Fraud Detection in Financial Businesses Using Data Mining Approaches

A Thesis Submitted to the Committee on Graduate Studies
in Partial Fulfillment of the Requirements for the Degree of Master of Science
in the

Faculty of Arts and Science

Peterborough, Ontario, Canada

Applied Modelling and Quantitative Methods (M.Sc.) Graduate Program

January 2020

ABSTRACT

Fraud Detection in Financial Businesses Using Data Mining Approaches

Anissa Nour Moudarres

The purpose of this research is to apply four methods on two data sets, a Synthetic dataset and a Real-World dataset, and compare the results to each other with the intention of arriving at methods to prevent fraud. Methods used include Logistic Regression, Isolation Forest, Ensemble Method and Generative Adversarial Networks. Results show that all four models achieve accuracies between 91% and 99% except Isolation Forest gave 69% accuracy for the Synthetic dataset.

The four models detect fraud well when built on a training set and tested with a test set. Logistic Regression achieves good results with less computational efforts. Isolation Forest achieve lower results accuracies when the data is sparse and not pre-processed correctly. Ensemble Models achieve the highest accuracy for both datasets. GAN achieves good results but overfits if a big number of epochs was used. Future work could incorporate other classifiers.

Keywords: Outliers, Isolation forest, Ensemble Method, Logistic Regression, SHARCNET, Compute Canada, GAN, Machine Learning, Feature Selection, Data Mining.

Acknowledgements

I owe many thanks to many people whose time and support made this thesis possible.

The first thanks go to my two supervisors Dr. Sabine McConnell and Dr. Richard Hurley for their patience, support, encouragement and help. Their support was essential in the completion of this thesis.

To my wonderful husband, Hashem, who gave me the power to carry on and the shoulder to cry on.

To my two little daughters: Salma and Haila who inspire me everyday with their cheerful laugh and positive energy.

To my parents who gave me a lot of support and help taking care of my kids and home while I am working on my paper.

To the woman that I consider a second mother who was always there pushing me forward to succeed my studies Dr. Deborah Berrill.

To my friends and fellow graduate students, Catharine west, Taylor Williams, Mark Weygang, Dongchul Lee, Cameron Champers, Rasha AlRaddadi and Vazken Minasyan. Whether in the grad office discussing a topic, solving a programming issue, or complaining about the workload and the lack of sleep. I enjoyed the time I spent with these exceptional people.

Big thanks to Kaggle competition website which helped me in coding many parts of my thesis work. Last but not least, I would like to thank all family, family in law and friends for all their love, support and prayers that always surrounded me.

This research was enabled in part by support provided by SHARCNET (www.sharcnet.ca) and Compute Canada (www.computecanada.ca).

Contents

List of Figures

List of Tables

List of Acronyms and Initialisms

AI	Artificial Intelligence
ML	Machine Learning
ANN	Artificial Neural Networks
GAN	Generative Adversarial Networks
SMOTE	Synthetic Minority Oversampleing Technique
PCA	Principal Component Analysis
ENN	Edited Nearest Neighbours
AUC	Area Under Curve
ROC	Receiver Operating Characteristic
SHARCNET	Shared Hierarchical Academic Research Computing Network
WGAN	Wasserstein Generative Adversarial Networks

Chapter 1

Introduction

1.1 What Is Fraud Detection

Fraud has been known since the beginning of mankind. With the introduction of new technologies, additional ways in which criminals may commit fraud are also introduced.

Fraud can cause billions of dollars of loss every year via fraudulent credit card transactions as the use of credit cards is prevalent in modern day society. For instance, in e-commerce the information about the card is sufficient to perpetrate a fraud [19]. Fraud affects banks, merchants and individuals through financial and non-financial losses. For example, if a cardholder is a victim of fraud through a certain company, he/she may no longer trust that company and end up choosing another company in which to do business.

Common ways to avoid fraud can be grouped into two categories: Fraud Prevention and Fraud Detection. In Fraud Prevention, technologies such as Address Verification Systems are used which, for example, verify addresses with zip codes. Similar approaches can be used for verifying a person through a Personal Identification Number. On the other hand, Fraud Detection is, given a set of credit card transactions, the process of identifying if a new authorized transaction belongs to the class of fraudulent or genuine transactions [47]. A Fraud Detection System (FDS) should not only detect fraud cases efficiently, but also be cost-effective in the sense that the cost invested in transaction screening should not be higher than the loss due to frauds [58].

Studies shows that screening only 2% of transactions can result in reducing fraud losses accounting for 1% of the total value of transactions. However, a review of 30% of transactions could reduce fraud losses drastically to 0.06% but increase the costs exorbitantly [9].

One solution to minimizing costs of detection is to use expert rules and statistical based models (e.g. Machine Learning), which can make a first scan for genuine and potential fraud then ask the investigators to review only the cases with high risk [9].

Financial fraud detection comes in a variety of styles:

- *Stolen card fraud:* In this case, the fraudster tries to use the card as much and as fast as he/she can. In detecting this case, we look for an unexpectedly high usage pattern of the credit card [58].

- *Absent card fraud:* In this case, the fraudster does not need the card itself, only information about a credit card. This kind of fraud usually occurs in e-business [58].

- *Application fraud:* This is where the fraudster uses a false personal information on a credit application. This kind of fraud is rare since most applications check personal information before allocating a card to an applicant [58].

1.2 How Fraud Affects Businesses

Since more and more people tend to use credit cards and e-transfer, instances of fraud are increasing and becoming more sophisticated. As a result, measuring the impact of fraud on businesses is more difficult.

The Association for Payment Clearing Services (APACS) has estimated that total losses through credit card fraud in the United Kingdom has grown rapidly from £122 million in 1997 to £440.3 million in 2010 [12]. According to the Nilson Report [6], global credit, debit, and prepaid card fraud losses reached $11.27 billion in 2012, up 14.6% over 2011. Gross fraud losses accounted for 5.22% of total volume, up from 5.07% in 2011. In 2012, fraud losses reached $5.33 billion in the USA alone [2]. According to the Lexis Nexis [28], in 2014 fraudulent card transactions worldwide have reached around $11 billion a year, with the USA accounting for about half of that.

The European Central Bank [64] reports that in 2012,€1 in every €2635 spent on credit and debit cards issued within *SEPA (the European Union, Iceland, Liechtenstein, Monaco, Norway and Switzerland)* was lost to fraud. The total value of fraud was estimated reaching €1.33 billion in 2012, registering an increase of 14.8% compared with 2011. In particular, 60% of fraudulent cases came from *card-not-present (CNP) payments* (i.e. payments via post, telephone or the internet), 23% from *point of sale (POS)* terminals and 17% from ATMs.

The introduction of the *EMV security standard* (chip on cards) has reduced fraud share (from 0.048% in 2008 to 0.038% in 2012) on the total number of transactions. However, from 2011 to 2012, CNP frauds have increased by 21%, following the growing of CNP payments, which rose by around 15% to 20% a year between 2008 and 2012 while other transactions rose by 4%.

The ECB report also shows that credit cards are more affected by fraud than debit cards, estimating that for every €1000 we have €1 of loss due to fraud in credit cards compared to €1 for every €5400 in debit card. Another interesting fact is that CNP fraud is usually more frequent in mature card markets where there is an absence of significant growth or a lack of innovation, whereas POS fraud is more common in less developed markets that have not reached a state of equilibrium [64]. The 2015 CyberSource report [69] shows that businesses are reluctant to adopt the 3-D Secure methods (online authentications based on a three-domain model: acquirer, issuer and interoperability), because it may lessen customer experience and increase the risk of customers abandoning their purchases.

1.3 Fraud Detection and Machine Learning

Most organizations still use rule based systems as their primary tool to detect fraud. Rules can do an excellent job of uncovering known patterns, but rules alone are not very effective at uncovering unknown schemes, adapting to new fraud patterns, or handling fraudsters increasingly sophisticated techniques. This is where Machine Learning can be useful for fraud detection [47]. There is no single Machine Learning algorithm or method that works the best for fraud detection. Getting the best results depend on applying multiple Machine Learning algorithms. Studies have shown that combining a variety of supervised and unsupervised methods can be more effective than any single method alone [47].

1.4 Relation with Existing Research

This thesis is the product of reading and applying the following:
Logistic Regression: A type of Machine Learning algorithms that is used to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

Isolation Forest Algorithm: Another type of Machine Learning algorithm that can be used for discovering anomaly patterns in the given data by building an ensemble of Isolation Trees for a given data set, the algorithm then detects the anomalies which typically are the instances that have short average path lengths in the Isolation

Trees [45].

Ensemble Classification and Extended Feature Selection: An approach that combines the feature selection with decision forest constructions by using an extended wrapper method that results in selecting the best and most efficient features [30].

Generative Adversarial Networks(GAN): These are Deep Learning techniques that build up multiple layers of abstraction in order to learn hierarchies of concepts. GANs have achieved considerable success in generating convincing examples [33].

The goal of this thesis is to apply these four approaches on two different datasets to determine the relative benefits of each approach.

1.5 Thesis Overview

In Chapter 2, we will provide the background needed to understand outlier detection along with corresponding data mining methods that will be applied. We will also provide some basic information about the types of Machine Learning algorithms that will be utilized. Chapter 3 focuses on applying the preprocessing methods to the desired datasets and then creating the models based on our selected algorithms. Chapter 4 presents the analysis of the results generated from the two datasets. Finally, in Chapter 5, we present our conclusions and suggest future work.

Chapter 2

Background

This chapter explains some basic concepts and terminology that will be used frequently in this research.

2.1 Outlier Analysis

An outlier is a data point which is significantly different from the remaining data [1]. Outliers are also referred to as *abnormalities*, *discordant*, *deviants*, or *anomalies* in data mining and statistics. Usually, an outlier has a low chance of occurrence within a given data set. Outlier analysis tries to find unusual patterns in any dataset. If one encounters a variable whose typical values display a certain kind of central tendency, or a certain kind of pattern, and then sees patterns that do not fit these typical ones, an abnormally in the data might exist [1]. Typically, outliers cause problems for parametric analyses. However, not everyone agrees that they always pose a problem, or what to do about them even if do [1].

We can categorize various types of outliers as follows: [1]

- Data entry errors (human errors)
- Experimental errors (data extraction or experiment planning/executing errors)
- Data processing errors (data manipulation or data set unintended mutations)
- Sampling errors (extracting or mixing data from wrong or various sources)
- Natural (not an error, but novelties in data)

In addition, outliers can be divided into two major types: *univariate outliers* that can be found when looking at a distribution of values in a single feature space, and *multivariate outliers* that can be found in a n-dimensional space (of n-features). Looking at distributions in n-dimensional spaces can be very difficult for the human brain, that is why we need to train a model to do it for us [7].

Two approaches are commonly used to deal with outliers, either to delete outliers from the sample, or replace the outlier value with one that is less extreme. Most of the time, these solutions cause problems such as biased parameter estimates and under weighted or eliminated valid values [7]. Other solutions are to use one of the popular methods for outlier detection such as [7]:

- Z-Score or Extreme Value Analysis (parametric)
- Probabilistic and Statistical Modeling (parametric)

- Linear Regression Models (PCA, LMS)

Studies continue to discover better techniques to detect outliers and work on optimizing the methods that are currently used. In this research, we will use some of these techniques and discuss their results and drawbacks.

2.2 Feature Selection

Feature Selection is the process of automatic or manual selection of features which contribute most to the prediction variable or output in which we are interested. With databases growing in size, Feature Selection becomes even more important [22]. In classification, a dataset usually includes a large number of features that may be relevant, irrelevant or redundant. Redundant and irrelevant features are not useful for classification and they might even reduce the efficiency of the classifier regarding the large search space. This phenomenon is referred to the curse of dimensionality [73], which is explained later in this chapter.

The benefits of Feature Selection include reducing the computational costs, saving storage space, facilitating model selection procedures for accurate prediction, and interpreting complex dependencies between variables [31]. The features that are selected not only optimize the classification accuracy, but also reduce the number of required data for achieving an optimum level of performance during the learning process [11, 71].

Feature Selection methods usually include a search strategy, assessment measure, stopping criterion, and validation of the results [22]. The search strategy is a method used for producing a subset of candidate features for assessment. An assessment measure is applied to evaluate the quality of the subset of candidate features [75]. The objective of the stopping criterion is to determine when a decision process should stop, and validation is the study of the validity of the selected features with real world datasets. Search strategy and assessment measure are the two key factors in the Feature Selection process [75]. One important class of methods that is used for Feature Selection is the Filter and Wrapper class [11]. We will apply one of these methods in our research.

2.3 Feature scaling

Feature scaling is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step. Two well known methods are usually used for re-scaling data. *Normalization* scales all numeric variables to the range [0,1]. One possible formula is given below [61]

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (2.1)$$

where x_{min} is the minimum value of x and x_{max} is the maximum value. The other, *standardization* transforms the data to have a zero mean and unit variance. An example is shown below [61]:

$$x_{new} = \frac{x - \mu}{\sigma} \quad (2.2)$$

where x is the data point, μ is the mean and σ is the standard deviation.

2.4 Imbalanced data

A dataset is imbalanced if the classification categories are not about equally represented [16]. Usually, there is a large amount of data/observations for one class (referred to as *the majority class*), and much fewer observations for one or more other classes (referred to as *the minority classes*). Blind application and optimization of learning algorithms on imbalanced data sets will lead to models that can produce poor results [66]. In following section, we will highlight the main approaches to deal with imbalance.

2.4.1 Sampling techniques

Sampling is a commonly used approach for selecting a subset of data objects to be analyzed. Sampling can be very useful in data mining, and it is preformed because it may be too expensive or time consuming to process all the data [67]. The key for effective sampling is the following: sampling must be representative. That is, the sample is representative if it has approximately the same property (of interest) as the original set of data. So, the best we can do is to choose a sampling scheme that guarantees a high probability of getting a representative sample. As discussed next, this involves choosing the appropriate sample size and sampling techniques [67].

2.4.1.1 Undersampling

Undersampling is a technique used to adjust the class distribution of a data set (i.e. the ratio between the different classes/categories represented) and is useful if the two classes are not equally represented in the dataset. When one class is underrepresented in a dataset, the data is said to be unbalanced. In such problems, it is the minority class that is of interest. When the data is unbalanced, standard machine learning algorithms that maximise overall accuracy tend to classify all observations as majority class instances. This translates into poor accuracy on the minority class (low recall), which is often the class of interest [20].

Several techniques can be applied to resolve this issue:

- *Random Undersampling*: This is one of the simplest strategies to handle imbalanced data by undersampling the majority class. Figure 2.1 shows the graphical representation of Random Undersampling. The blue and black data points which represent the data points that were removed were selected randomly from the majority class until the data is balanced. Removing data will reduce the strain on storage and also improve run time. However, removing data might lead to loss of useful information [39].

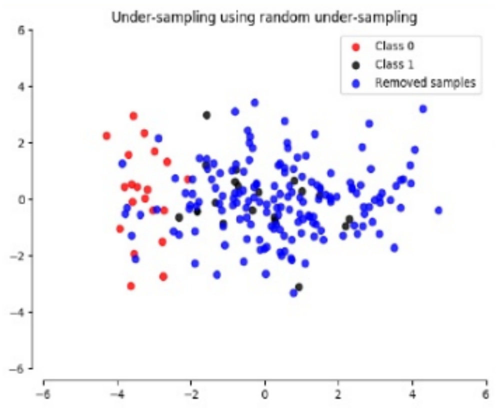


FIGURE 2.1: Graphical representation of Random Undersampling [39]

- *Cluster Centroids:* In this approach, a clustering technique is employed to re-sample the original training set into a smaller set of representative training exemplars, represented by weighted cluster centers and their target outputs [54].

- *NearMiss:* This is an undersampling technique that adds some heuristic rules to select samples. NearMiss heuristic rules are based on the nearest neighbors' algorithm, implementing three different types of heuristics. The first type selects samples from the majority class for which the average distance of the k nearest samples of the minority class is the smallest. The second type selects the samples from the majority class for which the average distance to the farthest samples of the negative class is the smallest. The third type is a two-step algorithm: first, for each minority sample, their nearest neighbors will be kept. then, the majority samples selected are the on for which the average distance to the k nearest neighbors is the largest [49].

2.4.1.2 Oversampling

Oversampling is another technique that is used to adjust the class distribution of a data set. There are a number of methods available to oversample a dataset that are used in a typical classification problem:

- *Random Oversampling:* This can be used to repeat some samples in order to balance the number of samples between the dataset. Random Oversampling involves supplementing the training data with multiple copies of some of the minority classes. Random Oversampling can be done more than once. This is one of the earliest proposed methods that is also proven to be strong [44]. Instead of duplicating every sample in the minority class, some of them may be randomly chosen with replacement.

- *Synthetic Minority Oversampling (SMOTE):* SMOTE is an oversampling approach in which the minority class is oversampled by creating *synthetic* examples rather than by oversampling with replacement [17]. To illustrate how this technique works, we will consider the following graph. SMOTE starts from a set of positive (green points) and negative (blue points) examples; It then selects a positive example (black) and its k nearest neighbors among the positives (yellow points, with k = 3), Finally one of the k nearest neighbours is randomly selected (brown point) and a new synthetic positive example is added, by randomly generating an example (red point) along the straight line that connects the black and brown points. The procedure depicted in (b) and (c) is repeated for all the positives, by adding each time a new synthetic example similar (in a Euclidean sense) to the other positive examples [62].

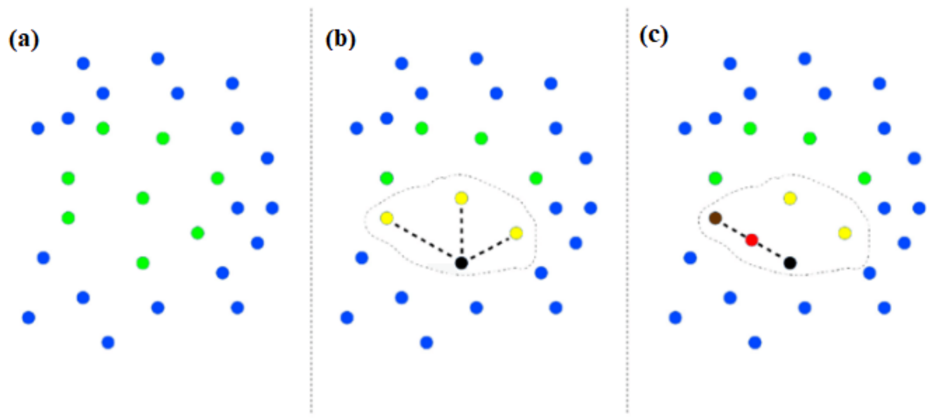


FIGURE 2.2: Graphical representation the SMOTE

2.4.1.3 Combined Oversampling and Undersampling

We will briefly talk about two methods that fall under this approach

- SMOTE-ENN: Basically, SMOTE-ENN is a statistical technique for increasing the number of cases in the dataset in a balanced way. The module works by generating new instances from existing minority cases that you supply as input. Basically, it is a class which performs oversampling using SMOTE and then cleaning using ENN (*Edited Nearest Neighbours*). Nearest neighbor editing aims to increase the classifier's generalization ability by removing noisy instances from the training set. [3]. Figure 2.3 illustrates how SMOTE-ENN works [3].

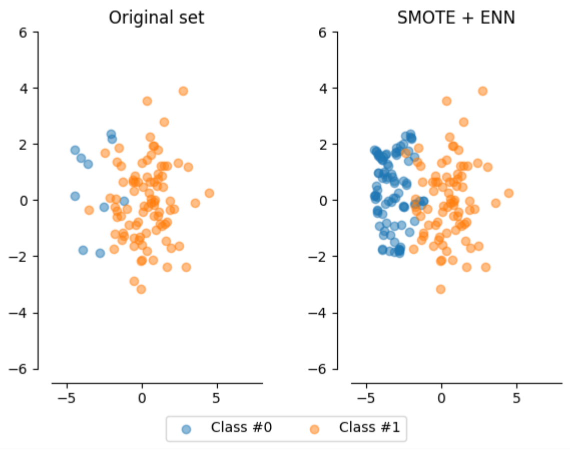


FIGURE 2.3: Before and after applying SMOTE+ENN

- *SMOTE-Tomek:* Another method to combine Oversampling and Undersampling is SMOTE-Tomek which is a class to perform Oversampling using SMOTE and cleaning using Tomek links. Tomek links remove unwanted overlap between classes where majority class links are removed until all minimally distanced nearest neighbor pairs are of the same class [16].

  Figure 2.4 below [16] shows how Tomek links to the Oversampled training set as a data cleaning method. Thus, instead of removing only the majority class examples that form Tomek links, examples from both classes are removed [4].

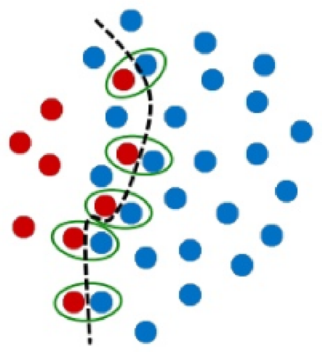


Figure 2.4: Removing examples from both classes using SMOTE-Tomek

2.5 Curse of Dimensionality

Another issue with high dimensional data is related to the curse of dimensionality [53]. In the case of input vectors with high dimensions, the learning process can be difficult. When a new dimension is added, the volume of the space increases and the data become sparse. This sparsity is problematic for any method that requires statistical significance. In order to obtain a statistically sound and reliable result, the amount of data needed to support the result often grows exponentially with the dimensionality [68]. Also, organizing and searching data often relies on detecting areas where objects form groups with similar properties. In high dimensional data, however, all objects appear to be sparse and dissimilar in many ways, which prevents common data organization strategies from being efficient [50]. One solution to the problem is feature extraction, where the data is transformed to a space with fewer dimensions. The transformation may be linear like with the Principal Component Analysis(PCA) [72] which is discussed later in this chapter.

2.6 Supervised Learning

In Supervised Learning, the algorithm performs the task of finding and inferring a function from the data that has been already classified to two or more classes [18]. The algorithm accepts a training set and produces an inferred function that can be used to classify new examples.

In general, a supervised algorithm needs an external component that classifies the input data first and organizes the data to be passed to the algorithm. The algorithm then is fed with the training data and the classification associated with it. The algorithm infers a classification function from the training set and is ready to accept new information to be classified. New data then is presented to the algorithm typically, based on the inferred function, the algorithm indicates the class of the formation. The algorithm typically does not learn more from the new data and the acquired knowledge is fixed.

The fundamental goal of Supervised Learning algorithms is to try to model relationships and dependencies between the target prediction output and the input features such that we can predict the output values for new data based on those relationships which it learned from the previous data sets. However, learning the training set may not be the best solution. This is because there are problems where we do not need to recall a solution, but to generalize the solution [8]. In other words, the algorithm will behave better if, instead of memorizing associations, it tries to find characteristics that make the classification possible. The problem of memorizing minor variations in the data that are not representation of the overall population is commonly referred as Overfitting [53]. Imbalanced datasets, which were previously explained, pose a difficulty that usually arises in Supervised Learning, especially when detecting outliers.

In the following sections, we introduce several methods that are often used for fraud detection and fall into the category of Supervised Learning approaches.

2.6.1 Ensemble Methods

Ensemble Methods are Machine Learning algorithms that construct a set of classifiers and then classify new data points by taking a (weighted) vote of their predictions [26]. Ensemble Methods are very compatible with unbalanced data and have demonstrated great performance in such settings [21]. The accuracy of the fraud detection model is a critical factor for a proper categorization of a fraudulent or legal case [46]. Advancements in Machine Learning suggest using a classifier ensemble instead of a single forecaster. Many studies show that an ensemble of classifiers will have better results than a single classifier [55].

On a study about classification algorithms indicates that ensemble methods considered to be more stable and, more importantly, predict better than single classifiers [43]. Research by M Young-Jong shows that ensemble methods are also known to reduce model bias and variance [37]. Finally, an article presented by S.Finlay has shown that model combination increases predictive accuracy [32]. Bagging, Boosting, and Random Forests are the most well-known examples of Ensemble Methods. Random Forests are very efficient for classification and regression problems [29]. A Random Forest is a collection of decision trees. Essentially, a decision tree splits the data into smaller data groups based on the features of the data until we have a small enough set of data that only has data points under one label [26]. Random forests can [25]

- handle binary features, categorical features, and numerical features.
- perform well with highly dimensional data since they rely on working with subsets of data.
- are robust to outliers and non-linear data.
- handle unbalanced data so the larger class will get a low error rate while the smaller class will have a larger error rate.
- Lead to low bias and moderate variance model. We say our model is biased if it systematically under or over predicts the target variable. In machine learning, this is often the result either of the statistical assumptions made by our model of choice or of bias in the training data. Variance, on the other hand captures

the generalizability of the model as it is a measure of how much our prediction would change if we trained it on different data. Because all the trees in random forest are averaged, the variance is averaged as well so that we have a low bias and moderate variance model.

2.6.2 Wrapper Methods

Wrapper Methods use the classifier as a black box and its performance as objective function for features subset assessment [13]. Wrapper approaches include a learning algorithm as assessment function. The Feature Selection criterion in Wrapper Methods is a forecasting function that finds a subset of the data with the highest performance [73]. Wrapper Methods are subdivided into exhaustive search, heuristic search, and random search.

- *Exhaustive search* such as *BFS (Breadth First Search)* enumerates all possible feature combinations. These methods are rarely used in practice since the time complexity would be $O(2^n)$ [48]. Non-exhaustive search methods are optimizations based on exhaustive search. For example, *Branch and Bound Search* saves time by cutting off branches that are unlikely to search for a solution better than the currently found optimal solution [10].

- *Heuristic search* which contains *SFS (Sequential Forward Selection)* and *SBS (Sequential Backward Selection).* The *SFS* starts from an empty set, then a feature x is added to the feature subset X so that the evaluation metric can be optimized. *SBS*, on the other hand, starts from the universal set and deletes a feature x each time. Both *SFS* and *SBS* are greedy algorithms that likely find a local optimum [73]. For the full feature set (of size m) and performing the search until the desired dimension d is reached, using SBS and SFS steps. The time complexity of these methods is $O(d)$ for SFS and $O(m - d)$ for SBS [57].

- *Random Search Methods* first randomly generate a subset of features and then apply other algorithms on that subset. For instance, *RGSS (Random Generation plus Sequential Selection)* perform *SFS* and *SBS* on a randomly selected subset of feature to jump out of the local optimum. However, random search methods depend on random factors so that experimental results are difficult to reproduce [10]. The complexity can be linear to the number of iterations in a random search, but experiments show that in order to find best feature subset, the number of iterations required is usually at least quadratic to the number of features [23].

Wrapper Methods usually provide the best performing feature set for a particular type of model [10]. However, they are very computationally intensive since for each subset a new model needs to be trained [10].

2.6.3 Logistic Regression

Logistic Regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary) [36]. It is named Logistic Regression because its underlying technique is quite similar to Linear Regression. The term *Logistic* is taken from the Logic function that is used in this method of classification. Like all regression analyses, the Logistic Regression is a predictive analysis. Logistic Regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. The importance of regression analysis lies in the fact that it provides a powerful statistical method that allows a business to examine the relationship between two or more variables of interest [35].

In order to deal with outliers, Logistic Regression uses a Sigmoid function. This mathematical function has a characteristic S shaped curve or Sigmoid curve. This function can carry any value between 0 and 1 as shown in Figure 2.5 and is defined by the following equation [35]

$$S(X) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1} \quad (2.3)$$

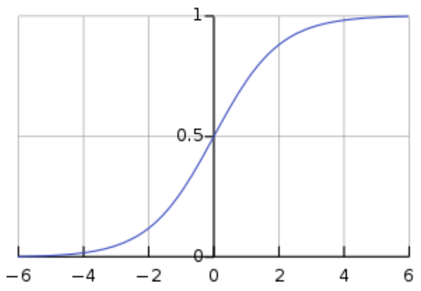


FIGURE 2.5: Sigmoid Curve

2.7 Unsupervised Learning

The goal of unsupervised learning is to find relationships in data that have no classification [53]. It relies on receiving unlabeled information and extracting from the data generalization features, if possible, of the relationships in the data.

There are several mechanisms to achieve this type of learning. One of the most common is clustering (also called automatic classification), where the data is grouped into sets in such way that all the elements of the group are similar. In general, clustering is not a specific technique, but it is a concept that can be achieved using different algorithms or combination of algorithms. In outlier detection, a simple unsupervised algorithm can be used to create a list of anomalies which can then be fed into an auditing process which then generates the true labels. Over time, when enough data labels are created, the unsupervised problem can be reformulated as a Supervised Machine Learning problem.

2.7.1 Isolation Forest

Isolation Forest is a method which in principle is similar to the well-known and Random Forest method by explicitly identifying anomalies instead of profiling normal data points. Like any tree ensemble method, Isolation Forests are built on the basis of decision trees. In these trees, partitions are created by first randomly selecting a feature and then selecting a random split value between the minimum and maximum value of the selected feature [45].

To understand the concept of Isolation Forest in detecting outliers, consider Figure 2.6 below. In a data-induced random tree, partitioning of instances is repeated recursively until all instances are isolated. This random partitioning produces noticeable shorter paths for anomalies since (a) the fewer anomalies result in a smaller number of partitions (i.e. shorter paths in a tree structure), and (b) instances with distinguishable attribute values are more likely to be separated in early partitioning. Hence, when a forest of random trees collectively produces shorter path lengths for some particular points, then they are likely to be anomalies [45]. Anomalies are more susceptible to isolation and hence have short path lengths. Given the Gaussian distribution in Figure 2.6 with 135 points, a normal point x_i requires twelve random partitions to be isolated while an anomaly x_0 requires only four partitions to be isolated [45].

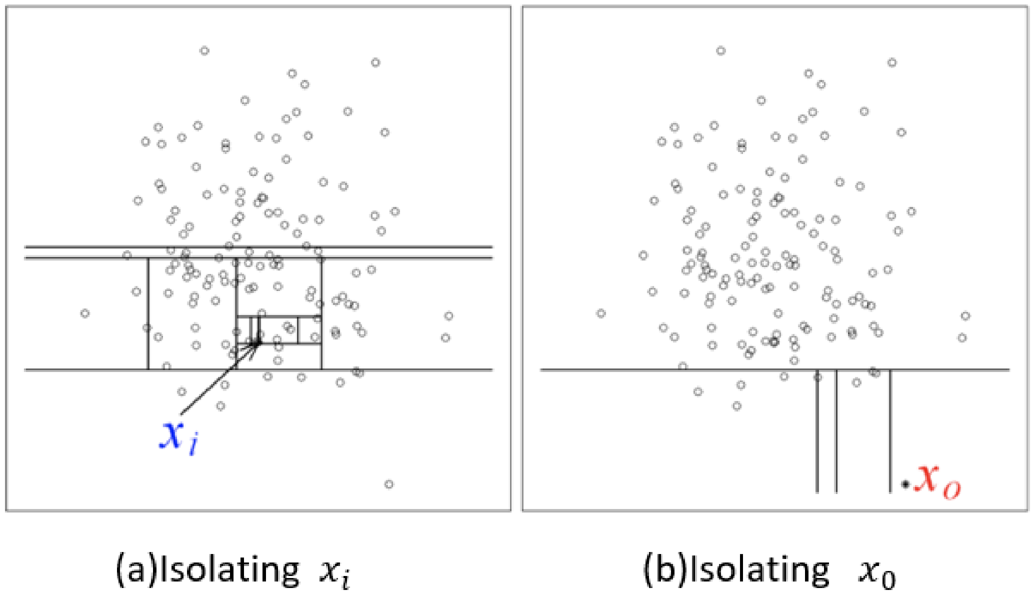


FIGURE 2.6: Difference between isolating a normal point and an abnormal point

As with other outlier detection methods, an anomaly score is required for decision making. In case of Isolation Forest, it is defined as:

$$S(x,n) = 2^{-\frac{E(h(x))}{c(n)}} \quad (2.4)$$

here *h(x)* is the path length of observation x, *c(n)* is the average path length of unsuccessful search in a Binary Search Tree and n is the number of external nodes [27]. Each observation is given an anomaly score, which is then used to compare samples.

Scores close to 1 indicates anomalies while, scores much smaller than 0.5 indicate normal observations. If all scores are close to 0.5, then the entire sample does not seem to have clearly distinct anomalies.

2.7.2 Generative Adversarial Networks

Generative Adversarial Networks (GAN) consists of two feed-forward neural networks: A *Generator G* and a *Discriminator D* competing against each other, with G producing new candidates and its adversary D evaluating their quality. Each of the two networks is usually a deep neural network with several layers connected in such as way that the output of the units in each layer becomes the input for the units in the layer immediately before [41].

The main idea in GANs (shown in 2.7) is to refine a generative model by making it confront an adversary, a discriminative model that has the goal of separating the generated examples from real ones. The generator takes random noise z as input, transforms it through a function and produces examples, while the discriminator learns to determine whether an example has been produced by the generator [33].

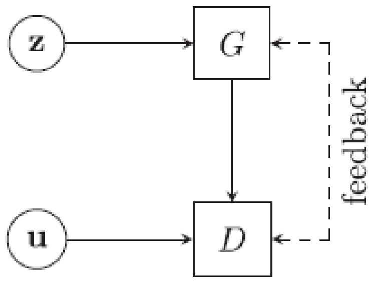


FIGURE 2.7: The generator G receives random noise z as input and the output is given to discriminator D, that distinguishes the examples produced by G from original data u [33]

To be more specific, the procedure of producing artificial candidates is explained with the following points:

- The generator's role is to produce new artificial candidates that are as close as possible to real data instances by learning the probability distribution of training data using the random noise z as input.
- The discriminator's role is to differentiate between real data and the artificial candidates
- Figure 2.8 shows how the trained generator G^* is merged with random noise z and its output is merged with the original training set X_t. The same classifier C is trained on the augmented C_a and the original training set C_0 [33].

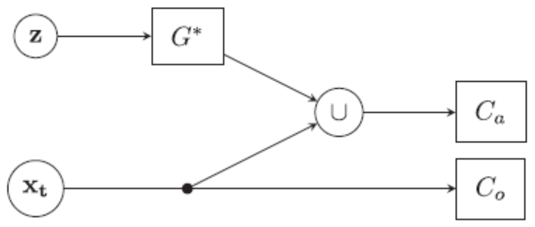


FIGURE 2.8: Producing artificial candidates

The training goal for the generator is to trick the discriminator into believing that generated examples are real. The discriminator is trained by minimizing its prediction error, whereas the generator is trained on the basis of maximizing the prediction error by the discriminator. This results in a competition between generator and discriminator that can be formalized by the following equation:

$$min_{\theta G} max_{\theta D}(E_{X\ p_D}[logD(x)] + E_{z|\ p_z}[log(1 - D(G(Z)))]) \quad (2.5)$$

where (p_D) is the data distribution, (p_Z) is the prior distribution of the generative network, and $\theta G\ /\theta D$ are the parameters of the (generator/discriminator) network. In other words, the goal of generator is to keep the difference between real and generated data to a minimum, whereas the discriminator aims to maximize the probability of distinguishing real data from generated ones [33].

2.7.3 Principal Component Analysis

This is a statistical process using orthogonal transformations to convert possibly correlated variables into linearly uncorrelated variables [53]. The orthogonal space after transformation is defined by having the first principal component with the largest variance with the constraint of uncorrelated components [53].

The idea to remove dimensionality is to drop the components that have lower variance and leave only those with great variance. Since the components removed are the ones with lower variance, the amount of information lost is controlled. The number of components removed may vary according to the amount of information needed. It is important to notice that this procedure is sensitive to the scaling of data. PCA reassuring the following steps [56].

(1) *Standardization:* The aim of this step is to standardize the range of the continuous initial variables so that each one of them contributes equally to the analysis. More specifically, the reason why it is critical to perform standardization prior to PCA as PCA is quite sensitive regarding the variances of the initial variables. That is, if there are large differences between the ranges of initial variables, those variables with larger ranges will dominate over those with small ranges. For example, a variable that ranges between 0 and 100 will dominate over a variable that ranges between 0 and 1, which will lead to biased results. So, transforming the data to a comparable scale can prevent this problem [56]. Mathematically, this can be done by subtracting the

mean and dividing by the standard deviation for each value of each variable.

$$z = \frac{value - mean}{standard \quad deviation} \qquad (2.6)$$

Once the standardization is done, all the variables will be transformed to the same scale.

(2) *Covariance Matrix Computation:* The aim of this step is to understand how the variables of the input data set vary from the mean with respect to each other, or in other words, to see if there is any relationship between them. Sometimes variables are highly correlated in such a way that they contain redundant information. So, in order to identify these correlations, we compute the covariance matrix [56]. The covariance matrix is a $p \times p$ symmetric matrix (where p is the number of dimensions) that has as entries the covariances associated with all possible pairs of the initial variables. For example, for a 3-dimensional data set with 3 variables x, y, and z, the covariance matrix is a 3×3 matrix of this from [56]:

$$\begin{pmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{pmatrix} \qquad (2.7)$$

Since the covariance of a variable with itself is its variance (i.e., Cov(a,a)=Var(a)), in the main diagonal (top left to bottom right) we actually have the variances of each initial variable. Since the covariance is commutative (Cov(a,b)= Cov(b,a)), the entries of the covariance matrix are symmetric with respect to the main diagonal which means that the upper and the lower triangular portions are equal. What matters is the sign of the covariance. So, if positive then the two variables increase or decrease together (correlated), if negative, one increases when the other decreases (inversely correlated) [56].

(3) The next step is to compute the *Eigenvectors* and *Eigenvalues* of the covariance matrix to identify the principal components. *Eigenvectors* and *Eigenvalues* are the linear algebra concepts that we need to compute from the covariance matrix in order to determine the principal components of the data. Principal components are new variables that are constructed as linear combinations or mixtures of the initial variables. These combinations are done in such a way that the new variables (i.e., principal components are uncorrelated and most of the information within the initial variables is squeezed or compressed into the first component). Thus, when you have 10-dimensional data, this gives you 10 principal components. What PCA tries to do is to put maximum possible information in the first component, then put the remaining information in the second until it reaches the maximum, so it turns to the third principal and so on. Figure 2.9 shows how PCA looks like at the end [56].

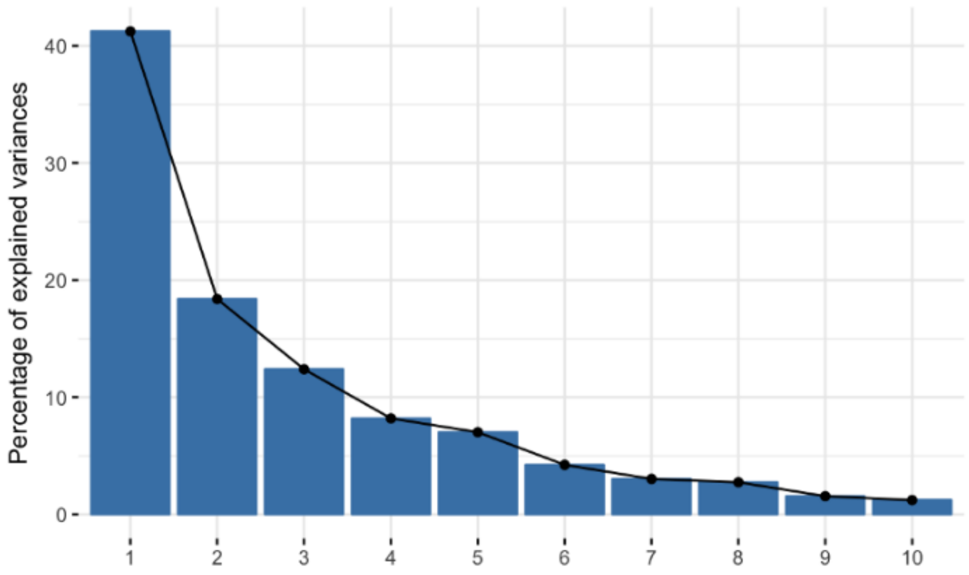


Figure 2.9: Principal components, percentage of variance (information) for by each PC

Organizing information in principal components this way will allow us to reduce dimensionality without losing much information. It accomplishes this by discarding the components with low information and considering the remaining components as the new variables. An important thing to realize here is that the principal components are less interpretable and do not have any real meaning since they are constructed as linear combinations of the initial variables.

Geometrically speaking, principal components represent the directions of the data that explain a maximal amount of variance, (i.e. the lines that capture most information of the data). The relationship between variance and information here is that the larger the variance carried by a line, the larger the dispersion of the data points along it, and the larger the dispersion along a line, the more the information it has. The idea is to think of principal components as new axes that provide the best angle to see and evaluate the data so that the differences between the observations are more visible.

There are as many principal components as variables in the data, principal components are constructed in such a manner that the first principal component accounts for the largest possible variance in the data set. The second principal component is calculated in the same way, with the condition that it is uncorrelated with, the first principal component and that it accounts for the next highest variance. This continues until a total of p principal components have been calculated, equal to the original number of variables [56].

Returning to *eigenvectors* and *eigenvalues*, we need to know is that they always come in pairs, so that every eigenvector has an eigenvalue. And their number is equal to the number of dimensions of the data. For example, for a three-dimensional data set, there are three variables, therefore there are three eigenvectors with three corresponding eigenvalues. It is eigenvectors and eigenvalues which are behind all the previous discussion, because the eigenvectors of the Covariance matrix are actually the directions of the axes where there is the most variance (most information) and what are called Principal Components. Eigenvalues are simply the coefficients attached to eigenvectors which give the amount of variance carried in each Principal Component. By ranking eigenvectors in order of their eigenvalues, highest to lowest, we get the principal components in order of significance [56].

Consider the following example [65]: suppose that the data set is 2-dimensional with 2 variables x,y and that the eigenvectors and eigenvalues of the covariance matrix are as follows:

$$v1 = \begin{vmatrix} 0.3778736 \\ \\ 0.7351785 \end{vmatrix} \quad \lambda 1 = 1.284028 \quad (2.8)$$

$$v2 = \begin{vmatrix} -0.7351785 \\ \\ 0.6778736 \end{vmatrix} \quad \lambda 2 = 0.04908323 \quad (2.9)$$

If we rank the eigenvalues in descending order, we get $\lambda 1 > \lambda 2$, that means the eigenvector that corresponds to the first principal component (PC1) is *v1* and the one that corresponds to the second component (PC2) is *v2* [65]. After having the principal components, to compute the percentage of variance (information) accounted for by each component, we divide the eigenvalue of each component by the sum of eigenvalues. If we apply this on the example above, we find that PC1 and PC2 carry respectively 96% and 4% of the variance of the data [65].

(4) *Feature vector:* As we saw in the previous step, computing the eigenvectors and ordering them by their eigenvalues in descending order, allow us to find the principal components in order of significance. For this step, we choose whether to keep all principal components or discard those of lesser significance (i.e. low eigenvalues), to create a matrix of vectors that is called the Feature Vector. So, the feature vector is simply a matrix that has the eigenvectors of the components that we decide to keep as columns. This is the first step towards dimensionality reduction because if we choose

to keep only p eigenvectors (components) out of n, the final data set will have only p dimensions.

Using the example from the Step(3), we can either form a feature vector with both of the eigenvectors $v1$ and$v2$:

$$\begin{vmatrix} 0.3778736 & -0.7351785 \\ 0.7351785 & 0.6778736 \end{vmatrix} \quad (2.10)$$

Discard the eigenvector *v2*, which is the one of lesser significance, and form a feature vector with *v1* only

$$\begin{vmatrix} 0.3778736 \\ 0.7351785 \end{vmatrix} \quad (2.11)$$

Discarding the eigenvector *v2* will reduce dimensionality by 1 where will cause a loss of information in the final data set. Given that *v2* was carrying only 4% of the information, the loss will be minimal as we still have 96% of the information that is carried by *v1* [65]. It is up to us to choose whether to keep all the components or discard the ones of lesser significance, depending on what is being considered. If we just want to describe the data in terms of new variables (principal components) that are uncorrelated without seeking to reduce dimensionality, leaving out lesser significant components is a recommended option [65].

(5) The last step is to recast the data the principal components axes. In the previous steps, apart from standardization, we do not make any changes to the data, we just

select the principal components and form the feature vector (i.e. the input data set remains always in terms of the original axes). In this step, the aim is to use the feature vector formed in step (4) to reorient the data from the original axes to the ones represented by the principal components. This can be done by multiplying the transpose of the original data set by the transpose of the feature vector [65].

$$FinalDataSet = FeatureVector^T \times StandardizedOriginalDataset^T \quad (2.12)$$

2.7.4 SHARCNET

The Shared Hierarchical Academic Research Computing Network (SHARCNET) is a consortium of 18 universities, colleges and research institutes providing a range of high performance computers and software, linked by an advanced fibre optics network [5]. Its overall aim is to promote the use of high-performance computing to accelerate the production of research results for researchers, Canadian industries, the economy and society in general. SHARCNET is a leading provider of this critical enabling technology and of training for highly qualified experts in computation. It plays a major role in supporting innovation in both academic and industrial research environments [5]. The SHARCNET clusters have a variety of storage systems available to users.

2.8 Summary of Chapter 2

In this chapter, we explained the basic concepts and terminology to be applied in the research. We talked about Outlier Analysis listing the types of outliers and the approaches used to deal with outliers. Then we defined Feature Selection illustrating its benefits and explaining more about Feature scaling and the terminology of normalization and standardization.

In this chapter, we also provide a detailed description of Sampling and Sampling techniques and explain the best time to use each. The chapter also includes definition of Curse of Dimensionality, Ensemble Methods, Wrapper Methods and Logistic Regression. We talked about Unsupervised Learning and explained Isolation Forest and GANs which falls under this conception. We finished our Chapter by explaining the PCA with an example and briefly talking about SHARCNET.

Chapter 3

Analysis, Modelling and Testing

This chapter describes the main concepts used for exploring and visualizing the data, and then illustrates the different algorithms used for training and testing the model. The actual methods for testing are described in this chapter with the results are analyzed in the following chapter.

3.1 Basic Steps

The main goal of this research is to check which machine learning algorithm is better to detect fraud in financial datasets. In order to do that, we use the following steps:

(1) Use two different financial datasets: one is a synthetic financial dataset and the other is a dataset from The European Central Bank. More details about the two datasets can be found in [74] and in [24].

(2) Use four machine learning algorithms to create four different models.

(3) Apply these four models on the two datasets.

(4) Compare the effectiveness and drawbacks of each model with the other.

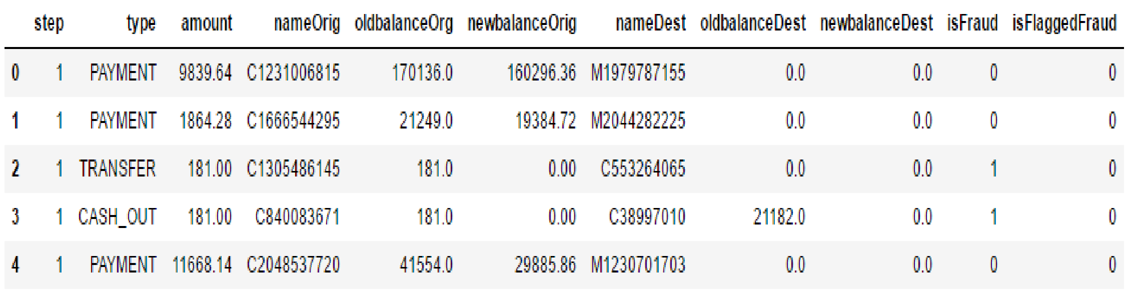
Before creating the models, the two datasets need to be preprocessed and prepared for modelling.

