

Analysis of Imbalanced Data

Project report submitted in partial fulfilment of the requirement for the degree

of

Bachelor of Technology

in

Information Technology

Under the supervision of

Dr. Hari Singh

Assistant Professor (Senior Grade)

By

Chandan Partap Singh (151456)

to



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Wagnaghat, Solan-173234,

Himachal Pradesh

May 2019

Certificate

I hereby declare that the work presented in this report entitled “**Analysis of Imbalanced Data**” in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from January 2019 to May 2019 under the supervision of **Dr. Hari Singh**.

The work done embodied in the report has not been appeased for the award of any other degree or diploma.

.....

(Student Signature)

Chandan Partap Singh

(151456)

This is to certify that the above affirmation made by the candidate is true to the best of my knowledge.

.....

Dr. Hari Singh

Assistant Professor (Senior Grade)

Department of Computer Science and Engineering and Information Technology

Dated:

Acknowledgement

I would like to take the opportunity to thank and express my deep sense of gratitude to my mentor and project guide Dr. Hari Singh for his immense support and valuable guidance without which it would not have been possible to reach this stage of my final year project.

I am also obliged to all my faculty members for their valuable support in their respective fields which helped me in reaching at this stage of my project. My thanks and appreciations also go to my colleagues who have helped me out with their abilities in developing the project.

Date.....

Chandan Partap Singh (151456)

Table of Contents

Certificate	i
Acknowledgement	ii
Table of Contents	iii
List of Figures	v
List if Tables	vi
Abstract	vii
Chapter 1. Introduction	1
1.1 Introduction	2
1.2 Problem Statement	3
1.3 Objective	4
1.4 Methodology	4
1.5 Organization of Thesis	5
Chapter 2. Literature Survey	6
2.1 Learning from Imbalanced Data Sets using various techniques	7
2.2 Using Random Forest to Learn Imbalanced Data	8
2.3 A Survey on Performance Measures in Machine Learning	9
Chapter 3. System Development	11
3.1 System Requirements	12
3.1.1 Software Requirements	12
3.1.2 Hardware Requirements	13
3.2 Existing Methodologies	13
3.2.1 Resampling Techniques	13
3.2.1.1 Random Under-Sampling	14
3.2.1.2 Random Over-Sampling	15

3.2.1.3	Synthetic Minority Over Sampling Technique	16
3.2.2	Algorithmic Ensemble Techniques	17
3.2.2.1	Logistic Regression	19
3.2.2.2	Random Forest	20
3.2.2.3	Naïve Bayes	21
3.3	Feature scaling	22
3.3.1	Methods	22
3.3.1.1	Min-Max Normalization	22
3.3.1.2	Mean-Normalization	23
3.3.1.3	Standardization	23
3.4	Combination of resampling and ensemble methods	23
Chapter 4. Implementation and Results		24
4.1	Dataset Used	25
4.2	Performance Parameters	26
4.2.1	F1 Score	26
4.3	Implementation	26
4.4	Results	28
4.5	Code	33
4.5.1	Without Feature Scaling	33
4.5.2	With Feature Scaling	36
Chapter 5. Conclusions		41
5.1	Conclusion	42
5.2	Future Scope	42
References		43

List of Figures:

Figure Number	Caption	Page Number
Figure 1.1	Visual representation of Imbalanced Data	2
Figure 1.2	Confusion Matrix	3
Figure 2.1	Example of an RUC Curve	10
Figure 3.1	Visual representation of Under and Over Sampling	13
Figure 3.2	Working of SMOTE	17
Figure 3.3	Basic Decision Tree working	18
Figure 3.4	Ensemble based methodologies working	18
Figure 3.5	Linear Regression	19
Figure 3.6	Graphs of Linear and Logistic Regression	20
Figure 3.7	How a Random Forest works?	21
Figure 4.1	Credit card Dataset Snapshot	25
Figure 4.2	Snapshot of the F1 scores of different algorithms without Feature scaling.	29
Figure 4.3	Confusion Matrix without Feature scaling	29
Figure 4.4	Snapshot of the F1 scores of different algorithms with Feature scaling	31

List of Tables:

Table Number	Caption	Page Number
Table 4.1	Table showing the F1 scores of different algorithms without Feature scaling	28
Table 4.2	Table showing the F1 scores of different algorithms with Feature scaling	30
Table 4.3	Table showing the F1 scores with Feature Scaling and without Feature Scaling	32

ABSTRACT

Due to the expansion of data in large scale organizations, it has become crucial to upgrade the data mining techniques for decision making. In some scenarios such as medical diagnosis, fraudulent detection, imbalanced data is not a unique feature. Any dataset can be identified as an imbalanced dataset if the number of instances in one class are significantly higher than the other one (around 10:1). For example, in a dataset, there are 100 instances in under Class A and 2 instances under Class B. Though many existing classification algorithms are there, but most of them are biased towards the majority class. In any normal scenario, it is normal, but in the areas such as medical diagnosis and fraud detection, the minority class is ignored and hence wrong outcomes are deduced. Resampling is a very popular method to tackle this issue. It involves generating synthetic instances (known as over-sampling) or removing instances (known as under-sampling). Some modern algorithms such as the Ensemble Classifiers such as Random Forest are also explained. In this report, I have demonstrated the existing algorithms and I have proposed a new model whose accuracy (calculated on the basis of F1 score) is better than the traditional models.

Chapter 1:
Introduction

Chapter 1: Introduction

1.1 Introduction

Imbalanced data set happens when there is an unequal portrayal of classes. Imbalanced data isn't generally an awful thing, and in genuine data sets, there is in every case some level of imbalance. There may not have a huge effect on your model execution if the dimension of irregularity is generally low. But, in specific regions, for example, fraud detection, medicinal determination and risk management, serious imbalanced class dispersion is normal, and in this way, is a concerning issue.

An vital task arises whilst we strive to analyse from imbalanced dataset in which the number of example of 1 (majority) class outnumbers the others. Conventional machine learning algorithms are usually biased to the class with higher instances, therefore generating poor predictive accuracy over the minority magnificence. In this report, a comparative study and a brand new technique that combines one of a kind algorithm is applied.

Imbalanced Class Distribution

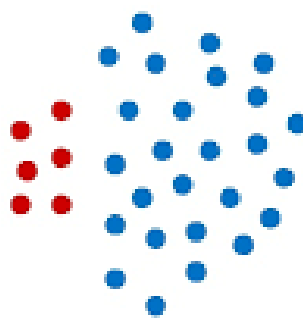


Figure 1.1: Visual representation of Imbalanced Data

1.2 Problem Statement

Companies that get affected by the problem of imbalanced datasets are working towards advanced analytics and machine learning algorithms to find the backdoor of this problem and eliminate it.

That said, the expansion of data at a very fast speed increases its size and distribution. The ratio of fraudulent transactions to the ordinary transactions is very low (1-2% from the dataset). The challenge is to increase the accuracy to identify a fraudulent transaction so that the loss users are facing can be minimized by the organizations/banks.

Traditional classifier algorithmic Decision Tree and Logistic Regression are biased to the class which has higher number of instances. The accuracy of such classifiers are hence reliable for majority class. Minority class, which has lower number of instances are usually ignored. Henceforth, the probability of misclassification is higher in case of such classifiers.

Confusion Matrix is one of the criteria on which we can evaluate any algorithm. Information of actual class and predicted class is represented in confusion matrix.

