Comparison of Various Machine Learning Techniques Based on Variable Selection under Imbalanced Data

1Mo Ahsan Ahmad, 2Umme Kulsum, 3Faizan Ansari, 4Mo Nafees, 5Ravindranath Sawane
1Student, 2Student, 3Student, 4Student, 5Student
1Bansal Institute of Science and Technology, Bhopal, MP,
2Aligarh Muslim University, Aligarh, UP,
3Jamia Millia Islamia, New Delhi,
4Rizvi College of Engineering, Mumbai,
5G H Raisoni Institute of Engineering and Technology, Nagpur

Abstract - **Classification in an imbalanced dataset is one of the challenges in statistical learning because many algorithms are designed to optimize overall accuracy without considering the relative class distribution. These algorithms are biased towards the majority class and tend to ignore the minority class, which is the class of interest for experimenters. This paper reviewed the data-level approach, algorithmic-level approach, and performance evaluation metrics for the classification of imbalanced data. Oversampling, undersampling, and hybrid sampling techniques are discussed along with Decision Trees Classifier, K-NN, Naive Bayes, Logistic Regression, Support Vector Machines, and Random Forest Classifier. The effectiveness of data-level approaches combined with algorithmic-level approaches to improve classification performance and compare the various machine learning classification algorithms of a "credit card fraud detection" dataset. Random Forest Classifier achieved the highest F-score and lowest log loss with and without resampling techniques.**

keywords - **Imbalanced data, Variable Selection, Oversampling, Undersampling, Supervised Learning, Credit Card Fraud Detection**

1. INTRODUCTION

An imbalanced classification problem is an example of a classification problem in which the distribution of examples across known classes is either biased or skewed. The distribution can vary from a slight bias to a severe imbalance where there is one example in the minority class and hundreds, thousands, or millions of examples in the majority class or classes. In other words, an imbalanced class distribution across when the data are highly skewed as samples from one (majority) class are higher in number than the other (minority) class, thus making it an imbalanced data set. Imbalanced class datasets are common in practical classification scenarios. However, this assumption is not true in many real-world applications. The differences between different misclassification errors can be large. For example, in the medical diagnosis of certain cancer, if the cancer is regarded as the positive class, and non-cancer (healthy) as negative, then missing cancer (the patient is actually positive but is classified as negative; thus, it is also called "false negative") is much more serious (thus expensive) than the false-positive error. The patient could lose his/her life because of a delay in the correct diagnosis and treatment. Similarly, if carrying a bomb is positive, then it is much more expensive to miss a terrorist who carries a bomb to a flight than to search for an innocent person. The imbalanced dataset problem occurs in many real-world applications such as text categorization, fault detection, fraud detection, oil-spill detection in satellite images, toxicology, cultural modeling, and medical diagnosis [1].

Such an error in diagnosis cause stress and further complications for the patient. Physicians cannot afford any incorrect diagnosis because this could severely affect the well-being of the patients and even change the course of the available treatment and medications. Therefore, it is crucial that a classification model be able to achieve a higher identification rate for the minority class in the datasets [2]. Many research papers on imbalanced data sets have agreed that because of this unequal class distribution, the performance of the existing classifiers tends to be biased toward the majority class. The reasons for the poor performance of the existing classification algorithms on imbalanced datasets are as follows: i) They are accuracy driven that is, their goal is to minimize the overall error to which the minority class contributes very little, and ii) They assume that there is equal distribution of data for all the classes [3] Classification problems are quite common in the machine learning world and in the classification problem we attempt to predict the class label by studying the input data or predictor where the target or response variable is a categorical variable in nature. The main goal of the classification models is to identify the category of a given dataset furthermore, after using feature selection and then resampling the dataset with the help of under-sampling, over-sampling, and hybrid sampling techniques and applying ML algorithms such as Random Forest Classifier, Decision Tree, and Logistic Regression to deal with imbalanced datasets. Feature selection is an important pre-processing step in machine learning and data science problems. It refers to data with a large number of features and relatively few observations. Feature selection was applied to identify the most relevant features in the data that could be used to learn the patterns of the target variable. Along

with other pre-processing steps such as data cleaning and normalization, feature selection transforms the raw data into a form that is suitable for machine learning algorithms. Technological advances over the last several years have enabled the collection of large quantities of data. The processing and analysis of data or big data have spurned rapid growth in data science techniques and methodologies [4] Feature selection has several benefits including improved computational efficiency, interpretability, and generalizability. A dataset with fewer features was computationally faster for analysis. Models that are based on fewer features are easier to interpret. Additionally, with fewer features, the model is less likely to overfit the data. Given the benefits of feature selection, it is important to develop fast and accurate algorithms for identifying relevant features in the data. In this study, we considered feature selection in the context of imbalanced class distribution. An imbalanced class distribution refers to a dataset in which instances of one class are significantly fewer than those of the other classes. In binary labeled class data, the minority and majority instances are referred to as positively and negatively labeled respectively. The issue of imbalanced data arises in the context of medical diagnostics, fraud detection, or any other situation involving rare events. Several solutions have been proposed to address the problem of feature selection in imbalanced data[5]