3.2 Preprocessing the Dataset

In order to prepare the data before we start modelling, basic steps need to be done. These steps are: Explore and analyze the data, visualize the data and prepare the data for modelling

3.2.1 Exploring and Analyzing the Synthetic Dataset

As we mentioned above, the first dataset we consider is a synthetic dataset. It was created with a specific reason: detect fraud in financial transactions. It contains around 23 Million records and 11 columns. Figure 3.1 below shows the first few records of this dataset with their features.



	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0

FIGURE 3.1: Head of synthetic dataset

Below is a list of the columns with a brief explanation:

amount- amount of the transaction in local currency.

nameOrig- customer who started the transaction.

oldbalanceOrg- initial balance before the transaction.

newbalanceOrig- new balance after the transaction.

nameDest- customer who is the recipient of the transaction.

oldbalanceDest- initial balance recipient before the transaction.

newbalanceDest- new balance recipient after the transaction.

isFraud- this has the transactions made by the fraudulent agents inside the simulation. In this specific dataset the fraudulent behavior of the agents aims to profit by taking control or customers accounts and try to empty the funds by transferring to another account and then cashing out of the system.

isFlaggedFraud- the business model aims to control large transfers from one account to another and flags illegal attempts. An illegal attempt in this dataset is an attempt to transfer more than $200,000 in a single transaction.

The data provided has the financial transaction data as well as the target variable *isFraud*, which is the actual fraud status of the transaction and *isFlaggedFraud*, which is the indicator the simulation uses to flag the transaction using some threshold. The goal should be how can we improve the threshold to capture the fraud transaction.

The dataset has five different types of transactions. The five types of transactions are:

CASH_OUT: where money is sent from a client to a merchant who then pays another customer in cash. The number of transactions for this type is 2237500 transactions.
PAYMENT: this includes a regular payment that the client pays in a certain amount of time such as mortgages. Number of transactions is 2151495.
CASH_IN: where the client put cash in his account. Number of this type of transaction is 1399284.
TRANSFER: where money is sent from a client to a customer online, the number of this kind of transaction is 532909 transactions.
DEBIT: where the client uses his debit card to pay. Number of transactions is 41432.
What is of concern for this dataset is to know which transaction types have fraudulent activity and which do not. In order to do that, we first need to study the relationship between types of transactions and the number of the fraudulent samples for each. This is shown in Figure 3.2:

```
The types that have fraudulent transactions are ['TRANSFER', 'CASH_OUT']

The number of fraudulent TRANSFERs = 4097

The number of fraudulent CASH_OUTs = 4116
```

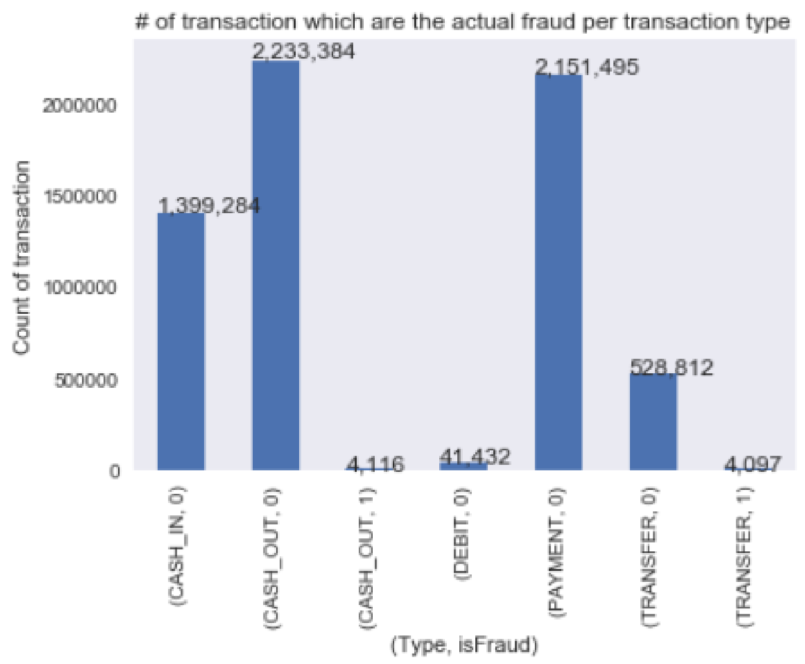


FIGURE 3.2: Number of transactions per transaction type

We can observe from Figure 3.2 that fraud occurs only in two of the five types of transactions:

TRANSFER where money is sent to a customer(fraudster) and, **CASH_OUT** where money is sent to a merchant who pays the customer (fraudster) in cash. What we also notice is that the number of fraudulent TRANSFER transaction is about the same as fraudulent CASH_OUT transaction. These observations imply that fraud is committed by first transferring funds to another account which subsequently cashes it out. We assume that ***isFraud*** indicates the actual fraud transactions whereas

isFlaggedFraud is what the system thinks it is a fraud and prevents the transaction due to some thresholds being triggered.

In the Figure 3.3, we plot the 5 types of transactions with the ***isFlaggedFraud*** and according to the results, only 16 transactions of the TRANSFER type are set as ***isFlaggedFraud*** with none were set as is ***isFlaggedFraud*** in the CASH_OUT type. This means that the system has suspected only 16 transactions that might be fraud.

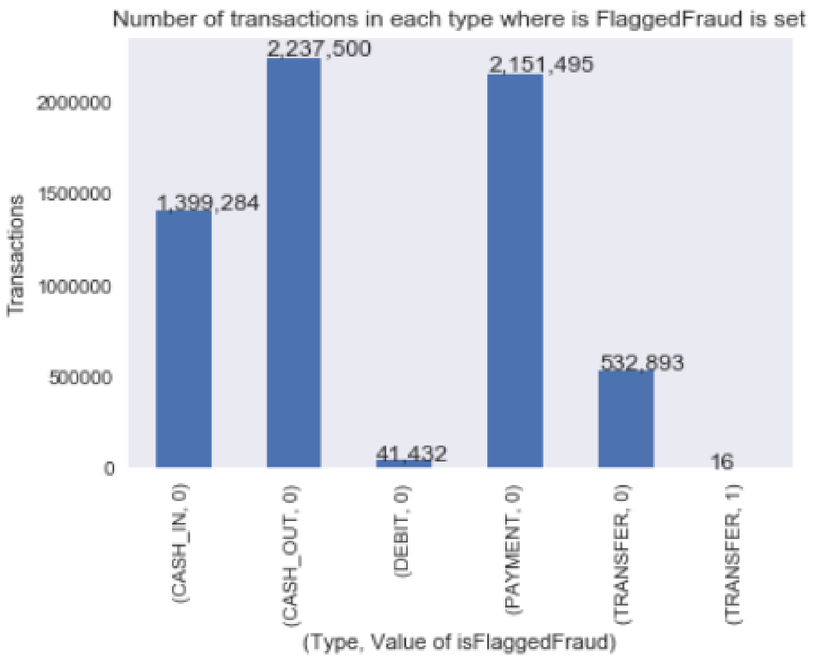


FIGURE 3.3: Number of transactions in each type where isFlaggedFraud is set

The preprocessing that this dataset can be summarized as following:

- Find the irrelevant features which can mislead and are not helpful in detecting fraud transactions and remove them.

- Correct the zero values which can be found in the feature ***newbalanceDest*** after a transaction is committed, and in ***oldbalanceDest*** before a transaction is committed.

3.2.1.1 Removing Irrelevant Features

We begin by discussing the feature ***isFlaggedFraud***. From its name we assume that this feature is responsible for indicating whether the transaction is most likely a fraud. In Appendix A we included several analyses in order to extract useful information about this feature. As a result, we arrived at the following conclusions:

- Duplicate customer names do not exist within transactions where ***isFlagged-Fraud*** is set, but duplicate customer names exist within transactions where ***isFlaggedFraud*** is not set.
- Originators of transactions that have the ***isFlaggedFraud*** set have only one transaction. Very few destination accounts of transactions have the ***isFlagged-Fraud*** set had more than one transaction.
- Since only 2 destination accounts with the ***isFlaggedFraud*** set have been destination accounts more than once, we can say that ***isFlaggedFraud*** is in-dependent of whether a destination account has been used before or not. Also, if we look at the relationship between ***isFlaggedFraud*** being set and *step col-umn* which is the column that shows the order of time that this transaction

happened, we can see that ***isFlaggedFraud*** is always on for all values of step so there is no relationship between those attributes.

- Finally, it seems that ***isFlaggedFraud*** was set 16 times in a meaningless way and thus we can discard that attribute in the dataset.

Two other attributes that should be discussed as to their importance are ***nameOrig*** and ***nameDest***. The question raised is: is there any need from these two attributes? The results extracted from the code in Appendix A show that there is no relationship between fraudulent transactions and ***nameOrig*** and ***nameDest*** features. As a result, we can say that ***nameOrig*** and ***nameDest*** features can be removed as they do not assist in the process of fraud detection.

3.2.1.2 Substituting Zero Values

The next step is to examine the zero values in the dataset. The data has several transactions with zero balances in the destination account, both before and after a non-zero amount is transacted. We need to determine if the value is an actual zero or is a missing value. From Appendix A, we can conclude the following:

- The percentage of transactions, where zero likely denotes a missing value, is much larger in fraudulent 50% compared to genuine transactions 0.06%. We assume that this there were no data entry errors (human errors) or experimental errors so most genuine transactions would be accurate while for fraudulent transactions, missing values might indicate that something is wrong with the transaction.

- The ***oldbalanceOrg*** and ***newbalanceOrg*** have values of 0 in some transactions where the amount being transacted is not 0. The percentages show that the number of transactions where this occurs is much smaller in fraudulent transactions 0.3% compared to genuine transactions 47%.

In order to address the issue of mixing fraudulent transactions and genuine transactions with missing values, we add two more features ***errorBalanceOrg*** and ***errorBalanceDest***. These two features will contain the same value as the amount being transacted in the transaction. The following formula are used to compute the amount:

$$errorBalanceOrg = (newbalanceOrig + amount) - oldbalanceorg \quad (3.1)$$

$$errorBalanceDest = (oldbalanceDest + amount) - newbalanceDest \quad (3.2)$$

3.2.2 Visualizing the Synthetic Dataset

We first concentrate on visualizing the new two features that we added to the dataset, ***errorBalanceOrig*** and ***errorBalanceDest***. These results are shown in Figure 3.4

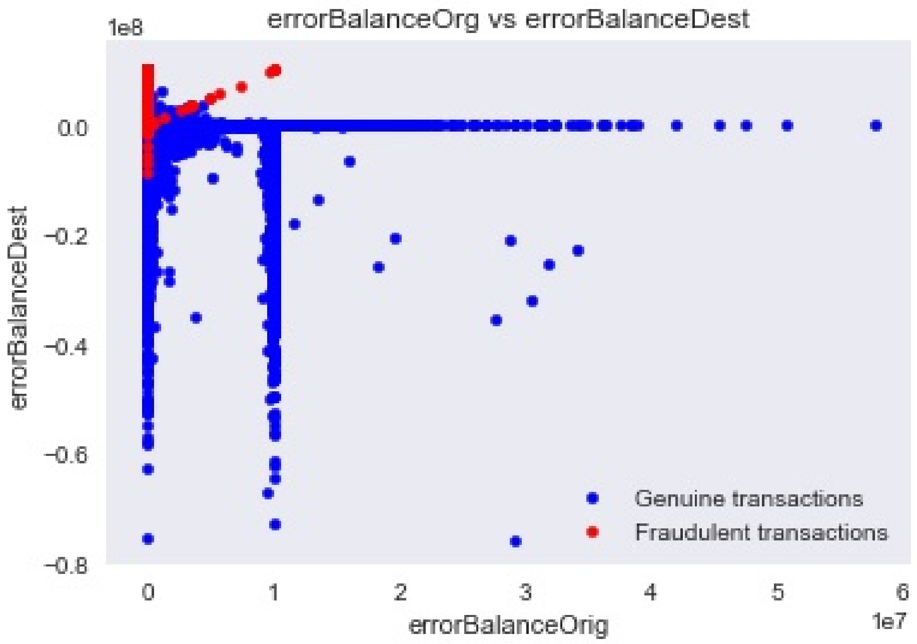


FIGURE 3.4: Genuine transactions vs Fraudulent transactions

The graph presents a comparison of genuine transactions and fraudulent transactions based on ***errorBalanceOrig*** and ***errorBalanceDest*** features. The genuine transactions, which are blue points, are concentrated in the top left corner of the graph where ***errorBalanceDest*** is larger than 0 whereas fraudulent transactions seem to appear more where ***errorBalanceDest*** is equal to or below 0.

The spread of errors in both the ***errorBalanceOrig*** and ***errorBalanceDest*** variables are large, however genuine transactions are much more likely to have an ***errorBalanceDest*** less than 0. On the other hand, fraudulent transactions are much more likely to have ***errorBalanceDest*** greater than 0. Figure 3.5 presents a three-dimensional examination of these two features

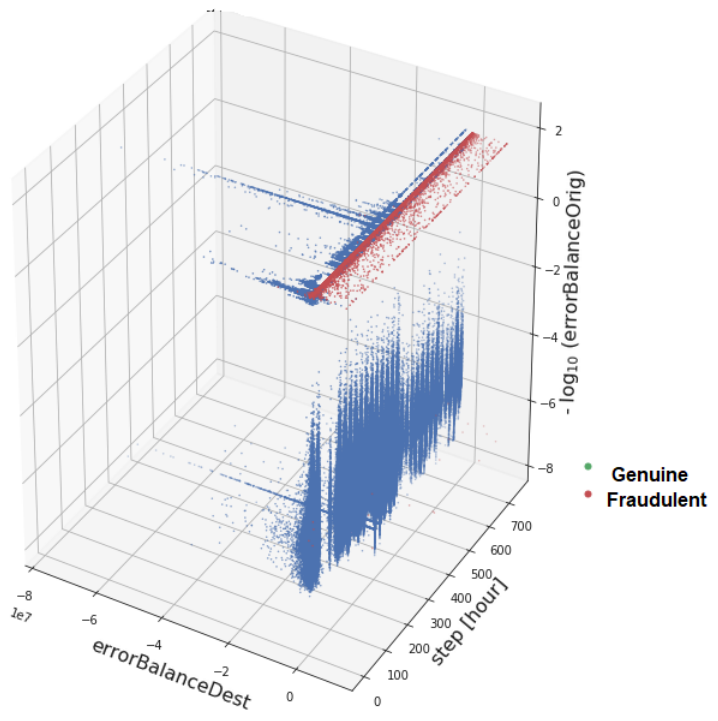


Figure 3.5: 3D plot for errorBalanceOrig and errorBalanceDest over time

By using both of the two error-based features, this plot is able to distinguish between fraud and non-fraud data as their dispersion is viewed over time, We see that the genuine transactions are likely distributed as strips while fraudulent transactions tend to be more homogeneously distributed from the time aspect. This might be caused by the fact that genuine transactions happen regularly with time, (i.e. a normal transaction happens in one step and another transaction from the same person happens in a second step). In fraudulent transactions however, this happen with no time order because a fraudster wants to complete as many transactions as possible.

The following *heatmap* in Figure 3.6 examines the respective correlations between the two kind of transactions.

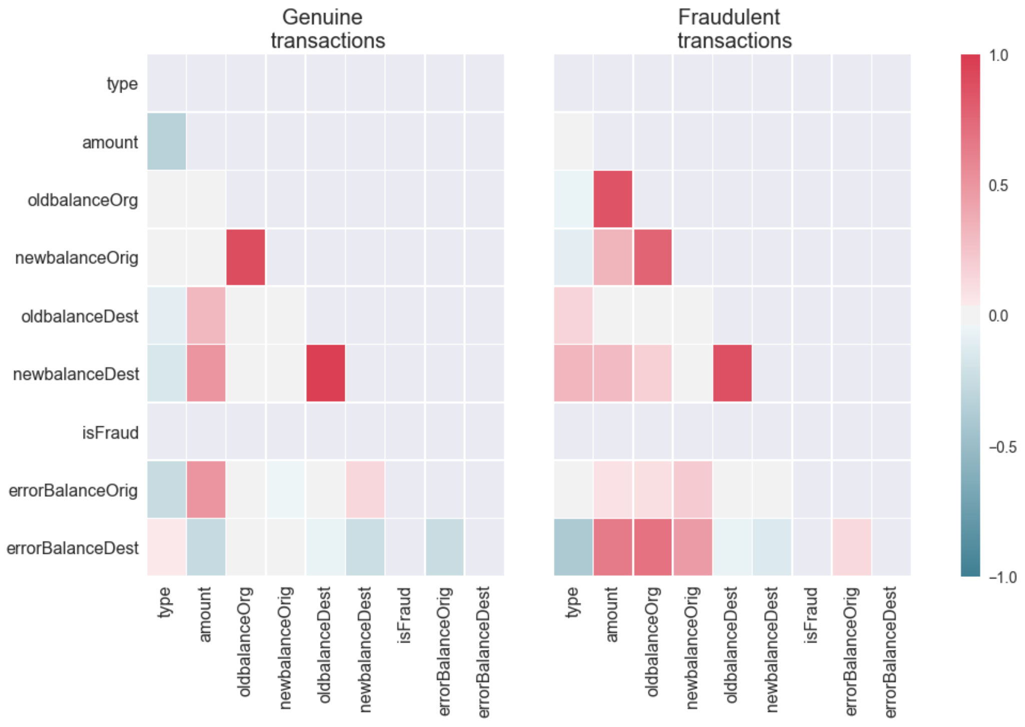


FIGURE 3.6: Heatmap for fraudulent and genuine transactions

We can see from Figure 3.6 that in fraudulent transactions, there is a strong correlation between ***amount*** and ***oldbalanceOrg*** in the fraudulent transactions, as the value of is larger than 0.25 and less than 0.5, while this correlation is weak in the genuine transactions the color is pale. Similar, in fraudulent transaction there is a strong coloration between ***errorBalanceDest*** and ***amount***, ***oldbalanceOrg*** and ***newBalanceOrig*** with the value that range from -1.0 to -0.25 while in genuine transactions, there is a weak correlation between ***errorBalanceDest*** and ***amount*** and no correlation between ***errorBalanceDest*** and both ***oldbalanceOrg*** and ***new-BalanceOrig***.

3.2.3 Preparing the Synthetic Dataset for Modelling

In order to test the data distribution, we divide the number of fraud transactions that we stored in an object *Xfraud* by the number of total transactions that we stored in an object *X*. The result shows the following:

skew = 0.002

The data is extremely skewed. In order to address this problem, we will oversample using SMOTE. Before applying oversampling, the counts in the genuine class is 193,538 while the counts in the fraud class is 5748. The oversampling will adjust the minority class distribution by adding more samples to it. The numbers after applying SMOTE are 1,933,538 for the genuine class and 1,933,538 for the fraud class.

3.2.4 Exploring and Analyzing the Real-World Dataset

The second dataset contains data from European credit card transactions. The dataset has been collected and analysed through a research collaboration between Worldline and the Machine Learning Group of ULB (Universit Libre de Bruxelles). It contains 284,808 records that have been collected over two days in 2013.

In this dataset, each transaction has a Boolean label assigned that indicates whether the transaction was in fact a fraudulent act. This labeling was performed by a team of human investigators who monitored the stream of transactions in near-real time. The dataset is highly unbalanced with the positive class (fraud) accounting for 0.172% of all transactions. Most features have been renamed for privacy reasons.

However, three features remain their names because they do not violate the privacy policy: *Time* which contains the seconds elapsed between each transaction and the first transaction in the dataset, *Amount* which is the amount of the transaction, and *Class* which indicates if the transaction was fraud or genuine. Features V1, V2,... V28 are the principal components obtained with PCA.

Table 3.1 shows specific descriptions for the two labels Genuine and Fraud

Distribution of the Two Classes		
Description	**Genuine**	**Fraud**
Count	284315	492
Mean	94838.2	80746.8
std	47484.0	47835.3
min	0	406
25%	54230	41241.5
50%	84711	75568.5
75%	139333	128483
max	172792	170348

Table 3.1: Numeric Description for Genuine and Fraud classes

As we mentioned earlier, the data is highly unbalanced: there are 492 transactions that were labeled as fraud and 284,315 as genuine transactions. Figure 3.7 contains a plot that shows Fraudulent transactions and Genuine transactions from the *Time* feature prospective.

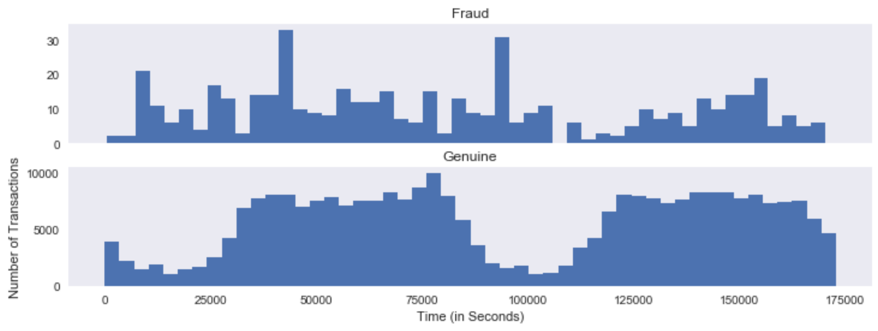


Figure 3.7: Compare Fraudulent to Genuine transactions in time

We can argue that fraudulent transactions are more uniformly distributed, while genuine transactions have a cyclical distribution. This could make it easier to detect a fraudulent transaction during at an off-peak time. Also, we can notice that there is some disconnection in the fraud plot while it is continuous in the genuine which could be related to the fact the fraud does not happens with any regularly genuine transaction.

Figure 3.8 shows a plot of the fraud and genuine transactions from the perspective of the Amount feature.

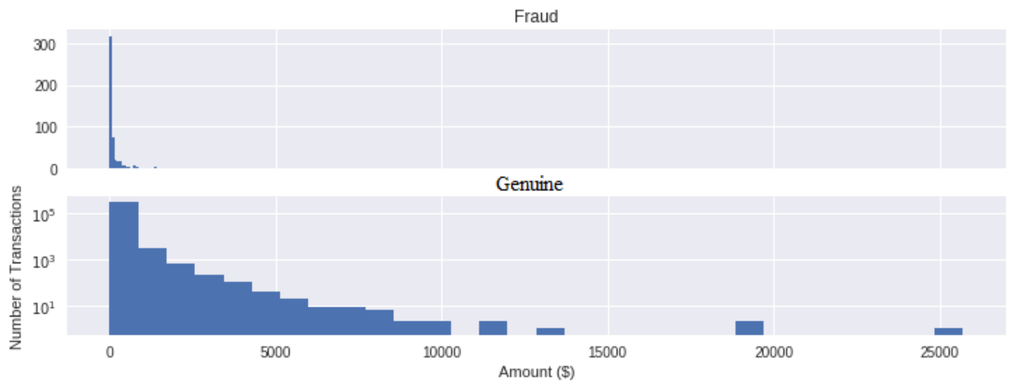


FIGURE 3.8: Fraudulent against Genuine transactions from the perspective Amount

Most transaction are Genuine transactions. They have continuous values start from 0 and reach 10,000. We can see few noncontinuous transactions with high values that reach $25,691

On the other hand, the maximum number of Fraud transactions carry low amount of money. The plot shows that the maximum amount of money a fraud transaction achieved is less than $2,125.

3.2.5 Visualizing the Real-World Dataset

We now take a closer look at the anonymized features in the second dataset by examining these features as a group using a correlation matrix, and then individually using a histogram, we begin with the correlation matrix of all the features which is shown in Figure 3.9

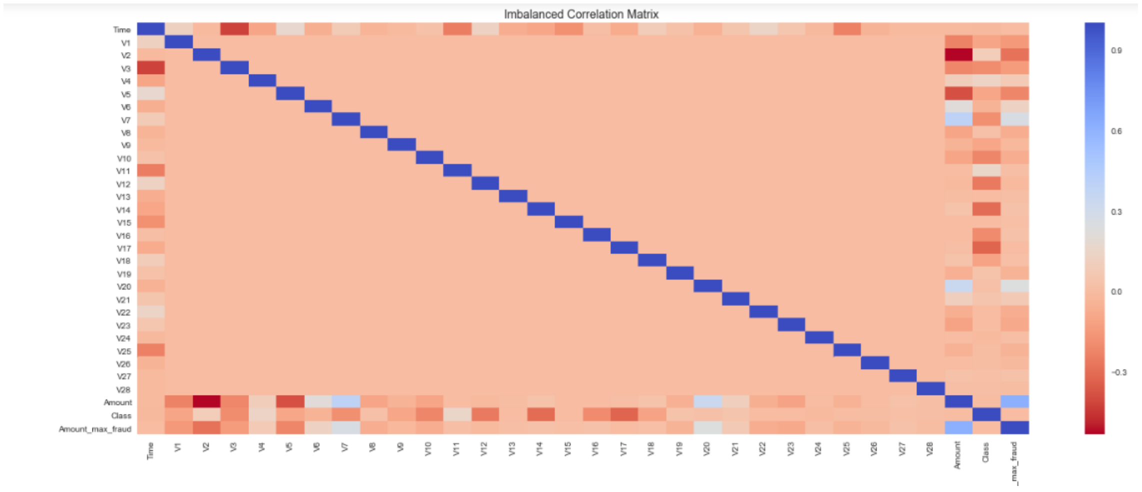


FIGURE 3.9: Correlation Matrix of the data

The blue color indicates the strong (positive) coloration between features. while the strong pink color indicates negative coloration. upon that we can say, V17, V14, V3, V12 and V10 are negatively correlated (notice the strong pink colour in the class row). The lower the values for these features, the more likely the end result will be a fraudulent transaction.

Positive Correlations: V2, V4, V11, and V19 tend to be positively correlated (notice the light pale pink colour in the class row). As the more these features have higher values, the more likely the end result will be a fraud transaction. Histograms for features (V1...V28) with the two classes (fraud and genuine) are shown in Figures 3.11-3.19

