ISSN No:-2456-2165

Enhancing Credit Card Fraud Detection with Regularized Generalized Linear Models: A Comparative Analysis of Down-Sampling and Up-Sampling Techniques

[1]Cyril Neba C., [2]Gerard Shu F. , [3]Adrian Neba F., [4]Aderonke Adebisi, [5]P. Kibet., [6]F.Webnda, [7]Philip Amouda A.
[1,3,4,5,6,7]Department of Mathematics and Computer Science, Austin Peay State University, Clarksville, Tennessee, USA
[2]Montana State University, Gianforte School of Computing, Bozeman, Montana, USA

Abstract:- This study highlights the problem of credit card fraud and the use of regularized generalized linear models (GLMs) to detect fraud. GLMs are flexible statistical frameworks that model the relationship between a response variable and a set of predictor variables. Regularization Techniques such as ridge regression, lasso regression, and Elasticnet can help mitigate overfitting, resulting in a more parsimonious and interpretable model. The study used a credit card transaction dataset from September 2013, which included 492 fraud cases out of 284,807 transactions.

The rising prevalence of credit card fraud has led to the development of sophisticated detection methods, with machine learning playing a pivotal role. In this study, we employed three machine learning models: Ridge Regression, Elasticnet Regression, and Lasso Regression, to detect credit card fraud using both Down-Sampling and Up-Sampling techniques. The results indicate that all three models exhibit accuracy in credit card fraud detection. Among them, Ridge Regression stands out as the most accurate model, achieving an impressive 98% accuracy with a 95% confidence interval between 97% and 99%. Following closely, Lasso Regression and Elasticnet Regression both demonstrate solid performance, with accuracy rates of 93.2% and 93.1%, respectively, and 95% confidence intervals ranging from 88% to 98%.

When considering the Up-Sampling technique, Ridge Regression maintains its position as the most accurate model, achieving an accuracy rate of 98.2% with a 95% confidence interval spanning from 97% to 99.4%. Elasticnet Regression follows with an accuracy rate of 94.1% and a confidence interval between 0.8959 and 0.9854, while Lasso Regression exhibits a slightly lower accuracy of 93.1% with a confidence interval from 0.88 to 0.982. all three machine learning models—Ridge Regression, Elasticnet Regression, and Lasso Regression—demonstrate competence in credit card fraud detection. Ridge Regression consistently outperforms the others in both Down-Sampling and Up-Sampling scenarios, making it a valuable tool for financial institutions to safeguard against credit card fraud threats in the United States.

Keywords:- *Machine Learning, Credit Card Transaction Fraud Detection, Regularized GLM, Ridge Regression, Elasticnet Regression, Lasso Regression, Down-Sampling and Up-Sampling.*

I. INTRODUCTION

A notable change in consumer financial services over the past few decades has been the growth of the use of credit cards, both for payments and as sources of revolving credit. From modest origins in the 1950s as a convenient way for the relatively well-to-do to settle restaurant and department store purchases without carrying cash, credit cards have become a ubiquitous financial product held by households in all economic strata (Credit Cards: Use and Consumer Attitudes, 1970-2000, 2000)".

Credit card fraud is a prevalent and challenging problem for financial institutions and consumers worldwide. In 2021, the Federal Trade Commission (FTC) fielded nearly 390,000 reports of credit card fraud, making it one of the most common kinds of fraud in the U.S. (Federal Trade Commission, 2020) and consistently ranks among the top consumer complaints, with thousands of cases reported each year. In 2020 alone, the FTC received over 2.2 million fraud reports, with identity theft and credit card fraud being among the most frequently cited issues (Federal Trade Commission, 2021).

Fraudulent transactions can cause significant financial losses, harm the reputation of financial institutions, and create inconvenience and stress for customers. Therefore, detecting and preventing credit card fraud is of utmost importance. According to the Nilson Report (December 2021), global credit card and debit card fraud resulted in losses of $28.58 billion during 2020, with card issuers and merchants incurring 88% and 12% of those losses, respectively. Card issuer losses occurred mainly at the point of sale from counterfeit cards while merchant losses occurred mainly on card-not-present (CNP) transactions. The report also noted that during 2020, credit card and debit card gross fraud losses accounted for roughly 6.81¢ per $100 in total volume, up from 6.78¢ per $100 in 2019. In 2020, the US accounted for 35.83% of the worldwide payment card fraud losses but generated only 22.40% of total volume. Finally, the Nilson Report predicted that over the next 10 years, card industry losses to fraud will collectively amount to $408.50 billion.

ISSN No:-2456-2165

II. METHODS OF CREDIT CARD FRAUD DETECTION

Detecting and preventing credit card fraud is an ongoing challenge, but various methods have been developed to mitigate its impact. One increasingly vital approach is the utilization of machine learning and data analytics. These techniques enable the analysis of vast datasets, identifying suspicious patterns and anomalies that may indicate fraudulent activity.

