Comparative Analysis of Two Anomaly Detection Algorithms

Abstract 3
Introduction 4
The dataset 5
Experimental Setup 6
Executing the code on your machine 7
1.Preprocessing and feature selection: 8
1.2 Dividing the dataset into training and testing sets 10
2. Artificially create a balanced dataset combining oversampling and undersampling techniques 12
3. Supervised method for Anomaly detection 14
Support Vector Machine (SVM) 15
3.1 Support Vector Machine on unbalanced dataset 16
3.2 Support Vector Machine on balanced dataset 18
4. Unsupervised method for anomaly detection 20
Isolation Forest 21
4.1 Isolation Forest on imbalanced dataset 21
4.1 Isolation Forest on balanced dataset 23
5. Analysis of the results 24
5.1 Comprehensive performance analysis of the supervised methods 24
5.2 Comprehensive performance analysis of the unsupervised methods 27
6. Comparative analysis of the algorithms chosen 30
Conclusion 31
References 32

Abstract

The ongoing dematerialization of data from the real world has a significant impact on the expansion of the traded data. Anomaly detection in this situation is progressively becoming a crucial data analysis task in order to identify aberrant data, which is of specific relevance and might demand action.
Recent developments in machine learning and other artificial intelligence techniques are making significant strides in this field. These methods have often been developed for balanced data sets or for data distributions that follow particular presumptions. The real applications, however, are more often faced with an unbalanced data distribution, where aberrant cases are typically quite rare while normal data are available in great abundance. This makes finding an anomaly comparable to looking for a needle in a haystack.

We create an experimental setup for comparative investigation of two different machine learning algorithms in their use with anomaly detection systems as part of a comparative study. We research their effectiveness while accounting for anomaly distribution in an unbalanced dataset.

Introduction

Anomaly detection in general involves detection of rare or non frequent events of data which deviate significantly from the normal readings. The non frequent rare events of data that are of trivial importance in detecting anomalous events in real life applications such as bank transactions[1], network traffic[2], s etc where majority of the events are normal and routinary. Anomaly detection can be even used in medical fields to monitor health metrics and generate preventive diagnostic results as well as caution inducing warnings.

The detection of these anomalous events are difficult due to two main reasons

1) These events are non frequent or rarely occurring
2) Even when they do occur, there is a high possibility that it could be a false positive as human beings are sometimes error prone under unfamiliar conditions.

The recent advances made in the field of artificial intelligence and machine learning had made it possible to detect these anomalous events through the usage of two learning paradigms

1) Supervised Learning
2) Unsupervised Learning

The supervised learning paradigm requires the observations to be labeled as anomalous or non anomalous. The objective here is to train a model to be able to separate the normal events from the anomalous events. The un supervised learning paradigm on the other hand requires an unlabeled observations as the aim here is to build a model that takes the proximity or density assumptions of the observations. It expects an event to be abnormal or anomalous if its different from the others according to its parameters[3].

The challenge with using these two anomaly detection approaches is that , most of the techniques devised using these paradigms have a certain degree of assumption about the data with respect to its distributions. This makes detection of real life anomalous events harder as they might constitute to only a minority percentage of distribution in the available observations. This type of a dataset is called an imbalance dataset as the number one class constitutes to only a minority when compared to the other class[4].

This affects the performance of the algorithms as there could be a possibility of these algorithms being of accuracy of upto 90% but its only in detecting the normal non anomalous events accurately. Thus even though the accuracy score could be of 90% since its not able to appropriately detect and classify the real anomalous events, the cost of such a bad prediction will be high in real life applications[4].

In this study , we use the “Credit card fraud detection dataset” provided by Kaggle [9]. The main objective of our work is to study the performance of supervised and unsupervised machine learning paradigms on this unbalanced labeled dataset. The performance of the two paradigms can be measured using the Accuracy as well as the Area Under ROC Curve (AuROC) metrics.

The dataset

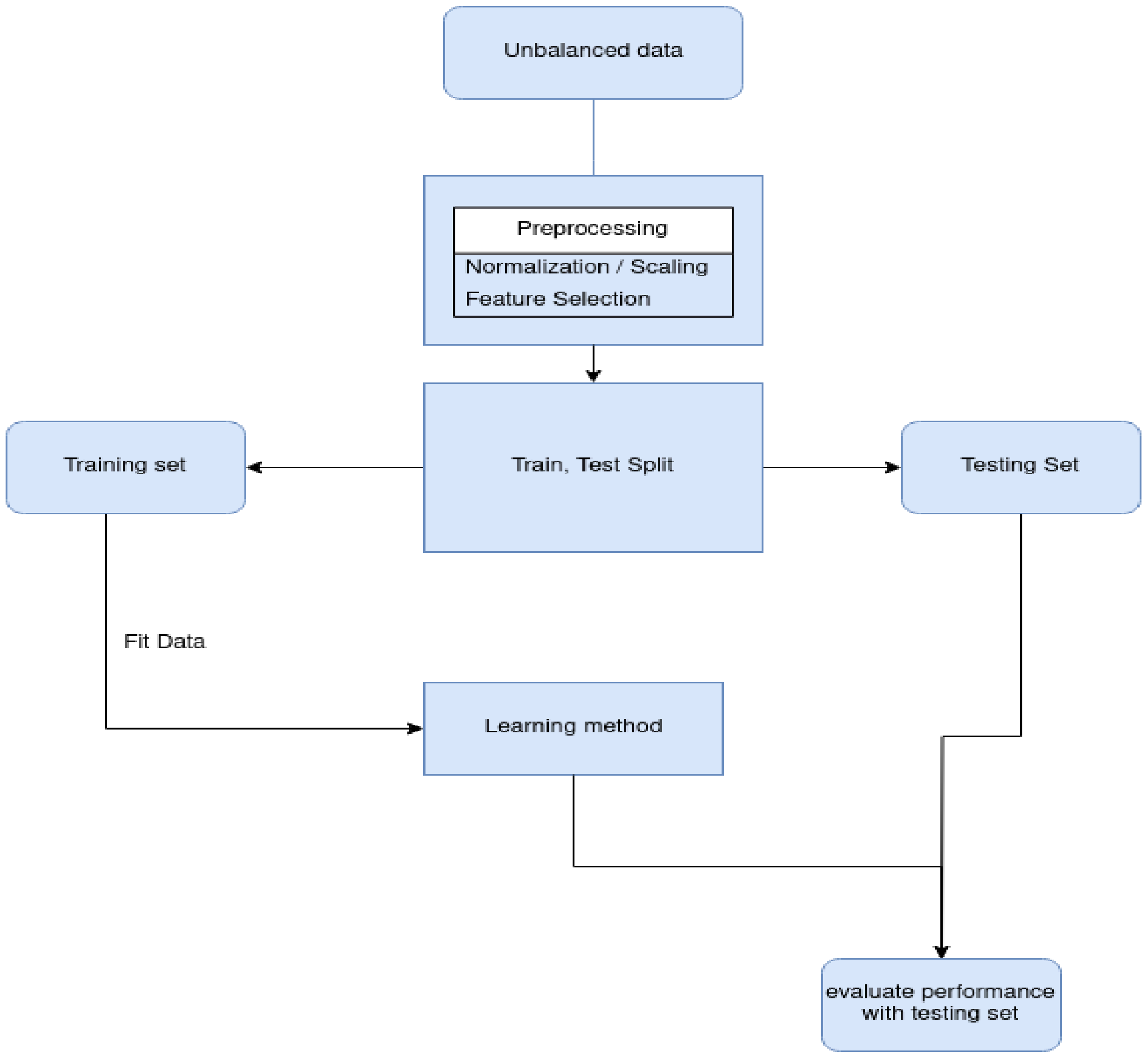
The data set we are using for our study is **Credit card fraud detection dataset**” provided by Kaggle [5].The dataset contains credit card transactions made by European cardholders in September 2013 . This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' is represented in the first column and it contains the seconds elapsed between each transaction and the first transaction in the dataset. The second last column of the dataset represents the feature 'Amount' is the transaction Amount. The last column represents the Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise. As previously mentioned features V1,V2,V3,.....V28 are principal components obtained with PCA. This is possibly sensitive information of the card holders such as the geo location of the transaction, the card type, race, ethnicity of the account holder etc which was transformed into values between -3 and 3.

Going forward we will be using the labels and terminology Fraud for Anomalous Class or observation and Not Fraud for Non Anomalous Classes or observations interchangeably.

Experimental Setup

The experimental setup consists of the following components



The steps are common for both supervised and unsupervised learning methods. The only difference being for unsupervised learning is that we will be dropping the class column also.