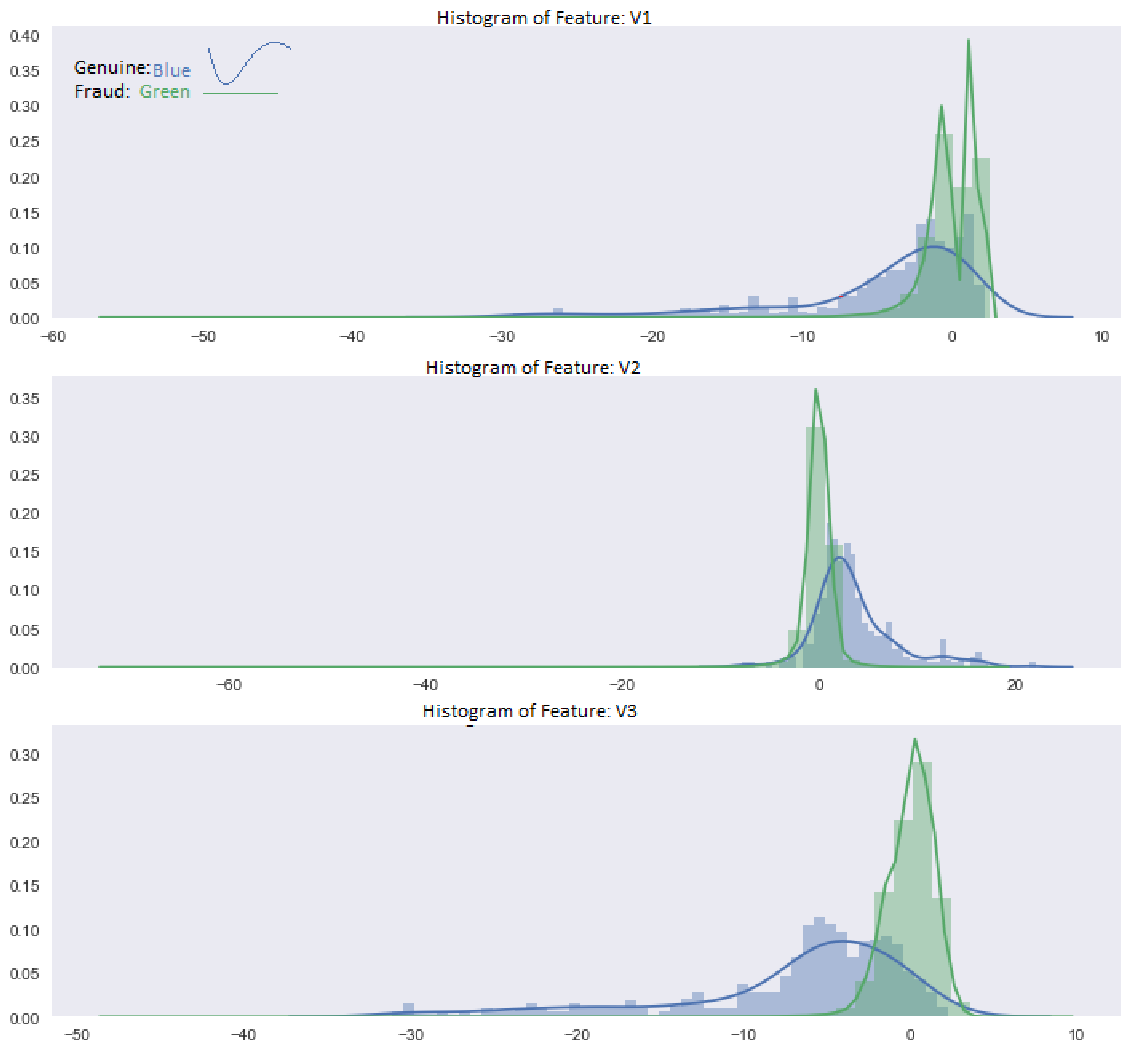


FIGURE 3.10: Distributions for Features V1,V2 and V3

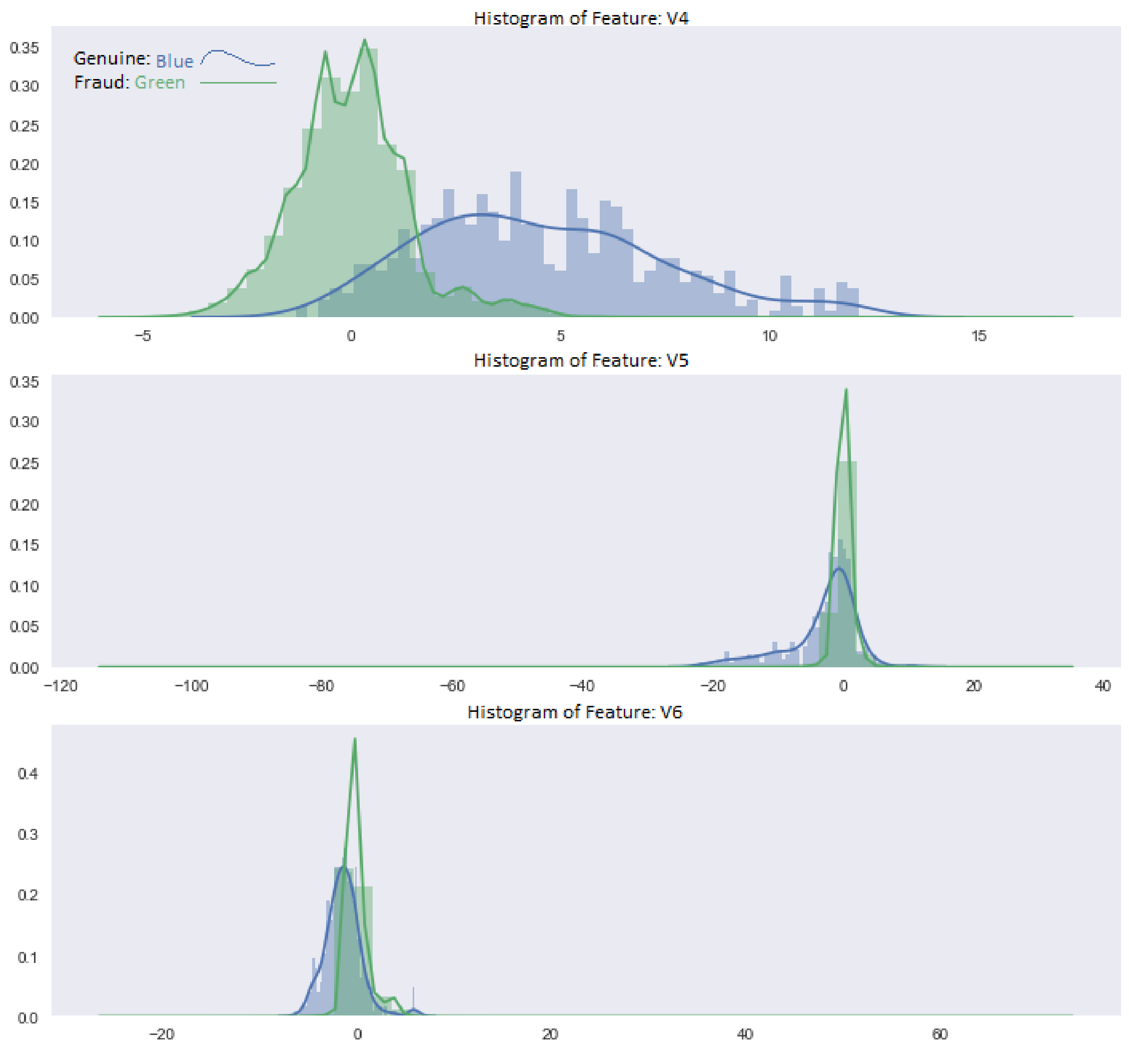


FIGURE 3.11: Distributions for Features V4, V5 and V6

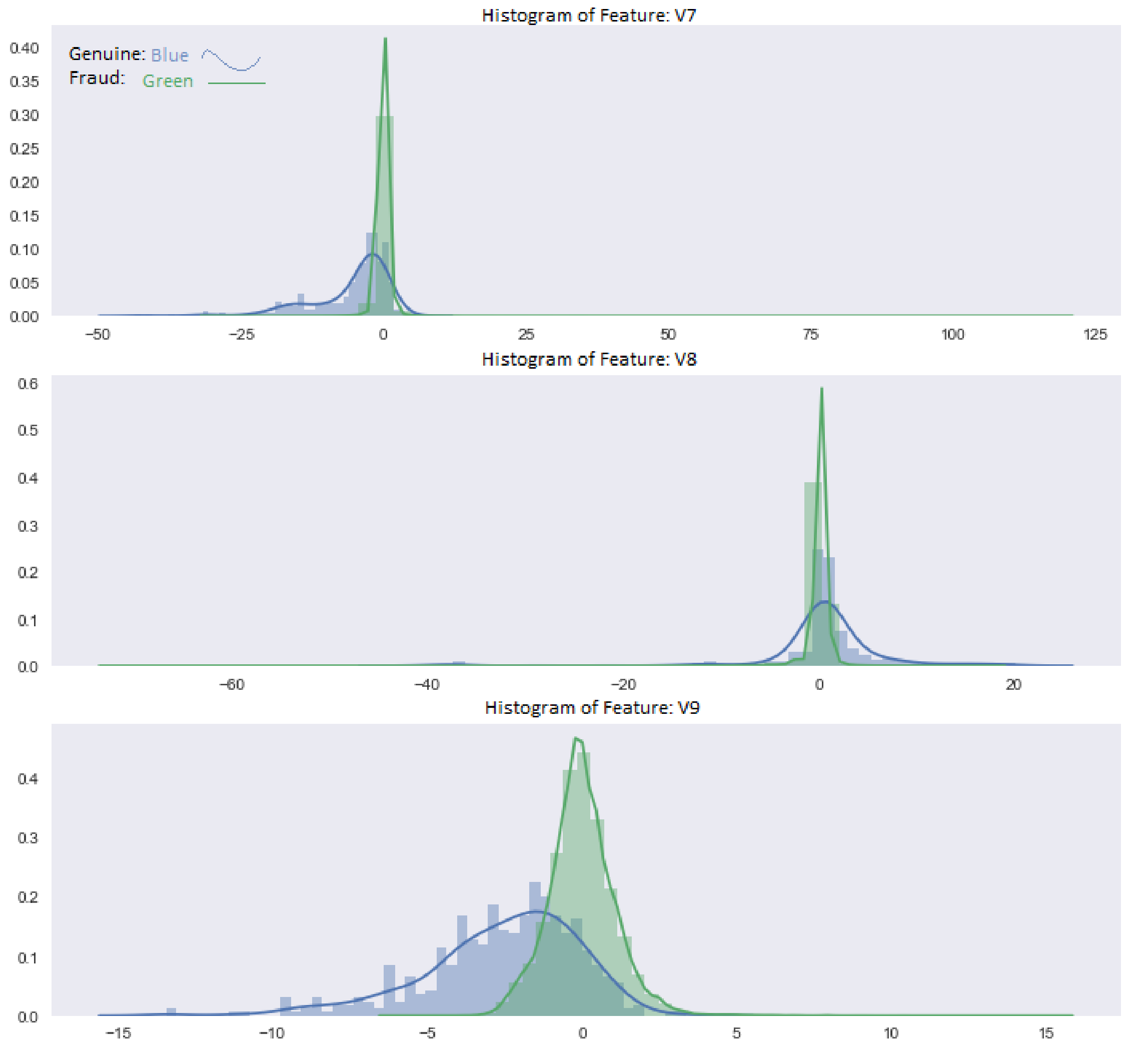


FIGURE 3.12: Distributions for Features V7, V8 and V9

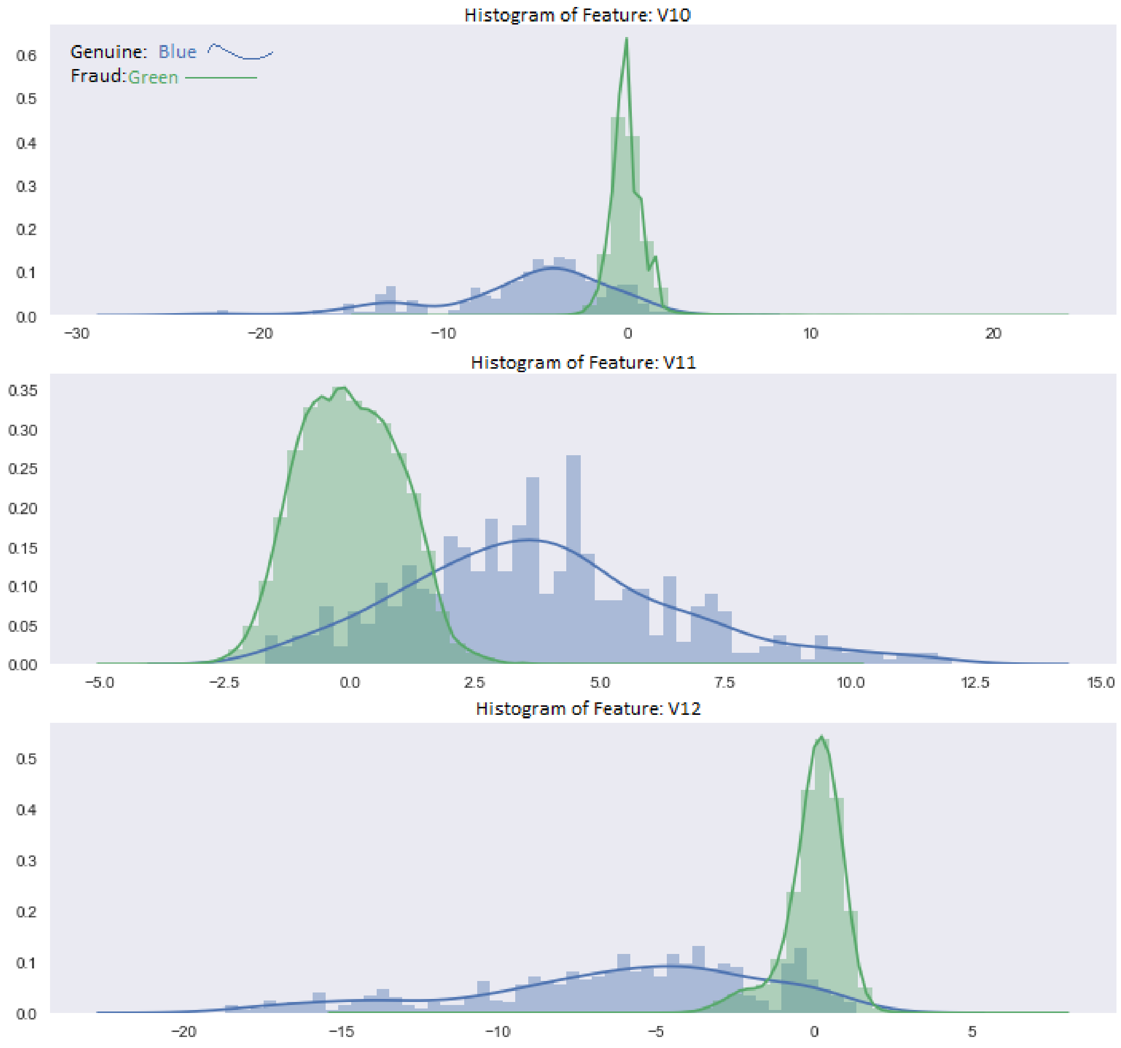


Figure 3.13: Distribution for Features V10, V11 and V12

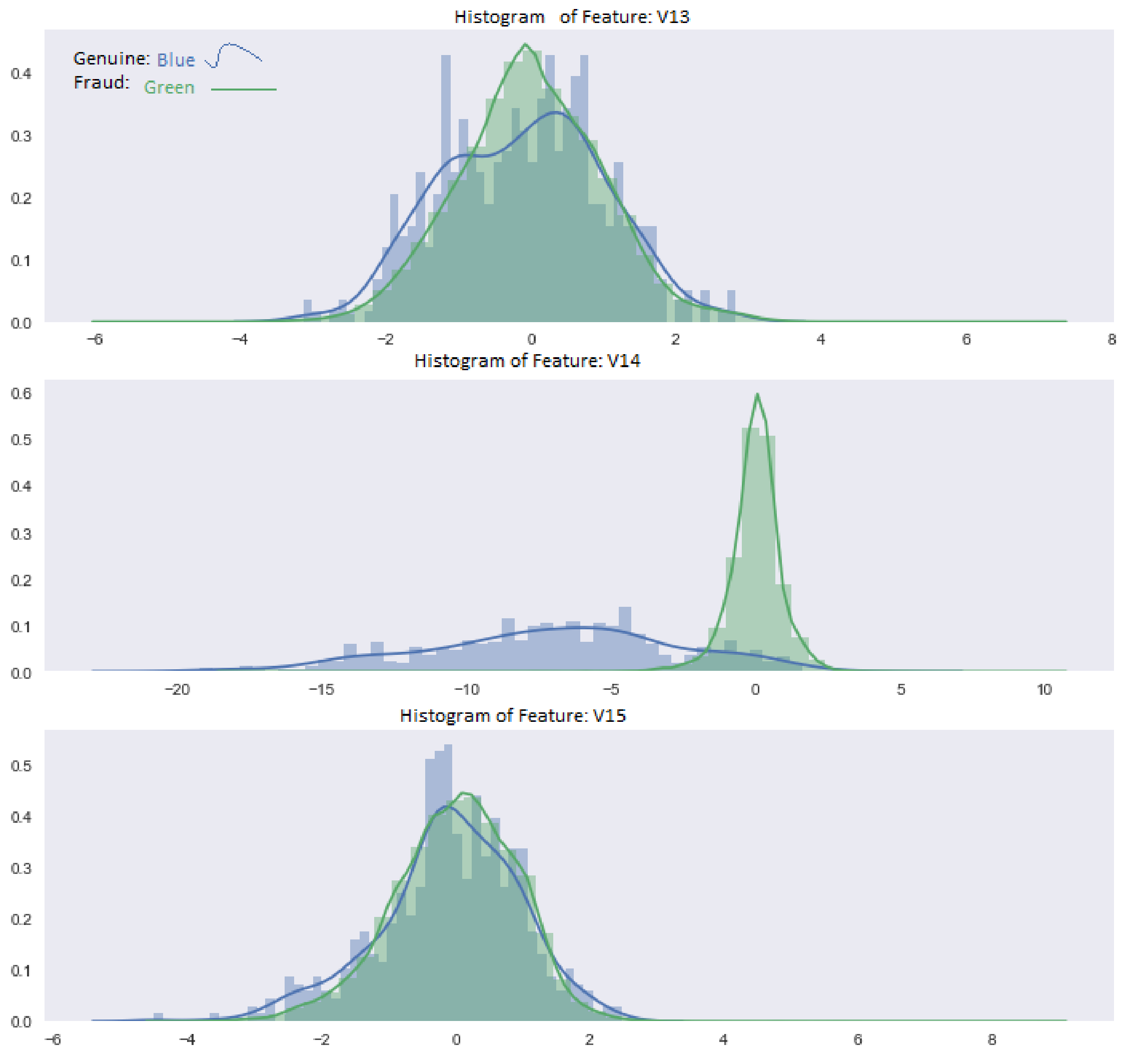


FIGURE 3.14: Distribution for Features V13,V14 and V15

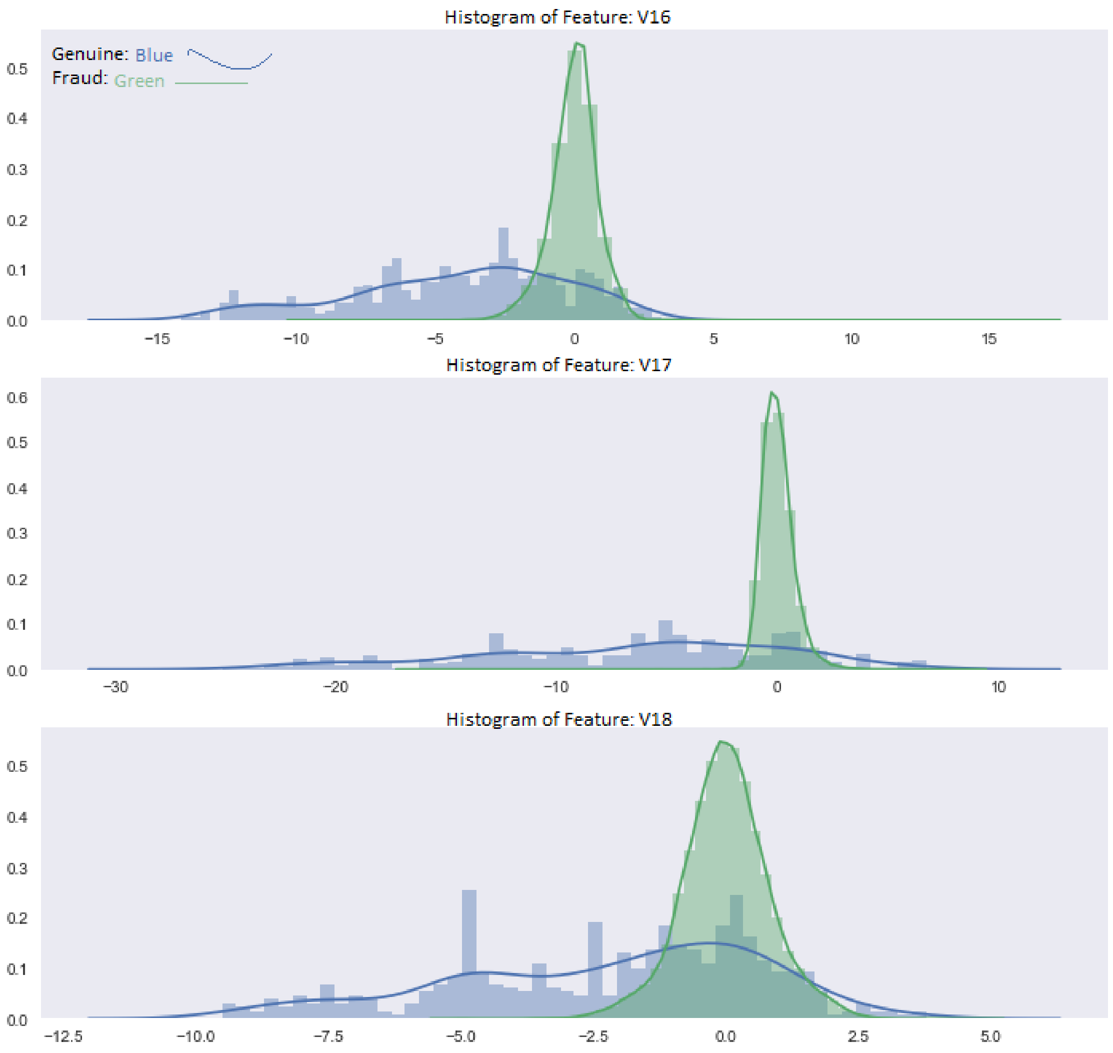


FIGURE 3.15: Distribution for Features V16,V17 and V18

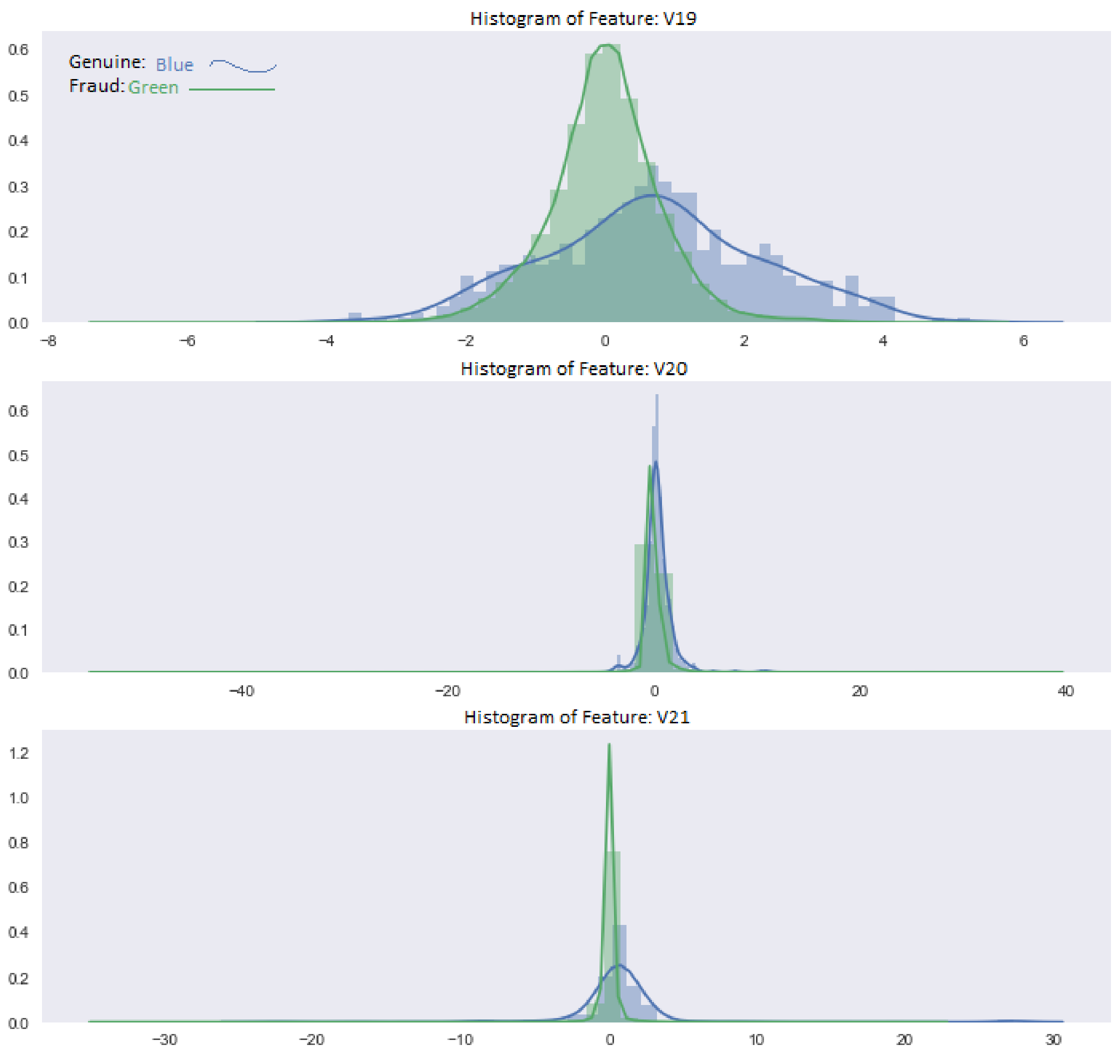


FIGURE 3.16: Distribution for Features V19,V20 and V21

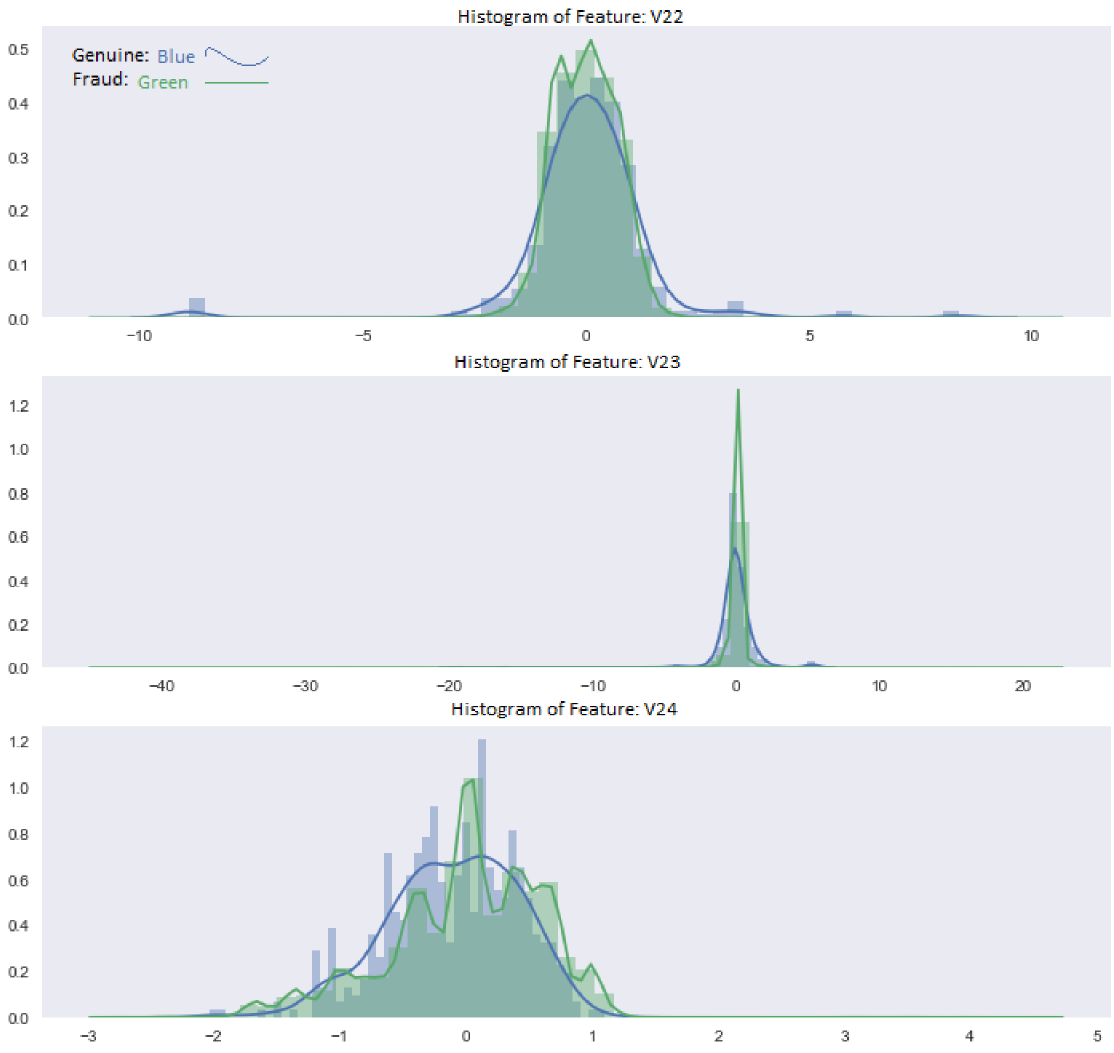


Figure 3.17: Distribution for Features V22,V23 and V24

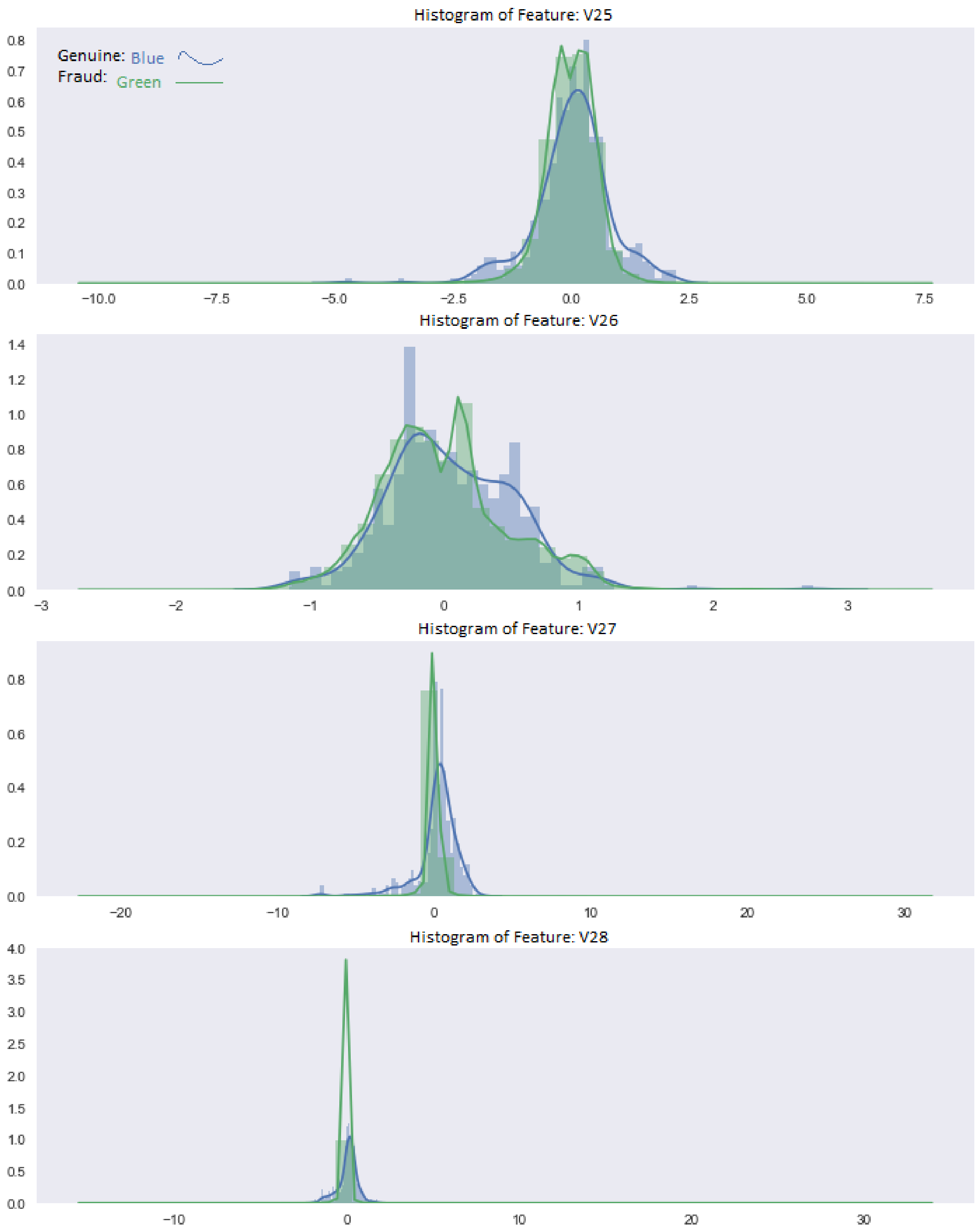


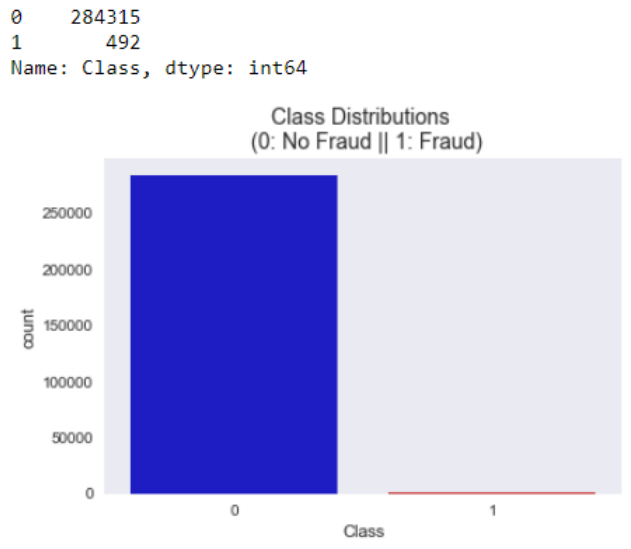
Figure 3.18: Distribution for Features V25, V26,V27 and V28

The distributions for Features V1, V2, V3, V4, V5,V6, V7, V9, V10, V11, V12, V14, V16, V17 and V18 are different for Genuine and Fraudulent cases. Thus, we can say that these features may carry important values that help determine if the transaction is fraud or genuine.

On the other hand, Features V8, V13, V15, V19... V28 have similar distribution for both labels which might imply these Features do not carry useful information that might affect fraud detection and could consider removing them from the dataset.

3.2.6 Preparing the Real-World Dataset for Modelling

We calculated the counts of both Genuine and Fraud class using Python. The result is shown in Figure 3.19.



```
0    284315
1       492
Name: Class, dtype: int64
```

FIGURE 3.19: Histogram shows the data is skewed

Looking at Figure 3.19 we can say that the data is extremely imbalanced. In this case, we handle the imbalance problem using Random Undersampling. Once this sampling technique has been selected, it is necessary to choose the sample size. Larger sample sizes increase the probability that a sample will be representative, but they also eliminate much of the advantage of sampling. Conversely, with smaller sample sizes, patterns may be missed or erroneous patterns can be detected [67]. We need to determine a sample size that would guarantee, with high probability, the desired outcome of good predictions. The result of Random Undersampling is shown in Figure 3.20 and 3.21:

```
Distribution of the Classes in the subsample dataset
1    0.5
0    0.5
Name: Class, dtype: float64
```

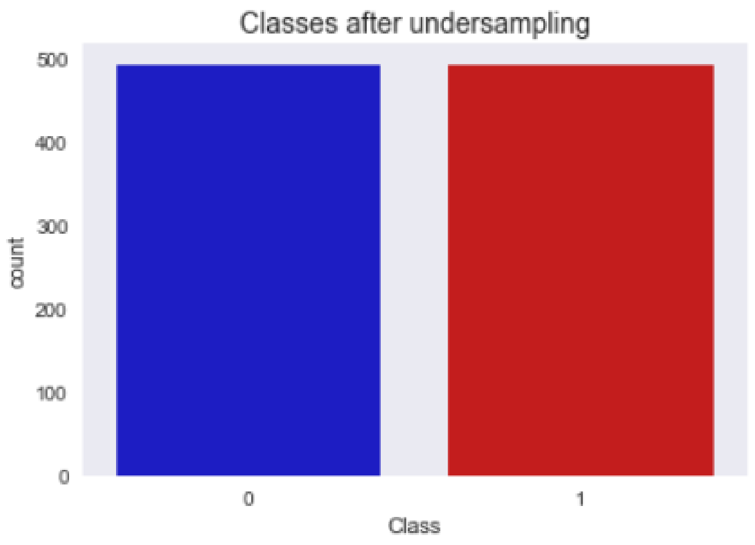


Figure 3.20: Histogram shows the data is now equally distributed

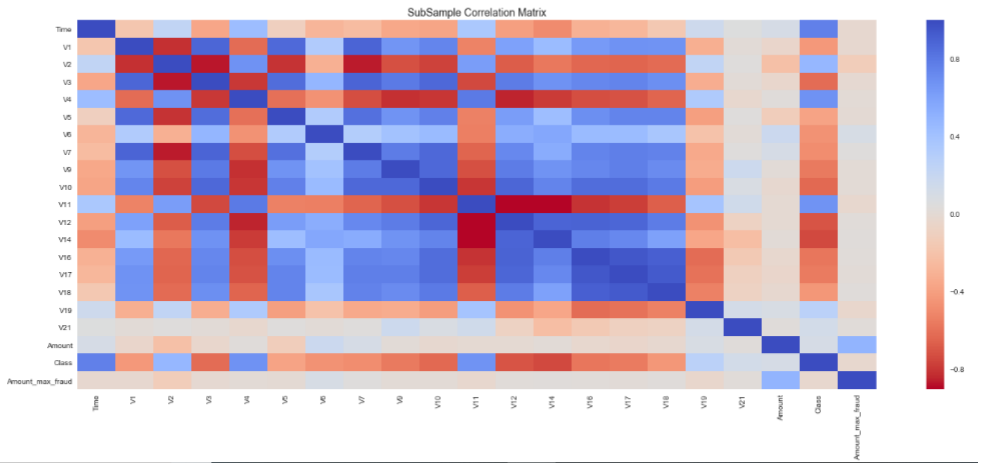


FIGURE 3.21: Correlation Matrix after sub-sampling

The correlation matrix above shows that the data now is fairly distributed compared to the Correlation matrix in Figure 3.9. We see positive coloration (blue color) between the class label and V2, V4 and V11. Our main aim is to know extreme outliers from features that have a high correlation with our classes. This will have a positive impact on the accuracy of our models.

3.3 Model Creation

Now the both datasets are prepared and ready for modelling, we will apply four different Machine Learning algorithms: Logistic Regression, Isolation Forest, Ensemble method and Generative Adversarial Networks and to each dataset and compare the results.

3.3.1 First Model: Logistic Regression

The first model, Logistic Regression, uses Sigmoid function to deal with outliers.

To begin the discussion, we start with an explanation of the standard logistic function. The logistic function is a Sigmoid function which map predicted values to probabilities by mapping any real value into another value between 0 and 1. It is defined as [38]:

$$S(X) = \frac{1}{1+e^{-x}} = \frac{e^{x}}{e^{x}+1} \quad (3.3)$$

A plot of the function is shown in Figure 3.22 [38]:

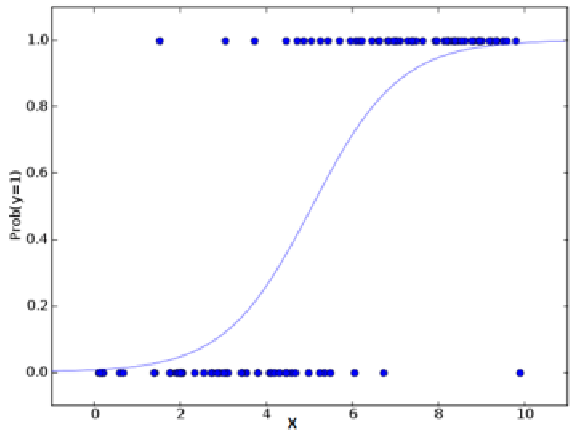


Figure 3.22: S curve for logistic regression

Logistic regression uses an equation as the representation, very much like linear regression

$$y = \frac{e^{(b0+b1*x)}}{1+e^{(b0+b1*x)}} \quad (3.4)$$

Where input values x are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value y. $b0$ is the bias or intercept term and $b1$ is the coefficient for the single input value x. Each column in input data has an associated b coefficient (a constant real value) that must be learned from training data.

When logistic regression model come across an outlier, it will detect it as shown in Figure 3.23 [38].

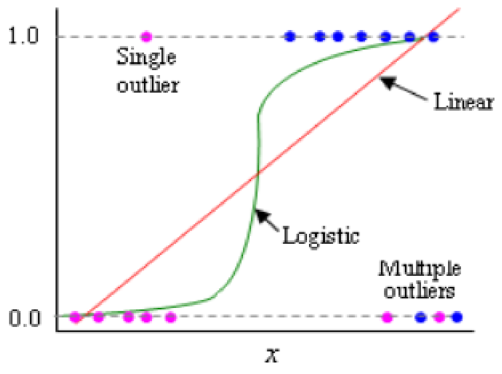
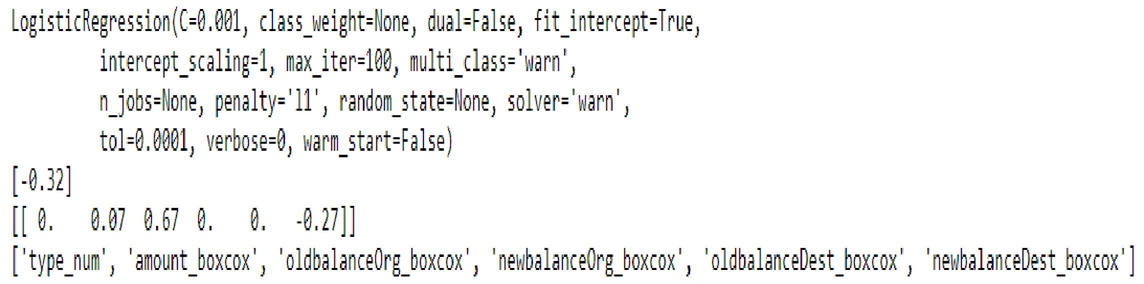


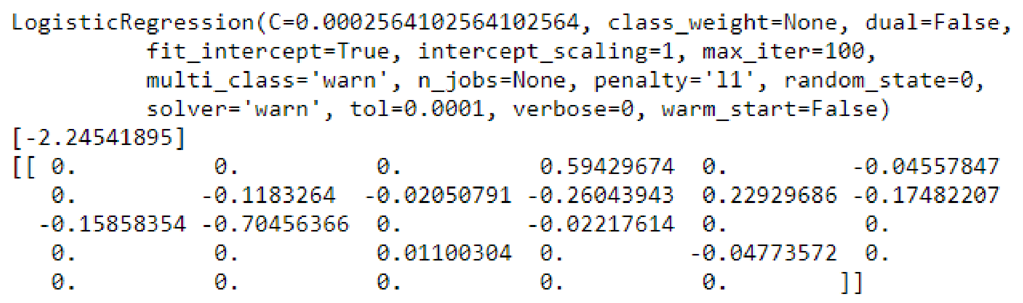
FIGURE 3.23: Outliers in Logistic Regression

Figure 3.24 and Figure 3.25 show the model creation with the corresponding parameters for each dataset.



```
LogisticRegression(C=0.001, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l1', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
[-0.32]
[[ 0.    0.07  0.67  0.    0.   -0.27]]
['type_num', 'amount_boxcox', 'oldbalanceOrg_boxcox', 'newbalanceOrg_boxcox', 'oldbalanceDest_boxcox', 'newbalanceDest_boxcox']
```

FIGURE 3.24: Logistic Regression Model, Synthetic Dataset



```
LogisticRegression(C=0.0002564102564102564, class_weight=None, dual=False,
          fit_intercept=True, intercept_scaling=1, max_iter=100,
          multi_class='warn', n_jobs=None, penalty='l1', random_state=0,
          solver='warn', tol=0.0001, verbose=0, warm_start=False)
[-2.24541895]
[[ 0.          0.          0.          0.59429674  0.         -0.04557847
   0.         -0.1183264  -0.02050791 -0.26043943  0.22929686 -0.17482207
  -0.15858354 -0.70456366  0.         -0.02217614  0.          0.
   0.          0.          0.01100304  0.         -0.04773572  0.
   0.          0.          0.          0.          0.        ]]
```

Figure 3.25: Logistic Regression Model, Real-World Dataset

3.3.2 Second Model: Isolation Forest

As it is mentioned in Chapter 2, Isolation Forest is an unsupervised learning algorithm that is commonly used in anomaly detection and scaled up to handle large, highly dimensional datasets. The goal is to return the anomaly score of each sample using the Isolation Forest algorithm [51]:

1- First, The Isolation Forest isolates observations by randomly selecting a feature.

2- Select a random split value between the maximum and minimum values of the selected feature.

3- The recursive partitioning can be represented by a tree structure.

4- Repeat the splitting. The number of splits required to isolate a sample is equivalent to the path length from the root node to the terminating node.

5- The path length, averaged over a forest of such random trees, is a measure of normality and our decision function.

Random partitioning produces noticeably shorter paths for anomalies since fewer instances of anomalies result in a smaller number of partitions (i.e. shorter paths in a tree structure). Hence, when a forest of random trees collectively produces shorter path lengths for particular samples, they are highly likely to be anomalies. [45]

Figure 3.26 illustrates the framework of the Isolation Forest algorithm [45]:

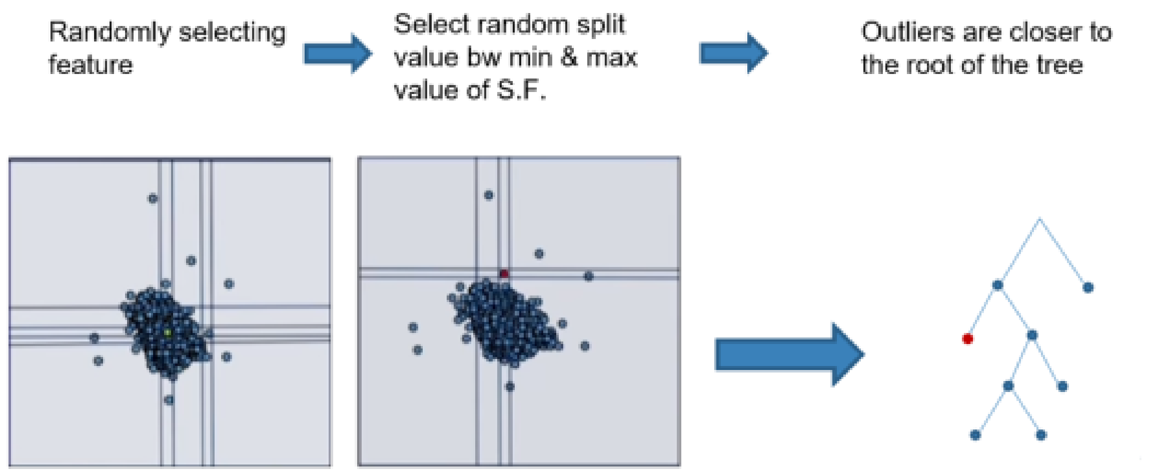

Randomly selecting feature
Select random split value bw min & max value of S.F.
Outliers are closer to the root of the tree

FIGURE 3.26: Illustration of Isolation Forest Algorithm

For each tree we get a sample of the data and then randomly select a dimension and randomly pick a value in that dimension. After that, we draw a straight line through the data at that value to split the data. We repeat until the tree is complete. We need to generate multiple trees to get a forest. Usually, anomalies will be isolated in only a few steps.

3.3.3 Third Model: Ensemble Method

The third model relies on an Ensemble method discussed in Section 2. Figure 3.27 illustrated the proposed model. The proposed method consists of two main parts: Feature Selection and Decision Forest construction. The first part includes creating a training dataset, selecting the best and the most efficient features by using Extended Wrapper method. The second part consists of dividing the dataset in several parts, creating a decision tree for each part, scoring each tree, and then choosing the best tree with the highest score in the decision forest.

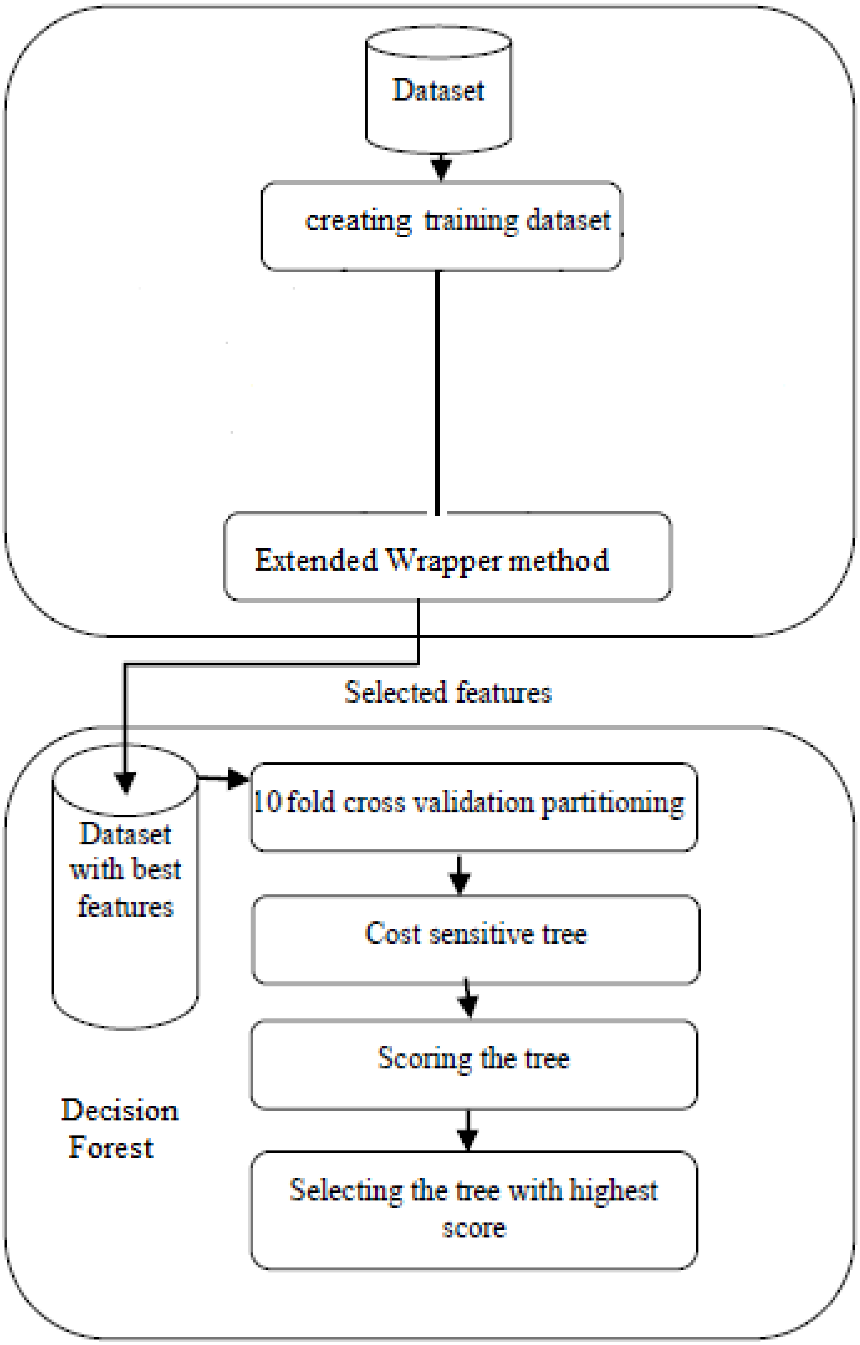

Dataset
creating training dataset
Extended Wrapper method
Selected features
Dataset with best features
10 fold cross validation partitioning
Cost sensitive tree
Scoring the tree
Selecting the tree with highest score
Decision Forest

FIGURE 3.27: Third model: Ensemble Method, adapted from [30]

3.3.3.1 Extended Wrapper Feature Selection

For this stage, the features are ranked based on the *Chi-squared filter*, *Gain Ratio* and *ReliefF*. These filters are appropriate and efficient filters for feature rankings [14]. The Chi-Squared Filter is a statistical test of independence to determine the dependency of two variables and is based on the X^2 statistics, it evaluates each feature based on the class labels separately.

The Gain Ratio filter is used to maximize information gain and ReliefF is a sample based filter that determines the volubility of a feature by repeated sampling and considering the value of a feature for discriminating a sample from a neighboring sample of a similar or a different class. Equations (3.4), (3.5), and (3.6) denote the Chi-Squared, Gain Ratio, and ReliefF filters, respectively [59]

$$X^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{\left(O_{ij} - E_{ij}\right)^2}{E_{ij}} \tag{3.4}$$

$$GR = \frac{IG}{H(X)} \tag{3.5}$$

$$W[A] == W[A] - \sum_{j=1}^{k} \frac{diff\left(A, R_i, H_j\right)}{m.k} \tag{3.6}$$
$$+ \sum_{C \neq class(R_i)} \frac{P(C)}{1 - P\left(class\left(R_i\right)\right)} \sum_{j=1}^{k} diff(A, R_i, M_j(C))$$

In (3.4), O_{i_j} is the resulting output when E_{i_j} is the target output. In (3.5), *IG* denotes the information gain [14]. The ReliefF measure, as denoted in (3.6) randomly selects an instance R_i and its k same class nearest neighbors, denoted by *Hj* and

k different class nearest neighbors denoted by $M_j(C)$. Then the ReliefF measure is updated for attribute A using the above-mentioned subsets. The contribution for each class is weighted with the prior probability of that class $P(C)$.

The second term is to ensure that the contribution of each class is in the range of [0, 1] and sums to 1 [59]. After the feature sets are created, the Chi-Squared, Gain Ratio, and ReliefF filters are integrated, and for each training subset, a candidate feature set is made. From the feature sets made by the three filters, the feature with the highest rank is selected. In order to choose the best features of different subsets, the candidate features of each subset that are selected based on their rank as classified by the C4.5[1] decision tree. The accuracy of the classifier is then determined. In case that accuracy of the classifier does not decrease, the feature is selected. However, if the feature being studied decreases the accuracy of the classifier, the feature is not selected, and the next feature is investigated (Pseudocode 1).

Algorithm 1 Selecting Best Features for each Phase

```
1: procedure Input: DATASET SELECTED FROM CANDIDATE FEATURE SET; Output: SELECTING THE BEST FEATURES.
2:     Steps 2 to 6 are repeated until all of candidate features are investigated.
3:     Classifier(C4.5) is made for each feature in candidate feature set.
4:     Classifier accuracy is calculated.
5:     The feature is selected and added to the best feature set if classifier accuracy does not decrease.
6:     Else the next feature is investigated.
7:     End.
```

[1] C4.5 is an algorithm used to generate a decision tree developed by Ross Quinlan. C4.5 is an extension of Quinlan's earlier ID3 algorithm. It became quite popular after ranking number 1 in the Top 10 Algorithms in Data Mining

3.3.3.2 Decision Forest

In this phase, the dataset with the selected features is divided into several non overlapping parts in order to create a decision forest (the number of parts in each decision tree is different) and with no overlap. For each part, a cost-sensitive decision tree is built. These types of trees take the misclassification costs (and possibly other types of cost) into consideration. Each tree is then ranked based on precision and F-Measure (a measure of test's accuracy). In the decision forest, the tree with the highest score is selected as the best one. For creating a cost-sensitive tree, the cost of each feature is calculated using *CS-Gini* [11]. The false negative and false positive decision costs are calculated using (3.7) and (3.8), respectively:

$$C_N = \left(\sum_{i=1}^{f} (C_{FN})_i \right) * \left(\frac{f}{n+f} \right)^2 \qquad (3.7)$$

Where *CN* is the total cost of wrong classification of legal transactions, *f* shows the number of frauds, *n* is the number of legal transactions, and the cost of a wrong classification of fraudulent transactions (CFN) is equal to 1.

$$C_P = n * C_{FP} * \left(\frac{n}{n+f} \right)^2 \qquad (3.8)$$

where *CP* is the total cost of a wrong classification for determining the transaction known as fraudulent. Also, the cost of a wrong classification of legal transactions (CFP) is 1.

After calculating the total cost of wrong classifications, the lowest cost of a wrong classification is selected as the feature cost using (3.9) [60]

$$\text{Cost(A)=min(C}_N\text{ ,C}_P\text{)} \quad (3.9)$$

Then the Gain Ratio is calculated for each Feature A using (3.10) [30].

In this equation, W shows the importance level of the feature:

$$\text{Rate(A)=}2^{Gain(A)}\text{-1/(Cost (A)+1)}^{W} \quad (3.10)$$

After calculating the Gain Ratio of each feature, the feature with the highest Gain Ratio is selected as the root of the tree. For the next step, the algorithm of the cost-sensitive decision tree is used, and the children of the root node are created. The algorithm is repeated for each child and shown in the following pseudocode

Algorithm 2 Decision Forest

```
1: procedure Input: Dataset with best features; Output: The best cost sensitive decision tree for fraud detection.
2:     Steps 2 to 6 are repeated until a tree with high score is selected.
3:     Training dataset is divided into several parts.
4:     Cost sensitive decision tree is made for each part of dataset.
5:     The tree with a high score between trees of decision forest is selected.
6:     End.
```

Figure 3.28 shows the best features selected from the second dataset which would be entered into the Decision Tree model.

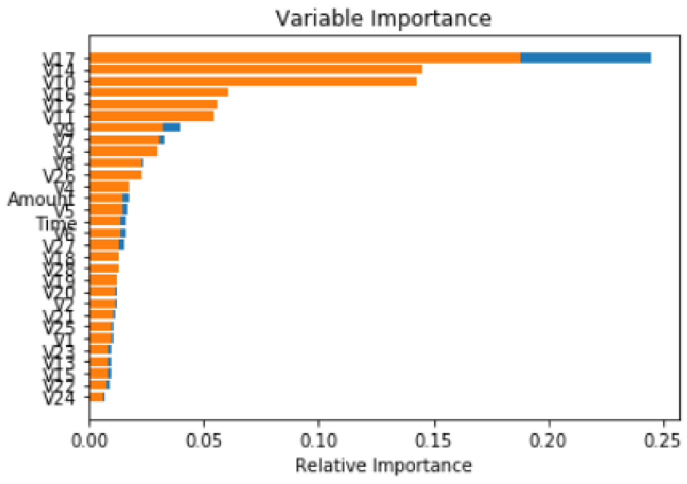


FIGURE 3.28: Most important features for the Real-World Dataset

Figure 3.28 shows that the most important features that has to be considered. For this example, V17 decreases the most entropy, and hence the most useful when splitting the plane. The decision tree will select splitting features in the order of most useful.

We can see from Figure 3.28 the first 3 features have high importance then importance reduced to half for the next 3 features then reduced again to the half for the 3 after until it reached a point where features have approximately the same value of low importance. The results and the accuracy of the model will be discussed in Chapter 4.

3.3.4 Fourth Model: Generative Adversarial Networks

As mentioned in Chapter 2, a GAN consists of two feed-forward neural networks: A Generator G and a Discriminator D competing against each other with G producing new candidates and its adversary D evaluating their quality.

The general imbalance problem has been dealt with by devising modified classifiers or by preprocessing data before any classification algorithm is applied. The purpose of training a GAN is to output mimicked minority class examples, which are then merged with training data into an augmented training set so that the effectiveness of a classifier can be improved [33]. Figure 3.29 that shows how Step 1 of the GAN framework is applied:

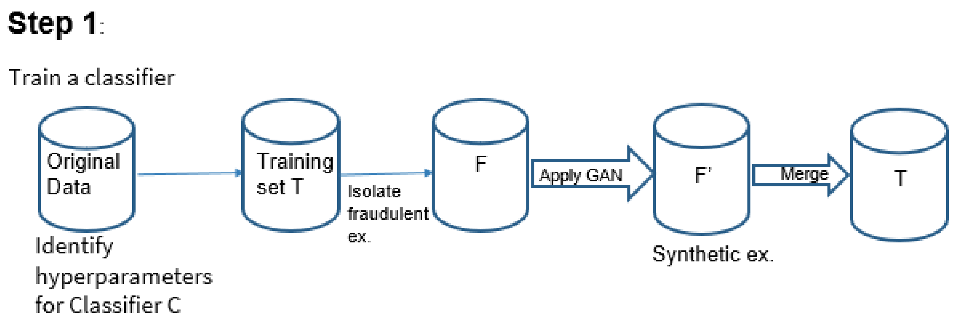
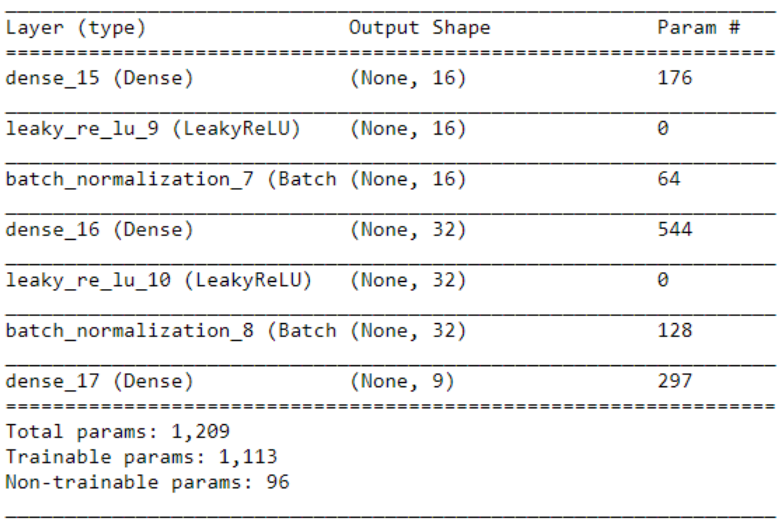


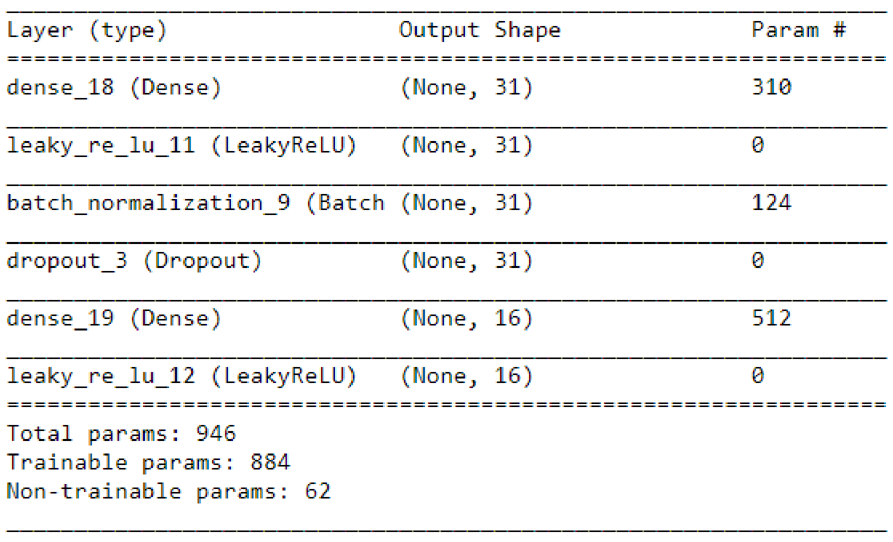
FIGURE 3.29: Step 1 in GAN Framework, adapted from [33]

In Step 2, we compare the performance (using the same testing set) of C trained on the augmented set(C_a) with the performance of the same discriminator trained on the original training set (C_0) [33]. Figure 3.30 and Figure 3.31 Show the Generator G and the Discriminator D for the Synthetic dataset:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_15 (Dense)             (None, 16)                176       
_________________________________________________________________
leaky_re_lu_9 (LeakyReLU)    (None, 16)                0         
_________________________________________________________________
batch_normalization_7 (Batch (None, 16)                64        
_________________________________________________________________
dense_16 (Dense)             (None, 32)                544       
_________________________________________________________________
leaky_re_lu_10 (LeakyReLU)   (None, 32)                0         
_________________________________________________________________
batch_normalization_8 (Batch (None, 32)                128       
_________________________________________________________________
dense_17 (Dense)             (None, 9)                 297       
=================================================================
Total params: 1,209
Trainable params: 1,113
Non-trainable params: 96
_________________________________________________________________
```

FIGURE 3.30: Creating G model, Synthetic Dataset

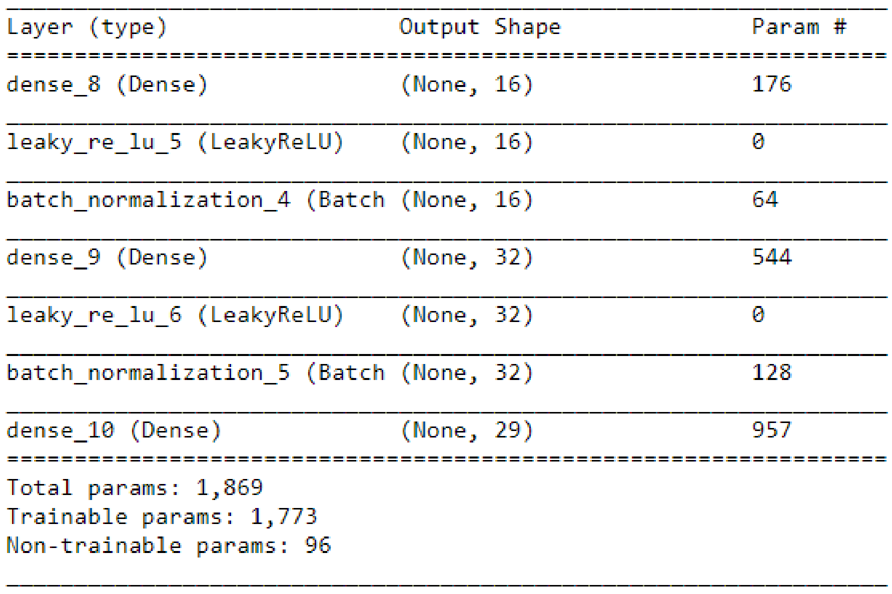


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_18 (Dense)             (None, 31)                310       
_________________________________________________________________
leaky_re_lu_11 (LeakyReLU)   (None, 31)                0         
_________________________________________________________________
batch_normalization_9 (Batch (None, 31)                124       
_________________________________________________________________
dropout_3 (Dropout)          (None, 31)                0         
_________________________________________________________________
dense_19 (Dense)             (None, 16)                512       
_________________________________________________________________
leaky_re_lu_12 (LeakyReLU)   (None, 16)                0         
=================================================================
Total params: 946
Trainable params: 884
Non-trainable params: 62
_________________________________________________________________
```

FIGURE 3.31: Creating D model, Synthetic Dataset

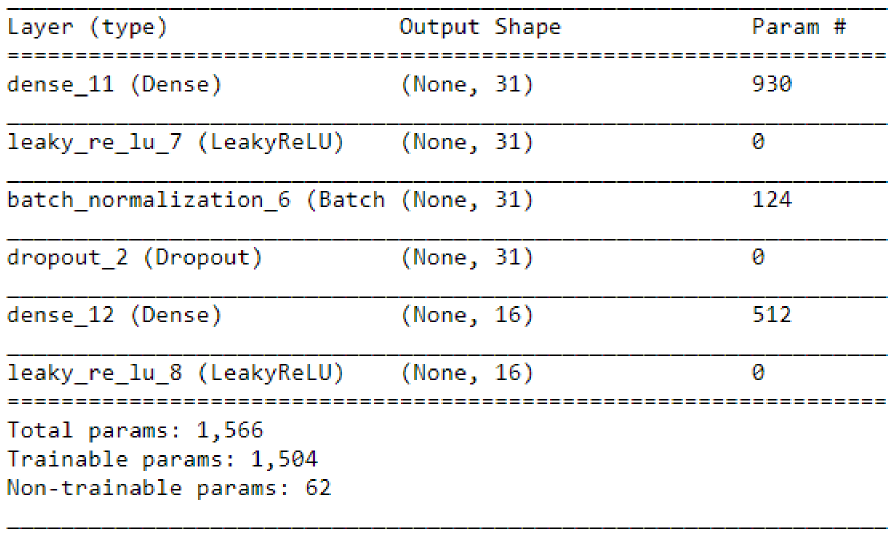
The results of applying the GAN on the Synthetic dataset have been included in Appendix B.