However, in this paper, I proposed the scikit-learn machine learning library which provides an implementation of the correlation statistic in the f_regression() function. This function can be used in a feature selection strategy, such as selecting the top k most relevant features (largest values) via the SelectKBest class. For resampling methods, the majority of the experiments are carried out using sklearn and imblearn [6] machine learning libraries in Python. The provided data set is based on transactions from European Cardholders that have been made during a two-day period in September 2013. In the data set, there are a total of 284,807 observations. There are some variables in the data set, a total of 31 variables in the data set three of which are Class, Time, and Amount. The class describes if the transaction is fraudulent or non-fraudulent. Time is the number of seconds elapsed between each transaction in the dataset. The amount is the total money spend in each transaction. The rest of the 28 variables are displayed with no description and have been transformed with Principal Component Analysis (PCA) to protect the sensitive information. It consists of transactions made by credit cards. This dataset has 492 fraud (class 1) and 284315(class 0) nonfraud transactions out of 284807 transactions. There are no missing values in the data set. The class distribution of highly uneven. There are only 0.17% of fraud transactions and 0.992% of the non-fraudulent transaction, Thus, the dataset is highly imbalanced.

2. Methods/Materials

2.1 Data Pre-processing

Data pre-processing is the technique of preparing (cleaning and organizing) the raw data to make it suitable for a building and training Machine Learning models and it improves the quality of the raw data. Raw data is often incomplete or inconsistent, containing numerous errors and/or lacking certain trends or behaviours. Thus, data pre-processing steps are taken to resolve these issues and enhance the quality of data to extract meaningful insights [7].

Data cleaning is the first step of data pre-processing which handles missing values, smooth noisy data, identifies and/or removes outliers, and resolves inconsistencies. If there are missing values in the data, techniques range from ignoring the existences of these missing values, manually filling the data, or obtaining new values by simple statistics such as average or correlations exist to handle such a situation. After cleaning an imbalanced dataset, resampling can be done to rebalance the class distribution. Resampling aims at correcting the problem of the distribution of the data set. This can be accomplished by over-sampling the minority class, under-sampling the majority class, or combining the under and over-sampling techniques in a systematic manner. These strategies allow the classifiers to receive training instances as if they belonged to a well-balanced data set. Thus, any bias of the classifier towards the majority class due to the different proportions of sample per class would be eliminated [8].

2.2 Methodology:

The proposed method is based on calculating the F1-scores of features using the machine learning algorithms i.e., Decision Tree Classifier, Random Forest Classifier etc. The specific details of the proposed feature selection algorithm are described below.

Let Y be the target class variable and $\{X\ i\}_{i=1}^{n}$ be the feature variables in a data set. For each feature $X\ i$:

1. Split the dataset (X_i, Y) into training and testing subsets according to 70/30 ratio.
2. Select the top k most relevant features via the SelectKBest class.
3. Fit the Machine Learning algorithms to the training set.
4. Calculate the predicted values $\hat{y}$ without sampling and after sampling the data of ML algorithms on the testing set.
5. alculate the F1-score using the true and predicted values y and $\hat{y}$ respectively on the testing set.
6. Comparison between different types of machine learning algorithms of their F-score.

2.3 Variable Selection:

Variable selection for the regression model is the process of identifying and selecting a subset of input(independent) variables that are most relevant to the target(dependent) variable.

Formally, $let\ (x_i, y_i),\ 1 \le i \le n,$ be pairs distributed as (X, Y) with probability measure P , where $y \in \{0, 1\}$ and $X = (x_i, \ldots, x_m)$ here X as a predictor or independent variable and **y** as a response or dependent variable and suppose we have 30 features(X), in our dataset and **y** as a response variable and it has two classes $0\ and\ 1$ then we use variable selection technique in order to we could select top k most relevant features. The scikit-learn machine learning library provides an

implementation of the correlation statistic in the f_regression() function. This function can be used in a variable selection strategy, such as selecting the top k most relevant features (largest values) via the "SelectKbest" class.

3. Sampling Methods:

Imbalanced datasets are those where there is a severe skew in the class distribution, such as 1:100 or 1:1000 etc. examples in the minority class to the majority class. Suppose we have a dataset "*credit card fraud detection*" which is a highly skewed dataset, or we could say imbalanced dataset. The dataset consists of transactions made by credit cards. This dataset has **492 fraud***(class 1)* **and 284315***(class 0)* **non fraud transactions out of 284807 transactions**. That makes it highly unbalanced, the positive class (frauds) account for 0.172% of all transactions. We can observe, this is highly skewed dataset. Then we apply some resampling techniques for resampling the imbalanced dataset into balanced dataset.

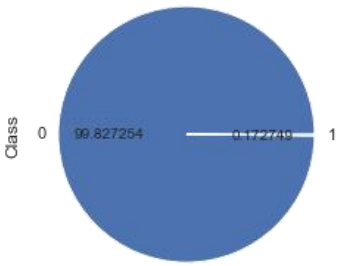

Fig1 : Original data (Imbalanced Dataset)

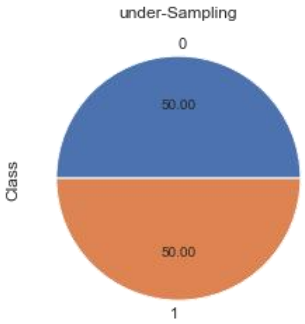

Fig 2 : After resampling the data (Balanced Dataset)

3.1 Over-Sampling

3.1.1 Random Over-Sampling

Random over-sampling (ROS) is the process of duplicating examples in the minority class to increase the size of the minority class. No new information is added in random over-sampling; therefore, it tends to overfit. Oversampling can be defined as adding more copies to the minority class. Oversampling can be a good choice when you don't have a ton of data to work with.