Actual	Predicted	
	Positive Class	Negative Class
Positive Class	True Positive(TP)	False Negative (FN)
Negative Class	False Positive (FP)	True Negative (TN)

Figure 1.2: Confusion Matrix

$$\text{Accuracy} = \frac{\text{TN} + \text{TP}}{\text{TN} + \text{FP} + \text{FN} + \text{TP}}$$

Having said that, accuracy is not an appropriate performance criterion if we are dealing with imbalanced dataset. For example: We have a dataset in which 98 instances are in Class A and 2 in Class B. A classifier can predict if an instance belongs to class A with 98% accuracy, but in case of imbalanced data, the remaining 2% is what matters.

1.3 Objective

Two broad objectives of this report are given below:

1. To detect a faulty transaction with higher accuracy:
Out of the most popular classification and resampling methods, the aim is to compare the combinations of resampling and classification techniques and to identify which of the combinations performs the best in terms of accuracy.
2. To further improve the computation speed of the algorithms used in the first objective.

1.4 Methodology

- 1.** To complete the first objective. I will perform Logistic Regression with Synthetic Minority Over-Sampling Technique (SMOTE), Logistic Regression with Random-Under-Sampling, Logistic Regression without SMOTE, Naïve Bayes with SMOTE, Naïve Bayes with Random-Under-Sampling, Naïve Bayes without SMOTE, Random Forest with SMOTE, Random Forest with Random-Under-Sampling, Random Forest without SMOTE.
- 2.** To improve the computation speed, normalizing/standardizing the dataset in some cases proves to be beneficial.

1.5 Organization of Thesis

In Chapter 2, I have discussed the literature survey. I have looked upon the various existing approaches that have been used for handling imbalanced data. Each one of them has its own unique features which distinguishes it from the rest.

In Chapter 3, I have explained the traditional algorithms that I have used in proposing my new model by integrating those algorithms.

In Chapter 4, I have explained my proposed strategy to handle imbalanced data in case of a credit card dataset and the comparative table results is shown.

In Chapter 5, I have thereby concluded my report and discussed the future scope of this project.

Chapter 2:
Literature Survey

Chapter 2: Literature Survey

2.1 Usamma Fayad, Gregorry Patetsky Sapiro and Pahrac Smith: Learning from Imbalanced Data Set using various techniques, Volume 39, Nov. 2015, Page 29-34

Resampling of the dataset has been a solution of handling imbalanced data since the problem came into existence. Resampling means either to increase the number of instances in the minority class (class with less number of objects under it) or removing instances from the majority class (class with more number of objects under it). There are many techniques under Resampling, mainly Random Over Sampling, Random Under-Sampling, Synthetic Minority Over Sampling Technique etc...

The advantage of using resampling techniques is that it is very easy to implement. However, some modern resampling techniques such as SMOTE, MSMOTE etc. have complex implementation which gives good results.

The limitations of Resampling techniques are that only the classification accuracy is improved. But the problem lies in the fact that classification accuracy is one of the most outdated performance criteria which is simply calculated by identifying how many of the predictions are correct without keeping in account the minority class.

Logistic Regression is a classification algorithm which is used to predict outcome using the sigmoid function. Sigmoid function takes real numbers as the input and the output lies in the range of 0 to 1 as probability lies in the range of 0 to 1.

Logistic Regression is easy to implement and is very efficient to train, also the conditional probabilities are calculated which can be useful in many scenarios.

However, Logistic regression shows poor results if the dimensionality of the dataset is high.

2.2 Xindongg Wou, Xigquan Zuh, Gongg-Xing Wuh, and Wey Dingh: “Using Random Forest to Learn Imbalanced Data” IEEE Transactions on Imbalanced Data Engineering, Volume26 Issue 1, January 2014 Pages 97-107

Random Forest is an ensemble method which can be used for both classification as well as regression analysis. It constructs multiple decision trees when the models re getting trained and the final outcome are decided on the basis of the individual results of the decision trees.

Basic parameters that are needed in Random Forest can be the total number of trees to be generated and the splitting criteria.

Random Forests has less variance than a single decision tree. They have high accuracy than existing classification algorithms. The disadvantage of using random forest is its complexity. They are much harder and time consuming to build than single decision trees.

Naïve Bayes is another type of classifier which is used to predict outcomes on the basis of probability. It is based on the Bayes theorem.

A Naïve Bayes model is easy to build which requires no complicated development, hence making it effective for large datasets with high dimensionality. Machine Learning revolves around in finding the best hypothesis, h , for the given data d . For classification, h could be the in which we want to add a new instance d . Using Bayes Theorem, the probability of a hypothesis can be calculated if the prior knowledge is given.

If the dataset is small, the Naïve Bayes does not come up very well. The precision and recall are set low and hence the F1 score is low, which does not make the Naive Bayes a reliable algorithm.

2.3 Apoorva Aggarwal, Boy Xi Ila Vosha, Oven Rambov Rebeca: “A Survey on Performance Measures in Machine Learning” IJCSMC, Vol. 4, Issue. 11, November 2015, pg.338 – 343.

To correctly evaluate the performance of any model, different performance metrics are available.

1. Confusion Matrix:

Confusion Matrix displays the complete performance of a model in the form of a matrix

There are 4 important terms:

- True Positives: Prediction is YES and actual output is also YES
- True Negative: Prediction is NO and actual output is also NO
- False Positive: Prediction is YES and actual output is NO
- False Negative: Prediction is NO and actual output is YES

2. F Score:

For datasets that have two classes, F1 score can be good performance criterion to score your model. It is defined as the harmonic mean of precision (fraction of relevant instances among the retrieved instances) and recall (fraction of relevant instances that have been retrieved over the total amount of relevant instances).

F score lies in the range of 0 to 1. 0 for the worst model and 1 for the best one.

3. AUC-ROC Curve:

Area Under the Curve-Receiver Operating Characteristics curve is a performance measurement criterion for checking a classification model's performance. ROC is a

probability curve and AUC represents the area under the curve. Greater the AUC, better is the model.

In ROC curve, True Positive Rate (TPR) is on y-axis and False Positive Rate (FPR) is on x-axis. Similar to F Score, 1 is for an excellent model and 0.5 for the worst, as it means that the model has no capability to classify a single instance correctly.

Calculation Formulae:

$$\text{TPR} = \text{TP} / (\text{FN} + \text{TP})$$

$$\text{FPR} = \text{FP} / (\text{FP} + \text{TN})$$

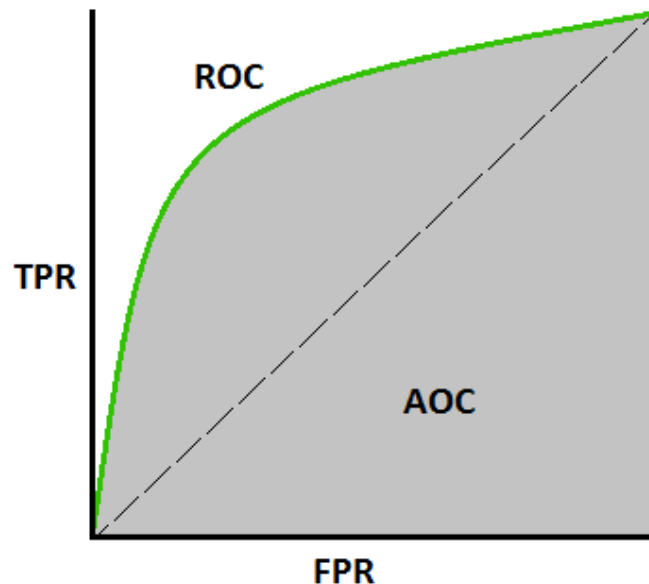


Figure 2.1: Example of an RUC Curve

However, the AUC does not interpret predictions as probabilities as AUC only cares about the ranking of your prediction scores and not their actual values. Therefore, you will not be able to express your uncertainty of a prediction, or simply speaking, AUC is insensitive to imbalanced datasets.