Figure 3.28 and Figure 3.29 show the Generator G and the Discriminator D for the Real-World dataset.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_8 (Dense)              (None, 16)                176
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 16)                0
_________________________________________________________________
batch_normalization_4 (Batch (None, 16)                64
_________________________________________________________________
dense_9 (Dense)              (None, 32)                544
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 32)                0
_________________________________________________________________
batch_normalization_5 (Batch (None, 32)                128
_________________________________________________________________
dense_10 (Dense)             (None, 29)                957
=================================================================
Total params: 1,869
Trainable params: 1,773
Non-trainable params: 96
_________________________________________________________________
```

FIGURE 3.32: Creating G model, Real-World Dataset



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_11 (Dense)             (None, 31)                930
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 31)                0
_________________________________________________________________
batch_normalization_6 (Batch (None, 31)                124
_________________________________________________________________
dropout_2 (Dropout)          (None, 31)                0
_________________________________________________________________
dense_12 (Dense)             (None, 16)                512
_________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 16)                0
=================================================================
Total params: 1,566
Trainable params: 1,504
Non-trainable params: 62
_________________________________________________________________
```

FIGURE 3.33: Creating D model, Real-World Dataset

The results of applying GAN on the Real-World dataset is shown in Appendix C.

3.4 Summary of Chapter 3

In this chapter, we explored the two datasets that we want to work on in details. We explained the different preprocessing techniques that need to be applied before applying the algorithms or the two datasets then used visualization techniques to explain the nature and the relationships of features in each dataset. We explained 4 different models: Logistic Regression, Isolation Forest, Ensemble Method and Generative Adversarial Networks separately and applied each one of them on the two datasets. We show the results in Chapter 4.

Chapter 4

Results

The analysis presented in this chapter is based on the data obtained from running the models as described in the previous chapter. Due to the size of results of the Generative Adversarial Networks, it has been summarized in this Chapter and included in full in Appendix B and C.

4.1 Evaluating the Models

We use the following assessment measures to evaluate the proposed models.

4.1.1 Measurements

We use a *Confusion Matrix* to present our results. The following measures are calculated based on the confusion matrix. *Accuracy*, *Recall*, *Precision* and *F-score*. The

Confusion Matrix shows the performance of the classification algorithm when assigning input data to different classes (see table 4.1) [60].

Confusion Matrix		
	Positive(Genuine)	Negative(Fraud)
Positive(Genuine)	TruePositive(TP)	FalseNegative(FN)
Negative(Fraud)	FalsePositive(FP)	TrueNegative(TN)

Table 4.1: Confusion Matrix

$$Recall = \frac{TP}{FN + TP} \quad (4.1)$$

the *Recall* shows the efficiency of the classifier in detecting the actual fraudulent transactions [60].

$$Precision = \frac{TP}{Tp + FP} \quad (4.2)$$

The *Precision* shows how reliable is the output form [60]

$$F - Measure = \frac{2 * Recall * Precision}{Recall + Precision} \quad (4.3)$$

The *F-measure(F-score)* is the harmonic mean of *Recall* and *Precision* measures [60].

$$Accuracy = \frac{TP + TN}{Tp + FP + TN + FN} \quad (4.4)$$

Accuracy provides the proportion of the total number of correctly identified objects that belong to the class. Although this measurement is generally acceptable, it does not work well if the number of negative cases is too high compared to the positive cases [42].

The *F-measure* is a more reliable measure for evaluating data mining systems with imbalanced classes because it is the harmonic mean of *Recall* and *Precision* measures. Therefore, this score takes both *false positives* and *false negatives* into account. Using the F-measure, both the *TP* and *TN* measures are equally treated when we have an unbalanced dataset.

4.1.2 General Evaluation

Each dataset is divided into training data and testing data. The test size is 0.3, which means that 70% of the data is used as training data while the remaining 30% is used as test data. Then each model is generated on the training data and evaluated on the test data.

4.1.3 Evaluation of the First Model (Logistic Regression)

The result of applying a Logistic Regression model on the Synthetic dataset is shown in Figure 4.1.

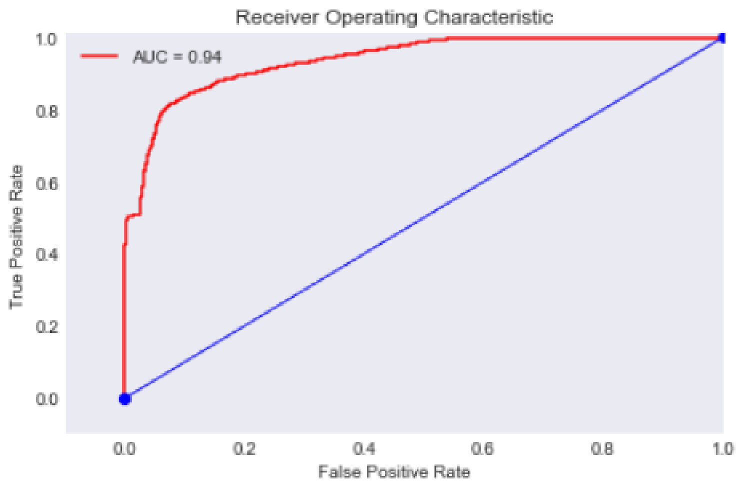
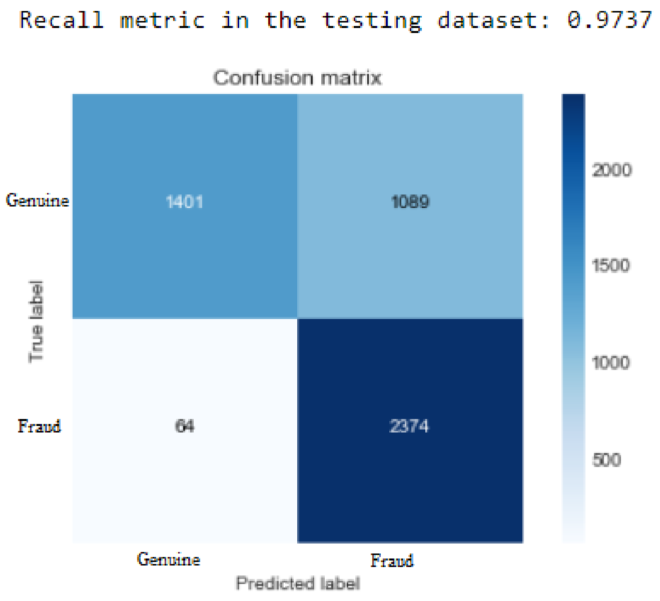


Figure 4.1: Logistic Regression accuracy and ROC curve, Synthetic Dataset, Model 1

For this Model, we used *ROC* Curve to compare the True Positive Rate vs False Positive Rate as well as calculating the *AUC*. The *ROC* is a measure of precision and recall at a particular threshold value whereas AUC is the area under the *ROC* curve. Taking into consideration that the closer the curve follows the left-hand border and then the top border of the *ROC* space, the more accurate the test is. On the contrary, the closer the curve comes to the 45-degree diagonal of the *ROC* space, the less accurate the test. To classify an instance with a regression model, the model first computes the probabilities of either class. If the probability of positive class is higher than a chosen threshold, this instance will be marked positive, and vice versa. Looking at Figure 4.1, the *ROC* curve is high and close to the left-hand border which indicates that the accuracy is high, and the calculation of the *AUC* gives a value of 0.94 which means the accuracy of this model is 94%.

The Confusion Matrix for the Synthetic dataset, shown in Figure 4.2, shows that 1401 transactions have been correctly predicted as genuine, and 64 are incorrectly predicted as genuine transactions. The number of correctly predicted fraudulent transactions is 2374 while the number of incorrectly predicted fraudulent transactions is 1089.



```
Recall metric in the testing dataset: 0.9737
```

Figure 4.2: Confusion Matrix- Synthetic Dataset- Model 1

Figure 4.3 below shows the Confusion Matrix after applying Logistic Regression on the Real-World dataset:

```
Recall metric in the testing dataset:0.8265
```

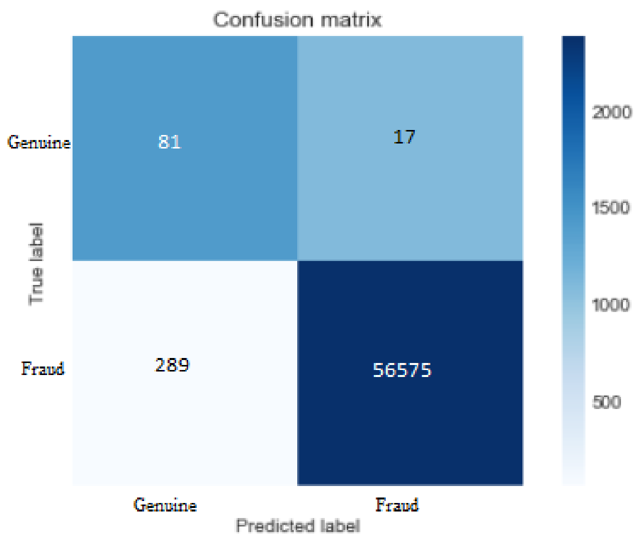
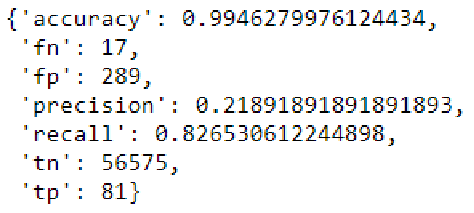


Figure 4.3: Confusion Matrix- Real-World Dataset- Model 1

Figure 4.21 shows the results of *Accuracy*, *Precision*, *Recall*, *TN*, *FN*, *TP*, *FP*. The results were calculated using Python. It shows that Model 1 achieves accuracy of 99.4%



```
{'accuracy': 0.9946279976124434,
 'fn': 17,
 'fp': 289,
 'precision': 0.21891891891891893,
 'recall': 0.826530612244898,
 'tn': 56575,
 'tp': 81}
```

Figure 4.4: Calculations- Real-World Dataset- Model 1

The results of the *ROC* curve for the Real-World dataset is shown in Figure 4.5:

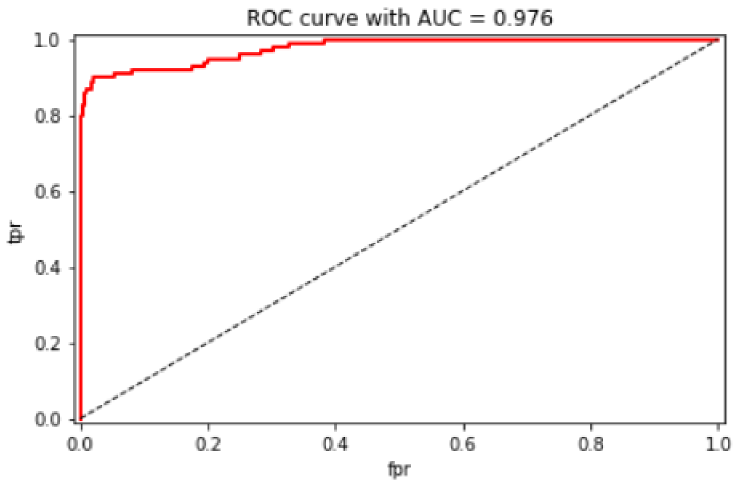


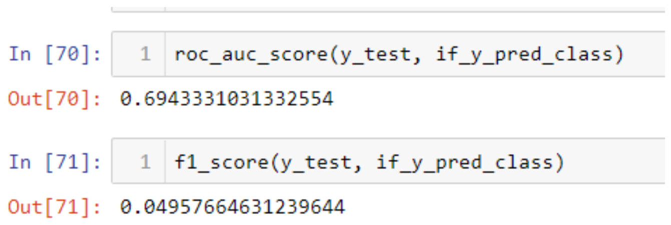
Figure 4.5: Logistic Regression ROC Curve- Real-World dataset- Model 1

The *ROC* curve starts with a big rise from the left-hand side until it reaches 0.9 then it starts to increase slowly until it reaches 0.976 which represents the accuracy of the model.

4.1.4 Evaluation of the Second Model (Isolation Forest)

To evaluate the second model, we will use *AUC*, *f1 score (F1-Measure)* and a *Confusion Matrix.*

The *AUC* or *Area Under Curve* is used in classification analysis in order to determine which of the used models predicts the classes best. An example of its application is *ROC* curves. Here, we use the ROC_ AUC score to check the accuracy of the model.



```
In [70]:  roc_auc_score(y_test, if_y_pred_class)
Out[70]: 0.6943331031332554

In [71]:  f1_score(y_test, if_y_pred_class)
Out[71]: 0.04957664631239644
```

Figure 4.6: Accuracy and f1-score- Synthetic Dataset- Model 2

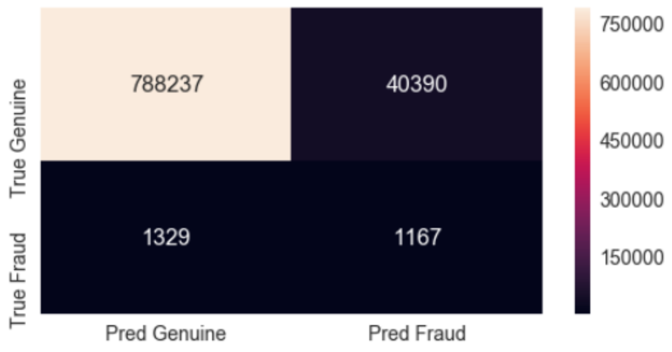
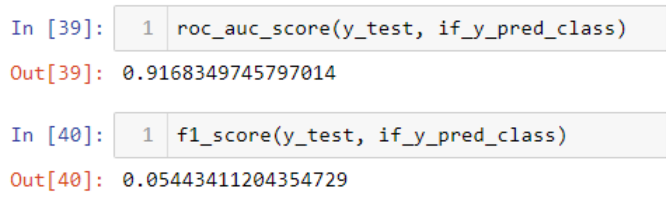


Figure 4.7: Confusion Matrix- Synthetic dataset- Model 2

The AUC score for the Real-World dataset was approximately 69%, with a F1 score around 0.049.

Usually if F1 score is high, both precision and recall of the classifier indicate good results. Here, the F1 score is close to 0 which indicates that the results are not good. We can not depend on F1 score for the last decision so we will use the Confusion Matrix for more details. The Confusion Matrix shows that Model 2 gives the numbers of the correctly predicted Genuine transactions as 788,237, while the number of false Genuine transactions is 1329. Similarly, the number of correctly predicted fraud transactions is 40,390 while the number of incorrectly predicted fraud transactions is 1167. These numbers are considered to be good overall results.

The accuracy achieved in the Real-World dataset is shown in figure 4.8:



```
In [39]:  1  roc_auc_score(y_test, if_y_pred_class)
Out[39]: 0.9168349745797014

In [40]:  1  f1_score(y_test, if_y_pred_class)
Out[40]: 0.05443411204354729
```

FIGURE 4.8: Accuracy and F-score- Real-World Dataset- Model 2

The AUC score for the Real-World dataset was approximately 91% accuracy which is high compared to the accuracy for the Synthetic dataset. The F1 score is roughly 0.05 which also consider to be bad because is close to 0. Figure 4.9 shows the Confusion Matrix for the Real-World dataset. As we can see, the number of correctly predicted Genuine transactions are 81153 while the number of incorrectly predicted Genuine transactions are only 17. Similarly, the number of correctly predicted fraud transactions is 4154 while the number of incorrectly predicted fraud transactions is 119. The Confusion Matrix shows good results which lead us to say that Model 2 gives good predicted results.

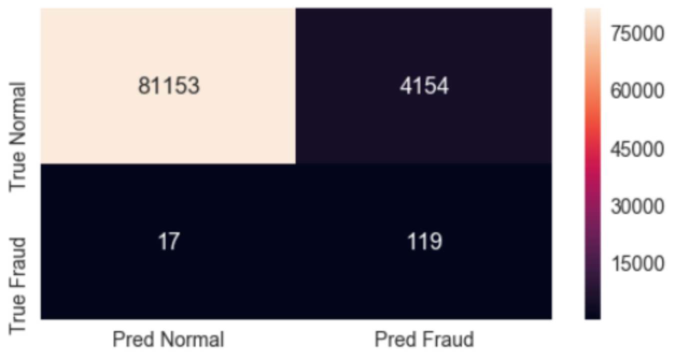


FIGURE 4.9: Confusion Matrix- Real-World Dataset- Model 2

4.1.5 Evaluation of the Third Model (Ensemble Method)

The results of applying the Ensemble Method are shown in Figure 4.10and Figure 4.13 for the two datasets respectively.

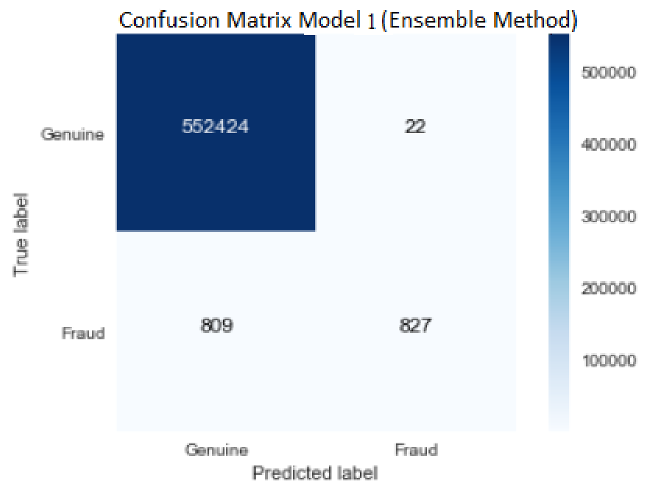
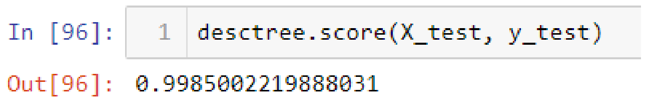


Figure 4.10: Confusion Matrix- Synthetic Dataset- Model 1

The Confusion Matrix shows that the Ensemble Method has resulted in 552424 true positive predictions (i.e. transactions that were genuine and were predicted to be genuine), and 827 true negative predictions, (i.e. transactions that were fraud and were predicted to be fraud fraud). However, the Ensemble method has also predicted 809 transactions as genuine which there were not and 22 transactions as fraudulent which also were actually genuine.

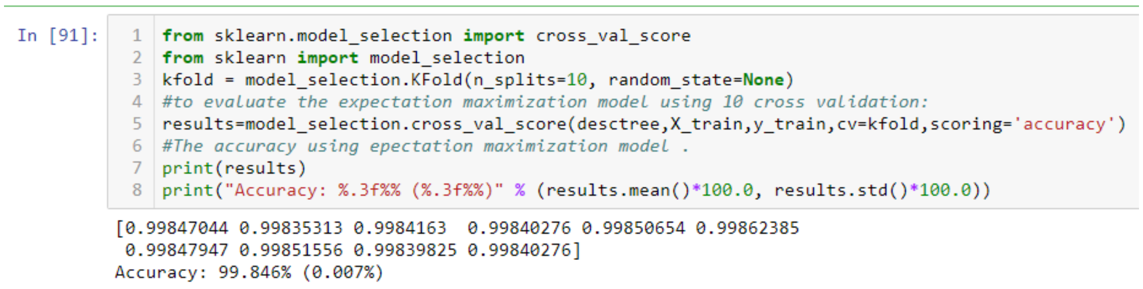
Figure 4.11 shows applying the instruction in Python to extract the accuracy number that the Ensemble method could achieve from the testing set. This number considered to be the best overall score.



```
In [96]:  desctree.score(X_test, y_test)
Out[96]:  0.9985002219888031
```

Figure 4.11: Accuracy- Synthetic dataset- Model 1

The accuracy score for the tree is 99.85% and this is before applying a 10-fold Cross Validation. The results after applying 10-fold Cross Validation are:



```
In [91]:  from sklearn.model_selection import cross_val_score
          from sklearn import model_selection
          kfold = model_selection.KFold(n_splits=10, random_state=None)
          #to evaluate the expectation maximization model using 10 cross validation:
          results=model_selection.cross_val_score(desctree,X_train,y_train,cv=kfold,scoring='accuracy')
          #The accuracy using epectation maximization model .
          print(results)
          print("Accuracy: %.3f%% (%.3f%%)" % (results.mean()*100.0, results.std()*100.0))

[0.99847044 0.99835313 0.9984163  0.99840276 0.99850654 0.99862385
 0.99847947 0.99851556 0.99839825 0.99840276]
Accuracy: 99.846% (0.007%)
```

Figure 4.12: 10 Cross Validation- Synthetic dataset- Model 1

The accuracy of the model is the average of the accuracy of each fold. The accuracy of each fold ranges between 99.840% and 99.850% which indicates that the Ensemble Method achieves a high accuracy of prediction.

As for the World dataset, checking the accuracy is also done by using the Tree Score, Confusion Matrix, and a 10-fold Cross Validation. The results are shown in 4.13:

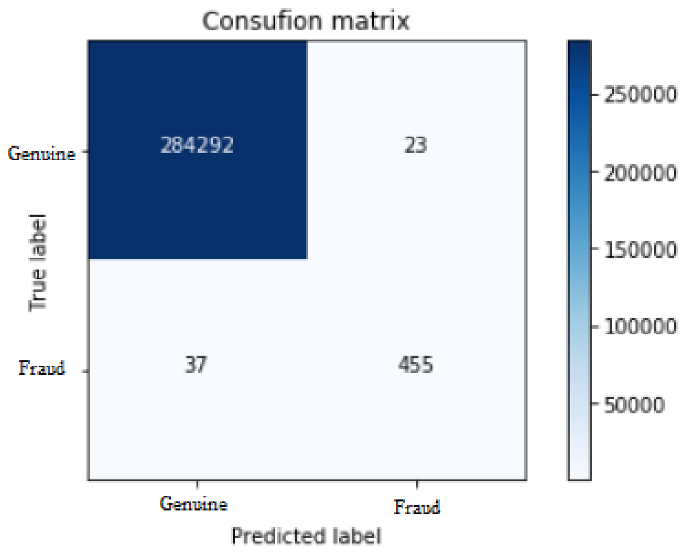
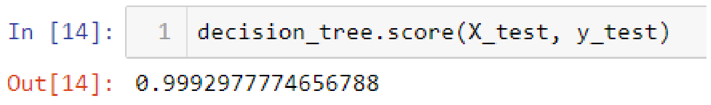


FIGURE 4.13: Confusion Matrix- Real-World Dataset- Model 1

The Confusion Matrix indicates that the number of *TP* are 284,292 which means that the method has predicted these genuine transactions as genuine, and the total number of *TN* fraud are 37. On the other hand, 23 transactions were predicted as genuine while they were actually fraud and 455 were fraud while they were predicted as genuine.

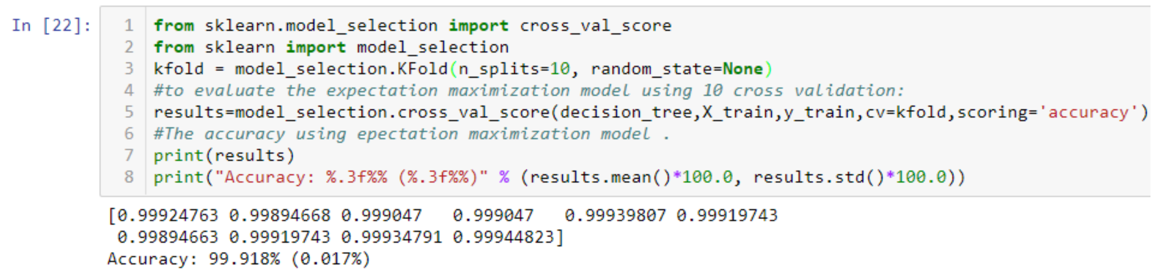
Figure 4.14 shows applying the instruction in Python to extract the accuracy score that the Ensemble method could achieve from the testing set. This number considered to be the best overall score.



```
In [14]:  decision_tree.score(X_test, y_test)
Out[14]:  0.9992977774656788
```

FIGURE 4.14: Accuracy- Real-World Dataset- Model 1

The accuracy scores 99.93%, and when applying 10 fold Cross Validation, the accuracy of each fold ranges between 99.8% and 99.9% which indicates that the Ensemble Method achieves a high accuracy of prediction in this dataset too.



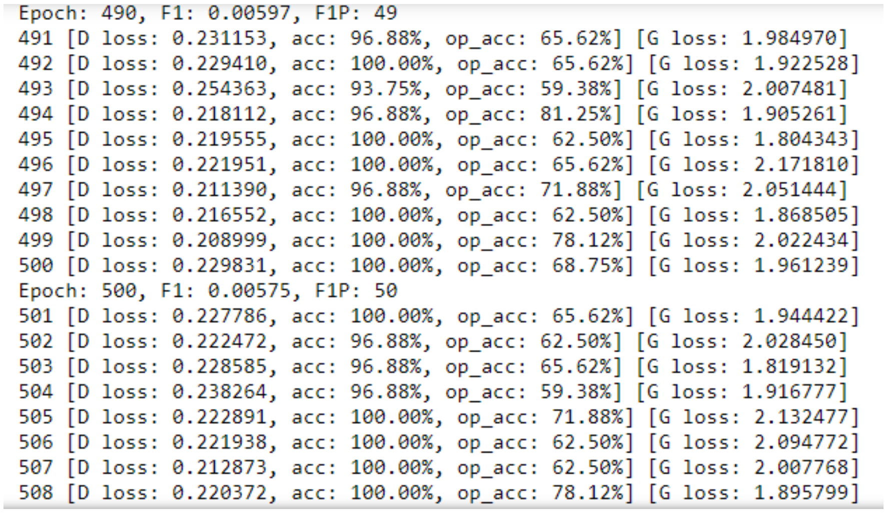
```
In [22]: from sklearn.model_selection import cross_val_score
         from sklearn import model_selection
         kfold = model_selection.KFold(n_splits=10, random_state=None)
         #to evaluate the expectation maximization model using 10 cross validation:
         results=model_selection.cross_val_score(decision_tree,X_train,y_train,cv=kfold,scoring='accuracy')
         #The accuracy using epectation maximization model .
         print(results)
         print("Accuracy: %.3f%% (%.3f%%)" % (results.mean()*100.0, results.std()*100.0))

[0.99924763 0.99894668 0.999047   0.999047   0.99939807 0.99919743
 0.99894663 0.99919743 0.99934791 0.99944823]
Accuracy: 99.918% (0.017%)
```

FIGURE 4.15: 10 Cross Validation- Real-World Dataset- Model 2

4.1.6 Evaluation of the Fourth Model (Generative Adversarial Network)

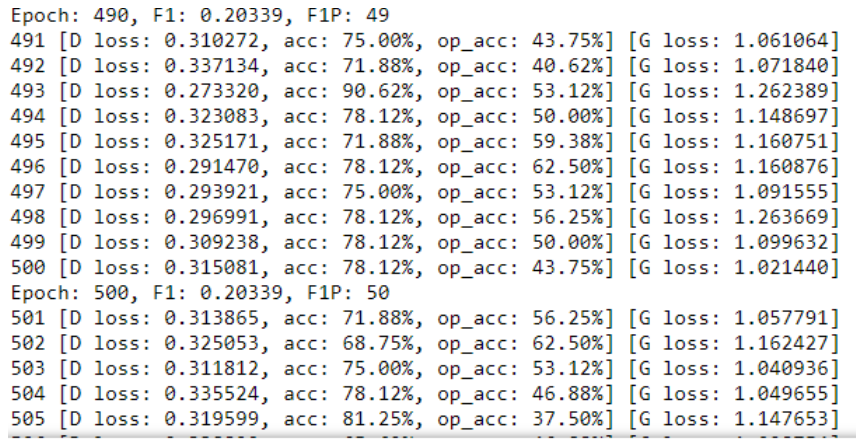
The results of GAN for the Synthetic and Real-World dataset are in Appendices B and C respectively. For each dataset 5000 epochs were selected . Initially, the accuracy is about 43% for the Synthetic dataset and 50% for the second one, after that the accuracy starts to increase with the number of epochs. When the model reaches Epoch 40, the accuracy reaches 75% for the Synthetic dataset and 50% for the second because the model starts to learn how to predict better.

Our results indicate that the higher number of iterations, the better are the results. For example, from Figure 4.16 and 4.17 by the time we reach Epoch 500, the accuracy is between 98% and 100% in the synthetic dataset and between 75% and 81% for the second respectively. The reason behind the improvement is that the model learnt how to predict correctly.



```
Epoch: 490, F1: 0.00597, F1P: 49
491 [D loss: 0.231153, acc: 96.88%, op_acc: 65.62%] [G loss: 1.984970]
492 [D loss: 0.229410, acc: 100.00%, op_acc: 65.62%] [G loss: 1.922528]
493 [D loss: 0.254363, acc: 93.75%, op_acc: 59.38%] [G loss: 2.007481]
494 [D loss: 0.218112, acc: 96.88%, op_acc: 81.25%] [G loss: 1.905261]
495 [D loss: 0.219555, acc: 100.00%, op_acc: 62.50%] [G loss: 1.804343]
496 [D loss: 0.221951, acc: 100.00%, op_acc: 65.62%] [G loss: 2.171810]
497 [D loss: 0.211390, acc: 96.88%, op_acc: 71.88%] [G loss: 2.051444]
498 [D loss: 0.216552, acc: 100.00%, op_acc: 62.50%] [G loss: 1.868505]
499 [D loss: 0.208999, acc: 100.00%, op_acc: 78.12%] [G loss: 2.022434]
500 [D loss: 0.229831, acc: 100.00%, op_acc: 68.75%] [G loss: 1.961239]
Epoch: 500, F1: 0.00575, F1P: 50
501 [D loss: 0.227786, acc: 100.00%, op_acc: 65.62%] [G loss: 1.944422]
502 [D loss: 0.222472, acc: 96.88%, op_acc: 62.50%] [G loss: 2.028450]
503 [D loss: 0.228585, acc: 96.88%, op_acc: 65.62%] [G loss: 1.819132]
504 [D loss: 0.238264, acc: 96.88%, op_acc: 59.38%] [G loss: 1.916777]
505 [D loss: 0.222891, acc: 100.00%, op_acc: 71.88%] [G loss: 2.132477]
506 [D loss: 0.221938, acc: 100.00%, op_acc: 62.50%] [G loss: 2.094772]
507 [D loss: 0.212873, acc: 100.00%, op_acc: 62.50%] [G loss: 2.007768]
508 [D loss: 0.220372, acc: 100.00%, op_acc: 78.12%] [G loss: 1.895799]
```

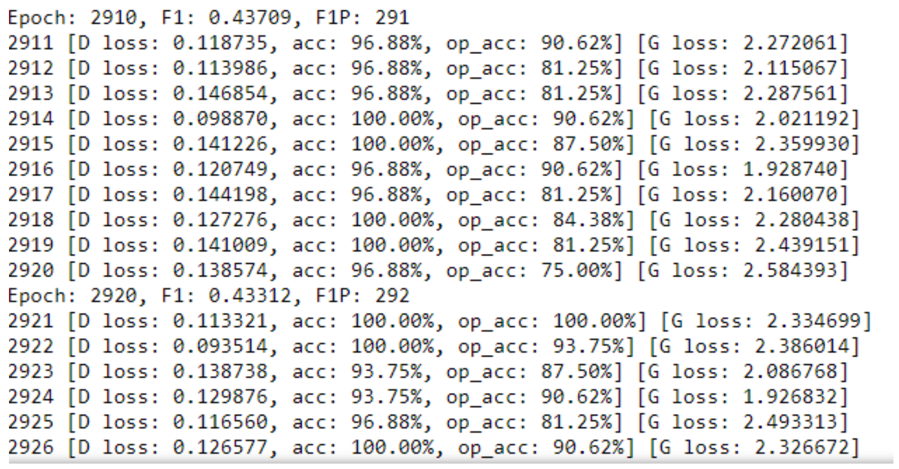
FIGURE 4.16: Plot the prediction accuracy across epochs- Synthetic Dataset, Model 4



```
Epoch: 490, F1: 0.20339, F1P: 49
491 [D loss: 0.310272, acc: 75.00%, op_acc: 43.75%] [G loss: 1.061064]
492 [D loss: 0.337134, acc: 71.88%, op_acc: 40.62%] [G loss: 1.071840]
493 [D loss: 0.273320, acc: 90.62%, op_acc: 53.12%] [G loss: 1.262389]
494 [D loss: 0.323083, acc: 78.12%, op_acc: 50.00%] [G loss: 1.148697]
495 [D loss: 0.325171, acc: 71.88%, op_acc: 59.38%] [G loss: 1.160751]
496 [D loss: 0.291470, acc: 78.12%, op_acc: 62.50%] [G loss: 1.160876]
497 [D loss: 0.293921, acc: 75.00%, op_acc: 53.12%] [G loss: 1.091555]
498 [D loss: 0.296991, acc: 78.12%, op_acc: 56.25%] [G loss: 1.263669]
499 [D loss: 0.309238, acc: 78.12%, op_acc: 50.00%] [G loss: 1.099632]
500 [D loss: 0.315081, acc: 78.12%, op_acc: 43.75%] [G loss: 1.021440]
Epoch: 500, F1: 0.20339, F1P: 50
501 [D loss: 0.313865, acc: 71.88%, op_acc: 56.25%] [G loss: 1.057791]
502 [D loss: 0.325053, acc: 68.75%, op_acc: 62.50%] [G loss: 1.162427]
503 [D loss: 0.311812, acc: 75.00%, op_acc: 53.12%] [G loss: 1.040936]
504 [D loss: 0.335524, acc: 78.12%, op_acc: 46.88%] [G loss: 1.049655]
505 [D loss: 0.319599, acc: 81.25%, op_acc: 37.50%] [G loss: 1.147653]
```

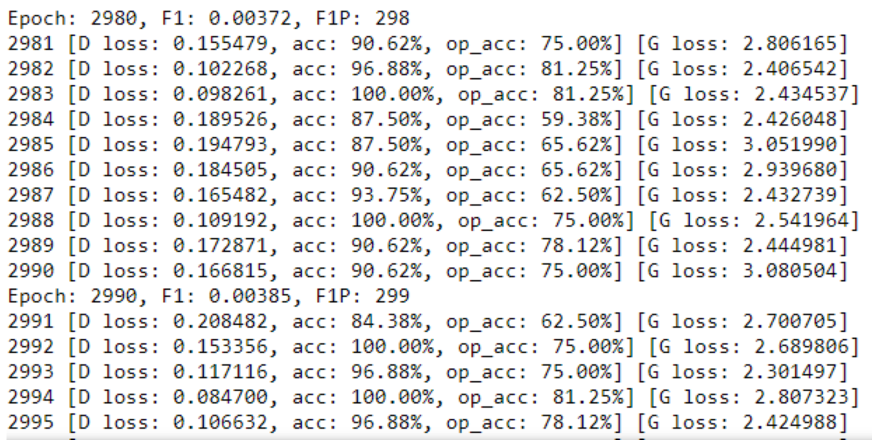
FIGURE 4.17: Plot the prediction accuracy across epochs- Real-World Dataset, Model 4

The accuracy continues to increase until it reaches between 98% and 100% by Epochs 1500-3000 for both datasets. Figures 4.18 and Figure 4.19 show the highest accuracy achieved for both datasets.



```
Epoch: 2910, F1: 0.43709, F1P: 291
2911 [D loss: 0.118735, acc: 96.88%, op_acc: 90.62%] [G loss: 2.272061]
2912 [D loss: 0.113986, acc: 96.88%, op_acc: 81.25%] [G loss: 2.115067]
2913 [D loss: 0.146854, acc: 96.88%, op_acc: 81.25%] [G loss: 2.287561]
2914 [D loss: 0.098870, acc: 100.00%, op_acc: 90.62%] [G loss: 2.021192]
2915 [D loss: 0.141226, acc: 100.00%, op_acc: 87.50%] [G loss: 2.359930]
2916 [D loss: 0.120749, acc: 96.88%, op_acc: 90.62%] [G loss: 1.928740]
2917 [D loss: 0.144198, acc: 96.88%, op_acc: 81.25%] [G loss: 2.160070]
2918 [D loss: 0.127276, acc: 100.00%, op_acc: 84.38%] [G loss: 2.280438]
2919 [D loss: 0.141009, acc: 100.00%, op_acc: 81.25%] [G loss: 2.439151]
2920 [D loss: 0.138574, acc: 96.88%, op_acc: 75.00%] [G loss: 2.584393]
Epoch: 2920, F1: 0.43312, F1P: 292
2921 [D loss: 0.113321, acc: 100.00%, op_acc: 100.00%] [G loss: 2.334699]
2922 [D loss: 0.093514, acc: 100.00%, op_acc: 93.75%] [G loss: 2.386014]
2923 [D loss: 0.138738, acc: 93.75%, op_acc: 87.50%] [G loss: 2.086768]
2924 [D loss: 0.129876, acc: 93.75%, op_acc: 90.62%] [G loss: 1.926832]
2925 [D loss: 0.116560, acc: 96.88%, op_acc: 81.25%] [G loss: 2.493313]
2926 [D loss: 0.126577, acc: 100.00%, op_acc: 90.62%] [G loss: 2.326672]
```

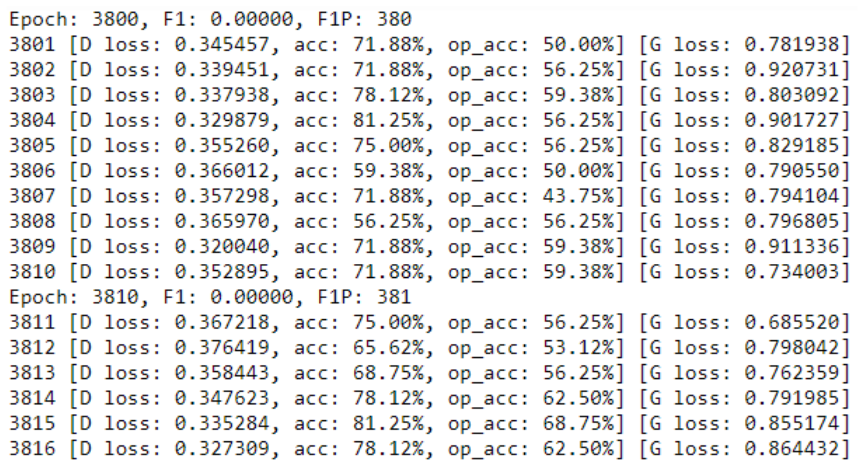
FIGURE 4.18: Plot the prediction accuracy across epochs- Synthetic Dataset, Model 4



```
Epoch: 2980, F1: 0.00372, F1P: 298
2981 [D loss: 0.155479, acc: 90.62%, op_acc: 75.00%] [G loss: 2.806165]
2982 [D loss: 0.102268, acc: 96.88%, op_acc: 81.25%] [G loss: 2.406542]
2983 [D loss: 0.098261, acc: 100.00%, op_acc: 81.25%] [G loss: 2.434537]
2984 [D loss: 0.189526, acc: 87.50%, op_acc: 59.38%] [G loss: 2.426048]
2985 [D loss: 0.194793, acc: 87.50%, op_acc: 65.62%] [G loss: 3.051990]
2986 [D loss: 0.184505, acc: 90.62%, op_acc: 65.62%] [G loss: 2.939680]
2987 [D loss: 0.165482, acc: 93.75%, op_acc: 62.50%] [G loss: 2.432739]
2988 [D loss: 0.109192, acc: 100.00%, op_acc: 75.00%] [G loss: 2.541964]
2989 [D loss: 0.172871, acc: 90.62%, op_acc: 78.12%] [G loss: 2.444981]
2990 [D loss: 0.166815, acc: 90.62%, op_acc: 75.00%] [G loss: 3.080504]
Epoch: 2990, F1: 0.00385, F1P: 299
2991 [D loss: 0.208482, acc: 84.38%, op_acc: 62.50%] [G loss: 2.700705]
2992 [D loss: 0.153356, acc: 100.00%, op_acc: 75.00%] [G loss: 2.689806]
2993 [D loss: 0.117116, acc: 96.88%, op_acc: 75.00%] [G loss: 2.301497]
2994 [D loss: 0.084700, acc: 100.00%, op_acc: 81.25%] [G loss: 2.807323]
2995 [D loss: 0.106632, acc: 96.88%, op_acc: 78.12%] [G loss: 2.424988]
```

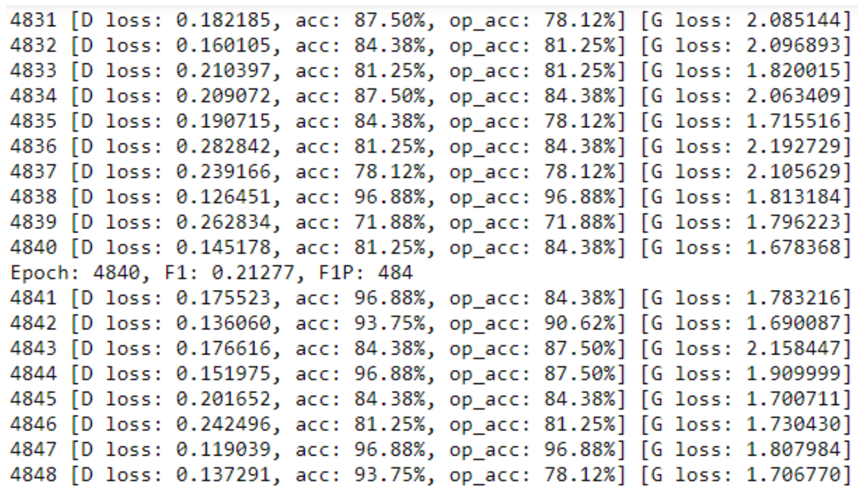
FIGURE 4.19: Plot the prediction accuracy across epochs- Real-World Dataset, Model 4

Then The accuracy starts to decrease by the time we reach Epoch 3800 in the Synthetic dataset and Epoch 4800 in the second dataset. The reason for these decrements in accuracy might be due to the fact that a large number of Epochs may result in overfitting. This can be seen in Figures 4.20 and 4.21:



```
Epoch: 3800, F1: 0.00000, F1P: 380
3801 [D loss: 0.345457, acc: 71.88%, op_acc: 50.00%] [G loss: 0.781938]
3802 [D loss: 0.339451, acc: 71.88%, op_acc: 56.25%] [G loss: 0.920731]
3803 [D loss: 0.337938, acc: 78.12%, op_acc: 59.38%] [G loss: 0.803092]
3804 [D loss: 0.329879, acc: 81.25%, op_acc: 56.25%] [G loss: 0.901727]
3805 [D loss: 0.355260, acc: 75.00%, op_acc: 56.25%] [G loss: 0.829185]
3806 [D loss: 0.366012, acc: 59.38%, op_acc: 50.00%] [G loss: 0.790550]
3807 [D loss: 0.357298, acc: 71.88%, op_acc: 43.75%] [G loss: 0.794104]
3808 [D loss: 0.365970, acc: 56.25%, op_acc: 56.25%] [G loss: 0.796805]
3809 [D loss: 0.320040, acc: 71.88%, op_acc: 59.38%] [G loss: 0.911336]
3810 [D loss: 0.352895, acc: 71.88%, op_acc: 59.38%] [G loss: 0.734003]
Epoch: 3810, F1: 0.00000, F1P: 381
3811 [D loss: 0.367218, acc: 75.00%, op_acc: 56.25%] [G loss: 0.685520]
3812 [D loss: 0.376419, acc: 65.62%, op_acc: 53.12%] [G loss: 0.798042]
3813 [D loss: 0.358443, acc: 68.75%, op_acc: 56.25%] [G loss: 0.762359]
3814 [D loss: 0.347623, acc: 78.12%, op_acc: 62.50%] [G loss: 0.791985]
3815 [D loss: 0.335284, acc: 81.25%, op_acc: 68.75%] [G loss: 0.855174]
3816 [D loss: 0.327309, acc: 78.12%, op_acc: 62.50%] [G loss: 0.864432]
```

FIGURE 4.20: Plot the prediction accuracy across epochs- Synthetic Dataset- Model 4



```
4831 [D loss: 0.182185, acc: 87.50%, op_acc: 78.12%] [G loss: 2.085144]
4832 [D loss: 0.160105, acc: 84.38%, op_acc: 81.25%] [G loss: 2.096893]
4833 [D loss: 0.210397, acc: 81.25%, op_acc: 81.25%] [G loss: 1.820015]
4834 [D loss: 0.209072, acc: 87.50%, op_acc: 84.38%] [G loss: 2.063409]
4835 [D loss: 0.190715, acc: 84.38%, op_acc: 78.12%] [G loss: 1.715516]
4836 [D loss: 0.282842, acc: 81.25%, op_acc: 84.38%] [G loss: 2.192729]
4837 [D loss: 0.239166, acc: 78.12%, op_acc: 78.12%] [G loss: 2.105629]
4838 [D loss: 0.126451, acc: 96.88%, op_acc: 96.88%] [G loss: 1.813184]
4839 [D loss: 0.262834, acc: 71.88%, op_acc: 71.88%] [G loss: 1.796223]
4840 [D loss: 0.145178, acc: 81.25%, op_acc: 84.38%] [G loss: 1.678368]
Epoch: 4840, F1: 0.21277, F1P: 484
4841 [D loss: 0.175523, acc: 96.88%, op_acc: 84.38%] [G loss: 1.783216]
4842 [D loss: 0.136060, acc: 93.75%, op_acc: 90.62%] [G loss: 1.690087]
4843 [D loss: 0.176616, acc: 84.38%, op_acc: 87.50%] [G loss: 2.158447]
4844 [D loss: 0.151975, acc: 96.88%, op_acc: 87.50%] [G loss: 1.909999]
4845 [D loss: 0.201652, acc: 84.38%, op_acc: 84.38%] [G loss: 1.700711]
4846 [D loss: 0.242496, acc: 81.25%, op_acc: 81.25%] [G loss: 1.730430]
4847 [D loss: 0.119039, acc: 96.88%, op_acc: 96.88%] [G loss: 1.807984]
4848 [D loss: 0.137291, acc: 93.75%, op_acc: 78.12%] [G loss: 1.706770]
```

FIGURE 4.21: Plot the prediction accuracy across epochs- Real-World Dataset, Model 4

4.2 Summary of Chapter 4

The four models have shown good overall results. The first model generates excellent results for the two datasets. in this model, I used *ROC* curve as an extra measure to guarantee that *ROC* is giving good same results as *Accuracy*, *Precision* and *Confusion Matrix*. In addition, the computation was not long in this model. The second model gave average result on the Synthetic dataset as the accuracy reached 69% and good result on the Real-World dataset as the accuracy reached 91%. this might be due to the nature of the dataset, specifically the size of the data. Also, computation on both datasets was long.

The third model gave good results in the beginning then it stayed stable for some epochs but finally we notice remarkable decrements as the accuracy started to decrease after a number of epochs. The reason for these decrements in accuracy might be due to the fact that a large number of Epochs may result in overfitting.

Model 4 gave excellent results for both datasets. The accuracy for Model 4 on both datasets was around 99.8%. Also, the 10 fold Cross Validation achieved good results that ranged around 99.8% in each split.

Table 4.2 and Table 4.3 show numeric descriptions of each model for both datasets. The tables include the accuracy, recall, positive predicted values and negative predicted values achievements for each model.