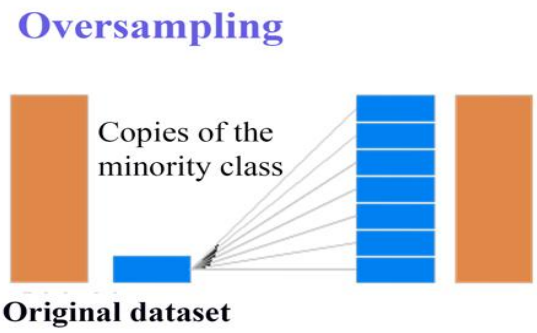

Fig 3

3.1.2 Synthetic Minority Over-sampling Techniques (SMOTE)

Smote is used to avoid overfitting of data and Chawla et al. (2002) proposed an intelligent oversampling technique called Synthetic Minority Over-sampling Technique (SMOTE) in which the minority class is over-sampled by creating synthetic examples rather than just randomly duplicating examples. this is the best approach as compared to random over-sampling to deal with imbalanced datasets. It works by randomly picking a point from the minority class and computing the k-nearest neighbors for this point and this point is called synthetic point. The synthetic points are added between the chosen points and its neighbors [9]. Andrew Estabrooks et al. proposed a multiple resampling method which selected the most appropriate re-sampling rate adaptively [10]. Taeho Jo et al. put forward a cluster-based over-sampling method which dealt with between-class imbalance and within-class imbalance simultaneously [11].

How SMOTE algorithm work!

i) Setting the minority class set A , for each $x \in A$ the **k-nearest neighbors of** x are obtained by calculating the **Euclidean distance** between x and every other sample in set

ii) Sampling rate N is set according to the imbalanced proportion. For each $x \in A$, N examples $(x_1, x_2, \ldots, x_n)$ are randomly selected from its k-nearest neighbors, and they construct the set A_1.

iii) For each example $x_k \in A_7$ (k=1, 2, 3…N), the following formula is used to generate a new example:

$$x' = x + rand(0, 1) \cdot |x - x_k|$$

where $rand(0, 1)$ represents the random number between 0 and 1.

3.2 Under-Sampling

3.2.1 Random Under-Sampling

Under-sampling balances the class distribution by decreasing the size of the majority class. This is done until the majority and minority class is balanced out. This method uses a subset of the majority class to train the classifier. Since many majority class examples are ignored, the training set becomes more balanced, and the training process becomes faster. The most common pre-processing technique is random majority under-sampling (RUS), IN RUS, Instances of the majority class are randomly discarded from the dataset. However, the main drawback of under-sampling is that potentially useful information contained in these ignored examples is neglected. There many ways attempt to improve upon the performance of random sampling, such as Tomek links, NearMiss sampling, etc.

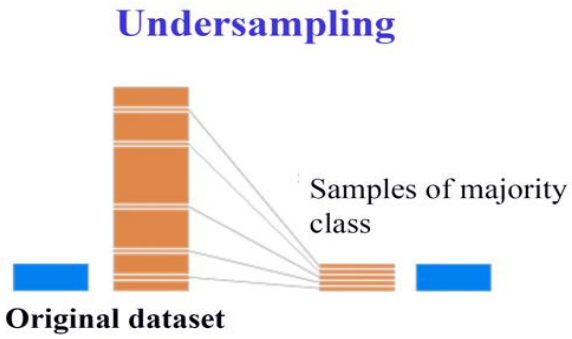

Fig 4

3.2.2 NearMiss sampling

NearMiss is an under-sampling technique. It aims to balance class distribution by randomly eliminating majority classes. When instances of two different classes are very close to each other, we remove the instances of the majority class to increase the spaces between the two classes. The method first finds the distances between all instances of the majority class and the instances of the minority class. The NearMiss which selects examples from the majority class that are close to three of the closest examples from the minority class and removes them. Random under-sampling and the NearMiss algorithm [12] are two of the most common techniques in this category. In the former the points are selected at random and in the latter the points nearest to the minority points are more likely to be selected. Here, majority class is to be under-sampled. Then **n** instances of the majority class that have the smallest distances to those in the minority class are selected. If there are k instances in the minority class, the nearest method will result in **k*n** instances of the majority class.

3.3 Hybrid Sampling

3.3.1 SMOTE+ENN (SMOTE + Edited Nearest Neighbor)

SMOTE + Edited Nearest Neighbor is a hybrid approach, mixture of oversampling and under-sampling techniques. First the algorithm over samples the minority class by the SMOTE, then Edited Nearest Neighbor (ENN) is used for under sampling. In ENN, under-sampling of the majority class is done by removing points whose class label differs from the majority of its k nearest neighbors [13]. ENN removes noise and avoids overfitting the data, however, ENN is not only limited to removing instances of majority class.