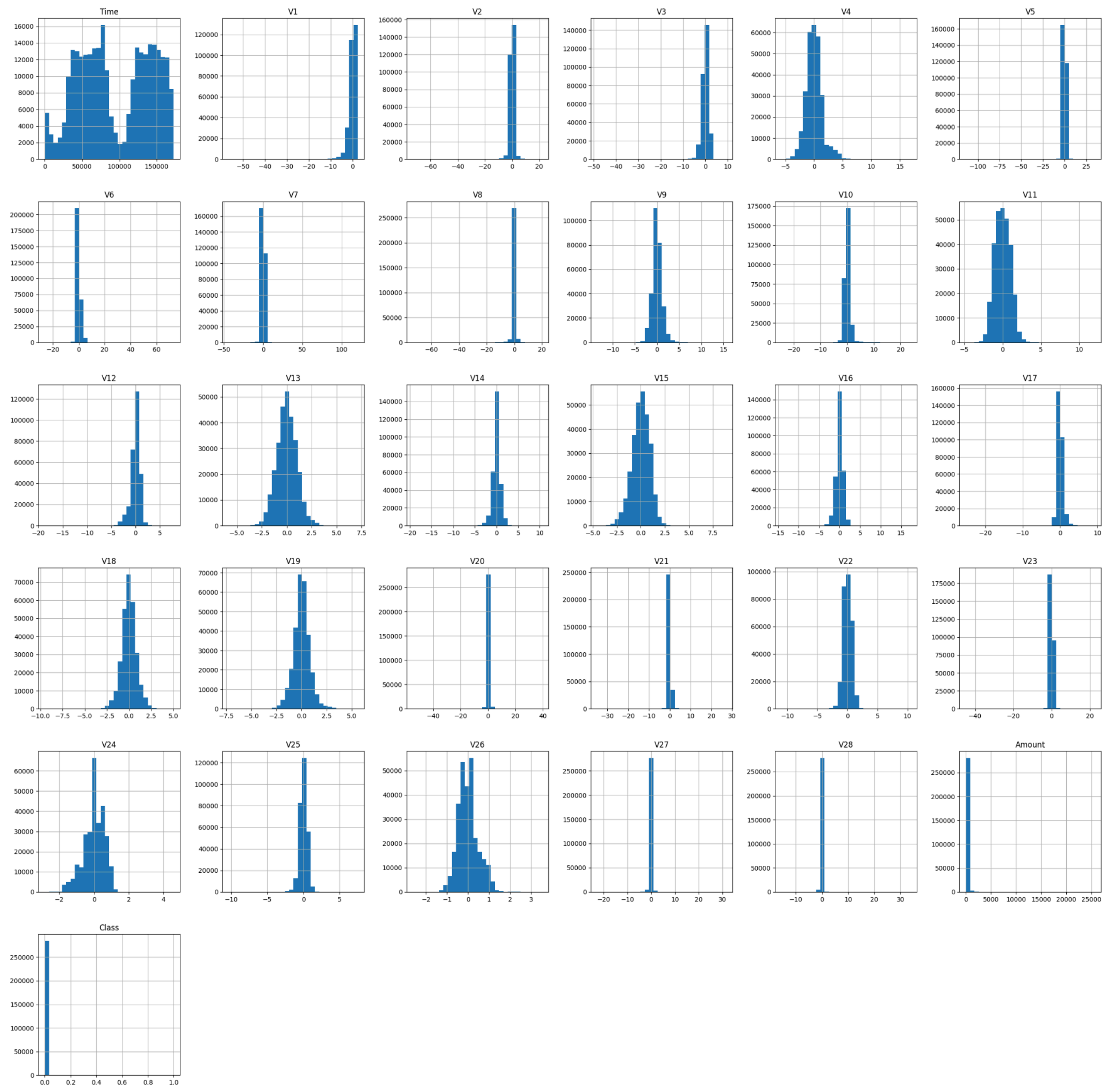
For evaluating the performance on a balanced dataset, we will be performing required operations on the training set only and evaluating the performance on the imbalanced testing set.

Executing the code on your machine

It is recommended to make use of the google collab platform to directly run and view the results of the study from the python code. Detailed information regarding how this can be done with google collab along are attached with this study.
This way of executing the code will eliminate all possibilities of unsupported dependencies or peer dependencies or even missing dependencies from affecting the execution of the project.
This method is also preferred as the library functions used for the study are resource heavy and might take some time for execution in a local environment when considering the application overheads and operating systems resource requirements into consideration.

1.Preprocessing and feature selection:

Plotting a simple histogram of the various attributes of the dataset results in the creation of the following histograms

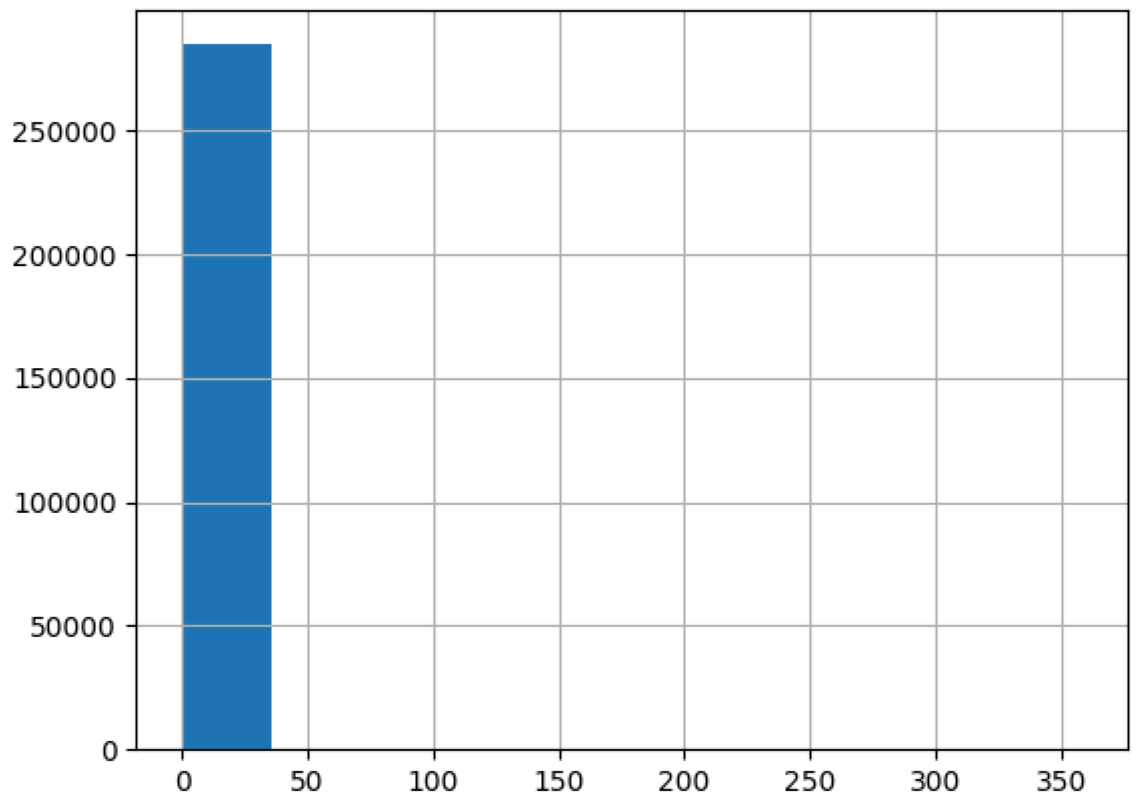


Here the time is the first plot and its a value that ranges between 0 and 48 hours converted to seconds and the next set of attributes from V1 to V28 have different distributions . Since the nature and type of the features represented by V1 to V28 are unknown , ignoring these are simply not an option and hence those features are to be retained.

One thing however thats evident from the histograms is that the amount column has values ranging between o and ~25000. Many machine learning algorithms perform better when numerical input variables are scaled to a standard range.This is because data often consists of many different input variables or features (columns) and each may have a different range of values or units of measure, such as feet, miles, kilograms, dollars, etc.

If there are input variables that have very large values relative to the other input variables, these large values can dominate or skew some machine learning algorithms. The result is that the algorithms pay most of their attention to the large values and ignore the variables with smaller values.One approach to standardizing input variables in the presence of outliers is to ignore the outliers from the calculation of the mean and standard deviation, then use the calculated values to scale the variable.This is called robust standardization or **robust data scaling**[18].

After the robust data scaling of the amount class, the values are now spread across a range of 0 to ~ 25 as shown by the histogram plot.