Synthetic Dataset				
Models	**Accuracy**	**Recall**	**PPV**	**NPV**
Logistic Regression	94%	0.9737	2374	1401
Isolation Forest	69%	0.951	788237	1167
Ensemble Method	99.85%	0.999	552424	827
GAN	98%	NA	NA	NA

Table 4.2: Numeric Description of the results for Real-World dataset

Real-World dataset				
Models	**Accuracy**	**Recall**	**PPV**	**NPV**
Logistic Regression	97%	0.8265	56575	81
Isolation Forest	91%	0.998	81153	119
Ensemble Method	99.93%	0.998	284292	455
GAN	96%	NA	NA	NA

Table 4.3: Numeric Description of the results for Real-World dataset

Chapter 5

Conclusions and Future Work

In this work, we applied four different models to two datasets to determine their effectiveness in detecting fraud. The first dataset is a synthetic dataset that contains around 23 million records and 11 columns. The data provided has the financial transaction data as well as the target variable *'isFraud'*, which is the actual fraud status of the transaction and *'isFlaggedFraud'*, which is the indicator the simulation uses to flag the transaction using some threshold. The second dataset contains data from European credit-card transactions. It contains 284,808 records that have been collected over two days in 2013.

For preparation of the datasets, they first required preprocessing, after which several attributes in each dataset had to be ignored as those are not carrying useful information for the analysis. We also found that the data in datasets were highly imbalanced. SMOTE was used to handle the imbalanced problem of Synthetic dataset and random undersampling was applied to the second one.

The first model obtained an accuracy of 94% on the Synthetic dataset and 97% on the Real-World.
The second model, Isolation Forest, obtained an accuracy of 69% on the Synthetic dataset and 91% for on the Real-World one. The reason that lies behind the difference in the result might be related to the different sizes of the datasets.
The third model utilized was the Extended Features Ensemble Method. This model achieved an accuracy of 99% for each dataset. In addition, 10-fold Cross Validation showed an accuracy of 99.8% and 99.9% respectively for each dataset.
Model four , Generative Adversarial Networks, obtained an overall accuracy of 43% on the Synthetic dataset and 50% on the Real-World with a small number of epochs. However, the model was able to obtain 99% for each dataset during the operation when the number of epochs increased to about 2900. This could be related to overfitting. The solution then is to decrease the number of epochs when rerunning the model.

In comparing the four models, we conclude the following:

- All four models seem to work properly in detecting fraud.
- Logistic Regression seems to be a good approach as it gives good results with less computational effort.
- Isolation Forest works well when the dataset is preprocessed appropriately.
- Ensemble Methods is the model that achieved the highest accuracy.

- GAN has great accuracy but requires a high number of epochs However, using many epochs might lead to the problem of overfitting.

5.1 Future work

We applied four different models to detect fraud, However, there are many other Machine Learning techniques that could still be applied for fraud detection.

An alternative is to use Extended Isolation Forest as an alternative to the regular Isolation Forest. Thus, instead of selecting a random feature and then a random value within the range of data it selects a random slope for the branch cut. The Extended Isolation Forest relies on the use of hyperplanes with random slopes (non-axis-parallel) for splitting the data in creating the binary search trees.
One extension to the Feature Ensemble Method would be to use other methods for selecting features, such as such as majority voting as well as applying other cost sensitive learning approaches such as Meta Learning. An example would be Empirical Thresholding.

An improvement of GAN might be achieved by reducing the epochs by using an optimiser such as Adam (0.0002, 0.5). Using Wasserstein GAN (WGAN) could also be an alternative to traditional GAN training. WGAN can improve the stability of learning, remove problems such as mode collapse, and provide meaningful learning curves that are useful for debugging and hyperparameter searches.

It may also be interesting for future work to apply our four models on other datasets which could open the door for more conclusions that predict better solutions.

Bibliography

[1] Aggarwal, Charu C. “Outlier analysis.” In *Data mining*. Springer, 2015, 237–263.

[2] Archambeault, Deborah S, Sarah Webber, and Janet Greenlee. “Fraud and corruption in US nonprofit entities: A summary of press reports 2008-2011.” *Nonprofit and Voluntary Sector Quarterly* 44, 6: (2015) 1194–1224.

[3] Barga, Roger, Valentine Fontama, Wee Hyong Tok, and Luis Cabrera-Cordon. *Predictive analytics with Microsoft Azure machine learning*. Springer, 2015.

[4] Batista, Gustavo EAPA, Ronaldo C Prati, and Maria Carolina Monard. “A study of the behavior of several methods for balancing machine learning training data.” *ACM SIGKDD explorations newsletter* 6, 1: (2004) 20–29.

[5] Bauer, Michael A. “High performance computing: the software challenges.” In *Proceedings of the 2007 international workshop on Parallel symbolic computation*. ACM, 2007, 11–12.

[6] Bedre-Defolie, Özlem, and Emilio Calvano. “Pricing payment cards.” *American Economic Journal: Microeconomics* 206–231.

[7] Ben-Gal, Irad. “Outlier detection.” In *Data mining and knowledge discovery handbook*, Springer, 2005, 131–146.

[8] Berry, Michael JA, and Gordon S Linoff. *Data mining techniques: for marketing, sales, and customer relationship management.* John Wiley & Sons, 2004.

[9] Bhatla, Tej Paul, Vikram Prabhu, and Amit Dua. “Understanding credit card frauds.” *Cards business review* 1, 6.

[10] Bontempi, Gianluca. “Structural feature selection for wrapper methods.” In *ESANN*. 2005, 405–410.

[11] Bouaguel, Waad, Ghazi Bel Mufti, and Mohamed Limam. “A fusion approach based on wrapper and filter feature selection methods using majority vote and feature weighting.” In *2013 International Conference on Computer Applications Technology (ICCAT)*. IEEE, 2013, 1–6.

[12] Button, Mark, Les Johnston, Kwabena Frimpong, and Geoff Smith. “New directions in policing fraud: The emergence of the counter fraud specialist in the United Kingdom.” *International Journal of the Sociology of Law* 35, 4: (2007) 192–208.

[13] Chandrashekar, Girish, and Ferat Sahin. “A survey on feature selection methods.” *Computers & Electrical Engineering* 40, 1: (2014) 16–28.

[14] Chang, Wen-Hsi, and Jau-Shien Chang. “An effective early fraud detection method for online auctions.” *Electronic Commerce Research and Applications* 11, 4: (2012) 346–360.

[15] Chau, S-C, and Ada Wai-Chee Fu. "Load balancing between computing clusters." In *Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies*. IEEE, 2003, 548–551.

[16] Chawla, Nitesh V. "Data mining for imbalanced datasets: An overview." In *Data mining and knowledge discovery handbook*, Springer, 2009, 875–886.

[17] Chawla, Nitesh V, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. "SMOTE: synthetic minority over-sampling technique." *Journal of artificial intelligence research* 16: (2002) 321–357.

[18] Cord, Matthieu, and Pádraig Cunningham. *Machine learning techniques for multimedia: case studies on organization and retrieval.* Springer Science & Business Media, 2008.

[19] Dal Pozzolo, Andrea, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, and Gianluca Bontempi. "Credit card fraud detection and concept-drift adaptation with delayed supervised information." In *2015 international joint conference on Neural networks (IJCNN)*. IEEE, 2015, 1–8.

[20] Dal Pozzolo, Andrea, Olivier Caelen, and Gianluca Bontempi. "When is undersampling effective in unbalanced classification tasks?" In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases.* Springer, 2015, 200–215.

[21] Dal Pozzolo, Andrea, Olivier Caelen, Yann-Ael Le Borgne, Serge Waterschoot, and Gianluca Bontempi. "Learned lessons in credit card fraud detection from

a practitioner perspective." *Expert systems with applications* 41, 10: (2014) 4915–4928.

[22] Dash, Manoranjan, and Huan Liu. "Feature selection for classification." *Intelligent data analysis* 1, 1-4: (1997) 131–156.

[23] Dash, Manoranjan, Huan Liu, and Hiroshi Motoda. "Consistency based feature selection." In *Pacific-Asia conference on knowledge discovery and data mining.* Springer, 2000, 98–109.

[24] DeCastro-García, Noemí, Ángel Luis Muñoz Castañeda, David Escudero García, and Miguel V Carriegos. "Effect of the Sampling of a Dataset in the Hyperparameter Optimization Phase over the Efficiency of a Machine Learning Algorithm." *Complexity* 2019.

[25] Del Río, Sara, Victoria López, José Manuel Benítez, and Francisco Herrera. "On the use of MapReduce for imbalanced big data using Random Forest." *Information Sciences* 285: (2014) 112–137.

[26] Dietterich, Thomas G. "Ensemble methods in machine learning." In *International workshop on multiple classifier systems.* Springer, 2000, 1–15.

[27] Ding, Zhiguo, and Minrui Fei. "An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window." *IFAC Proceedings Volumes* 46, 20: (2013) 12–17.

[28] Dumitrescu, George Cornel. "Bitcoin–A Brief Analysis of the Advantages and Disadvantages." *Global Economic Observer* 5, 2: (2017) 63–71.

[29] Elghazel, Haytham, Alex Aussem, and Florence Perraud. "Trading-off diversity and accuracy for optimal ensemble tree selection in random forests." In *Ensembles in Machine Learning Applications*, Springer, 2011, 169–179.

[30] Fadaei Noghani, F, and M Moattar. "Ensemble classification and extended feature selection for credit card fraud detection." *Journal of AI and Data Mining* 5, 2: (2017) 235–243.

[31] Feng, Dingcheng, Feng Chen, and Wenli Xu. "Supervised feature subset selection with ordinal optimization." *Knowledge-Based Systems* 56: (2014) 123–140.

[32] Finlay, Steven. "Multiple classifier architectures and their application to credit risk assessment." *European Journal of Operational Research* 210, 2: (2011) 368–378.

[33] Fiore, Ugo, Alfredo De Santis, Francesca Perla, Paolo Zanetti, and Francesco Palmieri. "Using generative adversarial networks for improving classification effectiveness in credit card fraud detection." *Information Sciences* .

[34] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[35] Grimm, Laurence G, and Paul R Yarnold. *Reading and understanding multivariate statistics.* American Psychological Association, 1995.

[36] Hellevik, Ottar. "Linear versus logistic regression when the dependent variable is a dichotomy." *Quality & Quantity* 43, 1: (2009) 59–74.

[37] Jordan, Michael I, and Robert A Jacobs. "Hierarchical mixtures of experts and the EM algorithm." *Neural computation* 6, 2: (1994) 181–214.

[38] Kleinbaum, David G, K Dietz, M Gail, Mitchel Klein, and Mitchell Klein. *Logistic regression*. Springer, 2002.

[39] Kotsiantis, Sotiris, Dimitris Kanellopoulos, Panayiotis Pintelas, et al. "Handling imbalanced datasets: A review." *GESTS International Transactions on Computer Science and Engineering* 30, 1: (2006) 25–36.

[40] Law, John. "Principal component analysis." *Journal of the Royal Statistical Society: Series D (The Statistician)* 36, 4: (1987) 432–432.

[41] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *nature* 521, 7553: (2015) 436.

[42] Lee, Jay. "Measurement of machine performance degradation using a neural network model." *Computers in Industry* 30, 3: (1996) 193–209.

[43] Lessmann, Stefan, Bart Baesens, Hsin-Vonn Seow, and Lyn C Thomas. "Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research." *European Journal of Operational Research* 247, 1: (2015) 124–136.

[44] Ling, Charles X, and Chenghui Li. "Data mining for direct marketing: Problems and solutions." In *Kdd*. 1998, volume 98, 73–79.

[45] Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation forest." In *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, 413–422.

[46] Louzada, Francisco, and Anderson Ara. "Bagging k-dependence probabilistic networks: An alternative powerful fraud detection tool." *Expert Systems with Applications* 39, 14: (2012) 11,583–11,592.

[47] Maes, Sam, Karl Tuyls, Bram Vanschoenwinkel, and Bernard Manderick. "Credit card fraud detection using Bayesian and neural networks." In *Proceedings of the 1st international naiso congress on neuro fuzzy technologies*. 2002, 261–270.

[48] Maldonado, Sebastián, and Richard Weber. "A wrapper method for feature selection using support vector machines." *Information Sciences* 179, 13: (2009) 2208–2217.

[49] Mani, Inderjeet, and I Zhang. "kNN approach to unbalanced data distributions: a case study involving information extraction." In *Proceedings of workshop on learning from imbalanced datasets*. 2003, volume 126.

[50] Marimont, RB, and MB Shapiro. "Nearest neighbour searches and the curse of dimensionality." *IMA Journal of Applied Mathematics* 24, 1: (1979) 59–70.

[51] Mehta, Swapneel, Prasanth Kothuri, and Daniel Lanza Garcia. "Anomaly Detection for Network Connection Logs." *arXiv preprint arXiv:1812.01941* .

[52] Mishra, Ankit, and Chaitanya Ghorpade. "Credit Card Fraud Detection on the Skewed Data Using Various Classification and Ensemble Techniques." In *2018 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*. IEEE, 2018, 1–5.

[53] Murphy, Kevin P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[54] Nguyen, Giang H, Abdesselam Bouzerdoum, and Son L Phung. "A supervised learning approach for imbalanced data sets." In *2008 19th International Conference on Pattern Recognition*. IEEE, 2008, 1–4.

[55] Opitz, David, and Richard Maclin. "Popular ensemble methods: An empirical study." *Journal of artificial intelligence research* 11: (1999) 169–198.

[56] O'Rourke, Norm, and Larry Hatcher. *A step-by-step approach to using SAS for factor analysis and structural equation modeling*. Sas Institute, 2013.

[57] Pudil, Pavel, Jana Novovičová, and Josef Kittler. "Floating search methods in feature selection." *Pattern recognition letters* 15, 11: (1994) 1119–1125.

[58] Quah, Jon TS, and M Sriganesh. "Real-time credit card fraud detection using computational intelligence." *Expert systems with applications* 35, 4: (2008) 1721–1732.

[59] Robnik-Šikonja, Marko, and Igor Kononenko. "Theoretical and empirical analysis of ReliefF and RReliefF." *Machine learning* 53, 1-2: (2003) 23–69.

[60] Sahin, Yusuf, Serol Bulkan, and Ekrem Duman. "A cost-sensitive decision tree approach for fraud detection." *Expert Systems with Applications* 40, 15: (2013) 5916–5923.

[61] Saitta, Sandro. "Standardization vs. Normalization. 2007." *URL: http://www. dataminingblog. com/standardization-vsnormalization (15.6. 2016.)* .

[62] Schubach, Max, Matteo Re, Peter N Robinson, and Giorgio Valentini. "Imbalance-aware machine learning for predicting rare and common disease-associated non-coding variants." *Scientific reports* 7, 1: (2017) 2959.

[63] Shlens, Jonathon. "A tutorial on principal component analysis." *arXiv preprint arXiv:1404.1100* .

[64] SMEs, IN. "GUIDELINES FOR IT SECURITY." .

[65] Smith, Lindsay I. "A tutorial on principal components analysis." Technical report, 2002.

[66] Spiegelhalter, David J, Nicola G Best, Bradley P Carlin, and Angelika Van Der Linde. "Bayesian measures of model complexity and fit." *Journal of the royal statistical society: Series b (statistical methodology)* 64, 4: (2002) 583–639.

[67] Tan, P, and M Kumar Steinbach. "V.(2005) Introduction to Data Mining.", 2011.

[68] Tan, Pang-Ning. *Introduction to data mining.* Pearson Education India, 2018.

[69] Turban, Efraim, David King, Jae Kyu Lee, Ting-Peng Liang, and Deborrah C Turban. "E-Commerce Security and Fraud Issues and Protections." In *Electronic Commerce*, Springer, 2015, 457–518.

[70] Visa, Sofia, Brian Ramsay, Anca L Ralescu, and Esther Van Der Knaap. "Confusion Matrix-based Feature Selection." *MAICS* 710: (2011) 120–127.

[71] Wang, Gang, Jian Ma, and Shanlin Yang. “An improved boosting based on feature selection for corporate bankruptcy prediction.” *Expert Systems with Applications* 41, 5: (2014) 2353–2361.

[72] Wang, Xuechuan, and Kuldip K Paliwal. “Feature extraction and dimensionality reduction algorithms and their applications in vowel recognition.” *Pattern recognition* 36, 10: (2003) 2429–2439.

[73] Xue, Bing, Mengjie Zhang, and Will N Browne. “Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms.” *Applied soft computing* 18: (2014) 261–276.

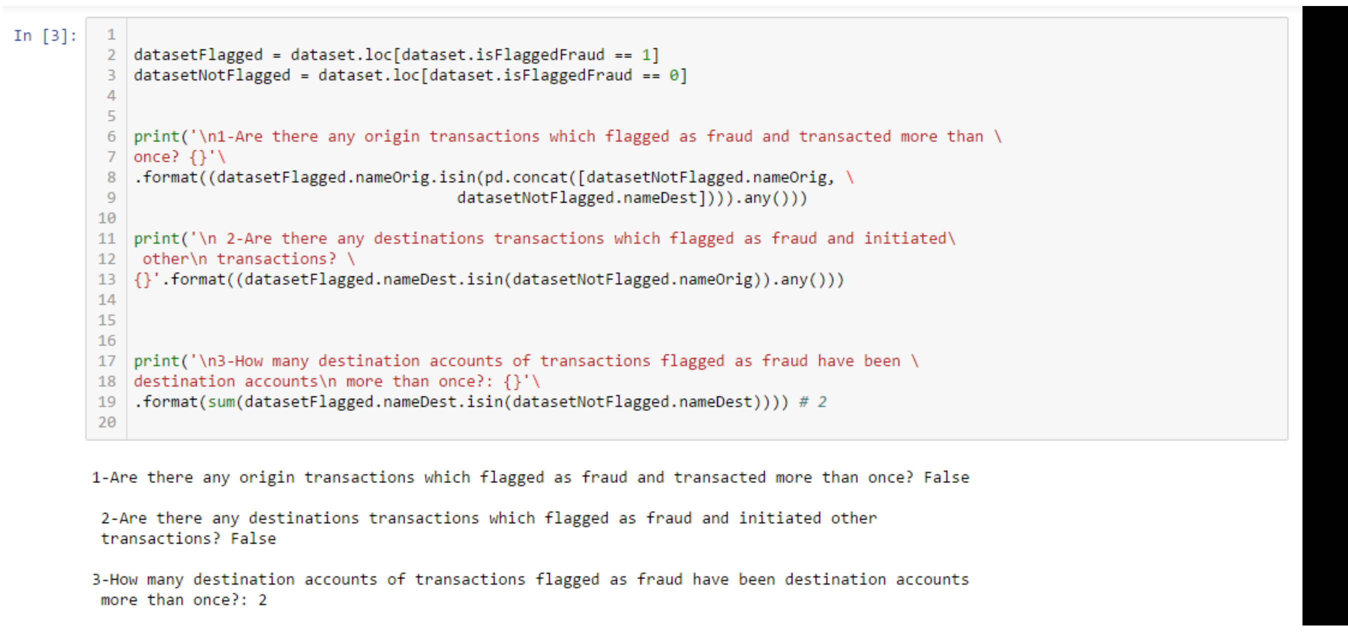
[74] Zareapoor, Masoumeh, Pourya Shamsolmoali, et al. “Application of credit card fraud detection: Based on bagging ensemble classifier.” *Procedia computer science* 48, 2015: (2015) 679–685.

[75] Zhao, Hua, and Keyun Qin. “Mixed feature selection in incomplete decision table.” *Knowledge-Based Systems* 57: (2014) 181–190.

Appendix A

Python code

A.1 Exploratory analysis for the first dataset



```
In [3]:
datasetFlagged = dataset.loc[dataset.isFlaggedFraud == 1]
datasetNotFlagged = dataset.loc[dataset.isFlaggedFraud == 0]


print('\n1-Are there any origin transactions which flagged as fraud and transacted more than \
once? {}'\
.format((datasetFlagged.nameOrig.isin(pd.concat([datasetNotFlagged.nameOrig, \
                                   datasetNotFlagged.nameDest]))).any()))

print('\n 2-Are there any destinations transactions which flagged as fraud and initiated\
 other\n transactions? \
{}'.format((datasetFlagged.nameDest.isin(datasetNotFlagged.nameOrig)).any()))



print('\n3-How many destination accounts of transactions flagged as fraud have been \
destination accounts\n more than once?: {}'\
.format(sum(datasetFlagged.nameDest.isin(datasetNotFlagged.nameDest)))) # 2

```

```
1-Are there any origin transactions which flagged as fraud and transacted more than once? False

 2-Are there any destinations transactions which flagged as fraud and initiated other
 transactions? False

3-How many destination accounts of transactions flagged as fraud have been destination accounts
 more than once?: 2
```

In [7]:
```
print('\nFraudulent TRANSFERs whose destination accounts are originators of \
genuine CASH_OUTs: \n\n{}'.format(datasetFraudTransfer.loc[datasetFraudTransfer.nameDest.\
isin(datasetNotFraud.loc[datasetNotFraud.type == 'CASH_OUT'].nameOrig.drop_duplicates())]))
print('\nFraudulent TRANSFER to C423543548 occured at step = 486 whereas \
genuine CASH_OUT from this account occured earlier at step = {}'.format(\
datasetNotFraud.loc[(datasetNotFraud.type == 'CASH_OUT') & (datasetNotFraud.nameOrig == \
                             'C423543548')].step.values))
```

```
Fraudulent TRANSFERs whose destination accounts are originators of genuine CASH_OUTs:

         step      type      amount     nameOrig  oldbalanceOrg  \
1030443    65  TRANSFER  1282971.57  C1175896731     1282971.57
6039814   486  TRANSFER   214793.32  C2140495649      214793.32
6362556   738  TRANSFER   814689.88  C2029041842      814689.88

         newbalanceOrig     nameDest  oldbalanceDest  newbalanceDest  isFraud  \
1030443             0.0  C1714931087             0.0             0.0        1
6039814             0.0   C423543548             0.0             0.0        1
6362556             0.0  C1023330867             0.0             0.0        1

         isFlaggedFraud
1030443               0
6039814               0
6362556               0

Fraudulent TRANSFER to C423543548 occured at step = 486 whereas genuine CASH_OUT from this account occured earlier at step = [1
85]
```

In [10]:
```
X = df.loc[(df.type == 1) | (df.type == 2)]
Y = X['isFraud']
Xfraud = X.loc[Y == 1]
XnonFraud = X.loc[Y == 0]
print('\nThe percentage of fraudulent transactions where \'oldbalanceDest\' = \
\'newbalanceDest\' = 0 although the \ntransacted \'amount\' is non-zero is: {}'.\
format(len(Xfraud.loc[(Xfraud.oldbalanceDest == 0) & \
(Xfraud.newbalanceDest == 0) & (Xfraud.amount)]) / (1.0 * len(Xfraud))))

print('\nThe percentage of genuine transactions where \'oldbalanceDest\' = \
newbalanceDest\' = 0 although the\n transacted \'amount\' is non-zero is: {}'.\
format(len(XnonFraud.loc[(XnonFraud.oldbalanceDest == 0) & \
(XnonFraud.newbalanceDest == 0) & (XnonFraud.amount)]) / (1.0 * len(XnonFraud))))
```

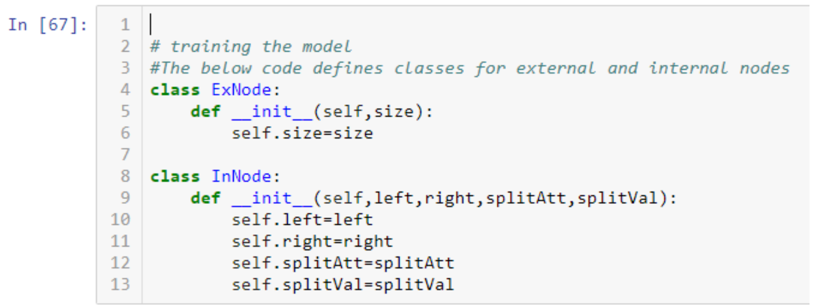
```
The percentage of fraudulent transactions where 'oldbalanceDest' = 'newbalanceDest' = 0 although the
transacted 'amount' is non-zero is: 0.4955558261293072

The percentage of genuine transactions where 'oldbalanceDest' = newbalanceDest' = 0 although the
 transacted 'amount' is non-zero is: 0.0006176245277308345
```

We can see that the percentage of transactions, where zero likely denotes a missing value, is much larger in fraudulent (50%) compared to genuine transactions (0.06%).

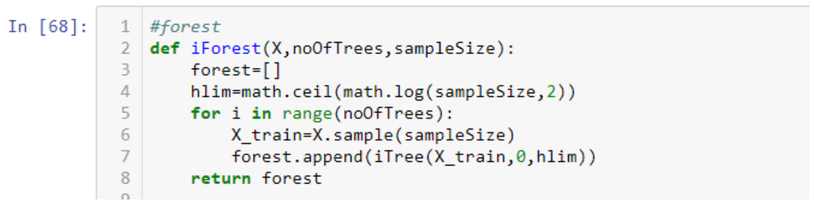
In [11]:
```
print('\nThe percentage of fraudulent transactions where \'oldbalanceOrg\' = \
\'newbalanceorg\' = 0 although the \ntransacted \'amount\' is non-zero is: {}'.\
format(len(Xfraud.loc[(Xfraud.oldbalanceOrg == 0) & \
(Xfraud.newbalanceOrig == 0) & (Xfraud.amount)]) / (1.0 * len(Xfraud))))

print('\nThe percentage of genuine transactions where \'oldbalanceDest\' = \
newbalanceDest\' = 0 although transacted \'amount\' is non-zero is: {}'.\
format(len(XnonFraud.loc[(XnonFraud.oldbalanceOrg == 0) & \
(XnonFraud.newbalanceOrig == 0) & (XnonFraud.amount)]) / (1.0 * len(XnonFraud))))
```

```
The percentage of fraudulent transactions where 'oldbalanceOrg' = 'newbalanceorg' = 0 although the
transacted 'amount' is non-zero is: 0.0030439547059539756

The percentage of genuine transactions where 'oldbalanceDest' = newbalanceDest' = 0 although the
 transacted 'amount' is non-zero is: 0.4737321319703598
```

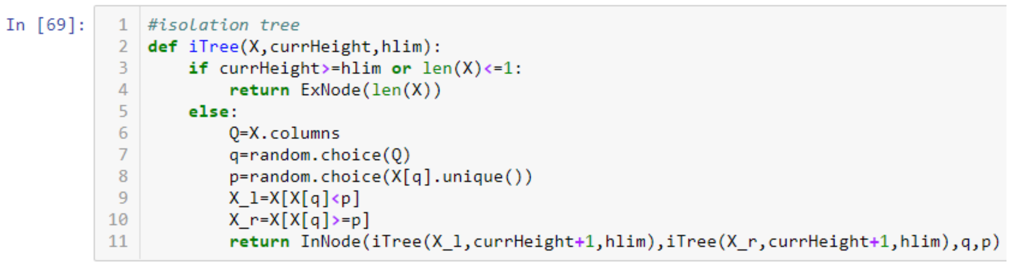
A.2 Applying Isolation forest



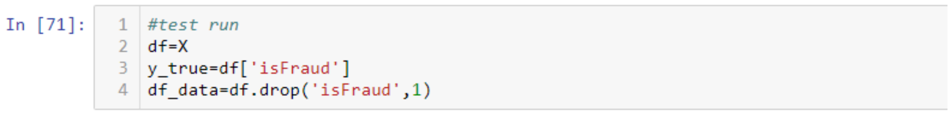
In [67]:
```python
|
# training the model
#The below code defines classes for external and internal nodes
class ExNode:
    def __init__(self,size):
        self.size=size

class InNode:
    def __init__(self,left,right,splitAtt,splitVal):
        self.left=left
        self.right=right
        self.splitAtt=splitAtt
        self.splitVal=splitVal
```



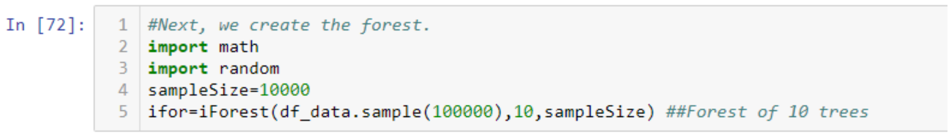
In [68]:
```python
#forest
def iForest(X,noOfTrees,sampleSize):
    forest=[]
    hlim=math.ceil(math.log(sampleSize,2))
    for i in range(noOfTrees):
        X_train=X.sample(sampleSize)
        forest.append(iTree(X_train,0,hlim))
    return forest
```



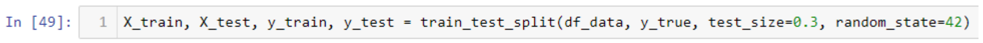
In [69]:
```python
#isolation tree
def iTree(X,currHeight,hlim):
    if currHeight>=hlim or len(X)<=1:
        return ExNode(len(X))
    else:
        Q=X.columns
        q=random.choice(Q)
        p=random.choice(X[q].unique())
        X_l=X[X[q]<p]
        X_r=X[X[q]>=p]
        return InNode(iTree(X_l,currHeight+1,hlim),iTree(X_r,currHeight+1,hlim),q,p)
```



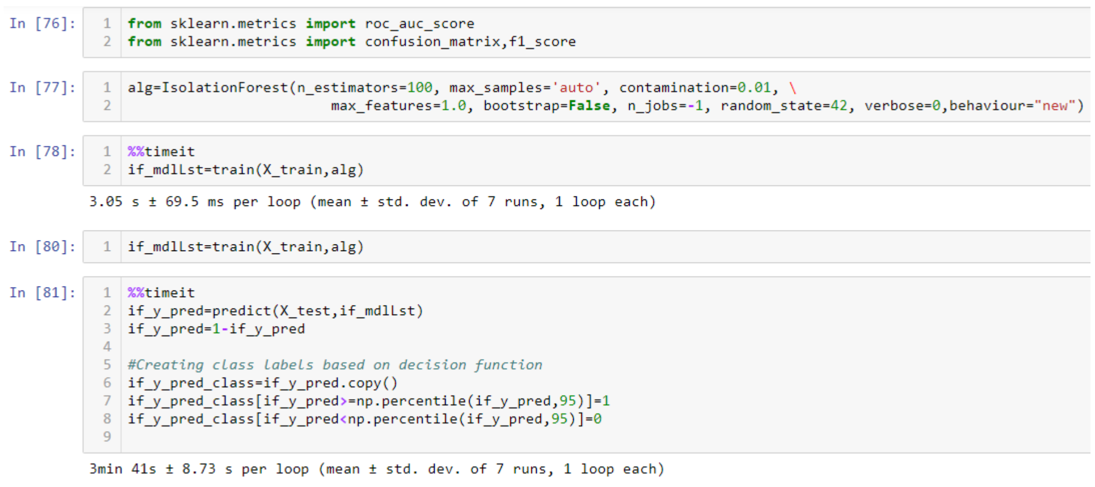
In [71]:
```python
#test run
df=X
y_true=df['isFraud']
df_data=df.drop('isFraud',1)
```



In [72]:
```python
#Next, we create the forest.
import math
import random
sampleSize=10000
ifor=iForest(df_data.sample(100000),10,sampleSize) ##Forest of 10 trees
```



In [49]:
```python
X_train, X_test, y_train, y_test = train_test_split(df_data, y_true, test_size=0.3, random_state=42)
```



```
In [76]: from sklearn.metrics import roc_auc_score
         from sklearn.metrics import confusion_matrix,f1_score
```

```
In [77]: alg=IsolationForest(n_estimators=100, max_samples='auto', contamination=0.01, \
                                 max_features=1.0, bootstrap=False, n_jobs=-1, random_state=42, verbose=0,behaviour="new")
```

```
In [78]: %%timeit
         if_mdlLst=train(X_train,alg)

         3.05 s ± 69.5 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

```
In [80]: if_mdlLst=train(X_train,alg)
```

```
In [81]: %%timeit
         if_y_pred=predict(X_test,if_mdlLst)
         if_y_pred=1-if_y_pred

         #Creating class labels based on decision function
         if_y_pred_class=if_y_pred.copy()
         if_y_pred_class[if_y_pred>=np.percentile(if_y_pred,95)]=1
         if_y_pred_class[if_y_pred<np.percentile(if_y_pred,95)]=0


         3min 41s ± 8.73 s per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Appendix B