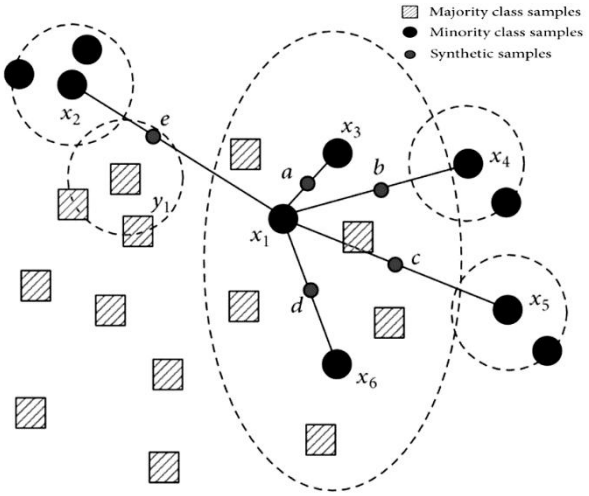

Fig 5

$$E[y_i \mid x_i, \beta] = p_i = \frac{e^{x_i\beta}}{1+e^{x_i\beta}}$$

where β is the vector of coefficients. Now assume that $x_{i0} = 1$ so that β_0 is included in the β vector. The logit transformation is the log of the odds and is:

$$\eta_i = \ln\left(\frac{p_i}{1-p_i}\right) = x_i\beta$$

The matrix form of the logit function is:

$$\eta_i = x_i\beta$$

3.3.2 SMOTE+TL (SMOTE + Tomek Links)

SMOTE + Tomek Links technique over samples the minority class with SMOTE and under samples the majority class with Tomek Link which removes the noise from the data.

4. Classification Models:

Some of the algorithms which are commonly used in improving the classification performance of imbalanced data are discussed in this section.

4.1 Logistic Regression

Logistic regression is similar to linear regression but is designed for studies where the response variable is a categorical value with two or more possible values. Logistic Regression is one of the basic and popular algorithms to solve a classification problem. It is named 'Logistic Regression' because its underlying technology is quite the same as Linear Regression. Logistic regression is a probability statistical model which is widely used for learning from the data because of its understandability, solid theoretical basics, and high generalization abilities. Logistic regression assumes that there's little to no multicollinearity among the independent variables. i.e., the variables are not too highly correlated. It is a linear model with a link function that maps the output of linear multiple regression to the posterior probability of class using the logistic sigmoid function [14].

Let $X \in R^{N\times d}$ be a matrix of data with *N* indicating the data points, *d* is the number of variables, and *y* is the categorical outcome for every row $x_i \in R^d$, where $i = 1 \ldots N$, $y_i = 1 or\ y_i = 0$. The rows of *X* can be considered a Bernoulli experiment with $E(y_i)$ as its expected value and p_i as probability and it can be expressed as:

$$E[y_i \mid x_i, \beta] = p_i = \frac{e^{x_i\beta}}{1+e^{x_i\beta}}$$

where β is the vector of coefficients. Now assume that $x_{i0} = 1$ so that β_0 is included in the β vector. The logit transformation is the log of the odds and is:

$$\eta_i = \ln\left(\frac{p_i}{1-p_i}\right) = x_i\beta$$

The matrix form of the logit function is:

$$\eta_i = x_i\beta$$

4.2 Decision Tree

Decision trees are an important type of algorithm for predictive modeling. They are known for their interpretability as they present a decision in a tree like structure, it is the foundation for advanced ensemble methods such as bagging, random forests, and gradient boosting.

The goal of using the Decision Tree is create a training model that can use to predict the class or value of the target variable. In a Decision Tree, for predicting a class label to a record, we start from the root of the tree. There are two major stages of decision trees – growth phase and pruning phase. In growth phase, decision trees are constructed using training data. We pick features that based on certain criterion best splits the instances into two subsets. Each of these two subsets will usually not contain instances of just one class, so we recursively split the subsets until the final subset contain examples from one class only. Thus, each internal node specifies a feature and a value that best separates the two classes. At the terminal node (leaf node), we make a final decision [15].

Important terminology related to Decision Tree:

i) Root Node: It represents the entire sample, and this further gets divided into two or more homogeneous sets.

ii) Splitting: It is a process of dividing a node into two or more sub-nodes. When sub-nodes splits into further sub-nodes then it is called Decision Node.

iii) Leaf/Terminal Node: Nodes that do not split into more nodes. Leaf nodes are the endpoint of a branch or the final output of a series of decisions. Decision Trees do not branch any further from a leaf node. With decision trees in machine learning, the features of the data are internal nodes, and the outcome is the leaf node.

iv) Pruning: When you selectively remove branches from a tree. The goal is to remove unwanted branches, improve the tree's structure, and direct new, healthy growth.
v) Branch/Sub Tree: A part of the entire decision tree is called a branch or sub-tree. For classification trees, there are two mathematical models which are used in selecting the splitting rule in extracting the decision trees the first one is *Gini Index* and another one is *Cross Entropy.*
Gini Index: It calculates the probability of a certain randomly selected feature that was classified incorrectly. The Gini Index varies between 0 and 1, where 0 represents the purity of the classification and 1 denotes the random distribution of elements among various classes. As we move down the decision tree, the Gini index decreases thus leading to better classification and best split at every node [16]. The process of splitting leaf nodes continues until a minimum number of examples in each leaf node are reached. A Gini Index of 0.5 shows that there is equal distribution of elements across some classes. As the Gini Index is non-parametric, it does not rely on data belonging to a particular type of distribution so it can be used for credit card fraud detection.
The mathematical formula of the Gini index is given below where p_r represents the probability and a is the number of classes:

$$G = \sum_{a=1}^{a} p_r\,(1 - p_r)$$
$$= \sum_{a=1}^{a} p_r - \sum_{a=1}^{a} p_r^2$$
$$= 1 - \sum_{a=1}^{a} p_r^2$$