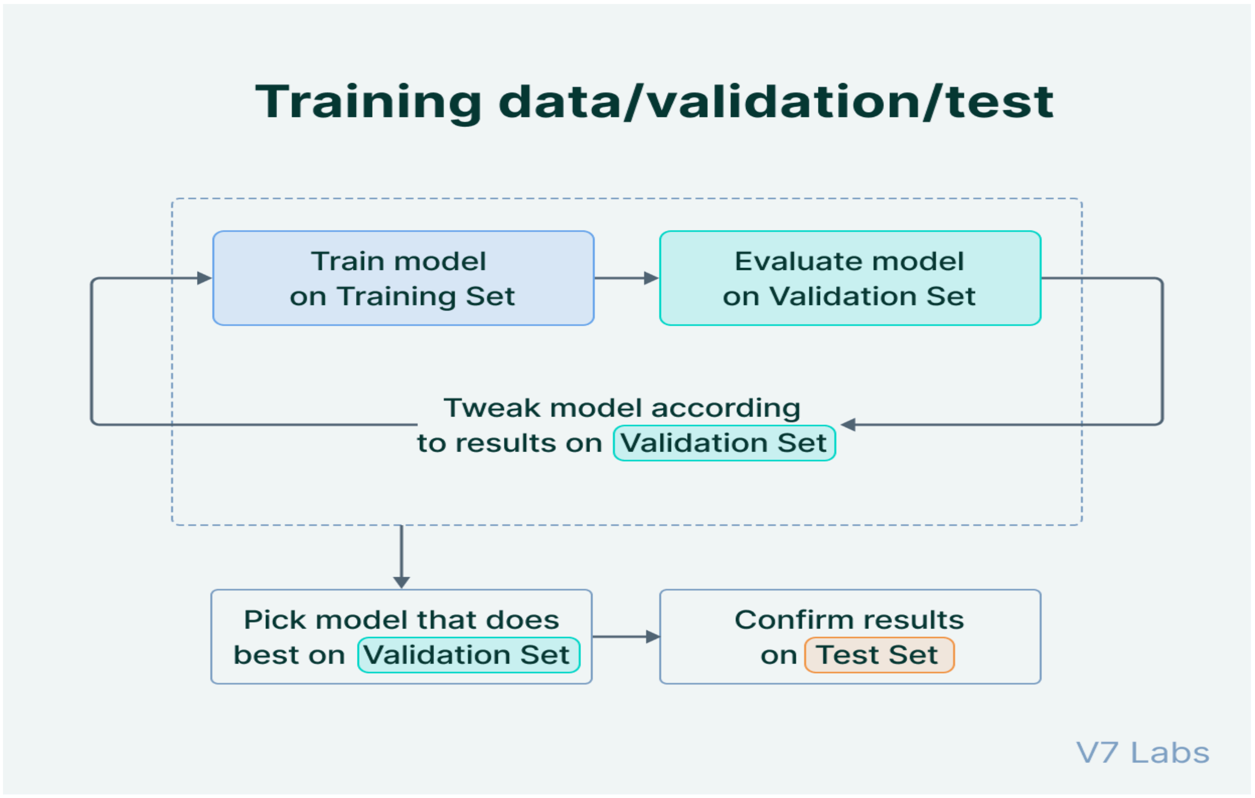
Another fact which is evident from the visualization of the available data distribution is that the time feature is split across a large range that is from 0 to (48*60*60). As the time feature is simple a measurement of time of transaction with respect to the time elapsed in that 2 day period in which the transactions are being recorded and as most of the transactions are simply non anomalous cases there is a high possibility of this attribute to contribute to the cause of forming certain biased judgements and hence we are removing that feature.

And thats it we are done with the pre processing and the feature selection step.

1.2 Dividing the dataset into training and testing sets

As a generic rule its is recommended to not use *the same dataset* for model training and model evaluation.If we wish to build a *reliable* machine learning model, then the data set it to be split into the training, validation, and test sets[17]

For training and testing purposes of our model, we should have our data broken down into three distinct dataset splits.


Training data/validation/test
Train model on Training Set
Evaluate model on Validation Set
Tweak model according to results on Validation Set
Pick model that does best on Validation Set
Confirm results on Test Set
V7 Labs

The training set:

It is the set of data that is used to train and make the model learn the hidden features/patterns in the data.

The validation set:

The validation set is a set of data, separate from the training set, that is used to validate our model performance during training.

The testing set:

The test set is a separate set of data used to test the model *after* completing the training.

It provides an unbiased final model performance metric in terms of accuracy, precision, etc. To put it simply, it answers the question of "*How well does the model perform?*"

In out study we will not be creating a validation set as such since it will even subdivide our already extremely imbalanced dataset into tinier fractions and also due to the fact that our end goal is to study the performances and not to select and evaluate the performance of the best performing model. We will be plotting the Precision-Recall Curve and evaluating the Area Under the Precision-Recall Curve as implemented by sklearn.metrics[19] against the testing set as our goal is to study the supervised and unsupervised learning methods under both balanced and imbalanced datasets .

First we shuffle the existing rows around in order to completely randomize the distribution. The pandas sample function has a random_state parameter which we will utilize here to ensure the reproducibility of the output. After this step we divide the dataset into test and train sets . We will not be keeping a dedicated validation set to first validate the performance on each model and then take the better performing one and put it against the testing set as our end goal is only to assess the performance of each model under different scenarios. Hence the dataset will be split as train and test.

```
train, test= new_df[:240000], new_df[240000:]
```

The distribution of data in the sets are as follows

Name of the set	# of non anomalous records	# of anomalous records
train	239584	416
test	44731	76

We convert these train and test sets into numpy arrays and they show 31 columns available for each of them of which the first 30 are going to be the predictive variables and the last one is going to be the output or the class. Keeping this in mind we split them as mentioned above into x_train, y_train and x_test, y_test .

2. Artificially create a balanced dataset combining oversampling and undersampling techniques

Imbalanced data is the term that we use to denote a dataset which has a characteristic ratio of a high number of negative class observations to a very small number of positive class observations. This makes it often trickier to come up with the right model to better predict the positive class observations as the models tend to ignore the minority positive class observations due to seeing the positive class observations more times in the data distribution than they come across the minority class observations.

In such cases we can't take a models accuracy as the determining factor as the accuracy score will be based on how many times the model predicted the non anomalous class observations in the testing set as a non anomalous observation and due to the number of non anomalous observations being numerically greater than anomalous observations in the dataset even if the model predicts wrong in most cases as non anomalous it would technically be correct and this accounts for the high accuracy[14].

To tackle this, we need to balance the dataset and for that we will be using the random undersampler implemented by imblearn.under_sampling[20] and random oversampler as implemented by imblearn.over_sampling[21] . After a lot of trial and error the optimum undersampling and oversampling sample ratios were decided around the following points

- Removing a large portion of the majority class through undersampling of the dataset could mean removing a huge chunk of the normal pattern in which the users do the transaction which the models could use to identify the normal scenarios leading to a greater number of false positives[6].
- Oversampling the minority class to the same number as the majority class would also mean that the models will have a difficult time figuring out the normal pattern of operations as well as the anomalous ones, this would mean that the model would have high false positives as well as false negatives[7].

Keeping these points in mind, it was decided to do the oversampling of the minority class to 4% of the number of positive observations and undersample the majority class by 3% with respect to the number of observations of the minority class.

Now one important thing to keep in mind here is that we will be doing the balancing only on the training set of the data only. The reason for taking this decision is that the real world data on which the model will be required to make predictions on will never be balanced most of the time. To get an idea of how well the model would perform on real world data we need to put it against the test against data that follows the real world distributions. Another factor is that during balancing events such as over sampling of the minority class is that data points with really minute differences between them could end up existing in both train and test sets. This would mean that the model will have an easy time doing the prediction on the testing set as it had seen some of the testing data points earlier while training.

3. Supervised method for Anomaly detection

Supervised learning is a machine learning methodology used in various fields such as finance, healthcare, cybersecurity, marketing and many more. In this type of learning methodology the model is trained on a labeled dataset. A dataset is said to be labeled when it has the tags or classes named to make the class distribution easier to understand for us as well as the model. The labeled dataset used in supervised learning consists of input features and corresponding output labels. The input features are the attributes or characteristics of the data that are used to make predictions, while the output labels are the desired outcomes or targets that the algorithm tries to predict.

The typical approach here is to split the data to training, testing and evaluation sets and use the training set to train the model and make the model predict with the unseen testing set and see the results

Supervised learning is divided into two main categories: regression and classification. In the case of regression the algorithm learns to predict a continuous output value such as the price of a house or the temprature of a particular day. In classification , the algorithm learns to predict a categorical output variable or class label such as whether a scenario of anomalous or not [8].

There are many supervised methods such as the Logistic regression (LR), Decision Tree (DT), Random Forest (RF), Gradient Boosted Tree (GBT), Naive Bayers (NB), Support Vector Machine (SVM), MultiLayer Perceptron (MLP).

Under many previous studies done in this field under similar conditions it was observed that the Support Vector Machine outperformed almost every other supervised learning algorithm available[9].

This is mainly due to the fact that SVMs were originally designed for binary classification, and they excel in such scenarios. When dealing with a strictly binary classification problem, SVMs can be particularly efficient and effective[10].

Hence we will be going forward with the SVM algorithm as implemented by sklearn.svm[16].