Results of applying GAN on the first dataset

```
0 [D loss: 0.388372, acc: 46.88%, op_acc: 34.38%] [G loss: 0.980453]

Epoch: 0, F1: 0.00395, F1P: 0

1 [D loss: 0.369019, acc: 65.62%, op_acc: 46.88%] [G loss: 1.029450]

2 [D loss: 0.383596, acc: 59.38%, op_acc: 28.12%] [G loss: 1.036421]

3 [D loss: 0.402863, acc: 50.00%, op_acc: 40.62%] [G loss: 1.055181]

4 [D loss: 0.387442, acc: 50.00%, op_acc: 34.38%] [G loss: 0.878463]

5 [D loss: 0.392806, acc: 62.50%, op_acc: 25.00%] [G loss: 1.259656]

6 [D loss: 0.395202, acc: 56.25%, op_acc: 37.50%] [G loss: 1.054989]
```

```
7 [D loss: 0.401006, acc: 56.25%, op_acc: 31.25%] [G loss: 1.006198]

8 [D loss: 0.377026, acc: 56.25%, op_acc: 31.25%] [G loss: 1.003893]

9 [D loss: 0.419331, acc: 53.12%, op_acc: 18.75%] [G loss: 1.079152]

10 [D loss: 0.389829, acc: 43.75%, op_acc: 40.62%] [G loss: 1.005554]

Epoch: 10, F1: 0.00379, F1P: 1

11 [D loss: 0.393789, acc: 65.62%, op_acc: 43.75%] [G loss: 0.986814]

12 [D loss: 0.396514, acc: 53.12%, op_acc: 31.25%] [G loss: 1.148525]

13 [D loss: 0.390434, acc: 50.00%, op_acc: 25.00%] [G loss: 1.019702]

14 [D loss: 0.355913, acc: 56.25%, op_acc: 40.62%] [G loss: 0.930842]

15 [D loss: 0.366537, acc: 50.00%, op_acc: 28.12%] [G loss: 1.138922]

16 [D loss: 0.366608, acc: 59.38%, op_acc: 28.12%] [G loss: 1.020915]

17 [D loss: 0.378144, acc: 71.88%, op_acc: 34.38%] [G loss: 1.129293]

18 [D loss: 0.397328, acc: 65.62%, op_acc: 40.62%] [G loss: 1.050206]

19 [D loss: 0.362097, acc: 68.75%, op_acc: 37.50%] [G loss: 1.080697]

20 [D loss: 0.385284, acc: 65.62%, op_acc: 28.12%] [G loss: 1.174599]

Epoch: 20, F1: 0.00384, F1P: 2

21 [D loss: 0.368288, acc: 65.62%, op_acc: 31.25%] [G loss: 1.096186]

22 [D loss: 0.377095, acc: 68.75%, op_acc: 40.62%] [G loss: 0.980444]

23 [D loss: 0.358401, acc: 65.62%, op_acc: 34.38%] [G loss: 0.872729]

24 [D loss: 0.379310, acc: 62.50%, op_acc: 21.88%] [G loss: 1.039133]

25 [D loss: 0.387150, acc: 53.12%, op_acc: 37.50%] [G loss: 1.100781]

26 [D loss: 0.401239, acc: 59.38%, op_acc: 34.38%] [G loss: 1.017507]

27 [D loss: 0.363280, acc: 78.12%, op_acc: 37.50%] [G loss: 1.017547]
```

```
28 [D loss: 0.391747, acc: 65.62%, op_acc: 31.25%] [G loss: 1.025161]

29 [D loss: 0.377832, acc: 56.25%, op_acc: 53.12%] [G loss: 0.951055]

30 [D loss: 0.385409, acc: 68.75%, op_acc: 34.38%] [G loss: 1.022846]

Epoch: 30, F1: 0.00379, F1P: 3

31 [D loss: 0.346957, acc: 81.25%, op_acc: 31.25%] [G loss: 1.100900]

32 [D loss: 0.346846, acc: 84.38%, op_acc: 28.12%] [G loss: 1.070017]

33 [D loss: 0.368737, acc: 75.00%, op_acc: 31.25%] [G loss: 1.259647]

34 [D loss: 0.370361, acc: 75.00%, op_acc: 28.12%] [G loss: 1.074761]

35 [D loss: 0.368185, acc: 65.62%, op_acc: 37.50%] [G loss: 0.992247]

36 [D loss: 0.375514, acc: 59.38%, op_acc: 34.38%] [G loss: 1.086171]

37 [D loss: 0.343374, acc: 68.75%, op_acc: 43.75%] [G loss: 0.952512]

38 [D loss: 0.372462, acc: 71.88%, op_acc: 28.12%] [G loss: 1.072056]

39 [D loss: 0.359133, acc: 75.00%, op_acc: 15.62%] [G loss: 1.113323]

40 [D loss: 0.379296, acc: 62.50%, op_acc: 31.25%] [G loss: 1.006910]

Epoch: 40, F1: 0.00353, F1P: 4

41 [D loss: 0.343790, acc: 78.12%, op_acc: 37.50%] [G loss: 1.028478]

42 [D loss: 0.349871, acc: 75.00%, op_acc: 28.12%] [G loss: 1.014985]

43 [D loss: 0.347341, acc: 71.88%, op_acc: 25.00%] [G loss: 1.011410]

44 [D loss: 0.366187, acc: 62.50%, op_acc: 31.25%] [G loss: 0.930188]

45 [D loss: 0.349913, acc: 59.38%, op_acc: 31.25%] [G loss: 1.093007]

46 [D loss: 0.351341, acc: 71.88%, op_acc: 43.75%] [G loss: 1.265074]

47 [D loss: 0.388178, acc: 71.88%, op_acc: 31.25%] [G loss: 1.103547]

48 [D loss: 0.367320, acc: 84.38%, op_acc: 31.25%] [G loss: 0.995281]
```

```
49 [D loss: 0.382987, acc: 75.00%, op_acc: 28.12%] [G loss: 1.090960]

50 [D loss: 0.344763, acc: 81.25%, op_acc: 34.38%] [G loss: 1.038033]

Epoch: 50, F1: 0.00398, F1P: 5

51 [D loss: 0.386140, acc: 62.50%, op_acc: 28.12%] [G loss: 1.064941]

52 [D loss: 0.343318, acc: 81.25%, op_acc: 46.88%] [G loss: 1.109327]

53 [D loss: 0.353072, acc: 75.00%, op_acc: 40.62%] [G loss: 1.227325]

54 [D loss: 0.370626, acc: 84.38%, op_acc: 12.50%] [G loss: 1.003191]

55 [D loss: 0.382076, acc: 81.25%, op_acc: 46.88%] [G loss: 1.095547]

56 [D loss: 0.351825, acc: 75.00%, op_acc: 34.38%] [G loss: 1.031944]

57 [D loss: 0.408638, acc: 68.75%, op_acc: 34.38%] [G loss: 1.022736]

58 [D loss: 0.371044, acc: 78.12%, op_acc: 40.62%] [G loss: 1.061581]

59 [D loss: 0.349691, acc: 68.75%, op_acc: 43.75%] [G loss: 1.226901]

60 [D loss: 0.363641, acc: 65.62%, op_acc: 31.25%] [G loss: 1.175121]

Epoch: 60, F1: 0.00384, F1P: 6

61 [D loss: 0.350298, acc: 75.00%, op_acc: 34.38%] [G loss: 1.204234]

62 [D loss: 0.355099, acc: 81.25%, op_acc: 21.88%] [G loss: 1.083098]

63 [D loss: 0.342901, acc: 87.50%, op_acc: 31.25%] [G loss: 1.060907]

64 [D loss: 0.351833, acc: 87.50%, op_acc: 40.62%] [G loss: 1.164078]

65 [D loss: 0.366630, acc: 78.12%, op_acc: 25.00%] [G loss: 1.064986]

66 [D loss: 0.356171, acc: 71.88%, op_acc: 37.50%] [G loss: 1.027031]

67 [D loss: 0.368864, acc: 75.00%, op_acc: 37.50%] [G loss: 1.003442]

68 [D loss: 0.342178, acc: 78.12%, op_acc: 31.25%] [G loss: 1.086747]

69 [D loss: 0.357443, acc: 75.00%, op_acc: 34.38%] [G loss: 1.095957]
```

```
70 [D loss: 0.330776, acc: 84.38%, op_acc: 31.25%] [G loss: 1.173773]
Epoch: 70, F1: 0.00413, F1P: 7
71 [D loss: 0.347356, acc: 84.38%, op_acc: 28.12%] [G loss: 1.204525]
72 [D loss: 0.350473, acc: 90.62%, op_acc: 40.62%] [G loss: 1.083327]
73 [D loss: 0.354294, acc: 71.88%, op_acc: 31.25%] [G loss: 1.110950]
74 [D loss: 0.353295, acc: 78.12%, op_acc: 34.38%] [G loss: 1.208734]
75 [D loss: 0.359541, acc: 81.25%, op_acc: 25.00%] [G loss: 0.994025]
76 [D loss: 0.370640, acc: 75.00%, op_acc: 28.12%] [G loss: 1.188369]
77 [D loss: 0.361537, acc: 71.88%, op_acc: 28.12%] [G loss: 1.109946]
78 [D loss: 0.340829, acc: 84.38%, op_acc: 25.00%] [G loss: 1.104137]
79 [D loss: 0.363370, acc: 84.38%, op_acc: 50.00%] [G loss: 1.043749]
80 [D loss: 0.353581, acc: 84.38%, op_acc: 34.38%] [G loss: 1.198173]
Epoch: 80, F1: 0.00406, F1P: 8
81 [D loss: 0.330877, acc: 81.25%, op_acc: 37.50%] [G loss: 1.156768]
82 [D loss: 0.329835, acc: 87.50%, op_acc: 25.00%] [G loss: 1.124117]
83 [D loss: 0.328976, acc: 90.62%, op_acc: 34.38%] [G loss: 1.157781]
84 [D loss: 0.338179, acc: 84.38%, op_acc: 37.50%] [G loss: 1.229280]
85 [D loss: 0.317814, acc: 84.38%, op_acc: 28.12%] [G loss: 1.287171]
86 [D loss: 0.357675, acc: 75.00%, op_acc: 34.38%] [G loss: 1.102918]
87 [D loss: 0.358966, acc: 81.25%, op_acc: 46.88%] [G loss: 1.174587]
88 [D loss: 0.327095, acc: 87.50%, op_acc: 37.50%] [G loss: 1.115591]
89 [D loss: 0.349651, acc: 87.50%, op_acc: 34.38%] [G loss: 1.100403]
90 [D loss: 0.328115, acc: 71.88%, op_acc: 31.25%] [G loss: 1.086079]
```

```
Epoch: 90, F1: 0.00420, F1P: 9

91 [D loss: 0.338331, acc: 78.12%, op_acc: 25.00%] [G loss: 1.176208]

92 [D loss: 0.336712, acc: 87.50%, op_acc: 40.62%] [G loss: 1.166511]

93 [D loss: 0.329621, acc: 90.62%, op_acc: 37.50%] [G loss: 1.216025]

94 [D loss: 0.336393, acc: 81.25%, op_acc: 28.12%] [G loss: 1.091293]

95 [D loss: 0.325472, acc: 81.25%, op_acc: 31.25%] [G loss: 1.289682]

96 [D loss: 0.307790, acc: 96.88%, op_acc: 37.50%] [G loss: 1.028445]

97 [D loss: 0.334575, acc: 84.38%, op_acc: 43.75%] [G loss: 1.089767]

98 [D loss: 0.374130, acc: 81.25%, op_acc: 28.12%] [G loss: 1.270608]

99 [D loss: 0.361722, acc: 84.38%, op_acc: 37.50%] [G loss: 1.168599]

100 [D loss: 0.339357, acc: 84.38%, op_acc: 46.88%] [G loss: 1.205003]


Epoch: 500, F1: 0.00575, F1P: 50

501 [D loss: 0.227786, acc: 100.00%, op_acc: 65.62%] [G loss: 1.94442]

502 [D loss: 0.222472, acc: 96.88%, op_acc: 62.50%] [G loss: 2.028450]

503 [D loss: 0.228585, acc: 96.88%, op_acc: 65.62%] [G loss: 1.819132]

504 [D loss: 0.238264, acc: 96.88%, op_acc: 59.38%] [G loss: 1.916777]

505 [D loss: 0.222891, acc: 100.00%, op_acc: 71.88%] [G loss: 2.13247]

506 [D loss: 0.221938, acc: 100.00%, op_acc: 62.50%] [G loss: 2.09477]

507 [D loss: 0.212873, acc: 100.00%, op_acc: 62.50%] [G loss: 2.00776]

508 [D loss: 0.220372, acc: 100.00%, op_acc: 78.12%] [G loss: 1.89579]

509 [D loss: 0.223827, acc: 96.88%, op_acc: 62.50%] [G loss: 1.914884]

510 [D loss: 0.239364, acc: 96.88%, op_acc: 78.12%] [G loss: 2.038812]
```

```
Epoch: 510, F1: 0.00597, F1P: 51

511 [D loss: 0.228686, acc: 100.00%, op_acc: 65.62%] [G loss: 2.06582]

512 [D loss: 0.233719, acc: 100.00%, op_acc: 62.50%] [G loss: 2.15259]

513 [D loss: 0.226409, acc: 96.88%, op_acc: 71.88%] [G loss: 1.870960]

514 [D loss: 0.221831, acc: 100.00%, op_acc: 65.62%] [G loss: 2.24392]

515 [D loss: 0.215504, acc: 100.00%, op_acc: 62.50%] [G loss: 2.11453]

516 [D loss: 0.221144, acc: 100.00%, op_acc: 46.88%] [G loss: 2.02267]

517 [D loss: 0.210474, acc: 100.00%, op_acc: 65.62%] [G loss: 2.18487]

518 [D loss: 0.222838, acc: 100.00%, op_acc: 59.38%] [G loss: 1.95171]

519 [D loss: 0.222735, acc: 96.88%, op_acc: 50.00%] [G loss: 1.847033]

520 [D loss: 0.228960, acc: 96.88%, op_acc: 59.38%] [G loss: 2.156384]

Epoch: 520, F1: 0.00595, F1P: 52

521 [D loss: 0.232600, acc: 100.00%, op_acc: 62.50%] [G loss: 2.06668]

522 [D loss: 0.231438, acc: 96.88%, op_acc: 56.25%] [G loss: 2.109175]

523 [D loss: 0.208763, acc: 100.00%, op_acc: 59.38%] [G loss: 2.03060]

524 [D loss: 0.226587, acc: 96.88%, op_acc: 65.62%] [G loss: 2.037552]

525 [D loss: 0.216548, acc: 96.88%, op_acc: 75.00%] [G loss: 1.821892]

526 [D loss: 0.217384, acc: 100.00%, op_acc: 53.12%] [G loss: 1.99881]

527 [D loss: 0.226414, acc: 96.88%, op_acc: 62.50%] [G loss: 1.896370]

528 [D loss: 0.222758, acc: 96.88%, op_acc: 71.88%] [G loss: 2.202181]

529 [D loss: 0.219848, acc: 100.00%, op_acc: 56.25%] [G loss: 2.17375]

530 [D loss: 0.229850, acc: 96.88%, op_acc: 65.62%] [G loss: 2.291156]

Epoch: 530, F1: 0.00593, F1P: 53
```

```
531 [D loss: 0.215865, acc: 100.00%, op_acc: 62.50%] [G loss: 2.04281]

532 [D loss: 0.197289, acc: 100.00%, op_acc: 65.62%] [G loss: 2.02647]

533 [D loss: 0.219943, acc: 96.88%, op_acc: 62.50%] [G loss: 1.990584]

534 [D loss: 0.224672, acc: 100.00%, op_acc: 59.38%] [G loss: 2.01251]

535 [D loss: 0.209343, acc: 100.00%, op_acc: 75.00%] [G loss: 1.84384]

536 [D loss: 0.227330, acc: 100.00%, op_acc: 65.62%] [G loss: 2.36470]

537 [D loss: 0.233489, acc: 93.75%, op_acc: 65.62%] [G loss: 2.018972]

538 [D loss: 0.207773, acc: 100.00%, op_acc: 71.88%] [G loss: 2.08678]

539 [D loss: 0.216393, acc: 96.88%, op_acc: 59.38%] [G loss: 2.111537]

540 [D loss: 0.218380, acc: 100.00%, op_acc: 50.00%] [G loss: 1.61455]

Epoch: 540, F1: 0.00586, F1P: 54

541 [D loss: 0.236222, acc: 100.00%, op_acc: 75.00%] [G loss: 2.05690]

542 [D loss: 0.215652, acc: 100.00%, op_acc: 68.75%] [G loss: 2.31078]

543 [D loss: 0.211352, acc: 100.00%, op_acc: 65.62%] [G loss: 2.01894]

544 [D loss: 0.211999, acc: 100.00%, op_acc: 65.62%] [G loss: 1.92498]

545 [D loss: 0.211139, acc: 100.00%, op_acc: 59.38%] [G loss: 2.41046]

546 [D loss: 0.196815, acc: 100.00%, op_acc: 75.00%] [G loss: 2.12359]

547 [D loss: 0.221216, acc: 100.00%, op_acc: 65.62%] [G loss: 2.00521]

548 [D loss: 0.242913, acc: 96.88%, op_acc: 56.25%] [G loss: 2.267944]

549 [D loss: 0.216693, acc: 96.88%, op_acc: 65.62%] [G loss: 2.081550]

550 [D loss: 0.196163, acc: 100.00%, op_acc: 78.12%] [G loss: 2.22762]

Epoch: 550, F1: 0.00597, F1P: 55

551 [D loss: 0.220153, acc: 100.00%, op_acc: 62.50%] [G loss: 2.11286]
```

```
552 [D loss: 0.228732, acc: 100.00%, op_acc: 65.62%] [G loss: 2.12226]

553 [D loss: 0.213657, acc: 96.88%, op_acc: 59.38%] [G loss: 2.013509]

554 [D loss: 0.212929, acc: 100.00%, op_acc: 68.75%] [G loss: 2.05641]

555 [D loss: 0.213805, acc: 96.88%, op_acc: 75.00%] [G loss: 1.986143]

556 [D loss: 0.201757, acc: 100.00%, op_acc: 68.75%] [G loss: 2.15366]

557 [D loss: 0.224333, acc: 100.00%, op_acc: 68.75%] [G loss: 2.00466]

558 [D loss: 0.224054, acc: 100.00%, op_acc: 71.88%] [G loss: 2.16446]

559 [D loss: 0.206841, acc: 100.00%, op_acc: 75.00%] [G loss: 2.12076]

560 [D loss: 0.217000, acc: 93.75%, op_acc: 65.62%] [G loss: 2.067472]

Epoch: 560, F1: 0.00593, F1P: 56

561 [D loss: 0.216138, acc: 96.88%, op_acc: 53.12%] [G loss: 2.264142]

562 [D loss: 0.219940, acc: 96.88%, op_acc: 78.12%] [G loss: 2.085659]

563 [D loss: 0.228012, acc: 96.88%, op_acc: 62.50%] [G loss: 2.073871]

564 [D loss: 0.199266, acc: 100.00%, op_acc: 65.62%] [G loss: 1.99099]

565 [D loss: 0.216650, acc: 100.00%, op_acc: 68.75%] [G loss: 2.05356]

566 [D loss: 0.213854, acc: 100.00%, op_acc: 65.62%] [G loss: 2.21450]

567 [D loss: 0.214523, acc: 100.00%, op_acc: 71.88%] [G loss: 2.19261]

568 [D loss: 0.212049, acc: 100.00%, op_acc: 65.62%] [G loss: 2.35149]

569 [D loss: 0.222211, acc: 100.00%, op_acc: 65.62%] [G loss: 2.00050]

570 [D loss: 0.224943, acc: 93.75%, op_acc: 84.38%] [G loss: 1.878095]

Epoch: 570, F1: 0.00592, F1P: 57

571 [D loss: 0.213443, acc: 100.00%, op_acc: 62.50%] [G loss: 2.29339]

572 [D loss: 0.223650, acc: 100.00%, op_acc: 78.12%] [G loss: 2.02655]
```

```
573 [D loss: 0.220934, acc: 96.88%, op_acc: 81.25%] [G loss: 2.256332]

574 [D loss: 0.219788, acc: 96.88%, op_acc: 75.00%] [G loss: 2.189319]

575 [D loss: 0.213233, acc: 96.88%, op_acc: 71.88%] [G loss: 2.112656]

576 [D loss: 0.205963, acc: 100.00%, op_acc: 84.38%] [G loss: 2.09018]

577 [D loss: 0.214882, acc: 96.88%, op_acc: 75.00%] [G loss: 2.081645]

578 [D loss: 0.215297, acc: 100.00%, op_acc: 62.50%] [G loss: 1.96156]

579 [D loss: 0.208544, acc: 100.00%, op_acc: 62.50%] [G loss: 2.19218]

580 [D loss: 0.204190, acc: 96.88%, op_acc: 78.12%] [G loss: 2.036752]

Epoch: 580, F1: 0.00586, F1P: 58

581 [D loss: 0.215754, acc: 100.00%, op_acc: 68.75%] [G loss: 2.31270]

582 [D loss: 0.221348, acc: 96.88%, op_acc: 68.75%] [G loss: 2.143085]

583 [D loss: 0.205514, acc: 100.00%, op_acc: 81.25%] [G loss: 2.34886]

584 [D loss: 0.203746, acc: 96.88%, op_acc: 68.75%] [G loss: 2.178059]

585 [D loss: 0.217072, acc: 100.00%, op_acc: 68.75%] [G loss: 2.07933]

586 [D loss: 0.212626, acc: 100.00%, op_acc: 84.38%] [G loss: 2.24100]

587 [D loss: 0.202491, acc: 100.00%, op_acc: 68.75%] [G loss: 2.12645]

588 [D loss: 0.215368, acc: 96.88%, op_acc: 71.88%] [G loss: 2.216654]

589 [D loss: 0.203691, acc: 100.00%, op_acc: 65.62%] [G loss: 2.32066]

590 [D loss: 0.221812, acc: 100.00%, op_acc: 65.62%] [G loss: 1.98377]

Epoch: 590, F1: 0.00584, F1P: 59

591 [D loss: 0.209065, acc: 100.00%, op_acc: 68.75%] [G loss: 2.00634]

592 [D loss: 0.209258, acc: 100.00%, op_acc: 71.88%] [G loss: 2.11220]

593 [D loss: 0.196960, acc: 100.00%, op_acc: 59.38%] [G loss: 2.24342]
```

```
594 [D loss: 0.203352, acc: 100.00%, op_acc: 75.00%] [G loss: 2.22218]

595 [D loss: 0.231012, acc: 96.88%, op_acc: 62.50%] [G loss: 2.232725]

596 [D loss: 0.191806, acc: 100.00%, op_acc: 78.12%] [G loss: 2.36922]

597 [D loss: 0.208465, acc: 96.88%, op_acc: 71.88%] [G loss: 1.925818]

598 [D loss: 0.229231, acc: 96.88%, op_acc: 75.00%] [G loss: 2.262008]

599 [D loss: 0.199340, acc: 100.00%, op_acc: 75.00%] [G loss: 2.10920]

600 [D loss: 0.239622, acc: 96.88%, op_acc: 62.50%] [G loss: 2.275958]

Epoch: 600, F1: 0.00595, F1P: 60

601 [D loss: 0.192857, acc: 100.00%, op_acc: 75.00%] [G loss: 1.96953]

602 [D loss: 0.224592, acc: 93.75%, op_acc: 62.50%] [G loss: 2.134129]

603 [D loss: 0.211953, acc: 93.75%, op_acc: 71.88%] [G loss: 2.413849]

604 [D loss: 0.195760, acc: 100.00%, op_acc: 65.62%] [G loss: 2.18795]

605 [D loss: 0.213272, acc: 96.88%, op_acc: 71.88%] [G loss: 2.338486]

606 [D loss: 0.204611, acc: 96.88%, op_acc: 68.75%] [G loss: 1.947411]

607 [D loss: 0.197507, acc: 100.00%, op_acc: 71.88%] [G loss: 2.52897]

608 [D loss: 0.189577, acc: 100.00%, op_acc: 78.12%] [G loss: 2.49939]

609 [D loss: 0.209974, acc: 100.00%, op_acc: 68.75%] [G loss: 2.23464]

610 [D loss: 0.204842, acc: 100.00%, op_acc: 71.88%] [G loss: 2.19266]

Epoch: 610, F1: 0.00592, F1P: 61

611 [D loss: 0.198953, acc: 100.00%, op_acc: 68.75%] [G loss: 2.28554]

612 [D loss: 0.212061, acc: 96.88%, op_acc: 75.00%] [G loss: 2.237705]

613 [D loss: 0.222665, acc: 100.00%, op_acc: 56.25%] [G loss: 2.24613]

614 [D loss: 0.205478, acc: 96.88%, op_acc: 71.88%] [G loss: 2.482052]
```

```
615 [D loss: 0.191576, acc: 100.00%, op_acc: 68.75%] [G loss: 2.11372]

616 [D loss: 0.206083, acc: 100.00%, op_acc: 78.12%] [G loss: 2.16430]

617 [D loss: 0.202101, acc: 100.00%, op_acc: 71.88%] [G loss: 2.37268]

618 [D loss: 0.201116, acc: 100.00%, op_acc: 65.62%] [G loss: 2.09126]

619 [D loss: 0.215358, acc: 96.88%, op_acc: 71.88%] [G loss: 2.518104]

620 [D loss: 0.215601, acc: 100.00%, op_acc: 65.62%] [G loss: 2.25407]

Epoch: 620, F1: 0.00460, F1P: 62

621 [D loss: 0.208782, acc: 96.88%, op_acc: 81.25%] [G loss: 2.304709]

622 [D loss: 0.189668, acc: 100.00%, op_acc: 68.75%] [G loss: 2.30750]

623 [D loss: 0.203803, acc: 100.00%, op_acc: 78.12%] [G loss: 2.22879]

624 [D loss: 0.188089, acc: 100.00%, op_acc: 81.25%] [G loss: 2.14154]

625 [D loss: 0.212060, acc: 96.88%, op_acc: 93.75%] [G loss: 2.085093]

626 [D loss: 0.209757, acc: 100.00%, op_acc: 68.75%] [G loss: 1.97303]

627 [D loss: 0.202033, acc: 100.00%, op_acc: 65.62%] [G loss: 2.18587]

628 [D loss: 0.193833, acc: 96.88%, op_acc: 68.75%] [G loss: 2.278641]

629 [D loss: 0.207159, acc: 96.88%, op_acc: 68.75%] [G loss: 2.131680]

630 [D loss: 0.207541, acc: 96.88%, op_acc: 75.00%] [G loss: 2.252702]

Epoch: 630, F1: 0.00568, F1P: 63

631 [D loss: 0.194335, acc: 100.00%, op_acc: 68.75%] [G loss: 2.17288]

632 [D loss: 0.187727, acc: 100.00%, op_acc: 71.88%] [G loss: 2.19066]

633 [D loss: 0.195920, acc: 100.00%, op_acc: 59.38%] [G loss: 2.34229]

634 [D loss: 0.209359, acc: 100.00%, op_acc: 59.38%] [G loss: 2.38725]

635 [D loss: 0.187845, acc: 100.00%, op_acc: 81.25%] [G loss: 1.99146]
```

```
636 [D loss: 0.205579, acc: 100.00%, op_acc: 75.00%] [G loss: 2.42847]

637 [D loss: 0.201360, acc: 96.88%, op_acc: 75.00%] [G loss: 2.314417]

638 [D loss: 0.207268, acc: 100.00%, op_acc: 68.75%] [G loss: 2.28121]

639 [D loss: 0.187956, acc: 100.00%, op_acc: 75.00%] [G loss: 2.18530]

640 [D loss: 0.195965, acc: 100.00%, op_acc: 65.62%] [G loss: 2.36995]

Epoch: 640, F1: 0.00588, F1P: 64

641 [D loss: 0.201963, acc: 100.00%, op_acc: 62.50%] [G loss: 2.43646]

642 [D loss: 0.193164, acc: 96.88%, op_acc: 75.00%] [G loss: 2.314856]

643 [D loss: 0.193296, acc: 100.00%, op_acc: 75.00%] [G loss: 2.37102]

644 [D loss: 0.208606, acc: 100.00%, op_acc: 71.88%] [G loss: 2.02897]

645 [D loss: 0.190155, acc: 100.00%, op_acc: 68.75%] [G loss: 2.27576]

646 [D loss: 0.221017, acc: 100.00%, op_acc: 56.25%] [G loss: 2.28114]

647 [D loss: 0.206166, acc: 100.00%, op_acc: 71.88%] [G loss: 2.33830]

648 [D loss: 0.204482, acc: 100.00%, op_acc: 68.75%] [G loss: 2.20985]

649 [D loss: 0.215241, acc: 96.88%, op_acc: 68.75%] [G loss: 2.205867]

650 [D loss: 0.186601, acc: 100.00%, op_acc: 81.25%] [G loss: 2.53859]


Epoch: 770, F1: 0.00513, F1P: 77

771 [D loss: 0.173991, acc: 100.00%, op_acc: 68.75%] [G loss: 2.36899]

772 [D loss: 0.204735, acc: 96.88%, op_acc: 75.00%] [G loss: 2.434993]

773 [D loss: 0.156707, acc: 100.00%, op_acc: 78.12%] [G loss: 2.47325]

774 [D loss: 0.183750, acc: 96.88%, op_acc: 75.00%] [G loss: 2.238315]

775 [D loss: 0.160724, acc: 100.00%, op_acc: 68.75%] [G loss: 2.61991]
```

```
776 [D loss: 0.170932, acc: 100.00%, op_acc: 81.25%] [G loss: 2.59117]

777 [D loss: 0.173348, acc: 100.00%, op_acc: 75.00%] [G loss: 2.06851]

778 [D loss: 0.191550, acc: 96.88%, op_acc: 81.25%] [G loss: 2.594759]

779 [D loss: 0.182244, acc: 96.88%, op_acc: 68.75%] [G loss: 2.634384]

780 [D loss: 0.166633, acc: 100.00%, op_acc: 75.00%] [G loss: 2.16744]

Epoch: 780, F1: 0.00590, F1P: 78

781 [D loss: 0.161917, acc: 100.00%, op_acc: 75.00%] [G loss: 2.36195]

782 [D loss: 0.175737, acc: 96.88%, op_acc: 62.50%] [G loss: 2.392230]

783 [D loss: 0.172566, acc: 100.00%, op_acc: 87.50%] [G loss: 2.63773]

784 [D loss: 0.169470, acc: 100.00%, op_acc: 65.62%] [G loss: 2.68355]

785 [D loss: 0.168693, acc: 100.00%, op_acc: 65.62%] [G loss: 2.49645]

786 [D loss: 0.185405, acc: 96.88%, op_acc: 75.00%] [G loss: 2.339440]

787 [D loss: 0.193712, acc: 96.88%, op_acc: 78.12%] [G loss: 2.340147]

788 [D loss: 0.159577, acc: 96.88%, op_acc: 75.00%] [G loss: 2.314955]

789 [D loss: 0.180192, acc: 96.88%, op_acc: 81.25%] [G loss: 2.409042]

790 [D loss: 0.152977, acc: 100.00%, op_acc: 75.00%] [G loss: 2.52291]

Epoch: 790, F1: 0.00591, F1P: 79

791 [D loss: 0.187713, acc: 100.00%, op_acc: 71.88%] [G loss: 2.79378]

792 [D loss: 0.183877, acc: 100.00%, op_acc: 62.50%] [G loss: 2.50817]

793 [D loss: 0.170140, acc: 100.00%, op_acc: 68.75%] [G loss: 2.50637]

794 [D loss: 0.169474, acc: 100.00%, op_acc: 65.62%] [G loss: 2.42272]

795 [D loss: 0.171308, acc: 96.88%, op_acc: 81.25%] [G loss: 2.682145]

796 [D loss: 0.190897, acc: 96.88%, op_acc: 71.88%] [G loss: 2.562903]
```

```
797 [D loss: 0.164758, acc: 100.00%, op_acc: 78.12%] [G loss: 2.32432]

798 [D loss: 0.174889, acc: 100.00%, op_acc: 65.62%] [G loss: 2.51514]

799 [D loss: 0.170453, acc: 100.00%, op_acc: 78.12%] [G loss: 2.55623]

800 [D loss: 0.156011, acc: 100.00%, op_acc: 75.00%] [G loss: 2.49671]

Epoch: 800, F1: 0.00582, F1P: 80

801 [D loss: 0.184436, acc: 100.00%, op_acc: 71.88%] [G loss: 2.49990]

802 [D loss: 0.160048, acc: 100.00%, op_acc: 65.62%] [G loss: 2.78225]

803 [D loss: 0.165907, acc: 100.00%, op_acc: 78.12%] [G loss: 2.12391]

804 [D loss: 0.213771, acc: 96.88%, op_acc: 68.75%] [G loss: 2.525781]

805 [D loss: 0.166143, acc: 100.00%, op_acc: 71.88%] [G loss: 2.54287]

806 [D loss: 0.187743, acc: 96.88%, op_acc: 78.12%] [G loss: 2.379322]

807 [D loss: 0.169722, acc: 100.00%, op_acc: 71.88%] [G loss: 2.14761]

808 [D loss: 0.168716, acc: 100.00%, op_acc: 75.00%] [G loss: 2.74791]

809 [D loss: 0.159669, acc: 100.00%, op_acc: 81.25%] [G loss: 2.66768]

810 [D loss: 0.154565, acc: 100.00%, op_acc: 87.50%] [G loss: 2.52468]

Epoch: 810, F1: 0.00590, F1P: 81

811 [D loss: 0.179278, acc: 100.00%, op_acc: 71.88%] [G loss: 2.64366]

812 [D loss: 0.154418, acc: 100.00%, op_acc: 81.25%] [G loss: 2.51530]

813 [D loss: 0.169489, acc: 100.00%, op_acc: 62.50%] [G loss: 2.41197]

814 [D loss: 0.154213, acc: 100.00%, op_acc: 81.25%] [G loss: 2.20485]

815 [D loss: 0.169686, acc: 100.00%, op_acc: 78.12%] [G loss: 2.34434]

816 [D loss: 0.175481, acc: 100.00%, op_acc: 75.00%] [G loss: 2.53689]

817 [D loss: 0.174666, acc: 96.88%, op_acc: 68.75%] [G loss: 2.563148]
```

```
818 [D loss: 0.159283, acc: 100.00%, op_acc: 90.62%] [G loss: 2.85816]

819 [D loss: 0.167059, acc: 100.00%, op_acc: 65.62%] [G loss: 2.54284]

820 [D loss: 0.158902, acc: 100.00%, op_acc: 65.62%] [G loss: 2.55513]

Epoch: 820, F1: 0.00596, F1P: 82

821 [D loss: 0.179659, acc: 96.88%, op_acc: 78.12%] [G loss: 2.595386]

822 [D loss: 0.186148, acc: 96.88%, op_acc: 71.88%] [G loss: 2.883087]

823 [D loss: 0.159830, acc: 100.00%, op_acc: 71.88%] [G loss: 2.35506]

824 [D loss: 0.184996, acc: 96.88%, op_acc: 78.12%] [G loss: 2.765959]

825 [D loss: 0.174522, acc: 96.88%, op_acc: 75.00%] [G loss: 2.543884]

826 [D loss: 0.174701, acc: 100.00%, op_acc: 75.00%] [G loss: 2.50068]

827 [D loss: 0.148663, acc: 100.00%, op_acc: 81.25%] [G loss: 2.52470]

828 [D loss: 0.156735, acc: 100.00%, op_acc: 68.75%] [G loss: 2.57185]

829 [D loss: 0.173894, acc: 100.00%, op_acc: 71.88%] [G loss: 2.77811]

830 [D loss: 0.160684, acc: 100.00%, op_acc: 71.88%] [G loss: 2.49640]

Epoch: 830, F1: 0.00594, F1P: 83

831 [D loss: 0.150408, acc: 100.00%, op_acc: 78.12%] [G loss: 2.52008]

832 [D loss: 0.189715, acc: 100.00%, op_acc: 65.62%] [G loss: 2.62644]

833 [D loss: 0.165654, acc: 100.00%, op_acc: 81.25%] [G loss: 2.58092]

834 [D loss: 0.177502, acc: 96.88%, op_acc: 75.00%] [G loss: 2.322166]

835 [D loss: 0.165379, acc: 100.00%, op_acc: 71.88%] [G loss: 2.42297]

836 [D loss: 0.158810, acc: 100.00%, op_acc: 75.00%] [G loss: 2.27342]

837 [D loss: 0.173272, acc: 96.88%, op_acc: 81.25%] [G loss: 2.551565]

838 [D loss: 0.155295, acc: 100.00%, op_acc: 78.12%] [G loss: 2.35333]
```

```
839 [D loss: 0.159714, acc: 100.00%, op_acc: 65.62%] [G loss: 2.65861]

840 [D loss: 0.160938, acc: 100.00%, op_acc: 62.50%] [G loss: 2.71340]

Epoch: 840, F1: 0.00595, F1P: 84

841 [D loss: 0.165816, acc: 96.88%, op_acc: 65.62%] [G loss: 2.173801]

842 [D loss: 0.152007, acc: 100.00%, op_acc: 71.88%] [G loss: 2.57540]

843 [D loss: 0.167245, acc: 100.00%, op_acc: 65.62%] [G loss: 2.41363]

844 [D loss: 0.187618, acc: 96.88%, op_acc: 75.00%] [G loss: 2.677176]

845 [D loss: 0.148417, acc: 100.00%, op_acc: 71.88%] [G loss: 2.88118]

846 [D loss: 0.172741, acc: 100.00%, op_acc: 71.88%] [G loss: 2.54928]

847 [D loss: 0.168215, acc: 96.88%, op_acc: 78.12%] [G loss: 2.478651]

848 [D loss: 0.161557, acc: 100.00%, op_acc: 81.25%] [G loss: 2.49077]

849 [D loss: 0.170915, acc: 100.00%, op_acc: 84.38%] [G loss: 2.63781]

850 [D loss: 0.143166, acc: 100.00%, op_acc: 71.88%] [G loss: 2.68570]

Epoch: 850, F1: 0.00583, F1P: 85

851 [D loss: 0.186713, acc: 96.88%, op_acc: 87.50%] [G loss: 2.744513]

852 [D loss: 0.155414, acc: 100.00%, op_acc: 62.50%] [G loss: 2.75683]

853 [D loss: 0.164392, acc: 100.00%, op_acc: 71.88%] [G loss: 2.70182]

854 [D loss: 0.163981, acc: 100.00%, op_acc: 71.88%] [G loss: 2.67865]

855 [D loss: 0.149853, acc: 100.00%, op_acc: 75.00%] [G loss: 2.48551]

856 [D loss: 0.186933, acc: 93.75%, op_acc: 71.88%] [G loss: 2.120328]

857 [D loss: 0.148601, acc: 100.00%, op_acc: 78.12%] [G loss: 2.67415]

858 [D loss: 0.178196, acc: 96.88%, op_acc: 75.00%] [G loss: 2.636909]

859 [D loss: 0.158518, acc: 100.00%, op_acc: 71.88%] [G loss: 2.54782]
```

```
860 [D loss: 0.144341, acc: 100.00%, op_acc: 78.12%] [G loss: 2.71086]

Epoch: 860, F1: 0.00589, F1P: 86

861 [D loss: 0.170711, acc: 100.00%, op_acc: 71.88%] [G loss: 2.49991]

862 [D loss: 0.172952, acc: 96.88%, op_acc: 68.75%] [G loss: 2.493163]

863 [D loss: 0.150169, acc: 100.00%, op_acc: 90.62%] [G loss: 2.51072]

864 [D loss: 0.158400, acc: 100.00%, op_acc: 71.88%] [G loss: 2.41688]

865 [D loss: 0.140554, acc: 100.00%, op_acc: 65.62%] [G loss: 2.53605]

866 [D loss: 0.209342, acc: 96.88%, op_acc: 68.75%] [G loss: 2.638603]

867 [D loss: 0.162454, acc: 96.88%, op_acc: 78.12%] [G loss: 2.854195]

868 [D loss: 0.169507, acc: 96.88%, op_acc: 78.12%] [G loss: 2.753211]

869 [D loss: 0.154792, acc: 100.00%, op_acc: 71.88%] [G loss: 2.95766]

870 [D loss: 0.154837, acc: 100.00%, op_acc: 68.75%] [G loss: 3.12860]

Epoch: 870, F1: 0.00596, F1P: 87

871 [D loss: 0.161246, acc: 100.00%, op_acc: 65.62%] [G loss: 2.48524]

872 [D loss: 0.154590, acc: 96.88%, op_acc: 75.00%] [G loss: 2.637994]

873 [D loss: 0.155594, acc: 96.88%, op_acc: 81.25%] [G loss: 2.877153]

874 [D loss: 0.140691, acc: 100.00%, op_acc: 71.88%] [G loss: 2.89378]

875 [D loss: 0.172924, acc: 96.88%, op_acc: 68.75%] [G loss: 2.762603]

876 [D loss: 0.147854, acc: 100.00%, op_acc: 62.50%] [G loss: 2.76543]

877 [D loss: 0.163415, acc: 100.00%, op_acc: 78.12%] [G loss: 2.68437]

878 [D loss: 0.155212, acc: 100.00%, op_acc: 78.12%] [G loss: 2.87360]

879 [D loss: 0.150384, acc: 100.00%, op_acc: 78.12%] [G loss: 2.49989]

880 [D loss: 0.154619, acc: 100.00%, op_acc: 75.00%] [G loss: 2.68889]
```

```
Epoch: 880, F1: 0.00558, F1P: 88

881 [D loss: 0.152607, acc: 100.00%, op_acc: 75.00%] [G loss: 2.75407]

882 [D loss: 0.151568, acc: 100.00%, op_acc: 75.00%] [G loss: 2.67220]

883 [D loss: 0.163659, acc: 100.00%, op_acc: 71.88%] [G loss: 2.77853]

884 [D loss: 0.200484, acc: 96.88%, op_acc: 68.75%] [G loss: 2.651265]

885 [D loss: 0.176133, acc: 96.88%, op_acc: 68.75%] [G loss: 2.632117]

886 [D loss: 0.154250, acc: 100.00%, op_acc: 75.00%] [G loss: 2.30345]

887 [D loss: 0.159515, acc: 100.00%, op_acc: 65.62%] [G loss: 2.68522]

888 [D loss: 0.186961, acc: 93.75%, op_acc: 71.88%] [G loss: 2.909969]

889 [D loss: 0.172699, acc: 93.75%, op_acc: 78.12%] [G loss: 2.648786]

890 [D loss: 0.145666, acc: 100.00%, op_acc: 75.00%] [G loss: 2.48069]

Epoch: 890, F1: 0.00595, F1P: 89

891 [D loss: 0.159853, acc: 100.00%, op_acc: 75.00%] [G loss: 2.30796]

892 [D loss: 0.167719, acc: 100.00%, op_acc: 65.62%] [G loss: 2.46811]

893 [D loss: 0.156654, acc: 100.00%, op_acc: 87.50%] [G loss: 2.84453]

894 [D loss: 0.146007, acc: 100.00%, op_acc: 75.00%] [G loss: 2.58545]

895 [D loss: 0.147059, acc: 100.00%, op_acc: 81.25%] [G loss: 2.69058]

896 [D loss: 0.154660, acc: 100.00%, op_acc: 84.38%] [G loss: 2.61505]

897 [D loss: 0.145762, acc: 100.00%, op_acc: 78.12%] [G loss: 2.78182]

898 [D loss: 0.174277, acc: 100.00%, op_acc: 78.12%] [G loss: 2.80462]

899 [D loss: 0.183227, acc: 96.88%, op_acc: 62.50%] [G loss: 2.802346]

900 [D loss: 0.149632, acc: 100.00%, op_acc: 81.25%] [G loss: 2.60471]

Epoch: 900, F1: 0.00598, F1P: 90
```

```
901 [D loss: 0.147803, acc: 100.00%, op_acc: 75.00%] [G loss: 2.75841]

902 [D loss: 0.145935, acc: 100.00%, op_acc: 62.50%] [G loss: 2.69421]

903 [D loss: 0.164265, acc: 96.88%, op_acc: 81.25%] [G loss: 2.632612]

904 [D loss: 0.141790, acc: 100.00%, op_acc: 81.25%] [G loss: 2.92877]

905 [D loss: 0.156982, acc: 100.00%, op_acc: 65.62%] [G loss: 2.79544]

906 [D loss: 0.167307, acc: 96.88%, op_acc: 84.38%] [G loss: 2.794965]

907 [D loss: 0.156154, acc: 96.88%, op_acc: 71.88%] [G loss: 2.784407]

908 [D loss: 0.150022, acc: 96.88%, op_acc: 71.88%] [G loss: 2.652630]

909 [D loss: 0.161181, acc: 96.88%, op_acc: 68.75%] [G loss: 2.907355]

910 [D loss: 0.142477, acc: 100.00%, op_acc: 71.88%] [G loss: 2.78753]

Epoch: 910, F1: 0.00592, F1P: 91

911 [D loss: 0.143291, acc: 100.00%, op_acc: 65.62%] [G loss: 2.81365]

912 [D loss: 0.155067, acc: 96.88%, op_acc: 59.38%] [G loss: 2.645769]

913 [D loss: 0.147565, acc: 100.00%, op_acc: 75.00%] [G loss: 2.44568]

914 [D loss: 0.152022, acc: 100.00%, op_acc: 68.75%] [G loss: 2.78658]

915 [D loss: 0.157501, acc: 100.00%, op_acc: 65.62%] [G loss: 2.90628]

916 [D loss: 0.153249, acc: 100.00%, op_acc: 71.88%] [G loss: 2.84531]

917 [D loss: 0.156107, acc: 100.00%, op_acc: 81.25%] [G loss: 2.88183]

918 [D loss: 0.143175, acc: 100.00%, op_acc: 87.50%] [G loss: 2.95837]

919 [D loss: 0.159521, acc: 96.88%, op_acc: 75.00%] [G loss: 2.474523]

920 [D loss: 0.139998, acc: 100.00%, op_acc: 78.12%] [G loss: 2.79561]

Epoch: 920, F1: 0.00595, F1P: 92

921 [D loss: 0.160731, acc: 96.88%, op_acc: 71.88%] [G loss: 2.711355]
```

```
922 [D loss: 0.165453, acc: 100.00%, op_acc: 75.00%] [G loss: 2.58031]

923 [D loss: 0.140312, acc: 100.00%, op_acc: 81.25%] [G loss: 2.61828]

924 [D loss: 0.179986, acc: 96.88%, op_acc: 68.75%] [G loss: 2.968562]

925 [D loss: 0.140407, acc: 100.00%, op_acc: 75.00%] [G loss: 2.46341]

926 [D loss: 0.146840, acc: 100.00%, op_acc: 84.38%] [G loss: 3.03179]

927 [D loss: 0.146025, acc: 100.00%, op_acc: 71.88%] [G loss: 2.50477]

928 [D loss: 0.163017, acc: 96.88%, op_acc: 78.12%] [G loss: 2.584504]

929 [D loss: 0.154404, acc: 100.00%, op_acc: 81.25%] [G loss: 2.77214]

930 [D loss: 0.146750, acc: 100.00%, op_acc: 81.25%] [G loss: 2.67308]

Epoch: 930, F1: 0.00592, F1P: 93

931 [D loss: 0.154164, acc: 100.00%, op_acc: 71.88%] [G loss: 2.49499]

932 [D loss: 0.143201, acc: 100.00%, op_acc: 75.00%] [G loss: 2.34748]

933 [D loss: 0.150782, acc: 100.00%, op_acc: 84.38%] [G loss: 3.29007]

934 [D loss: 0.131373, acc: 100.00%, op_acc: 84.38%] [G loss: 2.44759]

935 [D loss: 0.151295, acc: 96.88%, op_acc: 68.75%] [G loss: 2.740802]

936 [D loss: 0.173131, acc: 96.88%, op_acc: 81.25%] [G loss: 2.659576]

937 [D loss: 0.159093, acc: 100.00%, op_acc: 62.50%] [G loss: 2.69768]

938 [D loss: 0.142852, acc: 100.00%, op_acc: 71.88%] [G loss: 2.81949]

939 [D loss: 0.149605, acc: 96.88%, op_acc: 71.88%] [G loss: 2.595464]

940 [D loss: 0.144186, acc: 100.00%, op_acc: 81.25%] [G loss: 2.51602]

Epoch: 940, F1: 0.00596, F1P: 94

941 [D loss: 0.153057, acc: 100.00%, op_acc: 75.00%] [G loss: 2.63374]

942 [D loss: 0.152720, acc: 100.00%, op_acc: 71.88%] [G loss: 2.79579]
```

```
943 [D loss: 0.134607, acc: 100.00%, op_acc: 78.12%] [G loss: 2.61332]
944 [D loss: 0.160439, acc: 100.00%, op_acc: 71.88%] [G loss: 3.04911]
945 [D loss: 0.162264, acc: 100.00%, op_acc: 78.12%] [G loss: 2.49317]
946 [D loss: 0.173367, acc: 90.62%, op_acc: 75.00%] [G loss: 2.554070]
947 [D loss: 0.134614, acc: 100.00%, op_acc: 71.88%] [G loss: 2.77269]
948 [D loss: 0.142643, acc: 100.00%, op_acc: 65.62%] [G loss: 2.56888]
949 [D loss: 0.171050, acc: 96.88%, op_acc: 71.88%] [G loss: 2.763456]
950 [D loss: 0.151260, acc: 100.00%, op_acc: 65.62%] [G loss: 2.51832]
Epoch: 950, F1: 0.00597, F1P: 95
951 [D loss: 0.160532, acc: 96.88%, op_acc: 68.75%] [G loss: 2.637464]
952 [D loss: 0.143681, acc: 100.00%, op_acc: 71.88%] [G loss: 2.72086]
953 [D loss: 0.150102, acc: 100.00%, op_acc: 62.50%] [G loss: 2.61648]
954 [D loss: 0.170105, acc: 100.00%, op_acc: 62.50%] [G loss: 2.90207]
955 [D loss: 0.139616, acc: 100.00%, op_acc: 68.75%] [G loss: 2.83722]
956 [D loss: 0.163117, acc: 96.88%, op_acc: 65.62%] [G loss: 2.601409]
957 [D loss: 0.160549, acc: 96.88%, op_acc: 84.38%] [G loss: 2.880028]
958 [D loss: 0.162678, acc: 96.88%, op_acc: 71.88%] [G loss: 2.888992]
959 [D loss: 0.141467, acc: 100.00%, op_acc: 68.75%] [G loss: 2.61506]
960 [D loss: 0.173001, acc: 100.00%, op_acc: 75.00%] [G loss: 2.79942]
Epoch: 960, F1: 0.00597, F1P: 96
961 [D loss: 0.146643, acc: 96.88%, op_acc: 75.00%] [G loss: 2.833735]
962 [D loss: 0.170236, acc: 96.88%, op_acc: 84.38%] [G loss: 2.951139]
963 [D loss: 0.143358, acc: 100.00%, op_acc: 71.88%] [G loss: 3.20379]
```

```
964 [D loss: 0.155986, acc: 96.88%, op_acc: 62.50%] [G loss: 2.854616]

965 [D loss: 0.133463, acc: 100.00%, op_acc: 78.12%] [G loss: 2.33877]

966 [D loss: 0.138126, acc: 100.00%, op_acc: 68.75%] [G loss: 2.85885]

967 [D loss: 0.146839, acc: 96.88%, op_acc: 71.88%] [G loss: 2.921642]

968 [D loss: 0.137725, acc: 100.00%, op_acc: 56.25%] [G loss: 2.95927]

969 [D loss: 0.142748, acc: 96.88%, op_acc: 78.12%] [G loss: 3.142618]

970 [D loss: 0.131771, acc: 96.88%, op_acc: 81.25%] [G loss: 2.898473]

Epoch: 970, F1: 0.00598, F1P: 97

971 [D loss: 0.147863, acc: 100.00%, op_acc: 65.62%] [G loss: 3.04434]

972 [D loss: 0.126447, acc: 100.00%, op_acc: 78.12%] [G loss: 2.82548]

973 [D loss: 0.155419, acc: 96.88%, op_acc: 65.62%] [G loss: 2.695817]

974 [D loss: 0.153911, acc: 96.88%, op_acc: 59.38%] [G loss: 3.154489]

975 [D loss: 0.129128, acc: 100.00%, op_acc: 75.00%] [G loss: 3.08373]

976 [D loss: 0.135343, acc: 100.00%, op_acc: 65.62%] [G loss: 2.88627]

977 [D loss: 0.138945, acc: 100.00%, op_acc: 78.12%] [G loss: 2.73104]

978 [D loss: 0.135336, acc: 96.88%, op_acc: 78.12%] [G loss: 2.786193]

979 [D loss: 0.129094, acc: 100.00%, op_acc: 75.00%] [G loss: 2.73685]

980 [D loss: 0.158055, acc: 100.00%, op_acc: 56.25%] [G loss: 2.88983]

Epoch: 980, F1: 0.00597, F1P: 98

981 [D loss: 0.129325, acc: 100.00%, op_acc: 78.12%] [G loss: 2.65612]

982 [D loss: 0.154018, acc: 96.88%, op_acc: 78.12%] [G loss: 2.391577]

983 [D loss: 0.126372, acc: 100.00%, op_acc: 84.38%] [G loss: 2.46056]

984 [D loss: 0.153239, acc: 100.00%, op_acc: 68.75%] [G loss: 2.97581]
```

```
985 [D loss: 0.165457, acc: 96.88%, op_acc: 78.12%] [G loss: 2.790795]

986 [D loss: 0.146278, acc: 100.00%, op_acc: 65.62%] [G loss: 2.86041]

987 [D loss: 0.144510, acc: 100.00%, op_acc: 81.25%] [G loss: 2.81562]

988 [D loss: 0.139887, acc: 100.00%, op_acc: 65.62%] [G loss: 2.87630]

989 [D loss: 0.177649, acc: 90.62%, op_acc: 81.25%] [G loss: 2.585352]

990 [D loss: 0.136116, acc: 100.00%, op_acc: 75.00%] [G loss: 2.83559]

Epoch: 990, F1: 0.00595, F1P: 99

991 [D loss: 0.132939, acc: 100.00%, op_acc: 71.88%] [G loss: 2.53678]

992 [D loss: 0.151509, acc: 96.88%, op_acc: 71.88%] [G loss: 2.949676]

993 [D loss: 0.149904, acc: 96.88%, op_acc: 87.50%] [G loss: 2.994181]

994 [D loss: 0.138936, acc: 100.00%, op_acc: 84.38%] [G loss: 2.94328]

995 [D loss: 0.126061, acc: 100.00%, op_acc: 75.00%] [G loss: 3.09937]

996 [D loss: 0.132257, acc: 100.00%, op_acc: 75.00%] [G loss: 2.83499]

997 [D loss: 0.140715, acc: 100.00%, op_acc: 71.88%] [G loss: 2.93856]

998 [D loss: 0.160839, acc: 100.00%, op_acc: 75.00%] [G loss: 3.11455]

999 [D loss: 0.152124, acc: 100.00%, op_acc: 71.88%] [G loss: 2.95114]

1000 [D loss: 0.129632, acc: 100.00%, op_acc: 78.12%] [G loss: 3.0068]
```