- **Anomaly Detection**: Anomaly detection algorithms, such as clustering and autoencoders, are used to flag transactions that significantly deviate from the norm. These anomalies may indicate fraudulent activities, and machine learning models can assign risk scores to transactions based on their deviation from established patterns (Ahmed et al., 2016).
- **Behavioral Analysis**: Machine learning can analyze a user's behavior over time, creating a profile of their typical spending habits. Any sudden deviations from this profile can trigger alerts for further investigation (Ahmed et al., 2016).
- **Network Analysis**: Credit card fraud often involves coordinated efforts by criminal networks. Machine learning models can analyze transaction networks to identify links between seemingly unrelated accounts and transactions, uncovering hidden patterns (Gandomi & Haider, 2015).
- **Real-time Monitoring**: Machine learning models can operate in real-time, allowing for the immediate detection of potentially fraudulent transactions. This rapid response is crucial in preventing losses (Gandomi & Haider, 2015).
- **Natural Language Processing (NLP)**: NLP techniques can be employed to analyze text data, such as customer service interactions and transaction comments, to identify suspicious language or phrases associated with fraud (Dinakar & Nair, 2020).
- **Pattern Recognition**: Machine learning algorithms can be trained to recognize patterns associated with fraudulent transactions. By analyzing historical data, these models can identify deviations from typical spending behavior, such as unusual purchase locations, transaction amounts, or frequencies (Acar et al., 2017). One pattern recognition approach to detect credit card fraud is to use Machine Learning Techniques such as a generalized linear model (GLM), which is a flexible statistical framework that allows modeling the relationship between a response variable and a set of predictor variables. However, GLMs can suffer from overfitting, which occurs when the model is too complex and fits the noise in the data instead of the underlying signal. Regularization Techniques can help mitigate overfitting by adding a penalty term to the model's objective function that discourages large coefficients.

In this context, regularized forms of GLMs, such as ridge regression and lasso regression, can be useful tools for detecting credit card fraud. These Techniques allow the model to shrink the coefficients of less important predictors, leading to a more parsimonious and interpretable model that is less prone to overfitting. Furthermore, regularized GLMs can handle high-dimensional data with many predictors, a common scenario in credit card fraud detection, where there are numerous features that may be relevant to identifying fraudulent transactions.

For this study, we used the credit card transaction dataset from September 2013 which includes transactions made by European cardholders (Pozzolo, Caelen, Johnson, and Bontempi; 2015). This dataset covers a two-day period and includes 492 fraud cases out of a total of 284,807 transactions. The dataset is considered unbalanced because the positive class (frauds) accounts for only 0.172% of all transactions. The input variables in the dataset are numerical and have been transformed using PCA. The original features and additional background information about the data cannot be disclosed due to confidentiality concerns. The dataset includes 28 principal components obtained through PCA, and the 'Time' and 'Amount' features have not been transformed. 'Time' indicates the time in seconds between a given transaction and the first transaction in the dataset, while 'Amount' indicates the transaction amount and can be used for cost-sensitive learning. The response variable, 'Class', takes a value of 1 for fraud cases and 0 for non-fraud cases.

III. A GENERAL REVIEW ON CREDIT CARD FRAUD DETECTION TECHNIQUES

According to Hanagandi, Dhar, and Buescher (1996), historical information on credit card transactions was used to develop a fraud score model using a radial basis function network and a density-based clustering approach. The authors applied this methodology to a fraud detection problem and reported satisfactory preliminary results. The paper is considered an early example of using machine learning Techniques for fraud detection in credit card transactions, which has since become an important application area of machine learning and data analytics.

(Dorronsoro et al., 1997) developed an online system for detecting credit card fraud using a neural classifier, which was constructed using a nonlinear version of Fisher's discriminant analysis. The authors reported that the system is currently fully operational and can handle more than 12 million credit card operations per year, with satisfactory results obtained.

Bentley et al. (2000) proposed a genetic programming-based algorithm for classifying credit card transactions into suspicious and non-suspicious categories using logic rules. The algorithm was tested on a database of 4,000 transactions with 62 fields, and the most effective rule was selected based on its predictability. This algorithm has shown promise in detecting credit card fraud, particularly in the context of home insurance data. Nonetheless, given the constantly changing nature of fraud tactics, new and advanced fraud detection methods are continuously being developed to keep pace with this evolving field.

IV. METHODOLOGY

The methodology for building the fraud detection model involves data exploration and cleaning, data preprocessing, model building, model evaluation, and interpretation. The data was checked for missing values, outliers, and correlations between predictor variables.

ISSN No:-2456-2165

Standardization was used to avoid bias and the data was split into training and testing sets followed by modeling using down-sample and up-sample Techniques. Logistic regression with L1 regularization (Lasso Regularization), L2 regularization (Ridge Regularization) and a combination of L1 and L2 regularization also known as Elasticnet Regularization was used and hyperparameters were tuned using cross-validation. Performance metrics like accuracy, precision, recall, F1-score, ROC-AUC, etc. were used to evaluate the model, and the coefficients were interpreted to understand the impact of predictor variables.