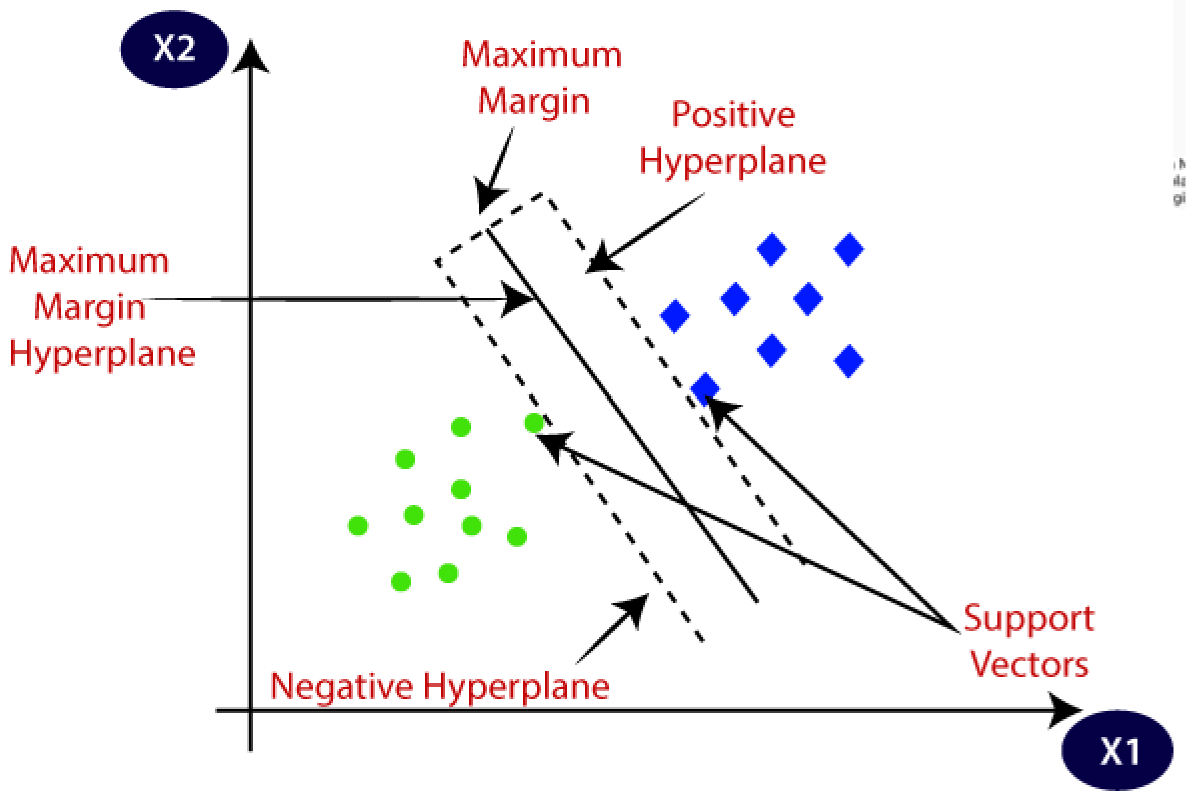
.

Support Vector Machine (SVM)

One of the most well-liked supervised learning algorithms, Support Vector Machine, or SVM, is used to solve Classification and Regression problems. However, it is primarily utilized for Machine Learning Classification problems.

The SVM algorithm's objective is to establish the best line or decision boundary that can divide n-dimensional space into classes, allowing us to quickly classify newly acquired information in the future. A hyperplane is the name given to this optimal decision boundary.

SVM selects the extreme vectors and points that aid in the creation of the hyperplane. Support vectors, which are used to represent these extreme instances, form the basis for the SVM method. Consider the diagram below, where a decision boundary or hyperplane is used to categorize two distinct categories:

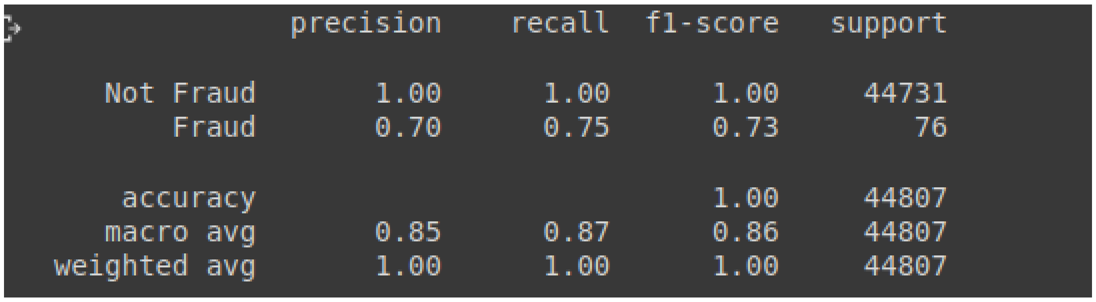


SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier[22].

3.1 Support Vector Machine on unbalanced dataset

The SVM algorithm was initialized with the random state parameter set to ensure that the results are re-creatable.
Then it was trained on the x_train, y_train training set we prepared earlier and was tested on the testing set. The classification report for SVM shows the following results



```
              precision    recall  f1-score   support

   Not Fraud       1.00      1.00      1.00     44731
       Fraud       0.70      0.75      0.73        76

    accuracy                           1.00     44807
   macro avg       0.85      0.87      0.86     44807
weighted avg       1.00      1.00      1.00     44807
```

Here is a breakdown of the metrics and what they indicate

- **Over all Accuracy**:The accuracy of nearly 100% is not of important to us in this particular scenario as the reason for the accuracy being so high is due to the imbalance of the dataset and the model repeatedly seeing non anomalous values greater number of times than the anomalous values, even if it were to blindly predict the observations to be non anomalous most of the time, it would technically still be correct in more than 99% of times.

Accuracy:1.0 = Total Number of Predictions / Number of Correct Predictions

- **Precision for "Fraud" Class**:This means that when the model predicts an instance as "Fraud," about 70% of those predictions are correct. The higher precision compared to previous cases indicates that the model is better at avoiding false positives (instances that are predicted as "Fraud" but are actually "Not Fraud").

  Precision:.70 = TP / TP + FP.

- **Recall for "Fraud" Class :** Recall, also known as sensitivity or true positive rate, indicates that the model is able to identify around 75% of the actual "Fraud" instances present in the dataset. This is a relatively good recall, showing that the model is sensitive to detecting the "Fraud" class instances.

  Recall: 0.75 =TP / TP + FN

- **F1-score for "Fraud" Class**:The F1-score is the harmonic mean of precision and recall. A value of 0.73 indicates a relatively good balance between precision and recall for the "Fraud" class.

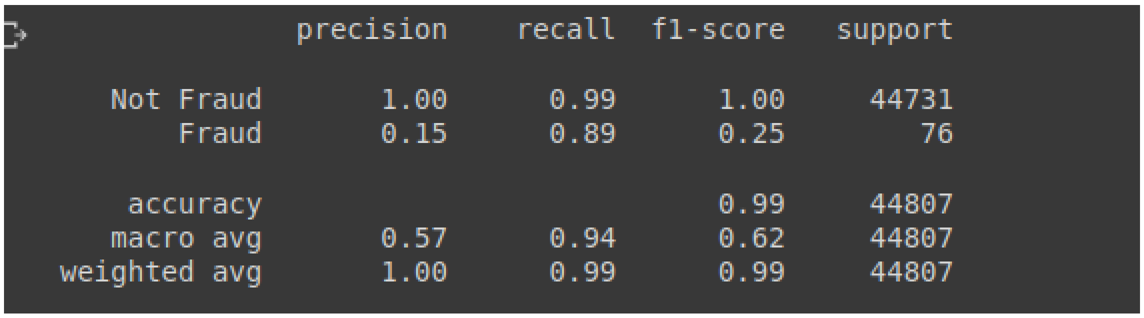
  F1-score: 0.73 = 2 * (Precision * Recall) / (Precision + Recall)

In summary, the SVM model demonstrates good precision, recall, and F1-score for the "Fraud" class on this imbalanced dataset.

3.2 Support Vector Machine on balanced dataset

With the initialization of the algorithm done with the random state,
this time the SVM is trained with the balanced training set and the prediction will be made on the un balanced testing set we made earlier.

The classification report has some rather surprising results for us



```
              precision    recall  f1-score   support

   Not Fraud       1.00      0.99      1.00     44731
       Fraud       0.15      0.89      0.25        76

    accuracy                           0.99     44807
   macro avg       0.57      0.94      0.62     44807
weighted avg       1.00      0.99      0.99     44807
```

The precision for the anomalous class has gone down significantly compared to the metrics from the non balanced dataset . This can be inferred as an increase in the number of false positives as precision = TP / TP + FP.

On the other hand the recall has improved and this means that the number of false negatives has decreased significantly Since Recall = TP / TP + FN

Here is a breakdown of the metrics and what they indicate

- **Over all Accuracy**: The overall accuracy of the model is quite high (99%). However, accuracy can be misleading on imbalanced datasets, as it can be skewed by the dominant class. In this case, the high accuracy might be driven by the accurate prediction of the majority class ("Not Fraud") rather than by effectively identifying the minority class ("Fraud").