Appendix C

Results of applying GAN on the Second dataset

```
Epoch: 0, F1: 0.00000, F1P: 0

1 [D loss: 0.467142, acc: 42.97%, op_acc: 34.38%] [G loss: 1.345880]

2 [D loss: 0.440062, acc: 41.41%, op_acc: 28.91%] [G loss: 1.373883]

3 [D loss: 0.411552, acc: 53.91%, op_acc: 25.00%] [G loss: 1.258146]

4 [D loss: 0.415113, acc: 49.22%, op_acc: 25.78%] [G loss: 1.313138]

5 [D loss: 0.437686, acc: 41.41%, op_acc: 37.50%] [G loss: 1.282372]

6 [D loss: 0.418401, acc: 52.34%, op_acc: 26.56%] [G loss: 1.279184]

7 [D loss: 0.406617, acc: 49.22%, op_acc: 34.38%] [G loss: 1.296972]

8 [D loss: 0.424078, acc: 46.88%, op_acc: 26.56%] [G loss: 1.288427]
```

```
9 [D loss: 0.422563, acc: 41.41%, op_acc: 34.38%] [G loss: 1.252043]

10 [D loss: 0.410838, acc: 46.88%, op_acc: 28.12%] [G loss: 1.25139]

Epoch: 10, F1: 0.00000, F1P: 1

11 [D loss: 0.389559, acc: 52.34%, op_acc: 25.00%] [G loss: 1.352537]

12 [D loss: 0.429091, acc: 45.31%, op_acc: 29.69%] [G loss: 1.219946]

13 [D loss: 0.422556, acc: 50.78%, op_acc: 28.91%] [G loss: 1.304943]

14 [D loss: 0.416407, acc: 47.66%, op_acc: 27.34%] [G loss: 1.269610]

15 [D loss: 0.412331, acc: 49.22%, op_acc: 27.34%] [G loss: 1.271494]

16 [D loss: 0.376889, acc: 50.00%, op_acc: 32.03%] [G loss: 1.262227]

17 [D loss: 0.407812, acc: 48.44%, op_acc: 31.25%] [G loss: 1.270771]

18 [D loss: 0.409088, acc: 50.00%, op_acc: 25.78%] [G loss: 1.189944]

19 [D loss: 0.380616, acc: 52.34%, op_acc: 32.81%] [G loss: 1.246342]

20 [D loss: 0.393808, acc: 42.97%, op_acc: 39.06%] [G loss: 1.213401]

Epoch: 20, F1: 0.00000, F1P: 2

21 [D loss: 0.397326, acc: 45.31%, op_acc: 32.81%] [G loss: 1.221194]

22 [D loss: 0.410294, acc: 50.78%, op_acc: 27.34%] [G loss: 1.273480]

23 [D loss: 0.418595, acc: 48.44%, op_acc: 28.12%] [G loss: 1.196183]

24 [D loss: 0.372465, acc: 53.91%, op_acc: 32.81%] [G loss: 1.264402]

25 [D loss: 0.388524, acc: 50.00%, op_acc: 31.25%] [G loss: 1.166969]

26 [D loss: 0.406078, acc: 50.00%, op_acc: 27.34%] [G loss: 1.232355]

27 [D loss: 0.391448, acc: 53.12%, op_acc: 35.16%] [G loss: 1.223101]

28 [D loss: 0.395794, acc: 52.34%, op_acc: 35.16%] [G loss: 1.245695]

29 [D loss: 0.380354, acc: 47.66%, op_acc: 35.94%] [G loss: 1.198040]
```

```
30 [D loss: 0.387251, acc: 53.12%, op_acc: 32.81%] [G loss: 1.169215]

Epoch: 30, F1: 0.00000, F1P: 3

31 [D loss: 0.368361, acc: 53.91%, op_acc: 29.69%] [G loss: 1.188592]

32 [D loss: 0.405142, acc: 51.56%, op_acc: 32.81%] [G loss: 1.141425]

33 [D loss: 0.392466, acc: 48.44%, op_acc: 36.72%] [G loss: 1.205584]

34 [D loss: 0.382772, acc: 50.00%, op_acc: 35.94%] [G loss: 1.125668]

35 [D loss: 0.360007, acc: 57.81%, op_acc: 39.06%] [G loss: 1.220386]

36 [D loss: 0.407571, acc: 45.31%, op_acc: 32.81%] [G loss: 1.216131]

37 [D loss: 0.379738, acc: 52.34%, op_acc: 32.03%] [G loss: 1.219781]

38 [D loss: 0.385179, acc: 48.44%, op_acc: 27.34%] [G loss: 1.161179]

39 [D loss: 0.381944, acc: 53.12%, op_acc: 35.16%] [G loss: 1.163719]

40 [D loss: 0.389445, acc: 47.66%, op_acc: 27.34%] [G loss: 1.187921]

Epoch: 40, F1: 0.00000, F1P: 4

41 [D loss: 0.366254, acc: 53.91%, op_acc: 32.03%] [G loss: 1.139721]

42 [D loss: 0.344178, acc: 53.12%, op_acc: 35.16%] [G loss: 1.240689]

43 [D loss: 0.364565, acc: 55.47%, op_acc: 37.50%] [G loss: 1.196322]

44 [D loss: 0.384085, acc: 51.56%, op_acc: 32.81%] [G loss: 1.152830]

45 [D loss: 0.365104, acc: 51.56%, op_acc: 36.72%] [G loss: 1.069800]

46 [D loss: 0.372538, acc: 56.25%, op_acc: 29.69%] [G loss: 1.166750]

47 [D loss: 0.360072, acc: 51.56%, op_acc: 36.72%] [G loss: 1.170370]

48 [D loss: 0.386586, acc: 50.78%, op_acc: 32.03%] [G loss: 1.205123]

49 [D loss: 0.371164, acc: 48.44%, op_acc: 40.62%] [G loss: 1.216956]

50 [D loss: 0.363316, acc: 56.25%, op_acc: 33.59%] [G loss: 1.123760]
```

```
Epoch: 50, F1: 0.00000, F1P: 5

51 [D loss: 0.378968, acc: 49.22%, op_acc: 29.69%] [G loss: 1.171417]

52 [D loss: 0.344709, acc: 58.59%, op_acc: 32.03%] [G loss: 1.169485]

53 [D loss: 0.368097, acc: 53.12%, op_acc: 31.25%] [G loss: 1.158283]

54 [D loss: 0.362526, acc: 54.69%, op_acc: 37.50%] [G loss: 1.182296]

55 [D loss: 0.369491, acc: 52.34%, op_acc: 34.38%] [G loss: 1.123484]

56 [D loss: 0.381465, acc: 50.78%, op_acc: 30.47%] [G loss: 1.172594]

57 [D loss: 0.360147, acc: 57.03%, op_acc: 39.84%] [G loss: 1.100466]

58 [D loss: 0.359869, acc: 56.25%, op_acc: 30.47%] [G loss: 1.135433]

59 [D loss: 0.366221, acc: 53.12%, op_acc: 28.91%] [G loss: 1.151569]

60 [D loss: 0.372707, acc: 52.34%, op_acc: 33.59%] [G loss: 1.214709]

Epoch: 60, F1: 0.00000, F1P: 6

61 [D loss: 0.368312, acc: 55.47%, op_acc: 32.03%] [G loss: 1.183355]

62 [D loss: 0.344258, acc: 53.91%, op_acc: 35.94%] [G loss: 1.160569]

63 [D loss: 0.377552, acc: 50.00%, op_acc: 35.94%] [G loss: 1.181042]

64 [D loss: 0.361852, acc: 53.12%, op_acc: 36.72%] [G loss: 1.124358]

65 [D loss: 0.362056, acc: 53.12%, op_acc: 34.38%] [G loss: 1.132425]

66 [D loss: 0.362534, acc: 51.56%, op_acc: 43.75%] [G loss: 1.147422]

67 [D loss: 0.370542, acc: 53.91%, op_acc: 25.78%] [G loss: 1.122947]

68 [D loss: 0.380926, acc: 49.22%, op_acc: 36.72%] [G loss: 1.117381]

69 [D loss: 0.369557, acc: 55.47%, op_acc: 35.94%] [G loss: 1.077481]

70 [D loss: 0.368012, acc: 53.91%, op_acc: 24.22%] [G loss: 1.066180]

Epoch: 70, F1: 0.00000, F1P: 7
```

```
71 [D loss: 0.379588, acc: 49.22%, op_acc: 35.16%] [G loss: 1.119493]

72 [D loss: 0.356292, acc: 56.25%, op_acc: 36.72%] [G loss: 1.084212]

73 [D loss: 0.357193, acc: 58.59%, op_acc: 34.38%] [G loss: 1.164393]

74 [D loss: 0.347406, acc: 56.25%, op_acc: 35.94%] [G loss: 1.159653]

75 [D loss: 0.369353, acc: 45.31%, op_acc: 35.94%] [G loss: 1.111808]

76 [D loss: 0.381800, acc: 50.78%, op_acc: 37.50%] [G loss: 1.124426]

77 [D loss: 0.359847, acc: 51.56%, op_acc: 37.50%] [G loss: 1.150707]

78 [D loss: 0.355877, acc: 53.91%, op_acc: 28.91%] [G loss: 1.069515]

79 [D loss: 0.367562, acc: 50.00%, op_acc: 34.38%] [G loss: 1.107454]

80 [D loss: 0.334205, acc: 59.38%, op_acc: 30.47%] [G loss: 1.150923]

Epoch: 80, F1: 0.00000, F1P: 8

81 [D loss: 0.348460, acc: 59.38%, op_acc: 32.81%] [G loss: 1.123273]

82 [D loss: 0.366012, acc: 54.69%, op_acc: 37.50%] [G loss: 1.174631]

83 [D loss: 0.351377, acc: 57.81%, op_acc: 37.50%] [G loss: 1.087580]

84 [D loss: 0.359605, acc: 54.69%, op_acc: 44.53%] [G loss: 1.147302]

85 [D loss: 0.365991, acc: 53.91%, op_acc: 35.16%] [G loss: 1.097752]

86 [D loss: 0.365358, acc: 54.69%, op_acc: 29.69%] [G loss: 1.129049]

87 [D loss: 0.345781, acc: 60.94%, op_acc: 43.75%] [G loss: 1.079568]

88 [D loss: 0.349165, acc: 60.16%, op_acc: 38.28%] [G loss: 1.104143]

89 [D loss: 0.355962, acc: 56.25%, op_acc: 35.16%] [G loss: 1.013063]

90 [D loss: 0.350434, acc: 62.50%, op_acc: 32.03%] [G loss: 1.084422]

Epoch: 90, F1: 0.03846, F1P: 9

91 [D loss: 0.350731, acc: 53.91%, op_acc: 42.19%] [G loss: 1.158438]
```

```
92 [D loss: 0.356539, acc: 51.56%, op_acc: 38.28%] [G loss: 1.134113]

93 [D loss: 0.328994, acc: 66.41%, op_acc: 35.16%] [G loss: 1.077497]

94 [D loss: 0.340964, acc: 55.47%, op_acc: 39.06%] [G loss: 1.176894]

95 [D loss: 0.320275, acc: 64.84%, op_acc: 35.16%] [G loss: 1.116780]

96 [D loss: 0.341723, acc: 59.38%, op_acc: 35.94%] [G loss: 1.130624]

97 [D loss: 0.333231, acc: 62.50%, op_acc: 39.06%] [G loss: 1.130290]

98 [D loss: 0.342751, acc: 61.72%, op_acc: 41.41%] [G loss: 1.154406]

99 [D loss: 0.334769, acc: 64.06%, op_acc: 36.72%] [G loss: 1.143267]

100 [D loss: 0.341712, acc: 63.28%, op_acc: 34.38%] [G loss: 1.088861]


Epoch: 700, F1: 0.61905, F1P: 70

701 [D loss: 0.273970, acc: 76.56%, op_acc: 48.44%] [G loss: 1.208564]

702 [D loss: 0.257934, acc: 82.81%, op_acc: 60.94%] [G loss: 1.228546]

703 [D loss: 0.259901, acc: 81.25%, op_acc: 53.91%] [G loss: 1.168388]

704 [D loss: 0.245468, acc: 86.72%, op_acc: 55.47%] [G loss: 1.178979]

705 [D loss: 0.267725, acc: 84.38%, op_acc: 53.91%] [G loss: 1.213567]

706 [D loss: 0.246280, acc: 89.84%, op_acc: 60.16%] [G loss: 1.162920]

707 [D loss: 0.264048, acc: 77.34%, op_acc: 49.22%] [G loss: 1.150116]

708 [D loss: 0.253943, acc: 83.59%, op_acc: 53.12%] [G loss: 1.270462]

709 [D loss: 0.258614, acc: 82.03%, op_acc: 57.81%] [G loss: 1.142994]

710 [D loss: 0.264892, acc: 85.94%, op_acc: 53.91%] [G loss: 1.230421]

Epoch: 710, F1: 0.56604, F1P: 71

711 [D loss: 0.245545, acc: 88.28%, op_acc: 57.81%] [G loss: 1.170244]
```

```
712 [D loss: 0.266359, acc: 78.91%, op_acc: 59.38%] [G loss: 1.250539]

713 [D loss: 0.264373, acc: 82.81%, op_acc: 46.88%] [G loss: 1.189223]

714 [D loss: 0.252420, acc: 85.16%, op_acc: 62.50%] [G loss: 1.166431]

715 [D loss: 0.262587, acc: 82.81%, op_acc: 56.25%] [G loss: 1.297352]

716 [D loss: 0.258857, acc: 83.59%, op_acc: 53.91%] [G loss: 1.168458]

717 [D loss: 0.257718, acc: 85.16%, op_acc: 53.12%] [G loss: 1.261794]

718 [D loss: 0.260605, acc: 81.25%, op_acc: 50.78%] [G loss: 1.227591]

719 [D loss: 0.243377, acc: 89.06%, op_acc: 59.38%] [G loss: 1.149416]

720 [D loss: 0.266184, acc: 81.25%, op_acc: 57.03%] [G loss: 1.220826]

Epoch: 720, F1: 0.27092, F1P: 72

721 [D loss: 0.247182, acc: 82.81%, op_acc: 48.44%] [G loss: 1.181571]

722 [D loss: 0.271552, acc: 77.34%, op_acc: 47.66%] [G loss: 1.195835]

723 [D loss: 0.261689, acc: 85.16%, op_acc: 53.91%] [G loss: 1.142213]

724 [D loss: 0.260063, acc: 84.38%, op_acc: 57.81%] [G loss: 1.213863]

725 [D loss: 0.270435, acc: 77.34%, op_acc: 52.34%] [G loss: 1.182143]

726 [D loss: 0.255273, acc: 81.25%, op_acc: 54.69%] [G loss: 1.222789]

727 [D loss: 0.263915, acc: 85.16%, op_acc: 53.91%] [G loss: 1.202248]

728 [D loss: 0.262080, acc: 83.59%, op_acc: 57.03%] [G loss: 1.191362]

729 [D loss: 0.258108, acc: 78.91%, op_acc: 57.03%] [G loss: 1.168001]

730 [D loss: 0.265966, acc: 81.25%, op_acc: 53.91%] [G loss: 1.168097]

Epoch: 730, F1: 0.15966, F1P: 73

731 [D loss: 0.257626, acc: 78.91%, op_acc: 56.25%] [G loss: 1.255620]

732 [D loss: 0.262793, acc: 81.25%, op_acc: 56.25%] [G loss: 1.262873]
```

```
733 [D loss: 0.271085, acc: 75.00%, op_acc: 51.56%] [G loss: 1.163301]

734 [D loss: 0.251713, acc: 86.72%, op_acc: 55.47%] [G loss: 1.140602]

735 [D loss: 0.255405, acc: 82.03%, op_acc: 52.34%] [G loss: 1.266286]

736 [D loss: 0.253172, acc: 82.03%, op_acc: 59.38%] [G loss: 1.203853]

737 [D loss: 0.264024, acc: 83.59%, op_acc: 59.38%] [G loss: 1.247801]

738 [D loss: 0.250002, acc: 85.94%, op_acc: 55.47%] [G loss: 1.251143]

739 [D loss: 0.252500, acc: 84.38%, op_acc: 57.81%] [G loss: 1.246314]

740 [D loss: 0.259037, acc: 82.03%, op_acc: 51.56%] [G loss: 1.222259]

Epoch: 740, F1: 0.18182, F1P: 74

741 [D loss: 0.256353, acc: 82.03%, op_acc: 49.22%] [G loss: 1.176602]

742 [D loss: 0.258720, acc: 82.81%, op_acc: 56.25%] [G loss: 1.229297]

743 [D loss: 0.244920, acc: 85.94%, op_acc: 57.03%] [G loss: 1.161960]

744 [D loss: 0.266282, acc: 78.12%, op_acc: 55.47%] [G loss: 1.228846]

745 [D loss: 0.256011, acc: 79.69%, op_acc: 59.38%] [G loss: 1.282872]

746 [D loss: 0.257530, acc: 82.03%, op_acc: 58.59%] [G loss: 1.238561]

747 [D loss: 0.258488, acc: 85.16%, op_acc: 55.47%] [G loss: 1.243407]

748 [D loss: 0.252177, acc: 85.16%, op_acc: 56.25%] [G loss: 1.228418]

749 [D loss: 0.256107, acc: 85.16%, op_acc: 54.69%] [G loss: 1.189259]

750 [D loss: 0.262793, acc: 80.47%, op_acc: 57.81%] [G loss: 1.254765]

Epoch: 750, F1: 0.08207, F1P: 75

751 [D loss: 0.270590, acc: 78.91%, op_acc: 58.59%] [G loss: 1.219727]

752 [D loss: 0.247644, acc: 85.16%, op_acc: 57.03%] [G loss: 1.234891]

753 [D loss: 0.250360, acc: 88.28%, op_acc: 48.44%] [G loss: 1.207373]
```

```
754 [D loss: 0.250306, acc: 89.84%, op_acc: 55.47%] [G loss: 1.258316]

755 [D loss: 0.253050, acc: 83.59%, op_acc: 67.19%] [G loss: 1.250992]

756 [D loss: 0.245221, acc: 87.50%, op_acc: 56.25%] [G loss: 1.181222]

757 [D loss: 0.256685, acc: 83.59%, op_acc: 60.16%] [G loss: 1.289905]

758 [D loss: 0.257698, acc: 80.47%, op_acc: 54.69%] [G loss: 1.265771]

759 [D loss: 0.245113, acc: 89.06%, op_acc: 56.25%] [G loss: 1.279315]

760 [D loss: 0.255687, acc: 82.03%, op_acc: 53.91%] [G loss: 1.260885]

Epoch: 760, F1: 0.43114, F1P: 76

761 [D loss: 0.229033, acc: 89.06%, op_acc: 58.59%] [G loss: 1.239542]

762 [D loss: 0.248291, acc: 84.38%, op_acc: 44.53%] [G loss: 1.285811]

763 [D loss: 0.249345, acc: 86.72%, op_acc: 57.03%] [G loss: 1.212575]

764 [D loss: 0.243815, acc: 89.84%, op_acc: 57.81%] [G loss: 1.290197]

765 [D loss: 0.237019, acc: 90.62%, op_acc: 60.94%] [G loss: 1.296971]

766 [D loss: 0.249055, acc: 84.38%, op_acc: 55.47%] [G loss: 1.266173]

767 [D loss: 0.249146, acc: 89.84%, op_acc: 62.50%] [G loss: 1.229435]

768 [D loss: 0.232441, acc: 87.50%, op_acc: 50.78%] [G loss: 1.214603]

769 [D loss: 0.237739, acc: 88.28%, op_acc: 53.91%] [G loss: 1.265310]

770 [D loss: 0.255434, acc: 82.03%, op_acc: 60.16%] [G loss: 1.280635]

Epoch: 770, F1: 0.41341, F1P: 77

771 [D loss: 0.251315, acc: 84.38%, op_acc: 55.47%] [G loss: 1.277451]

772 [D loss: 0.242467, acc: 86.72%, op_acc: 54.69%] [G loss: 1.271943]

773 [D loss: 0.246062, acc: 85.94%, op_acc: 55.47%] [G loss: 1.219514]

774 [D loss: 0.253903, acc: 81.25%, op_acc: 57.03%] [G loss: 1.229628]
```

```
775 [D loss: 0.242396, acc: 90.62%, op_acc: 60.16%] [G loss: 1.294233]

776 [D loss: 0.242275, acc: 87.50%, op_acc: 60.94%] [G loss: 1.263274]

777 [D loss: 0.241296, acc: 83.59%, op_acc: 53.91%] [G loss: 1.208019]

778 [D loss: 0.231437, acc: 88.28%, op_acc: 57.03%] [G loss: 1.301692]

779 [D loss: 0.251911, acc: 89.84%, op_acc: 59.38%] [G loss: 1.224513]

780 [D loss: 0.235101, acc: 87.50%, op_acc: 60.16%] [G loss: 1.260003]


Epoch: 790, F1: 0.12112, F1P: 79

791 [D loss: 0.242751, acc: 84.38%, op_acc: 57.81%] [G loss: 1.210089]

792 [D loss: 0.251046, acc: 90.62%, op_acc: 60.94%] [G loss: 1.235329]

793 [D loss: 0.238390, acc: 89.06%, op_acc: 61.72%] [G loss: 1.280262]

794 [D loss: 0.248438, acc: 82.81%, op_acc: 59.38%] [G loss: 1.242981]

795 [D loss: 0.240698, acc: 85.94%, op_acc: 57.81%] [G loss: 1.241056]

796 [D loss: 0.238086, acc: 88.28%, op_acc: 54.69%] [G loss: 1.217072]

797 [D loss: 0.241968, acc: 84.38%, op_acc: 53.12%] [G loss: 1.302303]

798 [D loss: 0.254227, acc: 82.81%, op_acc: 57.81%] [G loss: 1.323084]

799 [D loss: 0.239929, acc: 86.72%, op_acc: 60.94%] [G loss: 1.298827]

800 [D loss: 0.231899, acc: 86.72%, op_acc: 58.59%] [G loss: 1.294172]

Epoch: 800, F1: 0.02167, F1P: 80

801 [D loss: 0.238764, acc: 86.72%, op_acc: 62.50%] [G loss: 1.328436]

802 [D loss: 0.234579, acc: 89.84%, op_acc: 59.38%] [G loss: 1.272727]

803 [D loss: 0.250241, acc: 82.81%, op_acc: 58.59%] [G loss: 1.311297]

804 [D loss: 0.240749, acc: 86.72%, op_acc: 60.94%] [G loss: 1.210595]
```

```
805 [D loss: 0.241084, acc: 85.94%, op_acc: 58.59%] [G loss: 1.287291]

806 [D loss: 0.231249, acc: 85.94%, op_acc: 64.84%] [G loss: 1.302652]

807 [D loss: 0.229963, acc: 86.72%, op_acc: 60.16%] [G loss: 1.357216]

808 [D loss: 0.231737, acc: 83.59%, op_acc: 60.16%] [G loss: 1.270025]

809 [D loss: 0.241775, acc: 87.50%, op_acc: 63.28%] [G loss: 1.308101]

810 [D loss: 0.239789, acc: 89.84%, op_acc: 59.38%] [G loss: 1.302357]

Epoch: 810, F1: 0.04245, F1P: 81

811 [D loss: 0.244558, acc: 88.28%, op_acc: 57.03%] [G loss: 1.288499]

812 [D loss: 0.241177, acc: 89.84%, op_acc: 60.94%] [G loss: 1.342044]

813 [D loss: 0.241059, acc: 85.94%, op_acc: 57.03%] [G loss: 1.259164]

814 [D loss: 0.243532, acc: 92.19%, op_acc: 62.50%] [G loss: 1.280162]

815 [D loss: 0.228369, acc: 89.06%, op_acc: 63.28%] [G loss: 1.324009]

816 [D loss: 0.230659, acc: 90.62%, op_acc: 57.81%] [G loss: 1.286424]

817 [D loss: 0.228233, acc: 88.28%, op_acc: 58.59%] [G loss: 1.348555]

818 [D loss: 0.231975, acc: 88.28%, op_acc: 57.81%] [G loss: 1.225693]

819 [D loss: 0.222730, acc: 92.97%, op_acc: 68.75%] [G loss: 1.309636]

820 [D loss: 0.228098, acc: 90.62%, op_acc: 66.41%] [G loss: 1.313805]

Epoch: 820, F1: 0.01297, F1P: 82

821 [D loss: 0.246373, acc: 87.50%, op_acc: 56.25%] [G loss: 1.309121]

822 [D loss: 0.231774, acc: 86.72%, op_acc: 59.38%] [G loss: 1.241728]

823 [D loss: 0.236071, acc: 84.38%, op_acc: 67.97%] [G loss: 1.267977]

824 [D loss: 0.246304, acc: 87.50%, op_acc: 64.84%] [G loss: 1.314460]

825 [D loss: 0.244048, acc: 84.38%, op_acc: 57.81%] [G loss: 1.336302]
```

```
826 [D loss: 0.218040, acc: 90.62%, op_acc: 59.38%] [G loss: 1.287349]

827 [D loss: 0.238107, acc: 89.84%, op_acc: 58.59%] [G loss: 1.303073]

828 [D loss: 0.238809, acc: 86.72%, op_acc: 63.28%] [G loss: 1.278054]

829 [D loss: 0.224400, acc: 92.19%, op_acc: 64.06%] [G loss: 1.303020]

830 [D loss: 0.231695, acc: 88.28%, op_acc: 64.84%] [G loss: 1.337551]

Epoch: 830, F1: 0.01221, F1P: 83

831 [D loss: 0.249243, acc: 83.59%, op_acc: 52.34%] [G loss: 1.370661]

832 [D loss: 0.240082, acc: 85.94%, op_acc: 61.72%] [G loss: 1.333508]

833 [D loss: 0.238277, acc: 85.94%, op_acc: 59.38%] [G loss: 1.310955]

834 [D loss: 0.225107, acc: 92.19%, op_acc: 62.50%] [G loss: 1.256901]

835 [D loss: 0.244436, acc: 82.81%, op_acc: 67.19%] [G loss: 1.258790]

836 [D loss: 0.237193, acc: 90.62%, op_acc: 57.03%] [G loss: 1.281763]

837 [D loss: 0.241413, acc: 83.59%, op_acc: 64.06%] [G loss: 1.283474]

838 [D loss: 0.224269, acc: 87.50%, op_acc: 64.06%] [G loss: 1.262559]

839 [D loss: 0.239450, acc: 83.59%, op_acc: 64.84%] [G loss: 1.290556]

840 [D loss: 0.235585, acc: 86.72%, op_acc: 57.03%] [G loss: 1.287569]

Epoch: 840, F1: 0.00527, F1P: 84

841 [D loss: 0.243248, acc: 86.72%, op_acc: 67.19%] [G loss: 1.324806]

842 [D loss: 0.241393, acc: 89.06%, op_acc: 67.97%] [G loss: 1.279142]

843 [D loss: 0.240518, acc: 86.72%, op_acc: 60.94%] [G loss: 1.330524]

844 [D loss: 0.230132, acc: 88.28%, op_acc: 67.97%] [G loss: 1.349382]

845 [D loss: 0.227199, acc: 88.28%, op_acc: 64.84%] [G loss: 1.291792]

846 [D loss: 0.239673, acc: 85.94%, op_acc: 57.03%] [G loss: 1.226609]
```

```
847 [D loss: 0.243394, acc: 85.16%, op_acc: 60.16%] [G loss: 1.282964]

848 [D loss: 0.235961, acc: 91.41%, op_acc: 68.75%] [G loss: 1.244249]

849 [D loss: 0.239907, acc: 82.81%, op_acc: 61.72%] [G loss: 1.289389]

850 [D loss: 0.239225, acc: 82.81%, op_acc: 62.50%] [G loss: 1.254067]

Epoch: 850, F1: 0.00996, F1P: 85

851 [D loss: 0.236157, acc: 86.72%, op_acc: 67.97%] [G loss: 1.277538]

852 [D loss: 0.237244, acc: 89.84%, op_acc: 64.06%] [G loss: 1.349630]

853 [D loss: 0.239112, acc: 85.16%, op_acc: 61.72%] [G loss: 1.348577]

854 [D loss: 0.230141, acc: 91.41%, op_acc: 66.41%] [G loss: 1.323152]

855 [D loss: 0.237422, acc: 85.16%, op_acc: 56.25%] [G loss: 1.286867]

856 [D loss: 0.246667, acc: 88.28%, op_acc: 64.06%] [G loss: 1.357644]

857 [D loss: 0.226627, acc: 90.62%, op_acc: 64.06%] [G loss: 1.361973]

858 [D loss: 0.233528, acc: 86.72%, op_acc: 65.62%] [G loss: 1.296245]

859 [D loss: 0.236819, acc: 85.16%, op_acc: 64.06%] [G loss: 1.272057]

860 [D loss: 0.225900, acc: 95.31%, op_acc: 66.41%] [G loss: 1.277183]

Epoch: 860, F1: 0.00478, F1P: 86

861 [D loss: 0.246469, acc: 85.94%, op_acc: 61.72%] [G loss: 1.245772]

862 [D loss: 0.246721, acc: 85.16%, op_acc: 64.84%] [G loss: 1.267796]

863 [D loss: 0.262220, acc: 78.91%, op_acc: 57.03%] [G loss: 1.345001]

864 [D loss: 0.265785, acc: 82.81%, op_acc: 67.19%] [G loss: 1.351193]

865 [D loss: 0.232305, acc: 85.16%, op_acc: 60.16%] [G loss: 1.297333]

866 [D loss: 0.242391, acc: 89.06%, op_acc: 71.88%] [G loss: 1.343187]

867 [D loss: 0.238324, acc: 89.06%, op_acc: 62.50%] [G loss: 1.266564]
```

```
868 [D loss: 0.227644, acc: 89.06%, op_acc: 67.97%] [G loss: 1.329407]

869 [D loss: 0.226312, acc: 90.62%, op_acc: 65.62%] [G loss: 1.333394]

870 [D loss: 0.231776, acc: 85.94%, op_acc: 69.53%] [G loss: 1.285670]


Epoch: 1000, F1: 0.13578, F1P: 100

1001 [D loss: 0.222308, acc: 87.50%, op_acc: 67.19%] [G loss: 1.39428]

1002 [D loss: 0.232537, acc: 86.72%, op_acc: 57.81%] [G loss: 1.41239]

1003 [D loss: 0.232044, acc: 89.06%, op_acc: 64.84%] [G loss: 1.33978]

1004 [D loss: 0.229038, acc: 88.28%, op_acc: 69.53%] [G loss: 1.32122]

1005 [D loss: 0.226440, acc: 86.72%, op_acc: 64.06%] [G loss: 1.38091]

1006 [D loss: 0.231031, acc: 87.50%, op_acc: 64.84%] [G loss: 1.42875]

1007 [D loss: 0.208862, acc: 88.28%, op_acc: 67.19%] [G loss: 1.37109]

1008 [D loss: 0.230759, acc: 89.06%, op_acc: 60.94%] [G loss: 1.38492]

1009 [D loss: 0.237521, acc: 87.50%, op_acc: 59.38%] [G loss: 1.30113]

1010 [D loss: 0.242663, acc: 82.03%, op_acc: 59.38%] [G loss: 1.38869]

Epoch: 1010, F1: 0.26829, F1P: 101

1011 [D loss: 0.221317, acc: 88.28%, op_acc: 64.06%] [G loss: 1.41679]

1012 [D loss: 0.219401, acc: 90.62%, op_acc: 66.41%] [G loss: 1.39695]

1013 [D loss: 0.218739, acc: 87.50%, op_acc: 70.31%] [G loss: 1.39246]

1014 [D loss: 0.228483, acc: 89.84%, op_acc: 67.97%] [G loss: 1.42998]

1015 [D loss: 0.230029, acc: 89.84%, op_acc: 65.62%] [G loss: 1.46794]

1016 [D loss: 0.218670, acc: 88.28%, op_acc: 68.75%] [G loss: 1.46159]

1017 [D loss: 0.229189, acc: 89.84%, op_acc: 61.72%] [G loss: 1.39781]
```

```
1018 [D loss: 0.235904, acc: 89.84%, op_acc: 64.84%] [G loss: 1.39941]

1019 [D loss: 0.214666, acc: 87.50%, op_acc: 69.53%] [G loss: 1.33156]

1020 [D loss: 0.218251, acc: 90.62%, op_acc: 70.31%] [G loss: 1.37547]


Epoch: 1050, F1: 0.30556, F1P: 105

1051 [D loss: 0.220955, acc: 89.84%, op_acc: 68.75%] [G loss: 1.43188]

1052 [D loss: 0.213529, acc: 91.41%, op_acc: 71.09%] [G loss: 1.37872]

1053 [D loss: 0.221695, acc: 85.94%, op_acc: 74.22%] [G loss: 1.49647]

1054 [D loss: 0.228996, acc: 86.72%, op_acc: 66.41%] [G loss: 1.47554]

1055 [D loss: 0.218279, acc: 89.06%, op_acc: 69.53%] [G loss: 1.36148]


1057 [D loss: 0.215970, acc: 92.19%, op_acc: 67.19%] [G loss: 1.40440]

1058 [D loss: 0.224381, acc: 87.50%, op_acc: 65.62%] [G loss: 1.41781]

1059 [D loss: 0.219830, acc: 87.50%, op_acc: 68.75%] [G loss: 1.44738]

1060 [D loss: 0.221929, acc: 89.84%, op_acc: 66.41%] [G loss: 1.34906]

Epoch: 1080, F1: 0.35955, F1P: 108

1081 [D loss: 0.210246, acc: 91.41%, op_acc: 63.28%] [G loss: 1.44361]

1082 [D loss: 0.216443, acc: 88.28%, op_acc: 71.09%] [G loss: 1.43302]

1083 [D loss: 0.218390, acc: 89.06%, op_acc: 64.06%] [G loss: 1.38028]

1084 [D loss: 0.244240, acc: 82.03%, op_acc: 60.16%] [G loss: 1.44365]

1085 [D loss: 0.198676, acc: 89.06%, op_acc: 67.97%] [G loss: 1.44369]

1086 [D loss: 0.206851, acc: 92.19%, op_acc: 73.44%] [G loss: 1.44820]

1087 [D loss: 0.216971, acc: 87.50%, op_acc: 64.84%] [G loss: 1.44764]
```

```
1088 [D loss: 0.211500, acc: 92.97%, op_acc: 67.19%] [G loss: 1.42801]

1089 [D loss: 0.209783, acc: 91.41%, op_acc: 67.97%] [G loss: 1.47678]

1090 [D loss: 0.205377, acc: 91.41%, op_acc: 68.75%] [G loss: 1.42175]

Epoch: 1090, F1: 0.12892, F1P: 109

1091 [D loss: 0.216449, acc: 86.72%, op_acc: 68.75%] [G loss: 1.44607]

1092 [D loss: 0.219067, acc: 86.72%, op_acc: 62.50%] [G loss: 1.51941]

1093 [D loss: 0.198425, acc: 95.31%, op_acc: 67.97%] [G loss: 1.51074]

1094 [D loss: 0.204913, acc: 91.41%, op_acc: 75.00%] [G loss: 1.52139]

1095 [D loss: 0.203649, acc: 93.75%, op_acc: 72.66%] [G loss: 1.36236]

1096 [D loss: 0.195822, acc: 92.19%, op_acc: 65.62%] [G loss: 1.43254]

1097 [D loss: 0.213050, acc: 92.97%, op_acc: 70.31%] [G loss: 1.45145]

1098 [D loss: 0.217413, acc: 89.06%, op_acc: 61.72%] [G loss: 1.40149]

1099 [D loss: 0.216992, acc: 92.97%, op_acc: 67.97%] [G loss: 1.41071]

1100 [D loss: 0.213001, acc: 92.19%, op_acc: 64.84%] [G loss: 1.46727]

Epoch: 1100, F1: 0.24832, F1P: 110

1101 [D loss: 0.212600, acc: 89.84%, op_acc: 61.72%] [G loss: 1.43319]

1102 [D loss: 0.216924, acc: 90.62%, op_acc: 60.94%] [G loss: 1.45820]

1103 [D loss: 0.216597, acc: 90.62%, op_acc: 67.97%] [G loss: 1.485122

1104 [D loss: 0.221240, acc: 92.19%, op_acc: 67.97%] [G loss: 1.52052]

1105 [D loss: 0.203523, acc: 92.97%, op_acc: 72.66%] [G loss: 1.39889]

1106 [D loss: 0.204489, acc: 94.53%, op_acc: 77.34%] [G loss: 1.50921]

1107 [D loss: 0.200537, acc: 92.97%, op_acc: 73.44%] [G loss: 1.45338]

1108 [D loss: 0.199619, acc: 91.41%, op_acc: 65.62%] [G loss: 1.45882]
```

```
1109 [D loss: 0.210808, acc: 89.84%, op_acc: 67.19%] [G loss: 1.49735]

1110 [D loss: 0.211795, acc: 94.53%, op_acc: 71.09%] [G loss: 1.54874]

Epoch: 1110, F1: 0.12693, F1P: 111

1111 [D loss: 0.210726, acc: 91.41%, op_acc: 71.09%] [G loss: 1.55177]

1112 [D loss: 0.202197, acc: 92.97%, op_acc: 67.97%] [G loss: 1.47095]

1113 [D loss: 0.196434, acc: 89.84%, op_acc: 69.53%] [G loss: 1.51123]

1114 [D loss: 0.201657, acc: 91.41%, op_acc: 68.75%] [G loss: 1.58914]

1115 [D loss: 0.210086, acc: 91.41%, op_acc: 64.84%] [G loss: 1.45863]

1116 [D loss: 0.193106, acc: 93.75%, op_acc: 70.31%] [G loss: 1.53273]

1117 [D loss: 0.203728, acc: 90.62%, op_acc: 67.97%] [G loss: 1.51563]

1118 [D loss: 0.208105, acc: 92.19%, op_acc: 70.31%] [G loss: 1.52351]

1119 [D loss: 0.213231, acc: 92.97%, op_acc: 64.84%] [G loss: 1.55756]

1120 [D loss: 0.206735, acc: 85.94%, op_acc: 65.62%] [G loss: 1.50972]

Epoch: 1120, F1: 0.23948, F1P: 112

1121 [D loss: 0.203712, acc: 94.53%, op_acc: 70.31%] [G loss: 1.59978]

1122 [D loss: 0.203456, acc: 88.28%, op_acc: 69.53%] [G loss: 1.46210]

1123 [D loss: 0.200533, acc: 93.75%, op_acc: 67.19%] [G loss: 1.46818]

1124 [D loss: 0.222183, acc: 88.28%, op_acc: 67.19%] [G loss: 1.39674]

1125 [D loss: 0.212158, acc: 89.84%, op_acc: 68.75%] [G loss: 1.48143]

1126 [D loss: 0.203652, acc: 94.53%, op_acc: 71.88%] [G loss: 1.48994]

1127 [D loss: 0.195578, acc: 91.41%, op_acc: 65.62%] [G loss: 1.55869]

1128 [D loss: 0.196927, acc: 93.75%, op_acc: 61.72%] [G loss: 1.57270]

1129 [D loss: 0.191368, acc: 91.41%, op_acc: 69.53%] [G loss: 1.49431]
```

```
1130 [D loss: 0.204565, acc: 88.28%, op_acc: 78.12%] [G loss: 1.54552]

Epoch: 1130, F1: 0.07792, F1P: 113

1131 [D loss: 0.205292, acc: 92.19%, op_acc: 70.31%] [G loss: 1.53771]

1132 [D loss: 0.218049, acc: 87.50%, op_acc: 66.41%] [G loss: 1.57354]

1133 [D loss: 0.211027, acc: 94.53%, op_acc: 65.62%] [G loss: 1.45510]

1134 [D loss: 0.201996, acc: 89.84%, op_acc: 68.75%] [G loss: 1.54918]

1135 [D loss: 0.211001, acc: 87.50%, op_acc: 70.31%] [G loss: 1.45216]

1136 [D loss: 0.212393, acc: 91.41%, op_acc: 68.75%] [G loss: 1.52343]

1137 [D loss: 0.201547, acc: 95.31%, op_acc: 72.66%] [G loss: 1.41600]

1138 [D loss: 0.202018, acc: 96.09%, op_acc: 61.72%] [G loss: 1.47321]

1139 [D loss: 0.212943, acc: 90.62%, op_acc: 64.06%] [G loss: 1.49548]

1140 [D loss: 0.190821, acc: 93.75%, op_acc: 71.88%] [G loss: 1.46505]

Epoch: 1140, F1: 0.06080, F1P: 114

1141 [D loss: 0.201051, acc: 91.41%, op_acc: 69.53%] [G loss: 1.55103]

1142 [D loss: 0.217152, acc: 89.84%, op_acc: 70.31%] [G loss: 1.59306]

1143 [D loss: 0.218053, acc: 87.50%, op_acc: 66.41%] [G loss: 1.59488]

1144 [D loss: 0.202476, acc: 88.28%, op_acc: 62.50%] [G loss: 1.52498]

1145 [D loss: 0.203718, acc: 94.53%, op_acc: 65.62%] [G loss: 1.59648]

1146 [D loss: 0.186336, acc: 94.53%, op_acc: 75.00%] [G loss: 1.59943]

1147 [D loss: 0.191933, acc: 96.88%, op_acc: 67.19%] [G loss: 1.50303]

1148 [D loss: 0.194363, acc: 92.19%, op_acc: 76.56%] [G loss: 1.53717]

1149 [D loss: 0.208409, acc: 89.06%, op_acc: 67.19%] [G loss: 1.62062]

1150 [D loss: 0.206777, acc: 91.41%, op_acc: 73.44%] [G loss: 1.47273]
```

```
Epoch: 1150, F1: 0.01521, F1P: 115

1151 [D loss: 0.198371, acc: 90.62%, op_acc: 67.19%] [G loss: 1.61321]

1152 [D loss: 0.205665, acc: 87.50%, op_acc: 70.31%] [G loss: 1.58799]

1153 [D loss: 0.193298, acc: 91.41%, op_acc: 71.88%] [G loss: 1.59114]

1154 [D loss: 0.192280, acc: 94.53%, op_acc: 78.12%] [G loss: 1.52777]

1155 [D loss: 0.201438, acc: 89.06%, op_acc: 75.78%] [G loss: 1.54824]

1156 [D loss: 0.185604, acc: 95.31%, op_acc: 69.53%] [G loss: 1.52720]

1157 [D loss: 0.184719, acc: 96.09%, op_acc: 74.22%] [G loss: 1.51766]

1158 [D loss: 0.198326, acc: 92.97%, op_acc: 72.66%] [G loss: 1.56887]

1159 [D loss: 0.188869, acc: 94.53%, op_acc: 70.31%] [G loss: 1.52251]

1160 [D loss: 0.187886, acc: 93.75%, op_acc: 64.06%] [G loss: 1.55606]

Epoch: 1160, F1: 0.04857, F1P: 116

1161 [D loss: 0.194841, acc: 94.53%, op_acc: 70.31%] [G loss: 1.57923]

1162 [D loss: 0.192126, acc: 92.97%, op_acc: 66.41%] [G loss: 1.53467]

1163 [D loss: 0.183302, acc: 96.09%, op_acc: 71.88%] [G loss: 1.66233]

1164 [D loss: 0.184189, acc: 93.75%, op_acc: 66.41%] [G loss: 1.59285]

1165 [D loss: 0.180880, acc: 96.09%, op_acc: 76.56%] [G loss: 1.70809]

1166 [D loss: 0.203913, acc: 94.53%, op_acc: 65.62%] [G loss: 1.63189]

1167 [D loss: 0.196364, acc: 91.41%, op_acc: 69.53%] [G loss: 1.66986]

1168 [D loss: 0.190548, acc: 92.97%, op_acc: 67.97%] [G loss: 1.60546]

1169 [D loss: 0.189782, acc: 92.19%, op_acc: 71.88%] [G loss: 1.61989]

1170 [D loss: 0.184751, acc: 92.19%, op_acc: 69.53%] [G loss: 1.62327]

Epoch: 1170, F1: 0.00987, F1P: 117
```

```
1171 [D loss: 0.196055, acc: 90.62%, op_acc: 73.44%] [G loss: 1.53340]

1172 [D loss: 0.189313, acc: 94.53%, op_acc: 75.00%] [G loss: 1.61887]

1173 [D loss: 0.192712, acc: 93.75%, op_acc: 74.22%] [G loss: 1.69985]

1174 [D loss: 0.193485, acc: 92.97%, op_acc: 73.44%] [G loss: 1.64900]

1175 [D loss: 0.179837, acc: 94.53%, op_acc: 73.44%] [G loss: 1.63552]

1176 [D loss: 0.180239, acc: 94.53%, op_acc: 79.69%] [G loss: 1.67986]

1177 [D loss: 0.186160, acc: 92.97%, op_acc: 73.44%] [G loss: 1.53767]

1178 [D loss: 0.192920, acc: 92.19%, op_acc: 73.44%] [G loss: 1.58891]

1179 [D loss: 0.209125, acc: 89.84%, op_acc: 66.41%] [G loss: 1.62000]

1180 [D loss: 0.183757, acc: 92.97%, op_acc: 70.31%] [G loss: 1.59014]

Epoch: 1180, F1: 0.01357, F1P: 118

1181 [D loss: 0.180965, acc: 92.97%, op_acc: 73.44%] [G loss: 1.69469]

1182 [D loss: 0.181026, acc: 92.97%, op_acc: 69.53%] [G loss: 1.58509]

1183 [D loss: 0.198699, acc: 89.84%, op_acc: 68.75%] [G loss: 1.55336]

1184 [D loss: 0.180821, acc: 92.97%, op_acc: 72.66%] [G loss: 1.58404]

1185 [D loss: 0.181571, acc: 96.09%, op_acc: 67.19%] [G loss: 1.60918]

1186 [D loss: 0.175041, acc: 98.44%, op_acc: 68.75%] [G loss: 1.61485]

1187 [D loss: 0.187833, acc: 94.53%, op_acc: 67.19%] [G loss: 1.62214]

1188 [D loss: 0.190362, acc: 90.62%, op_acc: 69.53%] [G loss: 1.52013]

1189 [D loss: 0.198060, acc: 90.62%, op_acc: 73.44%] [G loss: 1.61183]

1190 [D loss: 0.196709, acc: 90.62%, op_acc: 70.31%] [G loss: 1.66647]

Epoch: 1190, F1: 0.03697, F1P: 119

1191 [D loss: 0.177232, acc: 96.88%, op_acc: 66.41%] [G loss: 1.68790]
```

```
1192 [D loss: 0.202225, acc: 89.06%, op_acc: 71.88%] [G loss: 1.68147]

1193 [D loss: 0.183099, acc: 92.97%, op_acc: 71.09%] [G loss: 1.68898]

1194 [D loss: 0.195333, acc: 92.97%, op_acc: 72.66%] [G loss: 1.66041]

1195 [D loss: 0.187615, acc: 93.75%, op_acc: 64.84%] [G loss: 1.64243]

1196 [D loss: 0.184253, acc: 95.31%, op_acc: 71.88%] [G loss: 1.62317]

1197 [D loss: 0.187456, acc: 95.31%, op_acc: 70.31%] [G loss: 1.64775]

1198 [D loss: 0.178950, acc: 92.97%, op_acc: 74.22%] [G loss: 1.59234]

1199 [D loss: 0.189897, acc: 89.06%, op_acc: 69.53%] [G loss: 1.66760]

1200 [D loss: 0.202765, acc: 92.19%, op_acc: 74.22%] [G loss: 1.64999]

Epoch: 1200, F1: 0.01634, F1P: 120

1201 [D loss: 0.179955, acc: 95.31%, op_acc: 76.56%] [G loss: 1.72914]

1202 [D loss: 0.196352, acc: 90.62%, op_acc: 70.31%] [G loss: 1.74466]

1203 [D loss: 0.195895, acc: 94.53%, op_acc: 71.09%] [G loss: 1.68433]

1204 [D loss: 0.171101, acc: 95.31%, op_acc: 78.12%] [G loss: 1.60821]

1205 [D loss: 0.189309, acc: 94.53%, op_acc: 80.47%] [G loss: 1.65672]

1206 [D loss: 0.187287, acc: 94.53%, op_acc: 65.62%] [G loss: 1.65748]

1207 [D loss: 0.198767, acc: 90.62%, op_acc: 78.91%] [G loss: 1.66656]

1208 [D loss: 0.167054, acc: 96.88%, op_acc: 76.56%] [G loss: 1.66155]

1209 [D loss: 0.187906, acc: 96.09%, op_acc: 75.78%] [G loss: 1.69504]

1210 [D loss: 0.175784, acc: 93.75%, op_acc: 74.22%] [G loss: 1.63193]

Epoch: 1210, F1: 0.00862, F1P: 121

1211 [D loss: 0.195728, acc: 89.84%, op_acc: 75.78%] [G loss: 1.58639]

1212 [D loss: 0.193532, acc: 92.19%, op_acc: 71.09%] [G loss: 1.67629]
```

```
1213 [D loss: 0.187973, acc: 91.41%, op_acc: 71.09%] [G loss: 1.67978]

1214 [D loss: 0.191566, acc: 90.62%, op_acc: 72.66%] [G loss: 1.66728]

1215 [D loss: 0.173045, acc: 95.31%, op_acc: 73.44%] [G loss: 1.64306]

1216 [D loss: 0.180091, acc: 97.66%, op_acc: 76.56%] [G loss: 1.64323]

1217 [D loss: 0.189475, acc: 90.62%, op_acc: 75.00%] [G loss: 1.71452]

1218 [D loss: 0.190526, acc: 92.97%, op_acc: 75.00%] [G loss: 1.69925]

1219 [D loss: 0.183534, acc: 95.31%, op_acc: 80.47%] [G loss: 1.72505]

1220 [D loss: 0.188790, acc: 93.75%, op_acc: 68.75%] [G loss: 1.61364]

Epoch: 1350, F1: 0.44720, F1P: 135

1351 [D loss: 0.168542, acc: 96.88%, op_acc: 80.47%] [G loss: 1.73419]

1352 [D loss: 0.161431, acc: 96.88%, op_acc: 88.28%] [G loss: 1.82230]

1353 [D loss: 0.167804, acc: 97.66%, op_acc: 84.38%] [G loss: 1.68920]

1354 [D loss: 0.164733, acc: 93.75%, op_acc: 85.94%] [G loss: 1.87572]

1355 [D loss: 0.171854, acc: 96.88%, op_acc: 84.38%] [G loss: 1.88716]

1356 [D loss: 0.166711, acc: 94.53%, op_acc: 79.69%] [G loss: 1.77717]

1357 [D loss: 0.170387, acc: 95.31%, op_acc: 75.78%] [G loss: 1.74140]

1358 [D loss: 0.180897, acc: 92.19%, op_acc: 78.12%] [G loss: 1.77230]

1359 [D loss: 0.170866, acc: 92.97%, op_acc: 76.56%] [G loss: 1.71007]

1360 [D loss: 0.168630, acc: 94.53%, op_acc: 84.38%] [G loss: 1.79441]

Epoch: 1360, F1: 0.33484, F1P: 136

1361 [D loss: 0.180332, acc: 90.62%, op_acc: 78.12%] [G loss: 1.79321]

1362 [D loss: 0.176693, acc: 93.75%, op_acc: 81.25%] [G loss: 1.74400]

1363 [D loss: 0.169910, acc: 95.31%, op_acc: 89.06%] [G loss: 1.83925]
```

```
1364 [D loss: 0.180690, acc: 92.19%, op_acc: 81.25%] [G loss: 1.71416]

1365 [D loss: 0.171752, acc: 93.75%, op_acc: 80.47%] [G loss: 1.92729]

1366 [D loss: 0.167079, acc: 97.66%, op_acc: 84.38%] [G loss: 1.79844]