Chapter 3:

System Development

Chapter 3: System Development

3.1 System Requirements

To run this project on a machine, the following software and hardware requirements are mandatory to meet:

3.1.1 Software Requirements:

1. Windows 10 (64-bit) or Windows 8/8.1 (x64 or x86 or Linux (tested on Ubuntu Linux)
2. I have used Python language for this project as Python contains unique libraries for machine learning such as SciPy or NumPy which makes it efficient for creating models based on linear algebra. The language is fun to use when working with machine learning algorithms and has easy syntax relatively. Training datasets can be done more efficiently than R.

3. Spyder

It is open source Integrated Development Environment (IDE) made for programming in Python. It is also an open-source platform, which invites a lot of programmers to test new features.

Spyder provides inbuilt important packages including NumPy, SciPy, Matplotlib, pandas, as well as other open source software.

Spyder is very easy to install compared to its alternatives (e.g., PyCharm). Spyder can be installed using Pip. It is also part of many Linux distributions package manager. The Variable Explorer in Spyder makes it very user-friendly.

Some other alternatives: PyCharm, VSCode

3.1.2 Hardware Requirements:

- Windows 8/8.1/10, Linux, Mac OSX
- Intel Core i3, i5, or i7 CPU with 2 or more cores, or AMD equivalent.
- 1.8 GHz or faster processor.
- Minimum 20 GB free space (depending on the features installed)
- Minimum 4 GB RAM

3.2 Existing Methodologies

3.2.1 Resampling Techniques:

A very popular technique for dealing with highly unbalanced datasets is called resampling. Resampling removes/adds virtual samples to the original dataset.

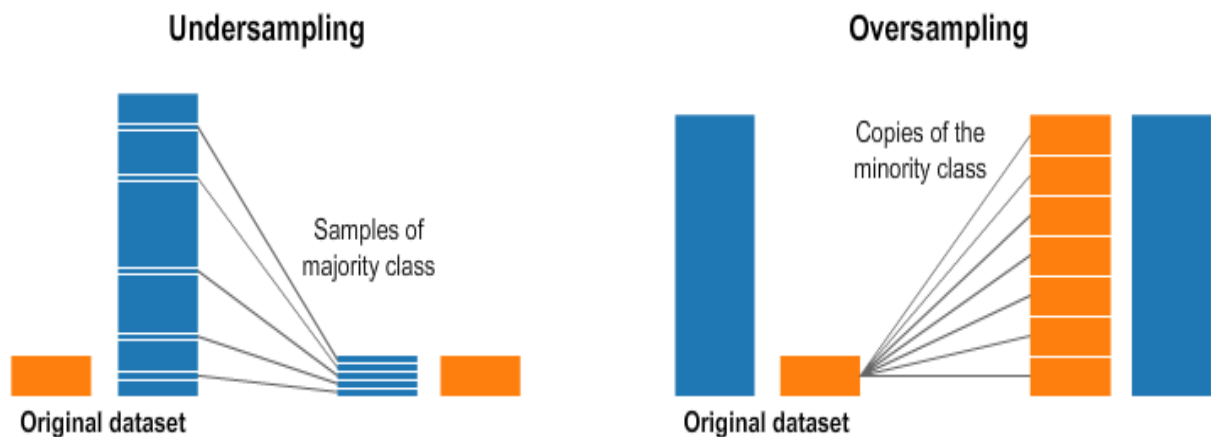


Figure 3.1: Visual representation of Under-Sampling and Oversampling

Though these techniques help in balancing the class, but they do have their limitations. The very basic implementation of Over-Sampling is to duplicate the records randomly from the minority class, which in some cases may cause overfitting. In under-sampling,

the basic method is removing records randomly from the minority class, which usually causes loss of information.

The main idea behind balancing classes is to either increase the frequency of the minority class or decrease the frequency of majority class, hence making the dataset balanced.

A few resampling techniques are given below:

3.2.1.1 Random Under-Sampling:

This method randomly removes instances from the majority class in order to make it balanced. This is done iteratively until the class becomes balanced.

For example,

T (total number of instances) = 100

F (fraudulent instances) = 2

NF Non Fraudulent instances = 98

Rate = 2 %

NF instances sampling (taking 10% samples) = 10 % of 98 = 9.8

After integrating all instances, total instances = 2 + 9.8 which comes around 12.

New Event Rate = $2/12 = 17\%$

Advantages:

- In some cases, it can improve the run time and memory problems as it reduces the number of samples when the training dataset is large.

Disadvantages:

- It can dispose of conceivably valuable data which could be significant for building classifiers.
- There are chances that the sample which is chosen is a biased sample, and hence it will not be the actual representation of the population, hence resulting in inaccurate results.

3.2.1.2 Random Over-Sampling:

Random Over-Sampling involves adding synthetic data with multiple copies of some of the minority classes. Over-Sampling can also be iteratively. This is one the method that has been in existence since a very long time. Over-Sampling can be done either by adding values with mean value or adding values randomly in the minority class.

For example,

T (total number of instances) = 100

F (fraudulent instances) = 2

NF Non Fraudulent instances = 98

Rate = 2 %

Duplicating 10 fraud observations 10 times.

Total Observations after oversampling = 100 + 98 = 198

New Event Rate = $100/198 = 49.9\%$

Benefits of using Oversampling:

- Main benefit of Over-Sampling over Under-Sampling is that there is no loss of information, hence outperforms Under-Sampling

Limitations:

- As Over-Sampling replicates from the minority class chances of overfitting increases.

3.2.1.3 Synthetic Minority Over Sampling Technique:

In this algorithm, synthetic samples are generated and then added to the original dataset. The synthetic samples are created by joining a line between the instances of the minority class and from this line, the synthetic samples are picked. These synthetic instances are afterwards added to the original dataset.

For example,

T (total number of instances) = 100

F (fraudulent instances) = 2

NF Non Fraudulent instances = 98

Rate = 2 %

5 Instances are taken from the minority class and are iterated 10 times.

After generating the samples:

Fraudulent Observations = 50

Non-Fraudulent Observations = 98

Event rate= $50/148 = 33.7\%$

Advantages:

- No loss of useful information.
- Solves the problem of overfitting which was caused by random Over-Sampling as the instances were generated more intelligently.

Disadvantages:

- SMOTE is not very efficient if the dataset has high number of dimension.
- When SMOTE is generating samples, it does takes neighbouring points from the other classes, which can sometimes increase noise.