  Accuracy: 0.99 = Total Number of Predictions / Number of Correct Predictions

- **Precision for "Fraud" Class**: This means that when the model predicts an instance as "Fraud," only about 15% of those predictions are correct. The low precision indicates that the model has a high number of false positives, where instances were predicted as "Fraud" but were actually "Not Fraud."

  Precision: 0.15 = TP / TP + FP.

- **Recall for "Fraud" Class :** Recall, also known as sensitivity or true positive rate, indicates that the model is able to identify around 89% of the actual "Fraud" instances present in the dataset. This is a relatively high recall, which means that the model is sensitive to detecting the "Fraud" class instances.

Recall: 0.89 =TP / TP + FN

- **F1-score for "Fraud" Class**:The F1-score is the harmonic mean of precision and recall. A low F1-score indicates that the model is not effectively balancing precision and recall for the "Fraud" class. In this case, it suggests that the model is achieving a compromise between precision and recall.

  F1-score: 0.25 = 2 * (Precision * Recall) / (Precision + Recall)

In summary, the SVM model achieves high recall for the "Fraud" class but at the expense of precision, leading to a low F1-score.

4. Unsupervised method for anomaly detection

Unsupervised learning is the training of a machine learning model using information which is not labeled or classified and allowing the algorithm to act on that information without any guidance[12]. Here the model groups the information according to similarities , patterns and differences without any prior training of data all by itself. The unsupervised learning is classified into two categories

- clustering : where it is possible to identify the inherent groupings in data
- Association : To form rules underlying amongst the different observations that describe a large portion of the dataset

Some of the most popular unsupervised learning methods are K-Means, KNN, Principal Component Analysis , DBSCAN etc
We will be using the Isolation Forest algorithm for this binary classification problem since Isolation Forest stands out as an exceptional choice for anomaly detection in binary distribution imbalanced datasets due to its innate ability to efficiently identify rare and unusual instances within such imbalanced distributions. In the context of imbalanced datasets, where one class significantly outnumbers the other, traditional unsupervised methods might struggle to detect the minority class anomalies effectively. Isolation Forest, with its random partitioning and isolation mechanism, excels at isolating anomalies regardless of their class distribution. By exploiting the concept of isolating anomalies through shorter paths in decision trees, Isolation Forest effectively sidesteps the issue of imbalanced classes. This unique characteristic allows it to discern anomalies more rapidly than normal instances, making it a potent tool to tackle the challenges of anomaly detection in binary imbalanced distributions where conventional techniques often fall short[11].
We are using Isolation Forest algorithm as implemented by sklearn.ensemble[15].

Isolation Forest

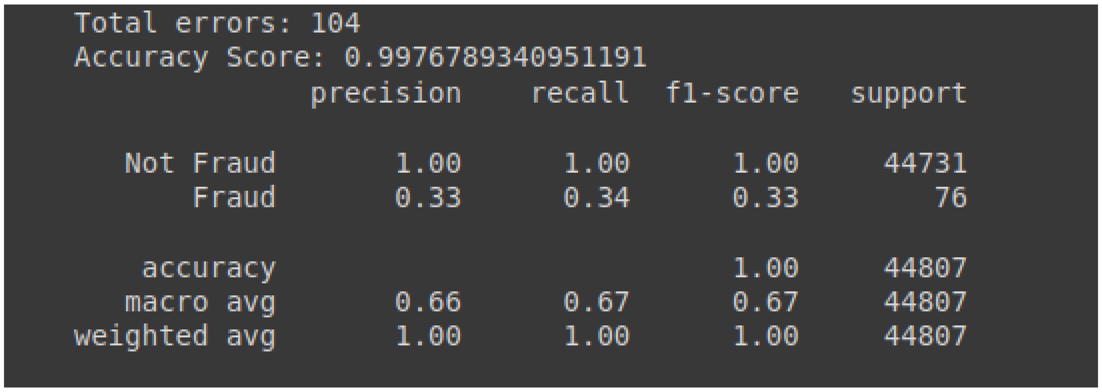
Isolation Forest is a distinctive anomaly detection algorithm that operates by harnessing the power of random partitioning and decision trees. It constructs a series of decision trees through a random selection of features and random splits, aiming to isolate anomalies as fewer instances require fewer splits to be isolated. By identifying anomalies through shorter paths in the trees, the algorithm efficiently separates rare and unusual instances from the bulk of normal data points. This mechanism inherently addresses the challenge of imbalanced datasets, where anomalies might be overshadowed by the majority class. Its capability to handle high-dimensional data, tackle both global and local anomalies, and perform well on large datasets further solidify its place as a versatile and effective tool in the realm of anomaly detection[13].

4.1 Isolation Forest on imbalanced dataset

The isolation forest algorithm is initialized with the random state : for ease of recreation of results , contamination factor:The amount of contamination of the data set, i.e. the proportion of outliers in the data set. Used when fitting to define the threshold on the scores of the samples, max samples: the length of the training set and it is fitted with the training set.
The model is fitted with the balanced training set and is tested against the untouched testing set.

The number of falsely predicted errors calculated using the number of classifications in the resultant prediction data that does not match with the testing set are displayed as Total Errors.

The classification report shows the following details of the models performance.



```
Total errors: 104
Accuracy Score: 0.9976789340951191
              precision    recall  f1-score   support

   Not Fraud       1.00      1.00      1.00     44731
       Fraud       0.33      0.34      0.33        76

    accuracy                           1.00     44807
   macro avg       0.66      0.67      0.67     44807
weighted avg       1.00      1.00      1.00     44807
```

Here's a breakdown of the metrics and what they indicate:

- **Accuracy Score**: The overall accuracy of the model is 0.9977, which might seem very high. However, accuracy can be misleading in highly imbalanced datasets. Since the majority class (Class 0, "Not Fraud") dominates the dataset, a high accuracy can be achieved by simply classifying all instances as the majority class. In such cases, accuracy alone is not a reliable metric for model performance.

  Accuracy: 0.97 = Total Number of Predictions / Number of Correct Predictions

- **Precision and Recall for "Fraud" Class**: The precision for the "Fraud" class is 0.33, and the recall is 0.34. Precision in this context means that when the model predicts an instance as "Fraud," only about 33% of those predictions are correct. Recall indicates that the model is able to identify around 34% of the actual "Fraud" instances present in the dataset

  Precision:0.33 = TP / TP + FP.

  Recall: 0.34 =TP / TP + FN

- **F1-score for "Fraud" Class**: The F1-score is 0.33, which is the harmonic mean of precision and recall. It considers both false positives (precision) and false negatives (recall). A low F1-score indicates that the model is not effectively balancing precision and recall for the "Fraud" class.

  F1-score: 0.33 = 2 * (Precision * Recall) / (Precision + Recall)

- **Total Errors**:This is the sum of false positives and false negatives, indicating the total number of incorrect predictions made by the model.

In summary, the Isolation Forest model struggles with the imbalanced nature of the dataset. It achieves high accuracy by correctly predicting the majority class, but its performance on the minority "Fraud" class is poor. This is a common issue with imbalanced datasets, as algorithms often favor the majority class due to its prevalence.

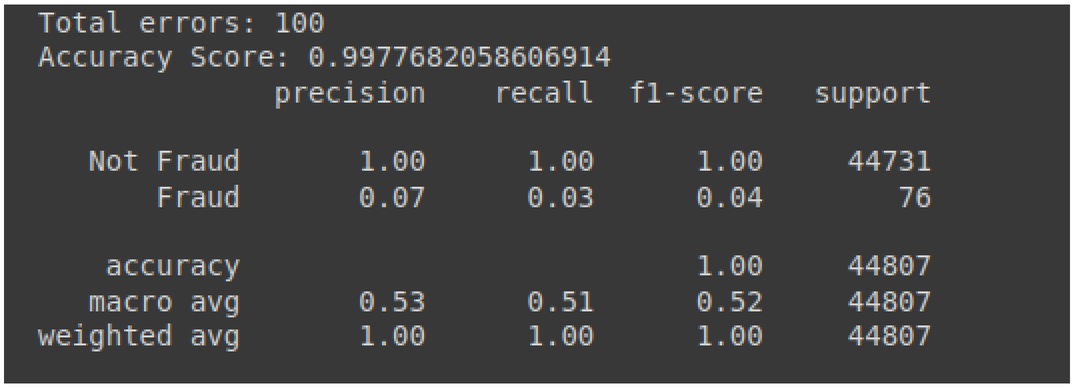
4.1 Isolation Forest on balanced dataset

Here also the model is initialized with the parameters, random state : to ensure ease of recreation of results contamination : The amount of contamination of the data set, i.e. the proportion of outliers in the data set. Used when fitting to define the threshold on the scores of the samples, max samples: the length of the training set and it is fitted with the training set.