Cross Entropy: Cross entropy - another technique to measure the impurity of the node. Just like Gini index, entropy value lies between 0 and 1. The smaller the value of entropy, the purer the node is. The formula for entropy is:

$$G = \sum_{a=1}^{a} (p_r) \log (p_r)$$

4.3 Random Forest
Random Forest classifier used to solve for classification problems. The random forest algorithm is made up of a collection of decision trees, and each tree in the ensemble is comprised of a data sample drawn from a training set with replacement, called the bootstrap sample. One of the most powerful and popular bagging technique is random forests. The core idea of ensemble-based learning algorithm is to combine diverse set of classifiers together to improvise the stability and predictive power of the model. The key advantage of the random forest is it can investigate nonlinear and hierarchical relationships between the predictors and the respons using an ensemble learning approach [17]. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression. Some features of a Random Forest Algorithm:

- It's more accurate than the decision tree algorithm.
- It provides an effective way of handling missing data.
- It can produce a reasonable prediction without hyper-parameter tuning.
- It solves the issue of overfitting in decision trees.
- In every random forest tree, a subset of features is selected randomly at the node's splitting point.

4.4 K- nearest Neighbors
K-Nearest Neighbors (KNN) is a non-parametric approach which does not take into consideration the underlying data distribution. KNN classification is one of the most fundamental and simple classification methods and it can be good choice for a classification study when there is little or no prior knowledge about the distribution of the data. KNN classification was developed from the need to perform discriminant analysis when reliable parametric estimates of probability densities are unknown or difficult to determine. KNN classifies an unknown observation to the class of majority among its k nearest neighbors' observations, as measured by a distance metric, in the training data. KNN uses the user defined value for ***k*** and distance metric. Usually, Euclidean distance is used to calculate the distance of the nearest neighbors.
It is common to select small and odd k but if ***k*** is too small, the classification accuracy will be degraded, and the interference in the noisy instances will be amplified [18]. If ***k*** is too large and the prototype belongs to the class with less training instances, excessive noisy instance will be selected as the nearest neighbor, resulting in the worse classification performance.
Minkowski Distance: The Minkowski distance or Minkowski metric is a metric in a normed vector space which can be considered as a generalization of both the Euclidean distance and the Manhattan distance. KNN classifier fundamentally relies upon the distance metric. Better the distance metric, better the classifier's performance. Most commonly used is Minkowski distance which can be described as:

$$Let\ X = (x_1, x_2, \ldots, x_n)\ and\ Y = (y_1, y_2, \ldots, yn) \in R^n$$
$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}}$$

where p is an integer.

i) If the value of $p = 1$, then it is known as Manhattan distance which is the sum of absolute difference between the points.

ii) If the $p = 2$, then the distance metric is called Euclidean distance which is the shortest distance between two points.

iii) When the value of p tends to ∞, it is known Chebyshev distance which is the maximum distance between two points.

4.5 Naive Bayes

Naive Bayes is a classification algorithm for binary (0 and 1) and multi-class classification problems. The technique is easiest to understand when described using binary or categorical input values. It is another popular algorithm for data mining applications due to its simplicity, easy implementation, and efficiency. Its simplicity comes from the assumption that features are independent given class. While the independence of features is a poor assumption, naïve bayes outperform more sophisticated machine learning classifiers. Naïve Bayes are based on Bayesian Classification – a process that estimates the probability of a new observation belonging to a predefined category, using a probability model defined according to Bayes' theory. The techniques assess the prior probabilities of each category based on a large set of training data, that are described by a number of variables, and assumes that classification could be estimated by calculating the conditional probabilities density function and the posteriori probability [19]. A posteriori probability is the probability of an outcome after taking into consideration the prior probability using Bayes' Theorem.

Suppose dataset contain n instances $x_i \; where \; i = 1, 2, \ldots n,$ which consists of u attributes, i.e., $x_i = (x_{i1}, x_{i2}, \ldots x_{in})$.