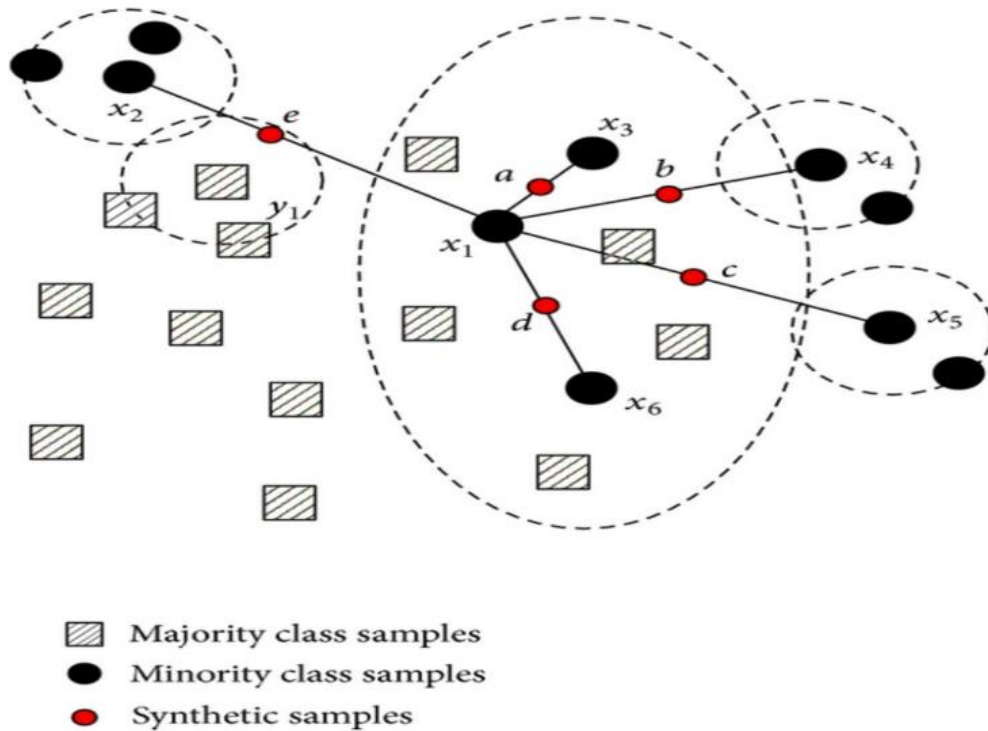


Figure 3.2: Working of SMOTE

3.2.2 Algorithmic Ensemble Techniques

Algorithmic ensemble methods are a technique in machine learning that unites different models to make one optimal predictive model. To better understand this, let me remind you the main idea behind machine learning as I will discuss some examples to why ensemble methods are used. Decision Trees can be used to make the definition clearer. The key idea behind these is to train multiple models in which each decision tree will predict or classify.

Ensemble methods help us increase stability of our final model and hence the individual models getting created will make one optimum model.

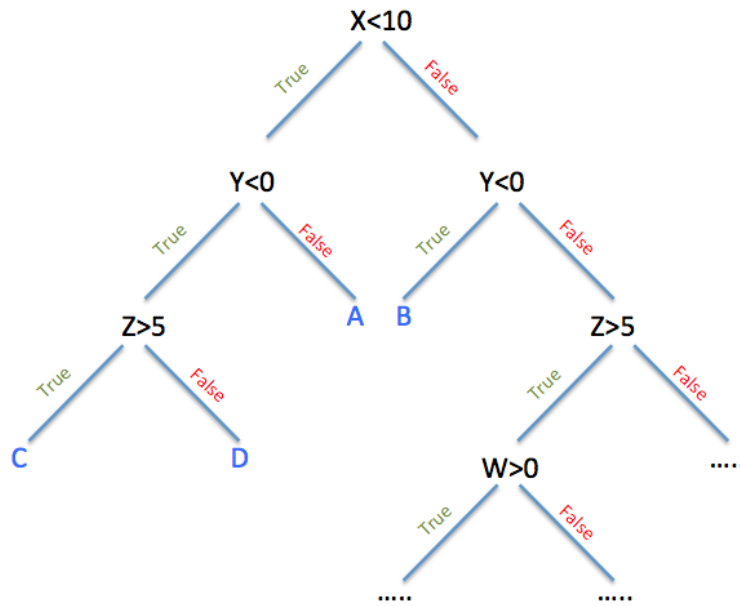


Figure 3.3: Basic decision tree working.

The basic working of decision tree is that it predicts a value based on the set of questions and conditions. In ensemble methods, instead on getting dependent on one Decision Tree, the prediction is made on the basis of multiple decision trees, determines what features are asked and makes a final predictor which is based on the integrated outcomes of the Decision Trees.

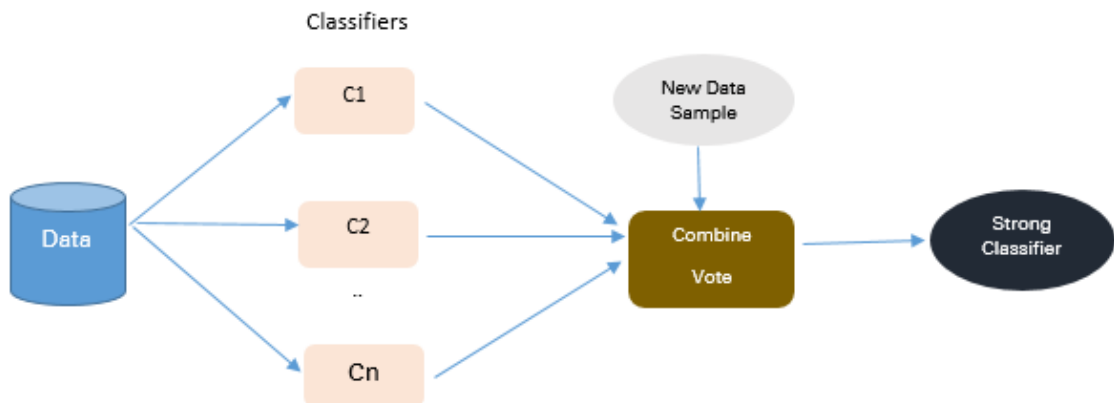


Figure 3.4: Ensemble based methodologies working

3.2.2.1 Logistic Regression

If the response variable is categorical, Logistic Regression can be used as a classifier. The role of Logistic Regression to find an association between probability of any random outcome and its features.

For example, if we want to know whether a student passed or fail in an examination in which we have the number of hours given as a feature.

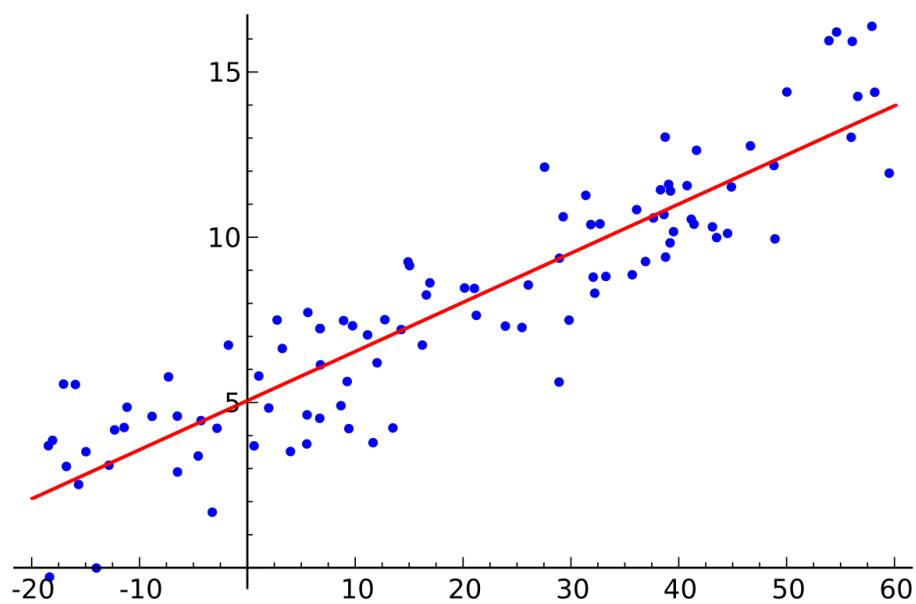


Figure 3.5 Linear Regression

3.2.2.2 Difference between Linear and Logistic Regression:

Logistic Regression is a classification algorithm, in which the outcome can have values from a limited set, i.e., only a limited number of possible values and in Linear Regression, the outcome is always continuous. Linear regression is actually a regression model, meaning it'll always give the output which will be continuous (not discrete).