1367 [D loss: 0.182308, acc: 94.53%, op_acc: 74.22%] [G loss: 1.82351]

1368 [D loss: 0.162273, acc: 96.09%, op_acc: 82.03%] [G loss: 1.88638]

1369 [D loss: 0.181199, acc: 93.75%, op_acc: 82.03%] [G loss: 1.73081]

1370 [D loss: 0.174823, acc: 92.97%, op_acc: 82.81%] [G loss: 1.70549]

Epoch: 1370, F1: 0.21512, F1P: 137

1371 [D loss: 0.177312, acc: 90.62%, op_acc: 85.16%] [G loss: 1.69891]

1372 [D loss: 0.167104, acc: 93.75%, op_acc: 82.81%] [G loss: 1.81781]

1373 [D loss: 0.172837, acc: 92.97%, op_acc: 80.47%] [G loss: 1.73955]

1374 [D loss: 0.174552, acc: 96.88%, op_acc: 84.38%] [G loss: 1.78231]

1375 [D loss: 0.165713, acc: 97.66%, op_acc: 82.81%] [G loss: 1.79945]

1376 [D loss: 0.178652, acc: 92.97%, op_acc: 83.59%] [G loss: 1.75277]

1377 [D loss: 0.163761, acc: 93.75%, op_acc: 87.50%] [G loss: 1.87115]

1378 [D loss: 0.169385, acc: 97.66%, op_acc: 81.25%] [G loss: 1.80540]

1379 [D loss: 0.170054, acc: 95.31%, op_acc: 85.16%] [G loss: 1.73364]

1380 [D loss: 0.164093, acc: 98.44%, op_acc: 79.69%] [G loss: 1.85241]

Epoch: 1380, F1: 0.11396, F1P: 138

1381 [D loss: 0.162601, acc: 92.97%, op_acc: 85.16%] [G loss: 1.85765]

1382 [D loss: 0.184412, acc: 91.41%, op_acc: 71.09%] [G loss: 1.80081]

1383 [D loss: 0.176580, acc: 92.97%, op_acc: 83.59%] [G loss: 1.71000]

1384 [D loss: 0.164878, acc: 95.31%, op_acc: 83.59%] [G loss: 1.86962]
```

```
1385 [D loss: 0.181684, acc: 92.19%, op_acc: 78.12%] [G loss: 1.93950]

1386 [D loss: 0.171957, acc: 94.53%, op_acc: 82.03%] [G loss: 1.89489]

1387 [D loss: 0.186974, acc: 89.84%, op_acc: 74.22%] [G loss: 1.88942]

1388 [D loss: 0.193980, acc: 92.19%, op_acc: 81.25%] [G loss: 1.73070]

1389 [D loss: 0.177286, acc: 92.19%, op_acc: 85.16%] [G loss: 1.79157]

1390 [D loss: 0.170665, acc: 95.31%, op_acc: 79.69%] [G loss: 1.77527]


Epoch: 1430, F1: 0.00426, F1P: 143

1431 [D loss: 0.178400, acc: 93.75%, op_acc: 81.25%] [G loss: 1.87387]

1432 [D loss: 0.186821, acc: 91.41%, op_acc: 78.12%] [G loss: 1.78457]

1433 [D loss: 0.167923, acc: 95.31%, op_acc: 80.47%] [G loss: 1.80346]

1434 [D loss: 0.180961, acc: 89.06%, op_acc: 77.34%] [G loss: 1.78544]

1435 [D loss: 0.170533, acc: 93.75%, op_acc: 79.69%] [G loss: 1.91138]

1436 [D loss: 0.175007, acc: 92.97%, op_acc: 84.38%] [G loss: 1.70851]

1437 [D loss: 0.176861, acc: 92.97%, op_acc: 80.47%] [G loss: 1.81167]

1438 [D loss: 0.161414, acc: 96.09%, op_acc: 83.59%] [G loss: 1.72910]

1439 [D loss: 0.158815, acc: 95.31%, op_acc: 83.59%] [G loss: 1.63294]

1440 [D loss: 0.166705, acc: 96.88%, op_acc: 82.03%] [G loss: 1.73114]

Epoch: 1440, F1: 0.00837, F1P: 144

1441 [D loss: 0.165710, acc: 94.53%, op_acc: 85.94%] [G loss: 1.81159]

1442 [D loss: 0.184819, acc: 91.41%, op_acc: 80.47%] [G loss: 1.72275]

1443 [D loss: 0.171089, acc: 94.53%, op_acc: 81.25%] [G loss: 1.73447]

1444 [D loss: 0.172002, acc: 94.53%, op_acc: 82.81%] [G loss: 1.67435]
```

```
1445 [D loss: 0.173466, acc: 95.31%, op_acc: 80.47%] [G loss: 1.76917]

1446 [D loss: 0.168157, acc: 96.09%, op_acc: 75.00%] [G loss: 1.90501]

1447 [D loss: 0.174250, acc: 93.75%, op_acc: 87.50%] [G loss: 1.85031]

1448 [D loss: 0.177070, acc: 92.19%, op_acc: 82.81%] [G loss: 1.86037]

1449 [D loss: 0.162835, acc: 97.66%, op_acc: 78.91%] [G loss: 1.81521]

1450 [D loss: 0.175922, acc: 95.31%, op_acc: 83.59%] [G loss: 1.77611]

Epoch: 1450, F1: 0.00528, F1P: 145

1451 [D loss: 0.160118, acc: 93.75%, op_acc: 82.81%] [G loss: 1.69905]

1452 [D loss: 0.159947, acc: 96.88%, op_acc: 80.47%] [G loss: 1.63061]

1453 [D loss: 0.172977, acc: 95.31%, op_acc: 76.56%] [G loss: 1.70066]

1454 [D loss: 0.174831, acc: 92.97%, op_acc: 83.59%] [G loss: 1.83185]

1455 [D loss: 0.176523, acc: 93.75%, op_acc: 79.69%] [G loss: 1.83113]

1456 [D loss: 0.159923, acc: 97.66%, op_acc: 82.03%] [G loss: 1.80283]

1457 [D loss: 0.172117, acc: 96.88%, op_acc: 76.56%] [G loss: 1.73458]

1458 [D loss: 0.180025, acc: 92.97%, op_acc: 75.00%] [G loss: 1.82496]

1459 [D loss: 0.171037, acc: 97.66%, op_acc: 75.78%] [G loss: 1.78071]

1460 [D loss: 0.174516, acc: 92.19%, op_acc: 82.81%] [G loss: 1.81473]

Epoch: 1460, F1: 0.00405, F1P: 146

1461 [D loss: 0.183690, acc: 90.62%, op_acc: 77.34%] [G loss: 1.72424]

1462 [D loss: 0.162657, acc: 97.66%, op_acc: 84.38%] [G loss: 1.81476]

1463 [D loss: 0.165948, acc: 96.88%, op_acc: 77.34%] [G loss: 1.79677]

1464 [D loss: 0.163302, acc: 96.09%, op_acc: 85.16%] [G loss: 1.91591]

1465 [D loss: 0.172688, acc: 93.75%, op_acc: 78.12%] [G loss: 1.79918]
```

```
1466 [D loss: 0.167120, acc: 96.09%, op_acc: 84.38%] [G loss: 1.68703]

1467 [D loss: 0.162189, acc: 96.09%, op_acc: 78.12%] [G loss: 1.81244]

1468 [D loss: 0.176656, acc: 94.53%, op_acc: 83.59%] [G loss: 1.65147]

1469 [D loss: 0.166899, acc: 96.88%, op_acc: 84.38%] [G loss: 1.84614]

1470 [D loss: 0.164066, acc: 98.44%, op_acc: 81.25%] [G loss: 1.86772]

Epoch: 1470, F1: 0.00389, F1P: 147

1471 [D loss: 0.172946, acc: 95.31%, op_acc: 77.34%] [G loss: 1.90161]

1472 [D loss: 0.176396, acc: 92.97%, op_acc: 74.22%] [G loss: 1.78985]

1473 [D loss: 0.177556, acc: 90.62%, op_acc: 78.91%] [G loss: 1.82821]

1474 [D loss: 0.165502, acc: 96.88%, op_acc: 80.47%] [G loss: 1.76177]

1475 [D loss: 0.170192, acc: 93.75%, op_acc: 80.47%] [G loss: 1.90747]

1476 [D loss: 0.159689, acc: 96.88%, op_acc: 85.94%] [G loss: 1.91469]

1477 [D loss: 0.164672, acc: 95.31%, op_acc: 80.47%] [G loss: 1.69898]

1478 [D loss: 0.158759, acc: 96.88%, op_acc: 85.94%] [G loss: 1.77828]

1479 [D loss: 0.181030, acc: 91.41%, op_acc: 75.00%] [G loss: 1.80432]

1480 [D loss: 0.171517, acc: 95.31%, op_acc: 82.81%] [G loss: 1.82314]

Epoch: 1480, F1: 0.00355, F1P: 148

1481 [D loss: 0.170967, acc: 94.53%, op_acc: 78.91%] [G loss: 1.84985]

1482 [D loss: 0.156428, acc: 98.44%, op_acc: 79.69%] [G loss: 1.90777]

1483 [D loss: 0.177717, acc: 94.53%, op_acc: 78.12%] [G loss: 1.85524]

1484 [D loss: 0.180288, acc: 92.97%, op_acc: 79.69%] [G loss: 1.81346]

1485 [D loss: 0.179610, acc: 92.97%, op_acc: 75.00%] [G loss: 1.78708]

1486 [D loss: 0.159800, acc: 93.75%, op_acc: 79.69%] [G loss: 1.92668]
```

```
1487 [D loss: 0.178180, acc: 94.53%, op_acc: 78.91%] [G loss: 1.90953]

1488 [D loss: 0.155116, acc: 96.09%, op_acc: 84.38%] [G loss: 1.89374]

1489 [D loss: 0.162037, acc: 95.31%, op_acc: 82.81%] [G loss: 1.93301]

1490 [D loss: 0.185864, acc: 92.19%, op_acc: 78.12%] [G loss: 1.95890]

Epoch: 1490, F1: 0.00345, F1P: 149

1491 [D loss: 0.168121, acc: 95.31%, op_acc: 80.47%] [G loss: 1.82829]

1492 [D loss: 0.165030, acc: 95.31%, op_acc: 78.91%] [G loss: 1.78613]

1493 [D loss: 0.172540, acc: 92.97%, op_acc: 78.12%] [G loss: 1.78444]

1494 [D loss: 0.171868, acc: 95.31%, op_acc: 78.12%] [G loss: 1.81260]

1495 [D loss: 0.167455, acc: 95.31%, op_acc: 85.16%] [G loss: 1.91176]

1496 [D loss: 0.173401, acc: 89.84%, op_acc: 75.00%] [G loss: 1.88658]

1497 [D loss: 0.163587, acc: 94.53%, op_acc: 78.91%] [G loss: 1.87814]

1498 [D loss: 0.173368, acc: 93.75%, op_acc: 80.47%] [G loss: 1.88621]

1499 [D loss: 0.166569, acc: 92.97%, op_acc: 76.56%] [G loss: 1.96082]

1500 [D loss: 0.180662, acc: 92.97%, op_acc: 74.22%] [G loss: 1.70281]

Epoch: 1500, F1: 0.00344, F1P: 150

1501 [D loss: 0.164539, acc: 93.75%, op_acc: 75.00%] [G loss: 1.86363]

1502 [D loss: 0.173898, acc: 92.97%, op_acc: 79.69%] [G loss: 1.92720]

1503 [D loss: 0.173664, acc: 92.19%, op_acc: 75.78%] [G loss: 1.93803]

1504 [D loss: 0.168451, acc: 90.62%, op_acc: 82.81%] [G loss: 1.73960]

1505 [D loss: 0.170535, acc: 91.41%, op_acc: 79.69%] [G loss: 1.91554]

1506 [D loss: 0.166121, acc: 95.31%, op_acc: 77.34%] [G loss: 1.90810]

1507 [D loss: 0.166912, acc: 95.31%, op_acc: 77.34%] [G loss: 1.81269]
```

```
1508 [D loss: 0.161112, acc: 96.09%, op_acc: 77.34%] [G loss: 1.96328]

1509 [D loss: 0.171383, acc: 94.53%, op_acc: 81.25%] [G loss: 1.88790]

1510 [D loss: 0.156444, acc: 96.09%, op_acc: 84.38%] [G loss: 1.74845]


Epoch: 1560, F1: 0.00344, F1P: 156

1561 [D loss: 0.157731, acc: 97.66%, op_acc: 82.03%] [G loss: 1.89535]

1562 [D loss: 0.154194, acc: 96.88%, op_acc: 80.47%] [G loss: 1.93517]

1563 [D loss: 0.156941, acc: 96.09%, op_acc: 80.47%] [G loss: 2.00602]

1564 [D loss: 0.163771, acc: 95.31%, op_acc: 84.38%] [G loss: 1.83766]

1565 [D loss: 0.151972, acc: 93.75%, op_acc: 85.16%] [G loss: 1.86252]

1566 [D loss: 0.147689, acc: 95.31%, op_acc: 81.25%] [G loss: 1.93298]

1567 [D loss: 0.147978, acc: 96.09%, op_acc: 82.03%] [G loss: 1.96786]

1568 [D loss: 0.153729, acc: 95.31%, op_acc: 83.59%] [G loss: 2.13876]

1569 [D loss: 0.149344, acc: 95.31%, op_acc: 82.81%] [G loss: 1.90259]

1570 [D loss: 0.167371, acc: 93.75%, op_acc: 78.12%] [G loss: 1.91169]

Epoch: 1570, F1: 0.00344, F1P: 157

1571 [D loss: 0.156064, acc: 96.09%, op_acc: 84.38%] [G loss: 1.88818]

1572 [D loss: 0.168274, acc: 91.41%, op_acc: 78.12%] [G loss: 1.90014]

1573 [D loss: 0.175530, acc: 90.62%, op_acc: 75.00%] [G loss: 2.02216]

1574 [D loss: 0.168083, acc: 93.75%, op_acc: 71.88%] [G loss: 1.92117]

1575 [D loss: 0.150297, acc: 93.75%, op_acc: 82.81%] [G loss: 1.85307]

1576 [D loss: 0.166922, acc: 93.75%, op_acc: 78.91%] [G loss: 1.84279]

1577 [D loss: 0.174624, acc: 92.97%, op_acc: 75.78%] [G loss: 1.90482]
```

```
1578 [D loss: 0.145170, acc: 96.09%, op_acc: 84.38%] [G loss: 2.04357]

1579 [D loss: 0.159709, acc: 93.75%, op_acc: 85.16%] [G loss: 2.08012]

1580 [D loss: 0.158806, acc: 96.88%, op_acc: 74.22%] [G loss: 1.96605]

Epoch: 1580, F1: 0.00344, F1P: 158

1581 [D loss: 0.173390, acc: 90.62%, op_acc: 79.69%] [G loss: 1.89917]

1582 [D loss: 0.164300, acc: 95.31%, op_acc: 79.69%] [G loss: 1.83705]

1583 [D loss: 0.177176, acc: 95.31%, op_acc: 78.91%] [G loss: 1.93471]

1584 [D loss: 0.144200, acc: 96.88%, op_acc: 83.59%] [G loss: 1.85019]

1585 [D loss: 0.146188, acc: 96.09%, op_acc: 82.03%] [G loss: 1.82029]

1586 [D loss: 0.148118, acc: 93.75%, op_acc: 86.72%] [G loss: 1.96881]

1587 [D loss: 0.153673, acc: 93.75%, op_acc: 85.94%] [G loss: 1.90121]

1588 [D loss: 0.176319, acc: 91.41%, op_acc: 84.38%] [G loss: 1.89177]

1589 [D loss: 0.168176, acc: 93.75%, op_acc: 80.47%] [G loss: 1.86967]

1590 [D loss: 0.164187, acc: 94.53%, op_acc: 82.81%] [G loss: 1.84678]

Epoch: 1590, F1: 0.00344, F1P: 159

1591 [D loss: 0.174558, acc: 92.97%, op_acc: 76.56%] [G loss: 2.00329]

1592 [D loss: 0.154446, acc: 96.09%, op_acc: 76.56%] [G loss: 1.84777]

1593 [D loss: 0.154562, acc: 96.09%, op_acc: 76.56%] [G loss: 1.80127]

1594 [D loss: 0.151781, acc: 96.09%, op_acc: 78.91%] [G loss: 1.88244]

1595 [D loss: 0.165377, acc: 92.19%, op_acc: 75.78%] [G loss: 1.81378]

1596 [D loss: 0.155842, acc: 95.31%, op_acc: 83.59%] [G loss: 1.78676]

1597 [D loss: 0.156240, acc: 95.31%, op_acc: 82.03%] [G loss: 1.84240]

1598 [D loss: 0.157107, acc: 94.53%, op_acc: 85.94%] [G loss: 1.81134]
```

```
1599 [D loss: 0.165182, acc: 92.19%, op_acc: 82.81%] [G loss: 1.96768]

1600 [D loss: 0.163487, acc: 94.53%, op_acc: 80.47%] [G loss: 1.97616]

Epoch: 1600, F1: 0.00344, F1P: 160

1601 [D loss: 0.152643, acc: 97.66%, op_acc: 81.25%] [G loss: 1.87631]

1602 [D loss: 0.162656, acc: 92.97%, op_acc: 81.25%] [G loss: 1.80826]

1603 [D loss: 0.169026, acc: 92.97%, op_acc: 81.25%] [G loss: 1.71642]

1604 [D loss: 0.160160, acc: 96.09%, op_acc: 78.91%] [G loss: 1.67340]

1605 [D loss: 0.170856, acc: 92.19%, op_acc: 83.59%] [G loss: 1.87548]

1606 [D loss: 0.151319, acc: 92.19%, op_acc: 79.69%] [G loss: 1.82956]

1607 [D loss: 0.159719, acc: 93.75%, op_acc: 81.25%] [G loss: 1.89123]

1608 [D loss: 0.157537, acc: 94.53%, op_acc: 83.59%] [G loss: 1.79186]

1609 [D loss: 0.169360, acc: 93.75%, op_acc: 83.59%] [G loss: 1.94105]

1610 [D loss: 0.180813, acc: 90.62%, op_acc: 78.12%] [G loss: 1.83612]

Epoch: 1610, F1: 0.00344, F1P: 161

1611 [D loss: 0.163730, acc: 93.75%, op_acc: 78.12%] [G loss: 1.84449]

1612 [D loss: 0.147582, acc: 98.44%, op_acc: 80.47%] [G loss: 1.79817]

1613 [D loss: 0.157584, acc: 95.31%, op_acc: 77.34%] [G loss: 1.82141]

1614 [D loss: 0.171386, acc: 90.62%, op_acc: 78.91%] [G loss: 1.70167]

1615 [D loss: 0.153559, acc: 93.75%, op_acc: 84.38%] [G loss: 1.83780]

1616 [D loss: 0.163076, acc: 94.53%, op_acc: 82.03%] [G loss: 1.82172]

1617 [D loss: 0.155748, acc: 93.75%, op_acc: 84.38%] [G loss: 1.92239]

1618 [D loss: 0.159975, acc: 98.44%, op_acc: 85.16%] [G loss: 1.81368]

1619 [D loss: 0.159722, acc: 94.53%, op_acc: 78.12%] [G loss: 1.88141]
```

```
1620 [D loss: 0.155535, acc: 96.88%, op_acc: 78.12%] [G loss: 1.78898]

Epoch: 1620, F1: 0.00344, F1P: 162

1621 [D loss: 0.159096, acc: 96.88%, op_acc: 75.78%] [G loss: 1.82439]

1622 [D loss: 0.160363, acc: 92.97%, op_acc: 83.59%] [G loss: 1.92195]

1623 [D loss: 0.158378, acc: 94.53%, op_acc: 78.12%] [G loss: 1.84634]

1624 [D loss: 0.161675, acc: 96.09%, op_acc: 79.69%] [G loss: 1.83655]

1625 [D loss: 0.161311, acc: 95.31%, op_acc: 75.00%] [G loss: 1.84186]

1626 [D loss: 0.141239, acc: 99.22%, op_acc: 85.94%] [G loss: 1.74284]

1627 [D loss: 0.156126, acc: 94.53%, op_acc: 73.44%] [G loss: 1.75222]

1628 [D loss: 0.164859, acc: 92.97%, op_acc: 81.25%] [G loss: 1.79983]

1629 [D loss: 0.162350, acc: 94.53%, op_acc: 83.59%] [G loss: 1.98515]

1630 [D loss: 0.155187, acc: 96.88%, op_acc: 78.12%] [G loss: 1.80439]

Epoch: 1630, F1: 0.00344, F1P: 163

1631 [D loss: 0.153987, acc: 97.66%, op_acc: 83.59%] [G loss: 1.88191]

1632 [D loss: 0.155487, acc: 94.53%, op_acc: 82.81%] [G loss: 1.84010]

1633 [D loss: 0.149879, acc: 97.66%, op_acc: 87.50%] [G loss: 1.96731]

1634 [D loss: 0.168438, acc: 92.97%, op_acc: 80.47%] [G loss: 1.96609]

1635 [D loss: 0.165178, acc: 94.53%, op_acc: 86.72%] [G loss: 1.83535]

1636 [D loss: 0.142970, acc: 97.66%, op_acc: 87.50%] [G loss: 1.87086]

1637 [D loss: 0.159806, acc: 95.31%, op_acc: 81.25%] [G loss: 1.87304]

1638 [D loss: 0.145092, acc: 97.66%, op_acc: 87.50%] [G loss: 1.82709]

1639 [D loss: 0.168475, acc: 91.41%, op_acc: 82.81%] [G loss: 1.95057]

1640 [D loss: 0.151222, acc: 97.66%, op_acc: 86.72%] [G loss: 1.80589]
```

```
Epoch: 1740, F1: 0.00343, F1P: 174

1741 [D loss: 0.154475, acc: 97.66%, op_acc: 78.91%] [G loss: 1.96785]

1742 [D loss: 0.154149, acc: 95.31%, op_acc: 80.47%] [G loss: 1.87745]

1743 [D loss: 0.147587, acc: 97.66%, op_acc: 78.12%] [G loss: 1.98141]

1744 [D loss: 0.152840, acc: 93.75%, op_acc: 83.59%] [G loss: 1.92821]

1745 [D loss: 0.143535, acc: 96.09%, op_acc: 85.94%] [G loss: 1.97154]

1746 [D loss: 0.155914, acc: 96.09%, op_acc: 82.81%] [G loss: 1.98118]

1747 [D loss: 0.160205, acc: 95.31%, op_acc: 80.47%] [G loss: 1.97231]

1748 [D loss: 0.145621, acc: 96.88%, op_acc: 82.03%] [G loss: 2.10041]

1749 [D loss: 0.157072, acc: 92.97%, op_acc: 81.25%] [G loss: 1.96233]

1750 [D loss: 0.161519, acc: 93.75%, op_acc: 79.69%] [G loss: 1.71465]

Epoch: 1750, F1: 0.00343, F1P: 175

1751 [D loss: 0.159743, acc: 93.75%, op_acc: 81.25%] [G loss: 1.95155]

1752 [D loss: 0.144326, acc: 94.53%, op_acc: 79.69%] [G loss: 1.86542]

1753 [D loss: 0.143734, acc: 95.31%, op_acc: 83.59%] [G loss: 1.93485]

1754 [D loss: 0.149351, acc: 97.66%, op_acc: 78.91%] [G loss: 1.90747]

1755 [D loss: 0.162883, acc: 94.53%, op_acc: 77.34%] [G loss: 1.91282]

1756 [D loss: 0.148359, acc: 96.88%, op_acc: 82.03%] [G loss: 1.96372]

1757 [D loss: 0.142789, acc: 96.88%, op_acc: 78.12%] [G loss: 2.08282]

1758 [D loss: 0.154301, acc: 96.88%, op_acc: 81.25%] [G loss: 1.86688]

1759 [D loss: 0.163814, acc: 94.53%, op_acc: 79.69%] [G loss: 1.82372]

1760 [D loss: 0.148359, acc: 95.31%, op_acc: 80.47%] [G loss: 1.82549]

Epoch: 1760, F1: 0.00343, F1P: 176
```

```
1761 [D loss: 0.131961, acc: 96.09%, op_acc: 84.38%] [G loss: 1.87079]

1762 [D loss: 0.171367, acc: 94.53%, op_acc: 82.81%] [G loss: 1.89790]

1763 [D loss: 0.136983, acc: 97.66%, op_acc: 81.25%] [G loss: 2.06633]

1764 [D loss: 0.152809, acc: 94.53%, op_acc: 85.16%] [G loss: 1.86168]

1765 [D loss: 0.150878, acc: 95.31%, op_acc: 79.69%] [G loss: 1.96278]

1766 [D loss: 0.148867, acc: 97.66%, op_acc: 78.91%] [G loss: 1.98701]

1767 [D loss: 0.151295, acc: 94.53%, op_acc: 81.25%] [G loss: 1.97921]

1768 [D loss: 0.147082, acc: 95.31%, op_acc: 83.59%] [G loss: 1.86201]

1769 [D loss: 0.130610, acc: 99.22%, op_acc: 85.16%] [G loss: 1.93381]

1770 [D loss: 0.150724, acc: 95.31%, op_acc: 77.34%] [G loss: 1.91141]

Epoch: 1770, F1: 0.00343, F1P: 177

1771 [D loss: 0.137075, acc: 95.31%, op_acc: 75.78%] [G loss: 1.90790]

1772 [D loss: 0.142241, acc: 97.66%, op_acc: 85.16%] [G loss: 1.88844]

1773 [D loss: 0.143163, acc: 93.75%, op_acc: 73.44%] [G loss: 1.85900]

1774 [D loss: 0.159322, acc: 91.41%, op_acc: 77.34%] [G loss: 1.76377]

1775 [D loss: 0.151012, acc: 94.53%, op_acc: 77.34%] [G loss: 1.87183]

1776 [D loss: 0.147480, acc: 96.88%, op_acc: 80.47%] [G loss: 1.76481]

1777 [D loss: 0.156575, acc: 94.53%, op_acc: 82.03%] [G loss: 1.94279]

1778 [D loss: 0.151194, acc: 96.88%, op_acc: 82.03%] [G loss: 1.92221]

1779 [D loss: 0.165664, acc: 93.75%, op_acc: 77.34%] [G loss: 1.93433]

1780 [D loss: 0.148884, acc: 95.31%, op_acc: 88.28%] [G loss: 1.94392]

Epoch: 1780, F1: 0.00343, F1P: 178

1781 [D loss: 0.139008, acc: 96.88%, op_acc: 85.94%] [G loss: 1.91520]
```

```
1782 [D loss: 0.150409, acc: 95.31%, op_acc: 82.03%] [G loss: 2.01243]

1783 [D loss: 0.152496, acc: 95.31%, op_acc: 80.47%] [G loss: 1.85726]

1784 [D loss: 0.153889, acc: 96.88%, op_acc: 79.69%] [G loss: 2.01104]

1785 [D loss: 0.150912, acc: 96.88%, op_acc: 82.03%] [G loss: 1.81766]

1786 [D loss: 0.143928, acc: 96.09%, op_acc: 80.47%] [G loss: 1.90497]

1787 [D loss: 0.144351, acc: 97.66%, op_acc: 85.16%] [G loss: 1.86505]

1788 [D loss: 0.149287, acc: 96.88%, op_acc: 79.69%] [G loss: 1.84019]

1789 [D loss: 0.149999, acc: 93.75%, op_acc: 82.03%] [G loss: 1.79892]

1790 [D loss: 0.152807, acc: 95.31%, op_acc: 78.12%] [G loss: 1.96392]

Epoch: 1790, F1: 0.00343, F1P: 179

1791 [D loss: 0.141368, acc: 95.31%, op_acc: 80.47%] [G loss: 1.86151]

1792 [D loss: 0.156482, acc: 93.75%, op_acc: 84.38%] [G loss: 1.80256]

1793 [D loss: 0.148221, acc: 94.53%, op_acc: 80.47%] [G loss: 1.84009]

1794 [D loss: 0.144662, acc: 96.88%, op_acc: 81.25%] [G loss: 1.89385]

1795 [D loss: 0.153349, acc: 92.97%, op_acc: 83.59%] [G loss: 1.85148]

1796 [D loss: 0.136259, acc: 96.88%, op_acc: 85.94%] [G loss: 1.96251]

1797 [D loss: 0.150582, acc: 96.09%, op_acc: 80.47%] [G loss: 1.92901]

1798 [D loss: 0.134088, acc: 97.66%, op_acc: 84.38%] [G loss: 1.78434]

1799 [D loss: 0.138903, acc: 96.88%, op_acc: 88.28%] [G loss: 1.82240]

1800 [D loss: 0.152005, acc: 93.75%, op_acc: 85.94%] [G loss: 1.86244]

Epoch: 1800, F1: 0.00343, F1P: 180

1801 [D loss: 0.167936, acc: 94.53%, op_acc: 81.25%] [G loss: 1.84559]

1802 [D loss: 0.147669, acc: 93.75%, op_acc: 84.38%] [G loss: 1.82136]
```

```
1803 [D loss: 0.153538, acc: 94.53%, op_acc: 85.94%] [G loss: 1.88280]

1804 [D loss: 0.154170, acc: 95.31%, op_acc: 81.25%] [G loss: 1.86025]

1805 [D loss: 0.140078, acc: 96.88%, op_acc: 85.94%] [G loss: 1.89375]

1806 [D loss: 0.141504, acc: 98.44%, op_acc: 85.94%] [G loss: 1.88387]

1807 [D loss: 0.140137, acc: 98.44%, op_acc: 83.59%] [G loss: 1.88242]

1808 [D loss: 0.147443, acc: 95.31%, op_acc: 82.81%] [G loss: 1.73609]

1809 [D loss: 0.143021, acc: 98.44%, op_acc: 82.81%] [G loss: 1.91885]

1810 [D loss: 0.142112, acc: 96.09%, op_acc: 88.28%] [G loss: 1.76840]

Epoch: 1810, F1: 0.00343, F1P: 181

1811 [D loss: 0.146266, acc: 96.88%, op_acc: 82.03%] [G loss: 1.76724]

1812 [D loss: 0.159523, acc: 96.09%, op_acc: 82.03%] [G loss: 1.93083]

1813 [D loss: 0.163560, acc: 92.19%, op_acc: 82.03%] [G loss: 1.84986]

1814 [D loss: 0.142483, acc: 96.09%, op_acc: 80.47%] [G loss: 1.88969]

1815 [D loss: 0.159271, acc: 96.09%, op_acc: 83.59%] [G loss: 1.95410]

1816 [D loss: 0.152164, acc: 94.53%, op_acc: 83.59%] [G loss: 1.79522]

1817 [D loss: 0.155782, acc: 95.31%, op_acc: 82.81%] [G loss: 1.87646]

1818 [D loss: 0.172924, acc: 93.75%, op_acc: 79.69%] [G loss: 1.73641]

1819 [D loss: 0.134324, acc: 98.44%, op_acc: 80.47%] [G loss: 1.81374]

1820 [D loss: 0.143636, acc: 96.09%, op_acc: 82.81%] [G loss: 2.00917]

Epoch: 1820, F1: 0.00343, F1P: 182

1821 [D loss: 0.143105, acc: 96.09%, op_acc: 82.03%] [G loss: 1.87383]

1822 [D loss: 0.137320, acc: 96.88%, op_acc: 87.50%] [G loss: 1.77935]

1823 [D loss: 0.153069, acc: 97.66%, op_acc: 83.59%] [G loss: 1.75470]
```

```
1824 [D loss: 0.151286, acc: 96.88%, op_acc: 82.81%] [G loss: 1.75618]

1825 [D loss: 0.166059, acc: 94.53%, op_acc: 81.25%] [G loss: 1.78856]

1826 [D loss: 0.141833, acc: 100.00%, op_acc: 84.38%] [G loss: 1.7646]

1827 [D loss: 0.154641, acc: 93.75%, op_acc: 80.47%] [G loss: 1.84587]

1828 [D loss: 0.142874, acc: 96.88%, op_acc: 78.91%] [G loss: 1.86209]

1829 [D loss: 0.156757, acc: 99.22%, op_acc: 87.50%] [G loss: 1.92325]

1830 [D loss: 0.138406, acc: 96.09%, op_acc: 84.38%] [G loss: 1.91270]

Epoch: 1830, F1: 0.00343, F1P: 183

1831 [D loss: 0.139494, acc: 99.22%, op_acc: 87.50%] [G loss: 1.87369]

1832 [D loss: 0.149211, acc: 95.31%, op_acc: 84.38%] [G loss: 1.86451]

1833 [D loss: 0.134011, acc: 99.22%, op_acc: 85.94%] [G loss: 1.85931]

1834 [D loss: 0.171899, acc: 93.75%, op_acc: 79.69%] [G loss: 1.89168]

1835 [D loss: 0.145032, acc: 94.53%, op_acc: 87.50%] [G loss: 1.83947]

1836 [D loss: 0.152750, acc: 96.09%, op_acc: 82.81%] [G loss: 1.88918]

1837 [D loss: 0.140712, acc: 97.66%, op_acc: 85.16%] [G loss: 1.91317]

1838 [D loss: 0.153289, acc: 94.53%, op_acc: 86.72%] [G loss: 1.95237]

1839 [D loss: 0.153141, acc: 93.75%, op_acc: 84.38%] [G loss: 1.73842]

1840 [D loss: 0.146593, acc: 92.97%, op_acc: 82.03%] [G loss: 1.76681]

Epoch: 1840, F1: 0.00343, F1P: 184

1841 [D loss: 0.143761, acc: 96.88%, op_acc: 85.16%] [G loss: 1.73745]

1842 [D loss: 0.147026, acc: 95.31%, op_acc: 82.81%] [G loss: 1.85901]

1843 [D loss: 0.152375, acc: 94.53%, op_acc: 89.84%] [G loss: 1.79659]

1844 [D loss: 0.142524, acc: 97.66%, op_acc: 89.06%] [G loss: 1.79743]
```

```
1845 [D loss: 0.144292, acc: 95.31%, op_acc: 84.38%] [G loss: 1.84190]

1846 [D loss: 0.163399, acc: 94.53%, op_acc: 85.94%] [G loss: 1.89060]

1847 [D loss: 0.135037, acc: 96.09%, op_acc: 85.94%] [G loss: 1.90327]

1848 [D loss: 0.143837, acc: 96.88%, op_acc: 89.06%] [G loss: 1.81604]

1849 [D loss: 0.143738, acc: 94.53%, op_acc: 85.94%] [G loss: 1.80951]

1850 [D loss: 0.164835, acc: 96.09%, op_acc: 85.16%] [G loss: 1.74795]

Epoch: 1850, F1: 0.00343, F1P: 185

1851 [D loss: 0.138583, acc: 97.66%, op_acc: 87.50%] [G loss: 1.82356]

1852 [D loss: 0.134778, acc: 96.09%, op_acc: 84.38%] [G loss: 1.86004]

1853 [D loss: 0.144856, acc: 98.44%, op_acc: 85.16%] [G loss: 1.78185]

1854 [D loss: 0.139079, acc: 97.66%, op_acc: 82.03%] [G loss: 1.75876]

1855 [D loss: 0.149335, acc: 95.31%, op_acc: 84.38%] [G loss: 1.78622]

1856 [D loss: 0.144453, acc: 96.09%, op_acc: 87.50%] [G loss: 1.76345]

1857 [D loss: 0.151344, acc: 96.09%, op_acc: 85.94%] [G loss: 1.81195]

1858 [D loss: 0.155419, acc: 95.31%, op_acc: 85.94%] [G loss: 1.83144]

1859 [D loss: 0.139666, acc: 96.88%, op_acc: 84.38%] [G loss: 1.83309]

1860 [D loss: 0.145865, acc: 95.31%, op_acc: 82.81%] [G loss: 1.84419]

Epoch: 1860, F1: 0.00343, F1P: 186

1861 [D loss: 0.138119, acc: 97.66%, op_acc: 86.72%] [G loss: 1.80992]

1862 [D loss: 0.136631, acc: 99.22%, op_acc: 82.81%] [G loss: 1.77909]

1863 [D loss: 0.145068, acc: 95.31%, op_acc: 88.28%] [G loss: 1.78365]

1864 [D loss: 0.145096, acc: 96.09%, op_acc: 84.38%] [G loss: 1.78777]

1865 [D loss: 0.137741, acc: 96.88%, op_acc: 85.16%] [G loss: 1.82399]
```

```
1866 [D loss: 0.155185, acc: 97.66%, op_acc: 82.81%] [G loss: 1.67872]

1867 [D loss: 0.143731, acc: 96.09%, op_acc: 86.72%] [G loss: 1.79878]

1868 [D loss: 0.134701, acc: 97.66%, op_acc: 87.50%] [G loss: 1.80710]

1869 [D loss: 0.158305, acc: 96.09%, op_acc: 78.12%] [G loss: 1.70882]

1870 [D loss: 0.138644, acc: 98.44%, op_acc: 85.94%] [G loss: 1.74428]

Epoch: 1870, F1: 0.00343, F1P: 187

1871 [D loss: 0.151737, acc: 93.75%, op_acc: 85.16%] [G loss: 1.73256]

1872 [D loss: 0.151489, acc: 96.09%, op_acc: 89.06%] [G loss: 1.75872]

1873 [D loss: 0.149616, acc: 98.44%, op_acc: 77.34%] [G loss: 1.84093]

1874 [D loss: 0.147635, acc: 94.53%, op_acc: 82.81%] [G loss: 1.79130]

1875 [D loss: 0.155119, acc: 96.09%, op_acc: 80.47%] [G loss: 1.80213]

1876 [D loss: 0.157120, acc: 94.53%, op_acc: 78.12%] [G loss: 1.72935]

1877 [D loss: 0.135565, acc: 93.75%, op_acc: 85.16%] [G loss: 1.78149]

1878 [D loss: 0.153444, acc: 94.53%, op_acc: 77.34%] [G loss: 1.77459]

1879 [D loss: 0.153680, acc: 94.53%, op_acc: 81.25%] [G loss: 1.76040]

1880 [D loss: 0.148749, acc: 93.75%, op_acc: 79.69%] [G loss: 1.76374]

Epoch: 1880, F1: 0.00343, F1P: 188

1881 [D loss: 0.154034, acc: 96.09%, op_acc: 82.81%] [G loss: 1.83931]

1882 [D loss: 0.138662, acc: 97.66%, op_acc: 85.16%] [G loss: 1.84075]

1883 [D loss: 0.129276, acc: 95.31%, op_acc: 85.16%] [G loss: 1.71309]

1884 [D loss: 0.134065, acc: 94.53%, op_acc: 85.94%] [G loss: 1.81065]

1885 [D loss: 0.161525, acc: 91.41%, op_acc: 81.25%] [G loss: 1.77185]

1886 [D loss: 0.137180, acc: 97.66%, op_acc: 86.72%] [G loss: 1.73514]
```

```
1887 [D loss: 0.148180, acc: 96.09%, op_acc: 86.72%] [G loss: 1.81839]

1888 [D loss: 0.149208, acc: 96.09%, op_acc: 85.16%] [G loss: 1.88843]

1889 [D loss: 0.153417, acc: 94.53%, op_acc: 80.47%] [G loss: 1.76025]

1890 [D loss: 0.160509, acc: 91.41%, op_acc: 85.16%] [G loss: 1.81539]

Epoch: 1890, F1: 0.00343, F1P: 189

1891 [D loss: 0.143557, acc: 96.09%, op_acc: 81.25%] [G loss: 1.59775]

1892 [D loss: 0.155712, acc: 96.88%, op_acc: 82.03%] [G loss: 1.75858]

1893 [D loss: 0.143301, acc: 96.88%, op_acc: 80.47%] [G loss: 1.84369]

1894 [D loss: 0.157950, acc: 94.53%, op_acc: 87.50%] [G loss: 1.76797]

1895 [D loss: 0.165020, acc: 94.53%, op_acc: 86.72%] [G loss: 1.82434]

1896 [D loss: 0.151724, acc: 95.31%, op_acc: 85.94%] [G loss: 1.85563]

1897 [D loss: 0.150456, acc: 95.31%, op_acc: 78.12%] [G loss: 1.77389]

1898 [D loss: 0.163240, acc: 96.88%, op_acc: 82.03%] [G loss: 1.85052]

1899 [D loss: 0.140144, acc: 99.22%, op_acc: 86.72%] [G loss: 1.83865]

1900 [D loss: 0.162312, acc: 93.75%, op_acc: 80.47%] [G loss: 1.80621]

Epoch: 1900, F1: 0.00343, F1P: 190

1901 [D loss: 0.148964, acc: 96.09%, op_acc: 86.72%] [G loss: 1.78993]

1902 [D loss: 0.150093, acc: 96.88%, op_acc: 87.50%] [G loss: 1.82901]

1903 [D loss: 0.156806, acc: 94.53%, op_acc: 75.78%] [G loss: 1.83772]

1904 [D loss: 0.128813, acc: 99.22%, op_acc: 85.94%] [G loss: 1.78730]

1905 [D loss: 0.152765, acc: 96.09%, op_acc: 88.28%] [G loss: 1.71357]

1906 [D loss: 0.141740, acc: 96.09%, op_acc: 88.28%] [G loss: 1.68554]

1907 [D loss: 0.142516, acc: 95.31%, op_acc: 85.16%] [G loss: 1.83414]
```

```
1908 [D loss: 0.153863, acc: 95.31%, op_acc: 85.16%] [G loss: 1.84225]

1909 [D loss: 0.140946, acc: 97.66%, op_acc: 86.72%] [G loss: 1.88298]

1910 [D loss: 0.140357, acc: 98.44%, op_acc: 83.59%] [G loss: 1.83259]

Epoch: 1910, F1: 0.00343, F1P: 191

1911 [D loss: 0.132278, acc: 97.66%, op_acc: 88.28%] [G loss: 1.71676]

1912 [D loss: 0.149545, acc: 94.53%, op_acc: 78.91%] [G loss: 1.82037]

1913 [D loss: 0.143778, acc: 96.88%, op_acc: 84.38%] [G loss: 1.73960]

1914 [D loss: 0.151317, acc: 92.19%, op_acc: 86.72%] [G loss: 1.86248]

1915 [D loss: 0.146400, acc: 96.88%, op_acc: 86.72%] [G loss: 1.75861]

1916 [D loss: 0.141693, acc: 96.09%, op_acc: 89.84%] [G loss: 1.74859]

1917 [D loss: 0.141042, acc: 97.66%, op_acc: 90.62%] [G loss: 1.84075]

1918 [D loss: 0.134913, acc: 96.09%, op_acc: 89.06%] [G loss: 1.91206]

1919 [D loss: 0.142586, acc: 92.97%, op_acc: 84.38%] [G loss: 1.82768]

1920 [D loss: 0.162569, acc: 93.75%, op_acc: 83.59%] [G loss: 1.82230]

Epoch: 1920, F1: 0.00343, F1P: 192

1921 [D loss: 0.136451, acc: 96.88%, op_acc: 85.94%] [G loss: 1.92376]

1922 [D loss: 0.158142, acc: 92.97%, op_acc: 82.03%] [G loss: 1.76281]

1923 [D loss: 0.145317, acc: 96.88%, op_acc: 82.81%] [G loss: 1.81496]

1924 [D loss: 0.147327, acc: 96.88%, op_acc: 86.72%] [G loss: 1.74482]

1925 [D loss: 0.138076, acc: 96.09%, op_acc: 83.59%] [G loss: 1.73226]

1926 [D loss: 0.139200, acc: 95.31%, op_acc: 91.41%] [G loss: 1.84974]

1927 [D loss: 0.149465, acc: 96.09%, op_acc: 90.62%] [G loss: 1.80055]

1928 [D loss: 0.147817, acc: 95.31%, op_acc: 89.06%] [G loss: 1.66045]
```

```
1929 [D loss: 0.157365, acc: 96.09%, op_acc: 79.69%] [G loss: 1.74211]

1930 [D loss: 0.161845, acc: 93.75%, op_acc: 84.38%] [G loss: 1.75102]

Epoch: 1930, F1: 0.00343, F1P: 193

1931 [D loss: 0.151665, acc: 98.44%, op_acc: 85.94%] [G loss: 1.72788]

1932 [D loss: 0.152304, acc: 95.31%, op_acc: 90.62%] [G loss: 1.72379]

1933 [D loss: 0.134811, acc: 96.88%, op_acc: 86.72%] [G loss: 1.71921]

1934 [D loss: 0.148880, acc: 94.53%, op_acc: 83.59%] [G loss: 1.88365]

1935 [D loss: 0.146779, acc: 96.88%, op_acc: 85.16%] [G loss: 1.74007]

1936 [D loss: 0.138611, acc: 96.88%, op_acc: 85.16%] [G loss: 1.92277]

1937 [D loss: 0.126016, acc: 99.22%, op_acc: 84.38%] [G loss: 1.84268]

1938 [D loss: 0.146728, acc: 98.44%, op_acc: 87.50%] [G loss: 1.80385]

1939 [D loss: 0.146565, acc: 96.88%, op_acc: 87.50%] [G loss: 1.93838]

1940 [D loss: 0.141233, acc: 96.88%, op_acc: 91.41%] [G loss: 1.86102]

Epoch: 1940, F1: 0.00343, F1P: 194

1941 [D loss: 0.134471, acc: 97.66%, op_acc: 84.38%] [G loss: 1.80227]

1942 [D loss: 0.133894, acc: 96.09%, op_acc: 86.72%] [G loss: 1.97530]

1943 [D loss: 0.128753, acc: 97.66%, op_acc: 85.16%] [G loss: 1.90283]

1944 [D loss: 0.147828, acc: 95.31%, op_acc: 86.72%] [G loss: 1.97921]

1945 [D loss: 0.127294, acc: 98.44%, op_acc: 89.84%] [G loss: 1.80221]

1946 [D loss: 0.136034, acc: 97.66%, op_acc: 89.84%] [G loss: 1.84799]

1947 [D loss: 0.143605, acc: 95.31%, op_acc: 89.84%] [G loss: 1.72765]

1948 [D loss: 0.135528, acc: 98.44%, op_acc: 84.38%] [G loss: 1.84209]

1949 [D loss: 0.130571, acc: 99.22%, op_acc: 89.06%] [G loss: 1.84516]
```

```
1950 [D loss: 0.137534, acc: 97.66%, op_acc: 86.72%] [G loss: 1.74623]

Epoch: 1950, F1: 0.00343, F1P: 195

1951 [D loss: 0.145173, acc: 98.44%, op_acc: 92.19%] [G loss: 1.83297]

1952 [D loss: 0.148643, acc: 95.31%, op_acc: 87.50%] [G loss: 1.88329]

1953 [D loss: 0.154572, acc: 96.09%, op_acc: 79.69%] [G loss: 1.94156]

1954 [D loss: 0.138082, acc: 97.66%, op_acc: 85.16%] [G loss: 1.90790]

1955 [D loss: 0.158501, acc: 92.97%, op_acc: 84.38%] [G loss: 1.76748]

1956 [D loss: 0.155329, acc: 95.31%, op_acc: 79.69%] [G loss: 1.83141]

1957 [D loss: 0.135647, acc: 96.88%, op_acc: 87.50%] [G loss: 1.93532]

1958 [D loss: 0.138893, acc: 98.44%, op_acc: 85.94%] [G loss: 1.83229]

1959 [D loss: 0.152990, acc: 92.19%, op_acc: 82.03%] [G loss: 1.77008]

1960 [D loss: 0.143183, acc: 96.09%, op_acc: 82.81%] [G loss: 1.85839]

Epoch: 1960, F1: 0.00343, F1P: 196

1961 [D loss: 0.135273, acc: 98.44%, op_acc: 89.84%] [G loss: 1.94964]

1962 [D loss: 0.138737, acc: 95.31%, op_acc: 82.81%] [G loss: 1.89276]

1963 [D loss: 0.117096, acc: 97.66%, op_acc: 91.41%] [G loss: 1.75525]

1964 [D loss: 0.139287, acc: 94.53%, op_acc: 86.72%] [G loss: 1.95286]

1965 [D loss: 0.153039, acc: 95.31%, op_acc: 89.84%] [G loss: 1.90582]

1966 [D loss: 0.126472, acc: 96.09%, op_acc: 89.06%] [G loss: 1.88851]

1967 [D loss: 0.132524, acc: 98.44%, op_acc: 86.72%] [G loss: 1.98459]

1968 [D loss: 0.120161, acc: 99.22%, op_acc: 91.41%] [G loss: 1.88490]

1969 [D loss: 0.120180, acc: 99.22%, op_acc: 92.97%] [G loss: 1.88248]

1970 [D loss: 0.144923, acc: 96.09%, op_acc: 87.50%] [G loss: 1.81320]
```

```
Epoch: 1970, F1: 0.00343, F1P: 197

1971 [D loss: 0.146025, acc: 96.09%, op_acc: 85.94%] [G loss: 1.87320]

1972 [D loss: 0.142844, acc: 96.88%, op_acc: 87.50%] [G loss: 1.89259]

1973 [D loss: 0.138237, acc: 96.88%, op_acc: 86.72%] [G loss: 1.91527]

1974 [D loss: 0.130662, acc: 96.88%, op_acc: 87.50%] [G loss: 1.91265]

1975 [D loss: 0.111030, acc: 99.22%, op_acc: 92.97%] [G loss: 1.87640]

1976 [D loss: 0.135589, acc: 99.22%, op_acc: 88.28%] [G loss: 1.92350]

1977 [D loss: 0.138296, acc: 96.09%, op_acc: 86.72%] [G loss: 1.84192]

1978 [D loss: 0.137814, acc: 96.09%, op_acc: 85.94%] [G loss: 1.88070]

1979 [D loss: 0.140351, acc: 95.31%, op_acc: 88.28%] [G loss: 1.78903]

1980 [D loss: 0.136159, acc: 97.66%, op_acc: 94.53%] [G loss: 1.83764]

Epoch: 1980, F1: 0.00343, F1P: 198

1981 [D loss: 0.143662, acc: 95.31%, op_acc: 87.50%] [G loss: 1.78125]

1982 [D loss: 0.136870, acc: 95.31%, op_acc: 86.72%] [G loss: 1.85106]

1983 [D loss: 0.114536, acc: 97.66%, op_acc: 89.84%] [G loss: 1.87929]

1984 [D loss: 0.149272, acc: 96.88%, op_acc: 89.06%] [G loss: 1.93498]

1985 [D loss: 0.140752, acc: 96.88%, op_acc: 82.81%] [G loss: 1.82396]

1986 [D loss: 0.133076, acc: 97.66%, op_acc: 90.62%] [G loss: 1.98347]

1987 [D loss: 0.121763, acc: 98.44%, op_acc: 88.28%] [G loss: 1.80779]

1988 [D loss: 0.138363, acc: 97.66%, op_acc: 82.81%] [G loss: 1.90205]

1989 [D loss: 0.139557, acc: 98.44%, op_acc: 82.03%] [G loss: 1.90156]

1990 [D loss: 0.129898, acc: 97.66%, op_acc: 89.06%] [G loss: 1.89199]

Epoch: 1990, F1: 0.00343, F1P: 199
```

```
1991 [D loss: 0.131333, acc: 96.88%, op_acc: 92.19%] [G loss: 1.86628]

1992 [D loss: 0.150975, acc: 95.31%, op_acc: 82.03%] [G loss: 1.87086]

1993 [D loss: 0.132550, acc: 96.88%, op_acc: 84.38%] [G loss: 1.91079]

1994 [D loss: 0.110268, acc: 97.66%, op_acc: 91.41%] [G loss: 1.95126]

1995 [D loss: 0.123761, acc: 98.44%, op_acc: 89.84%] [G loss: 1.95101]

1996 [D loss: 0.142888, acc: 97.66%, op_acc: 85.94%] [G loss: 1.98492]

1997 [D loss: 0.119540, acc: 97.66%, op_acc: 87.50%] [G loss: 1.92760]

1998 [D loss: 0.134068, acc: 97.66%, op_acc: 85.94%] [G loss: 1.87274]

1999 [D loss: 0.148692, acc: 96.88%, op_acc: 81.25%] [G loss: 1.86436]

2000 [D loss: 0.122028, acc: 98.44%, op_acc: 86.72%] [G loss: 1.87643]
```