Then the balanced training set is fitted to the model and prediction is made on the untouched testing set.

The number of falsely predicted errors calculated using the number of classifications in the resultant prediction data that does not match with the testing set are displayed as Total Errors.

The classification report shows the following details of the models performance



```
Total errors: 100
Accuracy Score: 0.9977682058606914
              precision    recall  f1-score   support

   Not Fraud       1.00      1.00      1.00     44731
       Fraud       0.07      0.03      0.04        76

    accuracy                           1.00     44807
   macro avg       0.53      0.51      0.52     44807
weighted avg       1.00      1.00      1.00     44807
```

Here's a breakdown of the metrics and what they indicate:

- **Accuracy**:The accuracy of 0.9978 might seem very high. However, accuracy can be misleading in highly imbalanced datasets. Since the majority class (Class 0, "Not Fraud") dominates the dataset, a high accuracy can be achieved by simply classifying all instances as the majority class.

  Accuracy: 0.9978 = Total Number of Predictions / Number of Correct Predictions

- **Precision for "Fraud" Class**:The precision score of 0.07 indicates that when the model predicts an instance as "Fraud," only around 7% of those predictions are actually correct.

  Precision: 0.07 = TP / TP + FP.

- **Recall for "Fraud" Class**:The recall score of 0.03 means that the model is able to identify only about 3% of the actual "Fraud" instances present in the dataset.

  Recall: 0.03 = TP / TP + FN

- **F1-score for "Fraud" Class**:The F1-score is the harmonic mean of precision and recall. A value of 0.04 indicates a low balance between precision and recall for the "Fraud" class.

  F1-score: 0.04 = 2 * (Precision * Recall) / (Precision + Recall)

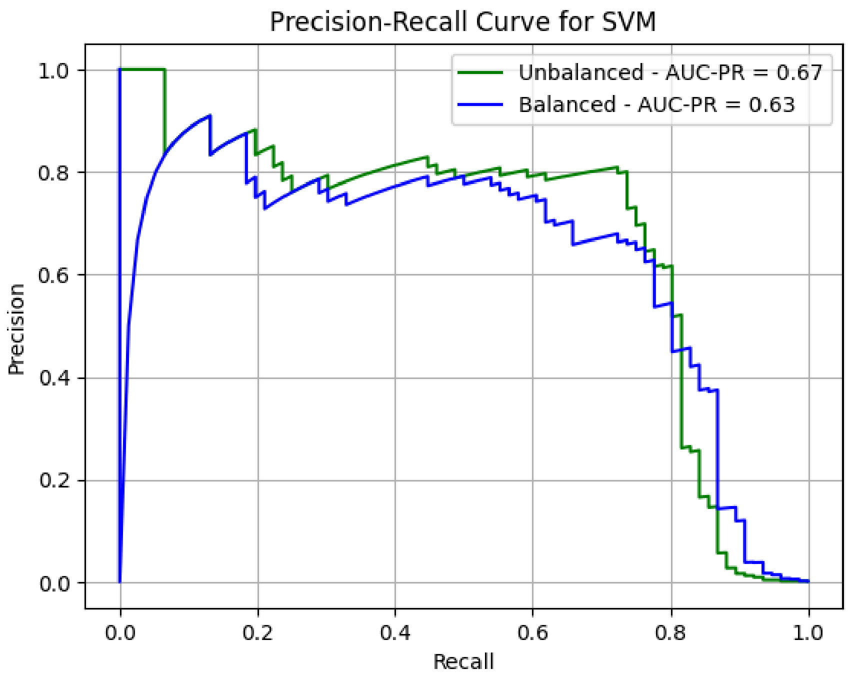
- **Total Errors**:This is the sum of false positives and false negatives, indicating the total number of incorrect predictions made by the model.

In summary, the Isolation Forest model exhibits low precision and recall for the "Fraud" class, resulting in a low F1-score. The accuracy is relatively high, but the total number of errors suggests room for improvement.

5. Analysis of the results

In this section of the study, we are comparing the performance of both supervised and unsupervised learning methods. This will be done in a systematic manner considering the relevant metrics of the classification report and most importantly the Precision Recall Curve and the area under the precision recall curve as the kaggle dataset team themselves recommend using this method due to the high imbalance of the dataset.

5.1 Comprehensive performance analysis of the supervised methods



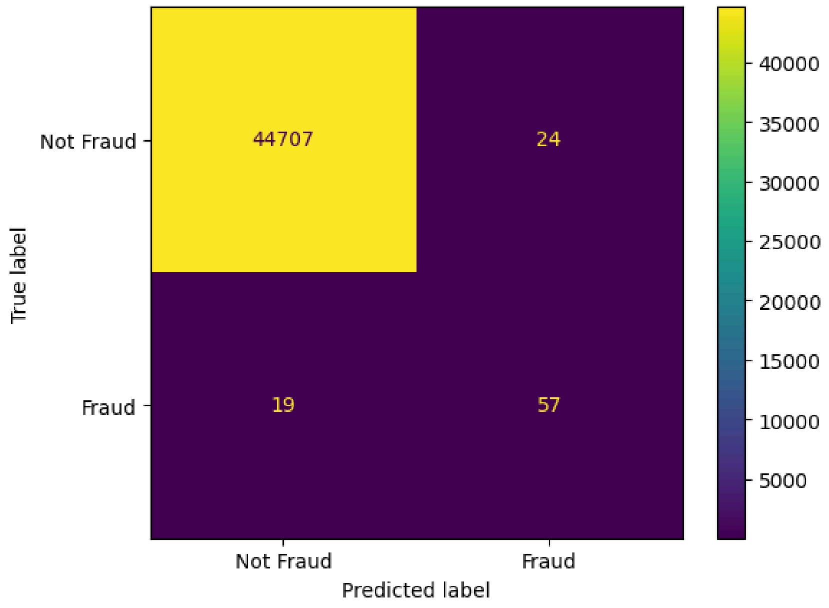
Unbalanced SVM Average Precision : 0.6766981994391814

Unbalanced SVM Area Under the Precision-Recall Curve (AUPRC) : 0.6742613630662013

Balanced SVM Average Precision : 0.6390975858055421

Balanced SVM Area Under the Precision-Recall Curve (AUPRC) : 0.6308534164967172

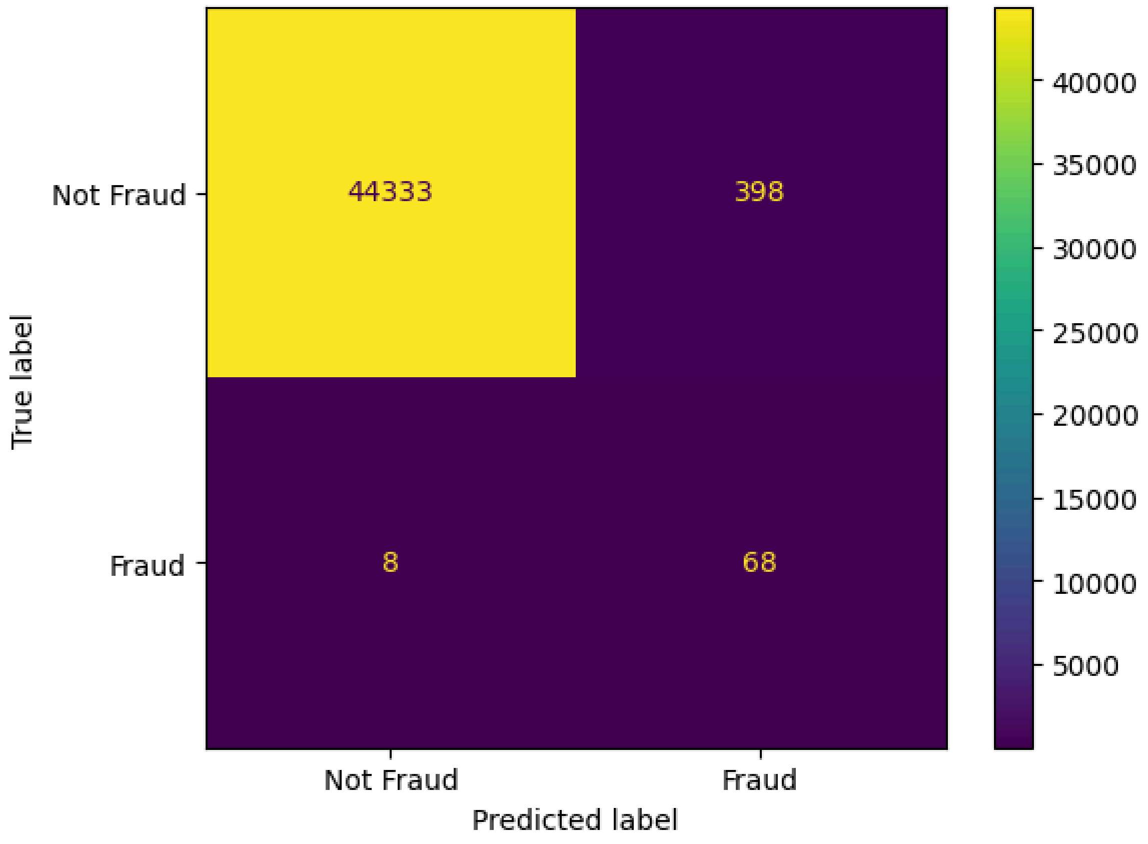
In the case of the **unbalanced dataset**, the SVM algorithm shows impressive results in terms of its precision recall and F1 score for the anomalous class. The confusion matrix shows that the model is effective at identifying the instances of the anomalous classes in the imbalanced data. But one important thing that needs to be considered here is the potential available impact of overfitting. The model shows almost a 100% accuracy which is due to the highly skewed nature of the dataset itself. Such a high accuracy indicates that the model is in fact fitting a lot of noise and giving more attention to the dominant class. This ultimately leads to overestimating and the models generalization ability to predict on new , yet unseen data.