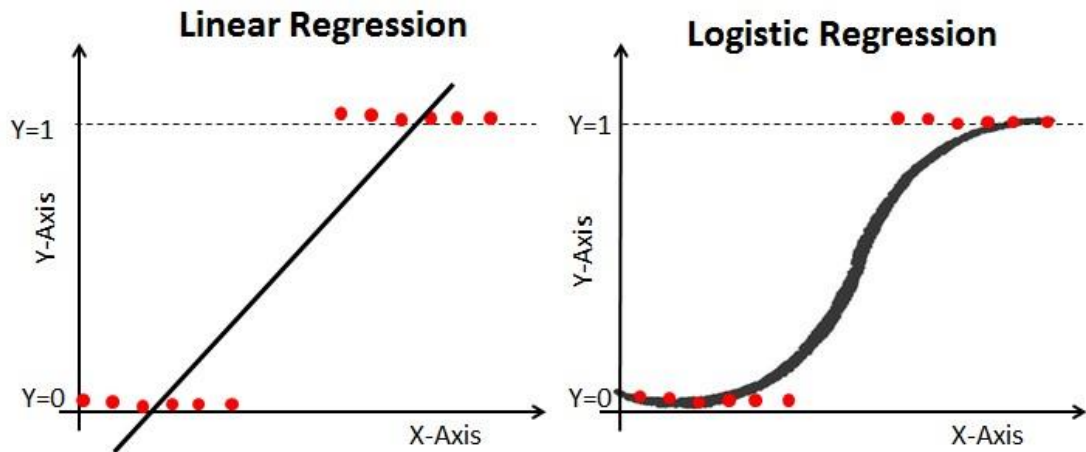


Figure-3.6: Graphs of Linear and Logistic Regression

3.2.2.3 Random Forest

Random Forest is a type of ensemble algorithm, meaning that it combines more than one algorithm for classification. For example, running SVM, Naïve Bayes and Decision Tree and the final consideration of class for an object is done on the bases of vote.

A random forest becomes more accurate if the number of decision trees is higher. One unique feature that Random Forest has is that it can work well both for classifications and regression analysis, which contributes to a large portion of Machine Learning. What makes Random Forest robust is that while growing the trees, it silently contributes to increasing randomness of a model. Rather than splitting from the most important feature, it chooses the splitting point based on the best feature among a set of features.

Total number of decision trees which are to be generated is the basic parameter to the Random Forest. Other parameters include minimum split, split criteria etc.

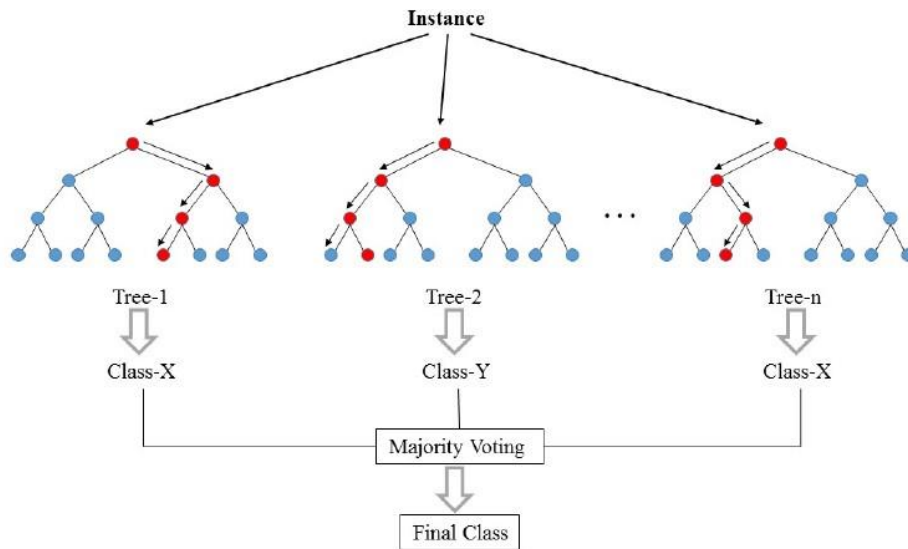


Figure 3.7: How a Random Forest works?

3.2.2.4 Naïve Bayes

Machine Learning revolves around in finding the best hypothesis, h , for the given data d . For classification, h could be the in which we want to add a new instance d . Using Bayes Theorem, the probability of a hypothesis can be calculated if the prior knowledge is given.

Bayes Theorem:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Likelihood
Class Prior Probability
Posterior Probability
Predictor Prior Probability

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

Here,

- $P(c|x)$ is the posterior probability of target given attribute.

- $P(x)$ is the prior probability of predictor.
- $P(c)$ is the probability of class,

3.3 Feature scaling

The features that are used to train the model have a deep relationship with the performance one can achieve. Feature scaling is a core topic of Data Science as it can affect the performance of the model. Feature Scaling is used to standardize the variables of a dataset. There are some features that are irrelevant/partially relevant and can hinder your model's performance. Feature scaling in some cases can be proved to be very beneficial.

Disadvantages:

- Feature scaling doesn't guarantee improved performance.
- In some cases, the performance/accuracy of your model may decrease

3.3.1 Methods

3.3.1.1 Min-Max Normalization:

It rescales the range of features in the range of $[-1, 1]$ or $[0, 1]$. It is one of the easiest implementation in which the range we want to target depends on the nature of the data.

General Formula:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)},$$

Where x is an original value and x' is the normalized value

3.3.1.2 Mean Normalization:

General Formula:

$$x' = \frac{x - \text{average}(x)}{\max(x) - \min(x)},$$

where x is an original value and x' is the normalized value.

3.3.1.3 Standardization:

It is a way of normalization in which we require mean, standard deviation as inputs,

General Formula:

$$x' = \frac{x - \bar{x}}{\sigma}$$

Where x is the original feature vector, $\bar{x} = \text{average}(x)$ is the sum of mean of that feature and σ is its standard deviation.

3.4 Combination of resampling and ensemble methods:

In my project. I have combined some of the resampling techniques with ensemble classifiers as resampling of imbalanced dataset will improve the overall accuracy of the classifiers. The classifier that I have made are listed below:

1. Logistic regression with SMOTE
2. Logistic Regression with Random-Under-Sampling
3. Logistic regression without SMOTE
4. Naïve Bayes regression with SMOTE
5. Naïve Bayes with Random-Under-Sampling
6. Naïve Bayes without SMOTE
7. Random Forest with SMOTE
8. Random Forest with Random-Under-Sampling
9. Random Forest without SMOTE

Chapter 4:

Implementations and Results

Chapter 4: Implementations and Results

4.1 Dataset Used

The datasets contain transactions made by credit cards in September 2013 by European cardholders. This dataset shows transactions that happened in two days, where 492 transactions out of 284,807 transactions are fraudulent. The dataset is profoundly unbalanced; the positive class represent 0.172% all things considered.

It contains just numerical information variables which are the consequence of a PCA transformation. Due to security issues of the users, the background information about the data is not provided. The only features which have not been converted with PCA are ‘Amount’ and ‘Time’.

Total 31 features are present in this dataset:

1. **V1, V2, V28** features are the principal components obtained after applying PCA.
2. **Amount** is the transaction amount.
3. **Time** represents the seconds elapsed between *n*th transaction and the first transaction in the dataset.

Time	Amount	Class	V1	V2	V3	V4	V5
0	149.62	0	-1.35981	-0.07278	2.536347	1.378155	-0.33832
0	2.69	0	1.191857	0.266151	0.16648	0.448154	0.060018
1	378.66	0	-1.35835	-1.34016	1.773209	0.37978	-0.5032
1	123.5	0	-0.96627	-0.18523	1.792993	-0.86329	-0.01031
2	69.99	0	-1.15823	0.877737	1.548718	0.403034	-0.40719
2	3.67	0	-0.42597	0.960523	1.141109	-0.16825	0.420987
4	4.99	0	1.229658	0.141004	0.045371	1.202613	0.191881
7	40.8	0	-0.64427	1.417964	1.07438	-0.4922	0.948934
7	93.2	0	-0.89429	0.286157	-0.11319	-0.27153	2.669599

Figure 4.1: Sample (Credit card Dataset Snapshot)

4.2 Performance Parameters

Evaluating a machine learning algorithm is necessary. Simply calculating the classification accuracy is not a valid criterion to judge our model. I have discussed two techniques below which are more reliable in evaluating my models:

4.2.1 F1 Score

Also known as F Score or F measure, it is used to test a model's accuracy. It takes into account the precision, p , and recall, r to calculate an overall score, which is done by taking the harmonic mean of p and r . p can be formulated as the number of correct positive results divided by the number of all positive results which are returned by the classifiers.