Each instance assumed belong to one class y ∈ {0, 1}. The formula for Bayes' theorem is as following:

$$P(y_j|x_i) = \frac{P(x_i|y_j)\,P(y_j)}{P(x_i)}$$

The numerator in the above equation is also written as:

Since Naive Bayes' assumption is that the individual x_i is independent from each other, thus:

$$P(x|y_j)\,P(y_j) = P(x_1|y_j).P(x_2|y_j)\ldots P(x_p|y_j)\,P(y_j)$$

$$P(y_j|x) = \prod_{k=1}^{u} \frac{P(x_k|y_j)\,P(y_j)}{P(x)}$$

Here *P(x)* acts as a scaling factor and ensure that the posterior probability $P(y_j|x_i)$ is properly scaled between 0 and 1 [20]. For classification purposes, we can simply calculate the value of the numerator for each class and select that class for which this value is maximum. The resulting class is also known as maximum a posteriori (MAP) class and Naïve Bayes implements this model. It is calculated as:

$$\hat{y} = \arg_{y_j} \max \left(\prod_{k=1}^{u} P(x_k|y_j)\,P(y_j) \right)$$

4.6 Support Vector Machines

Support vector Machines (SVM) is an adaptable, powerful algorithm that can be used for classification purposes. Due to its theoretical and practical advantages, solid mathematical background, high generalization capability, and ability to find global and non-linear classification solutions, SVMs have been very popular among statistical learning researchers [21]. SVM works by mapping data to a high-dimensional feature space by using different types of kernel functions so that data points can be categorized, even when the data are not otherwise linearly separable.

A support vector machine is a linear binary classifier that works by finding a hyperplane to separate two classes in a dataset. In fact, hard margin support vector machines work exactly like a perceptron, except that they optimise a different objective function in choosing the parameters. For the perceptron, all linear boundaries that separate the two classes are equally good, because the perceptron loss is zero for each of them. In the example below, however, it seems clear that line A is a better choice than line B.

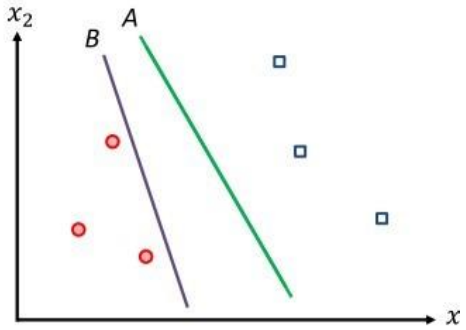

Fig 6

The support vector machine formalises this notion by finding the separating boundary that maximises the margin between classes.

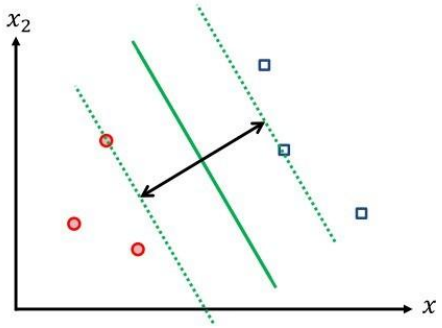

Fig 7

The points on the margin boundaries (the dotted lines) in the figure above are called the support vectors (they are a vector of observation variables). They play an important role in defining the margin width. The distance between these lines is called classification margins. The goal of the optimal classification hyperplane is to discriminate between the two classes correctly while maximizing the classification margins. Kernel function helps to transform the input samples into a high-dimensional space so that they can be classified linearly [22].

The hyperplane separating the class in the higher dimensional feature space of training data consists of N pairs $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N). x_i \in R^p \text{ and } y_i \in \{-1, 1\}$ can be represented:

$$\{x : f(x) = x^T\beta + \beta_0 = 0\}$$

If the classes are completely linearly separable, the separating hyperplane with the maximum classification margin (M) can be found by solving the following optimization problem:

$$\max_{\beta,\beta_0,||\beta||=1} M$$

$$subject\ to\ y_i(x_i^T\beta + \beta_0) \geq M,\ i = 1,\ldots,N$$

Where $||\beta|| = 1$ is a unit vector. When classes are not linearly separable i.e., they overlap even after mapping into a higher dimensional feature space, some constrains are added by introducing slack variables ξ_i which controls how many training points may lie on the wrong side of the hyperplane. The goal of this is to maximize the margins while minimizing the misclassification rate and variable C is the tuning parameter which controls the bias-variance trade-off. For the separable case, C = ∞

Confusion matrix calculates the accuracy. Accuracy is a metric that can be applied to classification tasks. It is an overall proportion of correct classification. It describes what percentage of test data is classified correctly [23]. There are some evaluation matrices which can be obtained from confusion matrix are Precision, Recall, and F-score.

Precision: Also called Positive predictive value. The ratio of correct positive predictions to the total predicted positives. It tells us how many of the correctly predicted cases turned out to be positive.

$$precision = \frac{TP}{TP + FP}$$

Recall: Also called Sensitivity, Probability of Detection, True Positive Rate. The technical definition of recall is the number of true positives divided by the number of true positives plus the number of false negatives. It tells us how many of the actual positive cases we were predict correctly with our model.

$$recall = \frac{TP}{TP + FN}$$

F-score: Precision and recall are combined into single score called F-score, which is the harmonic mean of precision and recall.

$$\text{F-score} = \frac{2(precision \times recall)}{precision + recall}$$

While working with imbalanced dataset, Precision Recall curve is plotted with recall on the x-axis and precision on the y-axis. Precision-Recall (PR) curves give a more informative picture of an algorithm's performance. Average Precision (AP) is approximately the area under the precision-recall curve. Average precision gives you mean precision at all possible thresholds and has values that lies between 0 and 1, and the higher is better. AP can be defined as:

$$AP = \int (precision(t) d[recall(t)])$$

Another evaluation criteria used to assess the performance of the classifier is the *log loss* function. Log loss quantifies the accuracy of a classifier. Thus, minimising the *Log loss* equates to maximising the accuracy of the classifier. The log loss function for binary classification can be defined as:

$$\log \quad loss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} \log (P_{ij})$$

where *N* is the total number of samples, *M* is the number of possible labels, and y_{ij} is a binary indicator. A perfect model will predict perfect probability and has the log loss of 0.0, a less ideal model will have a higher log loss. The log loss formula is equivalent to the formula of entropy, so optimizing the entropy gain (split criterion) is the same as optimizing the log loss (metric), but due to the fact that log loss is biased towards pure nodes, if we predict the wrong class with probability 1, the log loss becomes infinite to have a better generalization, such situations should be avoided which means that pure nodes should be avoided.