The Precision Recall curve is not optimal but it shows that its almost able to extend somewhat into the top right corner which shows that the model is high performing.

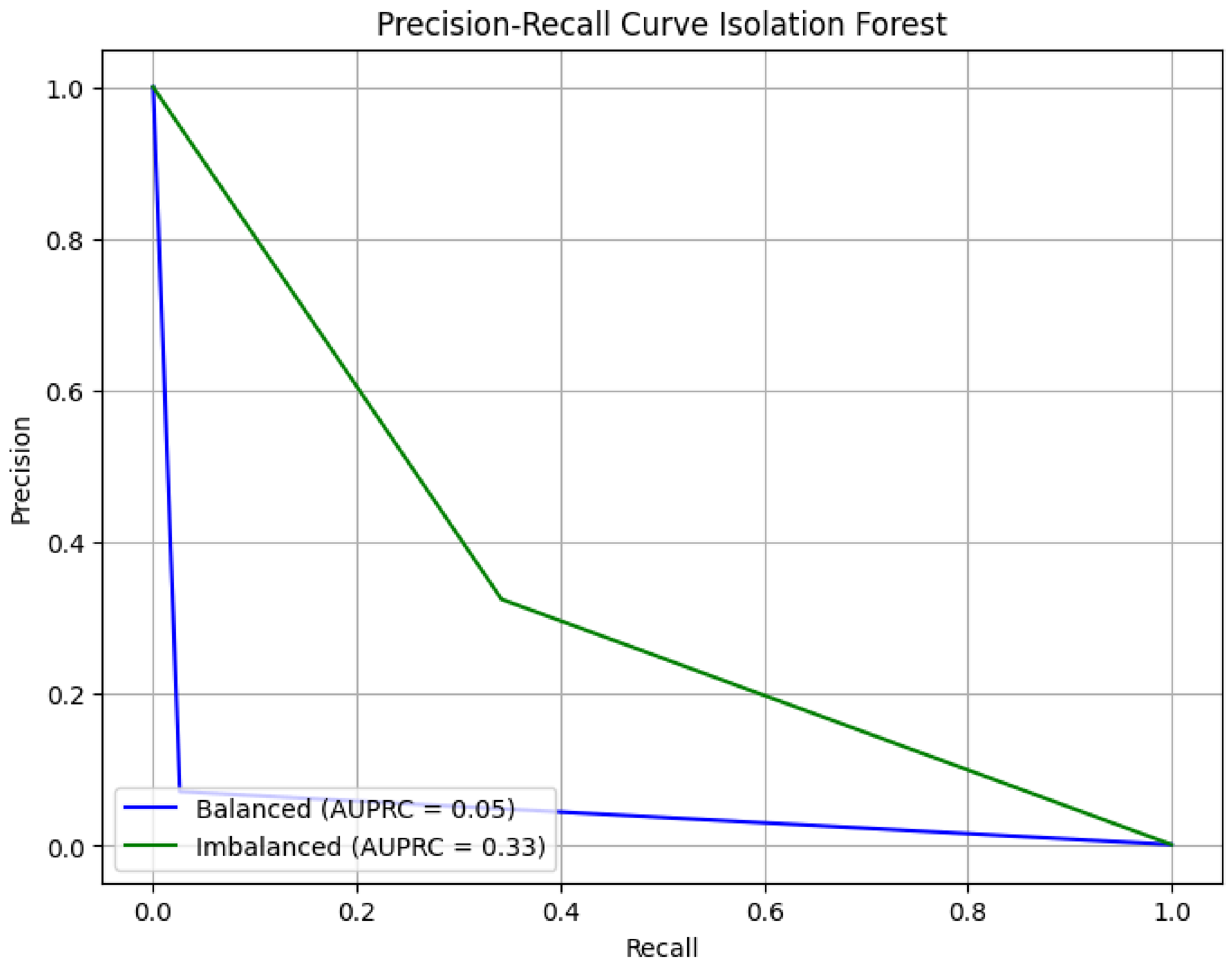
The average area under the precision recall curve is 0.67 which again is not optimal but relatively better as we will see in case of the balanced dataset.

the **Balanced Dataset** analysis reveals a more complex situation. The model achieves a high overall accuracy of 99%, which can be deceptive due to the class imbalance. The higher recall (0.89) suggests the model's sensitivity in detecting "Fraud" instances, but the trade-off comes in the form of low precision (0.15). This low precision implies that the model's propensity for false positives has increased, indicating potential overfitting on the balanced data. This aligns with the low F1-score (0.25), which indicates that the model's precision and recall are not well-balanced. The lower average precision and area under the precision-recall curve (AUPRC) also suggest that the model might not perform as well on unseen data.



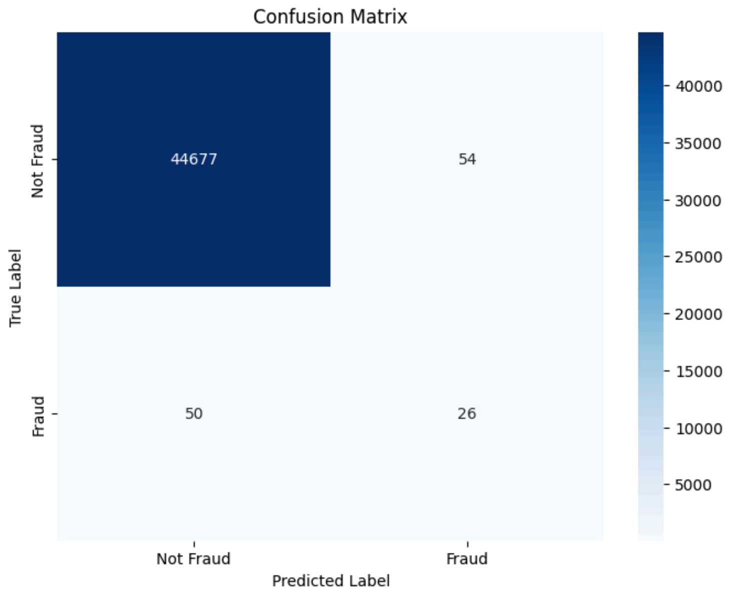
In both scenarios, the metrics hint at the possibility of overfitting. The unbalanced dataset's high accuracy and the balanced dataset's high recall and low precision can be signs of overfitting to the specific characteristics of each dataset. Overfitting could result in poor generalization to new data and ultimately lead to disappointing real-world performance.

5.2 Comprehensive performance analysis of the unsupervised methods

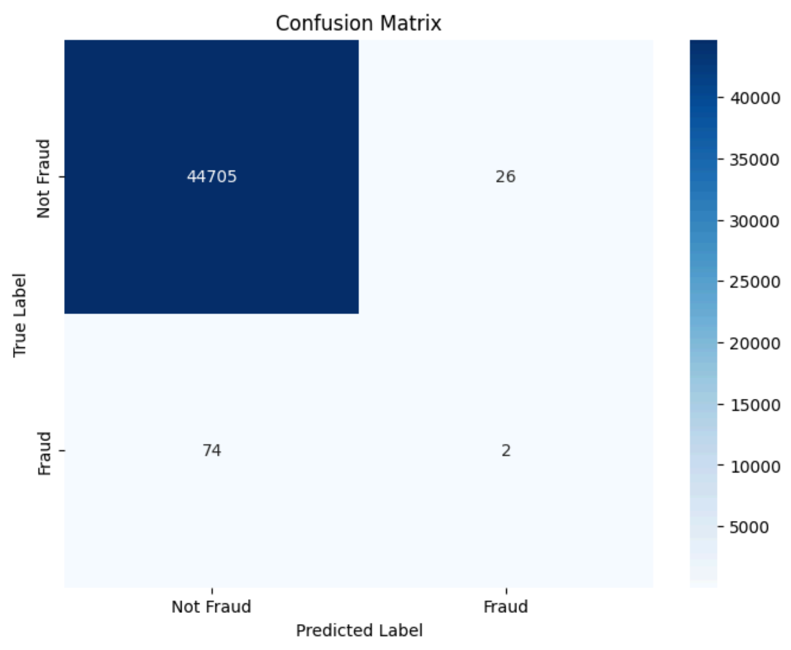


Area under the Precision-Recall Curve (AUPRC) - Balanced: 0.049697944282672
Area under the Precision-Recall Curve (AUPRC) - Imbalanced: 0.33411058011377454

The first scenario involves applying the Isolation Forest algorithm on an imbalanced dataset. The algorithm's performance is characterized by an impressive accuracy of 99.77%. This high accuracy, however, is somewhat deceptive in the context of an imbalanced dataset. The majority class ("Not Fraud") heavily dominates the data, which allows the algorithm to achieve this accuracy by simply predicting the majority class for most instances. This outcome emphasizes that accuracy, although a commonly used metric, can be misleading when faced with imbalanced data. The precision for the "Fraud" class is merely 0.33, indicating that when the model labels an instance as "Fraud," it's correct only about 33% of the time. Similarly, the recall score of 0.34 implies that the algorithm can only identify around 34% of actual "Fraud" instances. This scenario underscores the challenges faced by algorithms in such situations, where they tend to favor the majority class due to its prevalence. Despite the high accuracy, the Isolation Forest model's performance on the minority class remains suboptimal.