Mathematically,

$$F_1 = \left(\frac{\text{recall}^{-1} + \text{precision}^{-1}}{2} \right)^{-1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

One drawback of F1 score is that it gives equal importance to both precision and recall, which in many datasets is not always true.

4.3 Implementation

In my implementation I have combined different resampling and classification algorithms and then I have calculated the F1 scores and a comparative table is shown in the results section. Case 1 includes all the combinations of algorithms without Feature Scaling and in Case 2, Feature Scaling of the dataset were done first.

Case 1: Without Feature Scaling-

- **Logistic Regression with SMOTE:**

Applying resampling over the dataset using Synthetic Minority Over-Sampling Technique and then classifying it using Logistic Regression.

- **Logistic Regression with Random-Under-Sampling:**

Applying random Under-Sampling over the dataset and then classifying it using Logistic Regression.

- **Logistic Regression without SMOTE:**

Applying random over sampling over the dataset and then classifying it using Logistic Regression.

- **Naïve Bayes with SMOTE:**

Applying resampling over the dataset using Synthetic Minority Over-Sampling Technique and then classifying it using Naïve Bayes.

- **Naïve Bayes with Random-Under-Sampling:**

Applying random Under-Sampling over the dataset and then classifying it using Naïve Bayes.

- **Naïve Bayes without SMOTE:**

Applying random over sampling over the dataset and then classifying it using Naïve Bayes.

- **Random Forest with SMOTE:**

Applying resampling over the dataset using Synthetic Minority Over-Sampling Technique and then classifying it using Random Forest. This combination performs really well (comparison given in Results section).

- **Random Forest with Random-Under-Sampling:**

Applying random Under-Sampling over the dataset and then classifying it using Random Forest.

- **Random Forest without SMOTE:**

Applying random over sampling over the dataset and then classifying it using Random Forest.

Case 2: All the above mentioned algorithms were run again after normalizing the data using Feature Scaling. Results are shown in section 4.4.

4.4 Results

Case 1: Without Feature scaling-

Algorithm	F1 score
Logistic Regression with SMOTE	0.15113350125944586
Logistic Regression with Random-Under-Sampling	0.07468553459119497
Logistic Regression without SMOTE	0.6404494382022472
Naïve Bayes with SMOTE	0.25517241379310346
Naïve Bayes with Random-Under-Sampling	0.2375601926163724
Naïve Bayes without SMOTE	0.22752293577981647
Random Forest with SMOTE	0.85
Random Forest with Random-Under-Sampling	0.10296586457750419
Random Forest without SMOTE	0.8324324324324324

Table 4.1: Table showing the F1 scores of different algorithms without Feature scaling.

scoreLOG_SMOTE	float64	1	0.15113350125944586
scoreLOG_US	float64	1	0.07468553459119497
scoreLOG_woSMOTE	float64	1	0.6404494382022472
scoreNB_SMOTE	float64	1	0.25517241379310346
scoreNB_US	float64	1	0.2375601926163724
scoreNB_woSMOTE	float64	1	0.22752293577981647
scoreRF_SMOTE	float64	1	0.85
scoreRF_US	float64	1	0.10296586457750419
scoreRF_woSMOTE	float64	1	0.8324324324324324

Figure 4.2: Snapshot of the F1 scores of different algorithms without Feature scaling.

Important Points to consider:

4. In case of Random Forest with SMOTE, the F1 score comes out to be the highest with 0.85; meaning that it will successfully identify a fraudulent transaction with an accuracy of 85%. This is followed by Random Forest without SMOTE which has an accuracy of 83%, which itself is impressive.

5. Logistic Regression with Random-Under-Sampling performs poorest with only 7% accuracy, followed by Random Forest with Random-Under-Sampling with 15% accuracy.

6. Logistic Regression performs best when combined without SMOTE (F1 score comes to be 0.64), Naïve Bayes performs best with SMOTE (0.25 f1 score) and Random Forest performs best with SMOTE (0.85 f1 score).

cmLOG_SMOTE	int64	(2, 2)	[[55861 1000] [11 90]]
cmLOG_US	int64	(2, 2)	[[54513 2348] [6 95]]
cmLOG_woSMOTE	int64	(2, 2)	[[56841 20] [44 57]]
cmNB_SMOTE	int64	(2, 2)	[[56456 405] [27 74]]
cmNB_US	int64	(2, 2)	[[56413 448] [27 74]]
cmNB_woSMOTE	int64	(2, 2)	[[56479 382] [39 62]]
cmRF_SMOTE	int64	(2, 2)	[[56847 14] [16 85]]
cmRF_US	int64	(2, 2)	[[55267 1594] [9 92]]
cmRF_woSMOTE	int64	(2, 2)	[[56854 7] [24 77]]

Figure 4.3: Confusion Matrix without Feature scaling

Case 2: With Feature Scaling:

Algorithm	F1 score
Logistic Regression with SMOTE	0.003708327213981495
Logistic Regression with Random-Under-Sampling:	0.0042449459925188076
Logistic Regression without SMOTE	0.7356321839080459
Naïve Bayes with SMOTE	0.003539947076038764
Naïve Bayes with Random-Under-Sampling	0.003539947076038764
Naïve Bayes without SMOTE	0.12436731742588576
Random Forest with SMOTE	0.01663108214406113
Random Forest with Random-Under-Sampling	0.00401838110963019
Random Forest without SMOTE	0.8324324324324324

Table 4.2: Table showing the F1 scores of different algorithms with Feature scaling.

scoreLOG_SMOTE	float64	1	0.003708327213981495
scoreLOG_US	float64	1	0.0042449459925188076
scoreLOG_woSMOTE	float64	1	0.7356321839080459
scoreNB_SMOTE	float64	1	0.003539947076038764
scoreNB_US	float64	1	0.003539947076038764
scoreNB_woSMOTE	float64	1	0.12436731742588576
scoreRF_SMOTE	float64	1	0.01663108214406113
scoreRF_US	float64	1	0.00401838110963019
scoreRF_woSMOTE	float64	1	0.8324324324324324

Figure 4.4 : Snapshot of the F1 scores of different algorithms with Feature scaling.

The accuracy of this model is significantly deteriorated when I applied feature scaling using mean normalization. Below are some important deductions:

1. Random Forest with SMOTE, that performed best without feature scaling with 0.85 F1 score showed extremely poor results with feature scaling with only 0.016 F1 score. In case of Random Forest without SMOTE, the F1 score comes out to be the highest with 0.83.
2. Random Forest without SMOTE is unaffected with feature scaling and performs equivalent in both the cases.
3. Naïve Bayes with SMOTE and Naïve Bayes with Random-Under-Sampling performed poorest with 0.35% accuracy (i.e. 0.0035 F1 score).

We have seen that feature scaling is giving poor results in this case. A table is given in the next page for easier analysis.