5. Results

We compared the obtained results, I proposed variable selection techniques to select the topmost features and achieved results according to my methodology and used resampling techniques to handle the imbalanced data and then get balanced data and implemented different type of Machine Learning algorithms and got different results like F-score, recall and log loss. At first, we applied variable selection techniques in order to select topmost relevant features (largest values). And then implemented ML algorithms for results without sampling method and after that and after that sampling techniques have been applied before using our classification algorithms for improving the performance and then implemented different ML algorithms furthermore, I got some better results.

5.1 Abbreviations:

Models: Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), k-Nearest Neighbors (KNN), Naive Bayes (NB), Support Vector Classifier (SVC). *Sampling Methods:* Random Over Sampler (ROS), Synthetic Minority Over Sampling Techniques (SMOTE), Support Vector Machine Synthetic Minority Oversampling Technique (SVMSMOTE), Random Under Sampler (RUS), Edited Nearest Neighbor (ENN), Tomek Links (TL).

5.2 Without Sampling:

Models	F-score	recall	Log loss	accuracy
LR	0.68	0.57	0.029	0.99
DT	0.74	0.75	0.028	0.99
RF	0.82	0.77	0.017	0.99
KNN	**0.84**	0.78	0.153	0.99
NB	0.22	0.81	0.321	0.99
SVC	0.78	0.81	0.020	0.99

Table-2: ***F-score*** and recall without sampling techniques

5.3 After Resampling the data:

	Over-sampling			Under-sampling		
Models	ROS	SMOTE	SVMSMOTE	RUS	NearMiss	ENN
LR	0.95	0.94	0.99	0.93	0.91	0.91
DT	0.99	1	1	0.92	0.95	0.95
RF	**1**	1	1	0.92	0.96	0.96
KNN	0.99	1	1	0.91	0.91	0.91
NB	0.92	0.92	0.96	0.90	0.96	0.96
SVC	1	0.99	1	0.94	0.96	0.96

Table 3: F-***score*** of Over and Under-sampling techniques

We can notice that resampling technique improved the f-score while without any resampling techniques, the highest F-score was **0.84** of k-nearest neighbors model whereas after ROS, SMOTE, and SVMSMOTE the highest F-score achieved **1.00** for Random Forest. Thus, proving that resampling an imbalanced data set is an effective approach.

5.4 No sampling and Hybrid Sampling

		Hybrid Sampling	
Models	No sampling	SMOTE+ENN	SMOTE+TL
LR	0.68	0.94	0.94
DT	0.74	1	1
RF	0.82	**1**	1
KNN	0.84	1	1
NB	0.22	0.92	0.92

SVC	0.78	0.99	0.99

Table 4: ***F-score*** of no sampling and hybrid sampling techniques

5.5 Over-sampling and Under-sampling

	Over sampling			Under sampling		
Models	ROS	SMOTE	SVMSMOTE	RUS	NearMiss	ENN
LR	1.96	2.11	1.66	2.57	3.05	3.05
DT	0.01	0.07	0.01	2.73	1.92	1.92
RF	0.00	0.00	0.00	2.73	1.44	1.44
KNN	0.01	0.04	0.00	3.05	3.21	3.21
NB	1.49	2.63	1.41	3.37	1.60	1.60
SVC	0.14	0.22	0.20	2.08	1.44	1.44

Table 5: ***log loss*** of Over and Under-sampling techniques

Frome table 5, we can notice that the log loss of the resampling techniques. The log loss of the over sampling techniques is lower than the log loss of under sampling technique. The lowest log loss of **0.00** is for the RF model with ROS, SMOTE and SVMSMOTE, whereas the highest log loss of **3.37** is for the NB model with RUS as resampling technique.

5.6 No sampling and Hybrid Sampling

		Hybrid sampling	
Models	No sampling	SMOTE+ENN	SMOTE+TL
LR	0.029	2.09	2.08
DT	0.028	0.69	0.07
RF	**0.017**	**0.00**	0.01
KNN	0.153	0.00	0.04
NB	0.321	2.64	2.60
SVC	0.020	0.20	0.23

Table 6: ***log loss*** of no sampling and hybrid sampling techniques

From above table 6, we noticed that hybrid sampling techniques have lower log loss as compared to over-sampling, and under-sampling data set. The log loss of the data set with under-resampling was **2.73** for RF, and it decreased further to **0.01** after applying SMOTE+ENN and SMOTE+TL.

We can also be noticed from table 6 in some cases No sampling methods have less log loss that means our ML algorithms worked well, it may be happen after feature selection or variable selection techniques.