In the second scenario, the Isolation Forest algorithm is evaluated on a balanced dataset. Once again, the accuracy appears strikingly high at 99.78%. As with the imbalanced dataset, this high accuracy can be misleading given the dominance of the majority class. The precision and recall for the "Fraud" class, however, reveal a stark contrast to the first scenario. The precision is just 0.07, indicating that when the model predicts "Fraud," it's accurate only about 7% of the time. Similarly, the recall score of 0.03 highlights the model's limited ability to identify actual "Fraud" instances, covering just 3% of them. This outcome, although expected in an imbalanced context, highlights that even with a balanced dataset, the Isolation Forest model still struggles to effectively address the minority class. The F1-score of 0.04 underscores the model's inability to strike a balance between precision and recall for the "Fraud" class. Despite the seemingly high accuracy, the model's performance on the minority class is indicative of its limitations.



Both scenarios shed light on the Isolation Forest algorithm's behavior in the presence of imbalanced and balanced data. While it's evident that the algorithm can achieve high accuracy by capitalizing on the majority class, it struggles to perform well on the minority class. This suggests that the model is not inherently adept at identifying anomalies or outliers, which are often underrepresented in imbalanced datasets. The algorithm's precision-recall trade-off is pronounced, with its inability to strike a balance between correctly classifying anomalies and minimizing false positives. Despite the differences in dataset balance, the core challenge remains: effectively detecting anomalies while avoiding false positives.

Isolation Forest, as an ensemble-based anomaly detection algorithm, is designed to identify anomalies in a dataset by isolating them in a tree-based structure. However, it may not be the most suitable approach for addressing imbalanced or minority-class detection challenges. In such cases, clustering techniques, such as DBSCAN or Local Outlier Factor (LOF), could be more effective. Clustering methods identify anomalies based on their spatial distribution and density, enabling the model to capture the nuances of minority-class instances without being dominated by the majority class. These techniques can potentially enhance the Isolation Forest's weaknesses and offer more accurate anomaly detection, particularly when dealing with imbalanced datasets.

6. Comparative analysis of the algorithms chosen

Supervised Method - Support Vector Machine: The utilization of a supervised approach like the Support Vector Machine (SVM) offers distinct advantages in certain scenarios. One of its primary strengths is its ability to learn from labeled data, allowing it to make accurate predictions and classifications based on known patterns. This is particularly advantageous when a significant amount of labeled data is available for training, enabling the model to capture complex relationships within the data. The SVM's decision boundary is tailored to optimize generalization, which makes it effective in scenarios where the underlying distribution of the data is well-defined and there is a clear distinction between classes. Additionally, SVMs provide a level of interpretability through the support vectors, which can aid in understanding the model's decision-making process.

However, the supervised nature of SVMs introduces certain limitations. They heavily depend on the quality and representativeness of the labeled data, which can be a challenge when dealing with imbalanced datasets. The model's performance may suffer when the minority class is inadequately represented, leading to biased predictions and reduced accuracy. Furthermore, SVMs can be computationally intensive, especially when handling large datasets, as they involve solving complex optimization problems. This can impact the model's scalability and efficiency.

Unsupervised Method - Isolation Forest: Unsupervised methods like the Isolation Forest approach the problem from a different angle. They excel in scenarios where the underlying distribution is less defined and anomalies are the focus. Isolation Forest leverages the concept of isolating anomalies as a means of detection, making it effective when identifying rare instances that deviate from the norm. This approach doesn't require labeled data, which is advantageous when such data is scarce or expensive to acquire. Additionally, the method's ability to handle high-dimensional data and its relative insensitivity to outliers contribute to its robustness.

However, the reliance on unsupervised learning comes with its own set of trade-offs. The lack of labeled data means that the model might struggle to distinguish between anomalies and legitimate variations in the data, leading to false positives and negatives. Isolation Forest may struggle when the anomaly patterns are subtle or complex, as it works best with more evident deviations. The interpretability of unsupervised methods can be challenging, as their decisions are driven by intrinsic data properties rather than explicit class labels. This can limit the understanding of the underlying reasons for anomaly detection, especially when dealing with intricate datasets.

Conclusion

Anomaly detection is becoming an increasingly relevant task in various domains. In the age of the digital era where billions of transactions are taking place every day even the most sophisticated state of the art systems fail to prevent anomalous operations carried out by individuals due to the sheer volume of the data being generated in real time. The challenge with this real time data is that only a tiny strata of the data will contain these anomalous events and the model trained to look for these anomalous patterns needs to be extremely high performing with a high degree of recall measure as we have seen in our study. Other metrics such as the accuracy of the model can be even considered obsolete under this scenario as we should prioritize preventing an anomalous event from happening over a non anomalous event being marked as anomalous. Even if a non anomalous event is marked as anomalous by the model in real time data we have a large number of options available to check whether this is a true positive anomaly or not such as aligning with the user just like for example how email authentications such as email detect the user login in from a new ip and confirming with the user whether to proceed or not etc. In out study we saw that the supervised method Support Vector Machine outperformed the unsupervised method of Isolation Forest in almost all the metrics. In the study it was observed that the supervised method performed exceptionally well on the balanced data if we consider the recall value as the decisive factor for reasons mentioned earlier. There are lots of rooms for improvement in the study done , we could consider using techniques such as SMOTE-ENN for making a carefully balanced dataset and further perform the studies with other learning methods also. For future studies it can also be considered to look for a more recent dataset and consider performing the studies on it as the number of non anomalous transactions and anomalous events occurring in them in event a 4 hour period will be significantly higher in number than that of the time period from which this dataset was created.

References

1. Ahmed, M., Mahmood, A.N., Islam, M.R.: A survey of anomaly detection techniques in financial domain. Future Generation Computer Systems 55, 278–288 (2016)
2. Kumar, V.: Parallel and distributed computing for cybersecurity. IEEE Distributed Systems Online 6(10) (2005).
3. https://scikit-learn.org/stable/tutorial/basic/tutorial.html
4. https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data
5. https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
6. Japkowicz, N.: Class imbalance: are we focussing on the right issue?, In proc. International workshop on learning from imbalanced data-sets II, (2003)
7. https://ieeexplore.ieee.org/abstract/document/9078901/
8. https://www.geeksforgeeks.org/ml-types-learning-supervised-learning/
9. https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-019-1004-8/tables/4
10. https://dspace.mit.edu/handle/1721.1/7290
11. https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/icdm08b.pdf
12. https://www.ibm.com/blog/supervised-vs-unsupervised-learning/
13. Liu, Fei Tony, Ting, Kai Ming and Zhou, Zhi-Hua. "Isolation-based anomaly detection." ACM Transactions on Knowledge Discovery from Data (TKDD) 6.1 (2012)
14. Matos, L.M., Cortez, P., Mendes, R., Moreau, A.: A comparison of data-driven approaches for mobile marketing user conversion prediction. In: Jardim-Gonçalves, R., Mendonça, J.P., Jotsov, V., Marques, M., Martins, J., Bierwolf, R.E. (eds.) 9th IEE International Conference on Intelligent Systems, IS 2018, Funchal, Madeira, Portugal, September 25-27, 2018. pp.140–146. IEEE (2018).
15. https://scikit-learn.org/stable/modules/ensemble.html
16. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
17. https://www.v7labs.com/blog/train-validation-test-set
18. https://machinelearningmastery.com/robust-scaler-transforms-for-machine-learning/
19. https://scikit-learn.org/stable/modules/model_evaluation.html
20. https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling.RandomUnderSampler.html
21. https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.RandomOverSampler.html
22. https://serokell.io/blog/support-vector-machine-algorithm