Comparative table showing F1 scores:

Algorithm	F1 score (Without Feature Scaling)	F1 score (With Feature Scaling)
Logistic Regression with SMOTE	0.15113350125944586	0.003708327213981495
Logistic Regression with Random-Under-Sampling:	0.07468553459119497	0.0042449459925188076
Logistic Regression without SMOTE	0.6404494382022472	0.7356321839080459
Naïve Bayes with SMOTE	0.25517241379310346	0.003539947076038764
Naïve Bayes with Random-Under-Sampling	0.2375601926163724	0.003539947076038764
Naïve Bayes without SMOTE	0.22752293577981647	0.12436731742588576
Random Forest with SMOTE	0.85	0.01663108214406113
Random Forest with Random-Under-Sampling	0.10296586457750419	0.00401838110963019
Random Forest without SMOTE	0.8324324324324324	0.8324324324324324

Table 4.3: Table showing the F1 scores with Feature Scaling and without Feature Scaling

4.5 Code

4.5.1 Without Feature scaling:

```
import pandas as pd

dataset=pd.read_csv('creditcard.csv')
x=dataset.iloc[:, :-1].values
y=dataset.iloc[:, -1].values

#splitting the dataset
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.20,random_state=0)

#implementing logistic regression
from sklearn.linear_model import LogisticRegression
classifierLOG_woSMOTE=LogisticRegression(random_state=0)
classifierLOG_woSMOTE.fit(xtrain,ytrain)

#implementing naive bayes
from sklearn.naive_bayes import GaussianNB
classifierNB_woSMOTE=GaussianNB()
classifierNB_woSMOTE.fit(xtrain,ytrain)

#implementing random forest
from sklearn.ensemble import RandomForestClassifier
classifierRF_woSMOTE=RandomForestClassifier(n_estimators=10,criterion='entropy',r
andom_state=0)
classifierRF_woSMOTE.fit(xtrain,ytrain)

#predicting test results
ypredRF_woSMOTE=classifierRF_woSMOTE.predict(xtest)
ypredNB_woSMOTE=classifierNB_woSMOTE.predict(xtest)
```

```

ypredLOG_woSMOTE=classifierLOG_woSMOTE.predict(xtest)

#confusion matrix
from sklearn.metrics import confusion_matrix
cmRF_woSMOTE=confusion_matrix(ytest,ypredRF_woSMOTE)
cmNB_woSMOTE=confusion_matrix(ytest,ypredNB_woSMOTE)
cmLOG_woSMOTE=confusion_matrix(ytest,ypredLOG_woSMOTE)

#implementing f1 score
from sklearn.metrics import f1_score
scoreRF_woSMOTE = f1_score(ypredRF_woSMOTE, ytest)
scoreNB_woSMOTE = f1_score(ypredNB_woSMOTE, ytest)
scoreLOG_woSMOTE = f1_score(ypredLOG_woSMOTE, ytest)

#implementing SMOTE
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state=2)
xtrain_SMOTE, ytrain_SMOTE = sm.fit_sample(xtrain, ytrain.ravel())

#implementing random forest
classifierRF_SMOTE=RandomForestClassifier(n_estimators=10,criterion='entropy',ran
dom_state=0)
classifierRF_SMOTE.fit(xtrain_SMOTE,ytrain_SMOTE)

#implementing naive bayes
classifierNB_SMOTE=GaussianNB()
classifierNB_SMOTE.fit(xtrain_SMOTE,ytrain_SMOTE)

#implementing logistic regression
classifierLOG_SMOTE=LogisticRegression(random_state=0)
classifierLOG_SMOTE.fit(xtrain_SMOTE,ytrain_SMOTE)

#predicting test results

```

```

ypredRF_SMOTE=classifierRF_SMOTE.predict(xtest)
ypredNB_SMOTE=classifierNB_SMOTE.predict(xtest)
ypredLOG_SMOTE=classifierLOG_SMOTE.predict(xtest)

#confusion matrix
cmRF_SMOTE=confusion_matrix(ytest,ypredRF_SMOTE)
cmNB_SMOTE=confusion_matrix(ytest,ypredNB_SMOTE)
cmLOG_SMOTE=confusion_matrix(ytest,ypredLOG_SMOTE)

#implementing f1 score
scoreRF_SMOTE = f1_score(ypredRF_SMOTE, ytest)
scoreNB_SMOTE = f1_score(ypredNB_SMOTE, ytest)
scoreLOG_SMOTE = f1_score(ypredLOG_SMOTE, ytest)

#implementing Under-Sampling
from imblearn.under_sampling import RandomUnderSampler
us = RandomUnderSampler(random_state=0)
xtrain_US, ytrain_US = us.fit_sample(xtrain, ytrain.ravel())

#implementing naive bayes
classifierNB_US=GaussianNB()
classifierNB_US.fit(xtrain_US,ytrain_US)

#implementing logistic regression
classifierLOG_US=LogisticRegression(random_state=0)
classifierLOG_US.fit(xtrain_US,ytrain_US)

#implementing random forest
classifierRF_US=RandomForestClassifier(n_estimators=10,criterion='entropy',random_
state=0)
classifierRF_US.fit(xtrain_US,ytrain_US)

#predicting test results

```

```

ypredRF_US=classifierRF_US.predict(xtest)
ypredNB_US=classifierNB_US.predict(xtest)
ypredLOG_US=classifierLOG_US.predict(xtest)

#confusion matrix
cmRF_US=confusion_matrix(ytest,ypredRF_US)
cmLOG_US=confusion_matrix(ytest,ypredLOG_US)
cmNB_US=confusion_matrix(ytest,ypredNB_US)

#implementing f1 score
scoreRF_US = f1_score(ypredRF_US, ytest)
scoreLOG_US = f1_score(ypredLOG_US, ytest)
scoreNB_US = f1_score(ypredNB_US, ytest)

```

4.5.2 With Feature scaling:

```

# -*- coding: utf-8 -*-
import pandas as pd

dataset=pd.read_csv('creditcard.csv')
x=dataset.iloc[:, :-1].values
y=dataset.iloc[:, -1].values

#splitting the dataset
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.20,random_state=0)

#feature scaling
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
xtrain=sc.fit_transform(xtrain)
xtest=sc.fit_transform(xtest)

```

```

#implementing naive bayes
from sklearn.naive_bayes import GaussianNB
classifierNB_woSMOTE=GaussianNB()
classifierNB_woSMOTE.fit(xtrain,ytrain)

#implementing logistic regression
from sklearn.linear_model import LogisticRegression
classifierLOG_woSMOTE=LogisticRegression(random_state=0)
classifierLOG_woSMOTE.fit(xtrain,ytrain)

#implementing random forest
from sklearn.ensemble import RandomForestClassifier
classifierRF_woSMOTE=RandomForestClassifier(n_estimators=10,criterion='entropy',r
andom_state=0)
classifierRF_woSMOTE.fit(xtrain,ytrain)

#predicting test results
ypredRF_woSMOTE=classifierRF_woSMOTE.predict(xtest)
ypredNB_woSMOTE=classifierNB_woSMOTE.predict(xtest)
ypredLOG_woSMOTE=classifierLOG_woSMOTE.predict(xtest)

#confusion matrix
from sklearn.metrics import confusion_matrix
cmRF_woSMOTE=confusion_matrix(ytest,ypredRF_woSMOTE)
cmNB_woSMOTE=confusion_matrix(ytest,ypredNB_woSMOTE)
cmLOG_woSMOTE=confusion_matrix(ytest,ypredLOG_woSMOTE)

#implementing f1 score
from sklearn.metrics import f1_score
scoreRF_woSMOTE = f1_score(ypredRF_woSMOTE, ytest)
scoreNB_woSMOTE = f1_score(ypredNB_woSMOTE, ytest)
scoreLOG_woSMOTE = f1_score(ypredLOG_woSMOTE, ytest)