6. Conclusion:

Identifying credit card fraud is an essential for banks to avoid huge losses. It is becoming more and more difficult to detect credit card fraud, thus it is important to adopt a more efficient way to detect counterfeit credit card transactions. Random Forest was the best approach to handle class imbalance data as compared to other ML models. It enhanced the detection of credit card fraud substantially. In this paper, we considered a new approach to feature selection in the context of imbalanced class distribution followed by sampling methods. Concretely, we proposed a simple yet effective method to select the optimal subset of features based on Machine Learning algorithms F1-scores. Our method consists of constructing ML algorithms and calculating the corresponding F-score. The use of the F-score allows us to evaluate features based on their performance on the minority data. Additionally, Random Forest and Decision Trees as compared to other algorithms provide a simple and fast approach to calculate the F-scores. We tested the proposed method on datasets originated from different fields including credit card fraud detection, medical diagnostics, etc.

The range of different class ratios, number of features and sample size further diversified the experiments. As a result, the conclusions of the study can be applied in a wide variety of scenarios. The results of the experiments demonstrate that the proposed method is able to reduce the number of features while improving classification performance. Furthermore, the low computational complexity of the proposed algorithm makes it suitable for use with data. Given the discussed advantages of the proposed feature selection method we believe that it could be a valuable tool in pre-processing data with imbalanced class distribution. In addition, areas for further research Pretest and Post-shrinkage strategies can be applied for the model, based on Pretest and Shrinkage strategies to improve the estimation efficiency and these techniques improve the prediction performance for the selected model as well.

7. References:

[1] Szil´ard Vajda, Gernot A. ―Fink Strategies for Training Robust Neural Network Based Digit Recognizers on Unbalanced Data Set 2010, 12th International Conference on Frontiers in Handwriting Recognition.

[2] A. Ali, S. M. Shamsuddin, A. L. Ralescu, et al., "Classification with class imbalance problem: a review," Int. J. Advance Soft Compu. Appl, vol. 7, no. 3, pp. 176–204, 2015

[3] Nitesh V. Chawla, Nathalie Japkowicz, Aleksander Ko lcz ―Editorial: Special Issue on Learning from Imbalanced Data Sets‖ Sigkdd Explorations. Volume 6, Issue 1

[4] Olson DL, Shi Y, Shi Y (2007) Introduction to business data mining, vol 10. McGraw-Hill/Irwin, New York, pp 2250–2254

[5] Haixiang G, Yijing L, Shang J, Mingyun G, Yuanyue H, Bing G (2017) Learning from class-imbalanced data: review of methods and applications. Expert Syst Appl 73:220–239

[6] Lemaitre G, Nogueira F, Aridas CK (2017) Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning. J Mach Learn Res 18(1):559–563
[7] S. Ranganathan, K. Nakai, and C. Schonbach, Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics. Elsevier, 2018
[8] V. García, J. S. Sánchez, and R. A. Mollineda, "On the effectiveness of preprocessing methods when dealing with different levels of class imbalance," Knowledge-Based Systems, vol. 25, no. 1, pp. 13–21, 2012
[9] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," Journal of artificial intelligence research, vol. 16, pp. 321–357, 2002
[10] Andrew Estabrooks, Taeho Jo and Nathalie Japkowicz ―Multiple Resampling Method for Learning from Imbalanced Data Sets. Comprtational Intelligence 20 (1) (2004) 18-36
[11] Taeho Jo, Nathalie Japkowicz ―Class Imbalances versus Small Disjuncts‖. Sigkdd Explora-tions 6 (1) (2004) 40-49
[12] Mani I, Zhang I (2003) kNN approach to unbalanced data distributions: a case study involving information extraction. In: Proceedings of workshop on learning from imbalanced datasets, vol 126
[13] M. Zorkeflee, K. R. Ku-Mahamud, and A. Mohamed Din, "A conceptual model of enhanced undersampling technique," Unknown, 2014
[14] Edouard Duchesnay, Tommy Lofstedt, Feki Younes, "Statistical and Machine Learning in Python", Release 0.5, (oct 04, 2021), pp. 233-237
[15] H. Drucker and C. Cortes, "Boosting decision trees," in Advances in neural information processing systems, pp. 479–485, 1996
[16] J. Friedman, T. Hastie, and R. Tibshirani, The elements of statistical learning, vol. 1. Springer series in statistics New York, 2001
[17] Y. Everingham, J. Sexton, D. Skocaj, and G. Inman-Bamber, "Accurate prediction of sugarcane yield using a random forest algorithm," Agronomy for sustainable development, vol. 36, no. 2, pp. 27, 2016
[18] V. García, R. A. Mollineda, and J. S. Sánchez, "On the k-nn performance in a challenging scenario of imbalance and overlapping," Pattern Analysis and Applications, vol. 11, no. 3-4, pp. 269–280, 2008
[19] P. Tsangaratos and I. Ilia, "Comparison of a logistic regression and naïve bayes classifier in landslide susceptibility assessments: The influence of models complexity and training dataset size," Catena, vol. 145, pp. 164– 179, 2016
[20] R. Batuwita and V. Palade, "Class imbalance learning methods for support vector machines," Unknown, 2013
[21] Y. Huang and L. Zhao, "Review on landslide susceptibility mapping using support vector machines," Catena, vol. 165, pp. 520–529, 2018
[22] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," Information sciences, vol. 250, pp. 113–141, 2013