```

```

#implementing SMOTE
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state=2)
xtrain_SMOTE, ytrain_SMOTE = sm.fit_sample(xtrain, ytrain.ravel())

#feature scaling
sc_SMOTE=StandardScaler()
xtrain_SMOTE=sc_SMOTE.fit_transform(xtrain_SMOTE)

#implementing naive bayes
classifierNB_SMOTE=GaussianNB()
classifierNB_SMOTE.fit(xtrain_SMOTE,ytrain_SMOTE)

#implementing logistic regression
classifierLOG_SMOTE=LogisticRegression(random_state=0)
classifierLOG_SMOTE.fit(xtrain_SMOTE,ytrain_SMOTE)

#implementing random forest
classifierRF_SMOTE=RandomForestClassifier(n_estimators=10,criterion='entropy',ran
dom_state=0)
classifierRF_SMOTE.fit(xtrain_SMOTE,ytrain_SMOTE)

#predicting test results
ypredRF_SMOTE=classifierRF_SMOTE.predict(xtest)
ypredNB_SMOTE=classifierNB_SMOTE.predict(xtest)
ypredLOG_SMOTE=classifierLOG_SMOTE.predict(xtest)

#confusion matrix
cmRF_SMOTE=confusion_matrix(ytest,ypredRF_SMOTE)
cmNB_SMOTE=confusion_matrix(ytest,ypredNB_SMOTE)
cmLOG_SMOTE=confusion_matrix(ytest,ypredLOG_SMOTE)

#implementing f1 score

```

```
scoreRF_SMOTE = f1_score(ypredRF_SMOTE, ytest)
scoreNB_SMOTE = f1_score(ypredNB_SMOTE, ytest)
scoreLOG_SMOTE = f1_score(ypredLOG_SMOTE, ytest)
```

```
#implementing Under-Sampling
```

```
from imblearn.under_sampling import RandomUnderSampler
us = RandomUnderSampler(random_state=0)
xtrain_US, ytrain_US = us.fit_sample(xtrain, ytrain.ravel())
```

```
sc_US=StandardScaler()
xtrain_US=sc_US.fit_transform(xtrain_US)
```

```
#implementing naive bayes
```

```
classifierNB_US=GaussianNB()
classifierNB_US.fit(xtrain_US,ytrain_US)
```

```
#implementing logistic regression
```

```
classifierLOG_US=LogisticRegression(random_state=0)
classifierLOG_US.fit(xtrain_US,ytrain_US)
```

```
#implementing random forest
```

```
classifierRF_US=RandomForestClassifier(n_estimators=10,criterion='entropy',random_
state=0)
classifierRF_US.fit(xtrain_US,ytrain_US)
```

```
#predicting test results
```

```
ypredRF_US=classifierRF_US.predict(xtest)
ypredNB_US=classifierNB_US.predict(xtest)
ypredLOG_US=classifierLOG_US.predict(xtest)
```

```
#confusion matrix
cmRF_US=confusion_matrix(ytest,ypredRF_US)
cmNB_US=confusion_matrix(ytest,ypredNB_US)
cmLOG_US=confusion_matrix(ytest,ypredLOG_US)

#implementing f1 score
scoreRF_US = f1_score(ypredRF_US, ytest)
scoreNB_US = f1_score(ypredNB_US, ytest)
scoreLOG_US = f1_score(ypredLOG_US, ytest)
```


Chapter 5:

Conclusions

Chapter 5: Conclusions

5.1 Conclusions

Credit cards have become a great source of money for fraudsters. This is due to the increase in the use of credit cards. Everyone prefers using credit cards because it makes our life easy. These methods that I have explained prove accurate in deducting fraudulent transaction and minimizing the number of false alert. If this algorithm is applied into bank credit card fraud detection system, the probability of fraud transactions can be predicted soon after credit card transactions. And a series of anti-fraud strategies can be adopted to prevent banks from great losses and reduce risks. Data imbalance is the most common problem. Existing classification algorithms underperform on the imbalance data, so we need to pre-process the data and make it balanced. Methods for making data balance are Sampling and Cost sensitive learning. At data level, sampling is the most common approach to deal with imbalanced data. over- sampling clearly appears as better than under-sampling for local classifiers, whereas some under-sampling strategies outperform Over-Sampling when employing classifiers with global learning.

However, making a hybrid model by combining a classification algorithm with resampling (any method) shows some positive results.

5.2 Future Scope

The findings obtained here may not be generalized to the global fraud detection 25problem. As future work, some effective algorithm which can perform well for the classification problem with variable misclassification costs could be developed.

References

1. *Usamma Fayad, Gregorrry Patetsky Sapiro and Pahraic Smith:*
“Learning from Imbalanced Data Set using various techniques”, Volume 39, Nov. 2015, Page 29-34
2. *Xindongg Wou, Xigquan Zuh, Gongg-Xing Wuh, and Wey Dingh:*
“Using Random Forest to Learn Imbalanced Data” IEEE Transactions on Imbalanced Data Engineering, Volume26 Issue 1, January 2014 Pages 97-107
3. *Apoorva Aggarwal, Boy Xi Ila Vosha, Oven Rambov Rebeca:*
“A Survey on Performance Measures in Machine Learning” IJCSMC, Vol. 4, Issue. 11, November 2015, pg.338 – 343.
4. *C. Phuah, D. Lee, P. Smith, A. Gayler:*
“A quick introduction to Feature Scaling” Proceedings of the International Conference on System Science, 2005.
5. *. Chao Chenn:*
“Using Random Forest to Learn Imbalanced Data”, March 2013
6. *H. Han, W. Y. Wang, B. H. Mao:*
“SMOTE: A new Over-Sampling method in imbalanced data sets learning,” International Conference on Intelligent Computing (ICIC 2005), Aug. 2005, pp. 878-887
7. *David A. Cieslak, Nitesh V. Chawla:*
“Learning Decision Trees for Unbalanced Data”, 2008

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: 10 May 2019

Type of Document (Tick): PhD Thesis M.Tech Dissertation/ Report B.Tech Project Report Paper

Name: Chandan Parthap Singh Department: Information Technology Enrolment No 151456

Contact No. 8629009953 E-mail. cp0000@ymail.com

Name of the Supervisor: Dr. Hari Singh

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters):
ANALYSIS OF IMBALANCED DATA.

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/ revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.


Complete Thesis/Report Pages Detail:

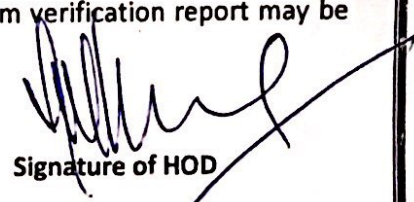
- Total No. of Pages = 51
- Total No. of Preliminary pages = ~~4~~ 7
- Total No. of pages accommodate bibliography/references = 1


(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found Similarity Index at 13.0.....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) 
19/5/19

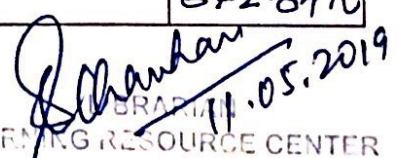

Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
<u>10-05-19</u>	<ul style="list-style-type: none"> • All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String 	<u>13%</u>	Word Counts	<u>5031</u>
Report Generated on		Submission ID	Character Counts	<u>32787</u>
<u>11-05-19</u>		<u>1128753681</u>	Total Pages Scanned	<u>50</u>
			File Size	<u>672.84K</u>

Checked by 
Name & Signature


LIBRARIAN 11.05.2019
LEARNING RESOURCE CENTER
Jaypee University of Information Technology
Waknaghat, Distt, U.S.P. (Uttarakhand Pradesh)
Pin Code: 173235

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com