A. Exploratory Data Analysis

This dataset contains 31 variables, including the response variable (Class). The first 30 variables (V1 to V30) represent numerical variables that have been transformed using PCA. The variables V1 to V28 represent the principal components, while V29 and V30 are the residuals from the PCA transformation. The Amount variable is a numerical variable representing the transaction amount, and the Class variable is a binary variable representing whether the transaction is fraudulent or not (1 for fraud, 0 for not fraud).

Table 1: Structure of Variables

Variable	**Description**
V1 to V28	Numeric variables representing different aspects of the transaction such as amount, time, location, etc.
Amount	Numeric variable representing the amount of the transaction.
Class	Binary variable indicating whether the transaction was fraudulent (1) or not (0).

➢ *Checking Missing Data*

The following R snipet was used to extract all rows from the "credit_card" data frame that have missing values and looking at the output, we realize that the dataset has no missing values.

```
credit_card[!complete.cases(credit_card),]
[1] Time  V1    V2    V3    V4    V5    V6    V7    V8    V9    V10   V11   V12   V13   V14   V15   V16
[18] V17   V18   V19   V20   V21   V22   V23   V24   V25   V26   V27   V28   Amount Class

<0 rows> (or 0-length row.names)
```

To confirm the absence of missing values in the dataset, the missing values in a data frame were visualized using the "naniar" package in R and again, we realize that there are no missing values in the dataset as can be seen on the plot below.

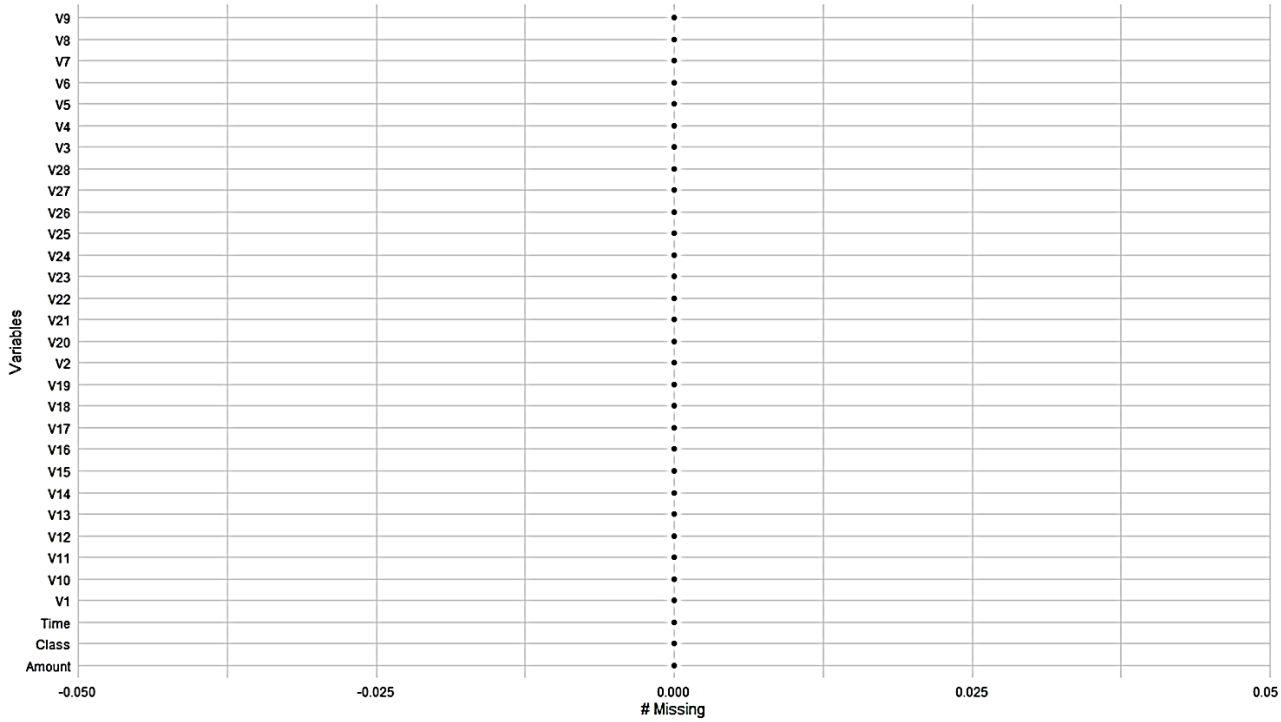

Fig. 1: Visualization of Missing Values in the dataset

➢ *Frequency Distribution of "Class" Variable*

The dataset is highly imbalanced, the positive class (frauds) accounts for just 0.1727486% of total transactions (that is 492 out of a total of 284807 transactions) which is not suitable for this project hence we will need to balance the data before proceeding with our Logistic regression.

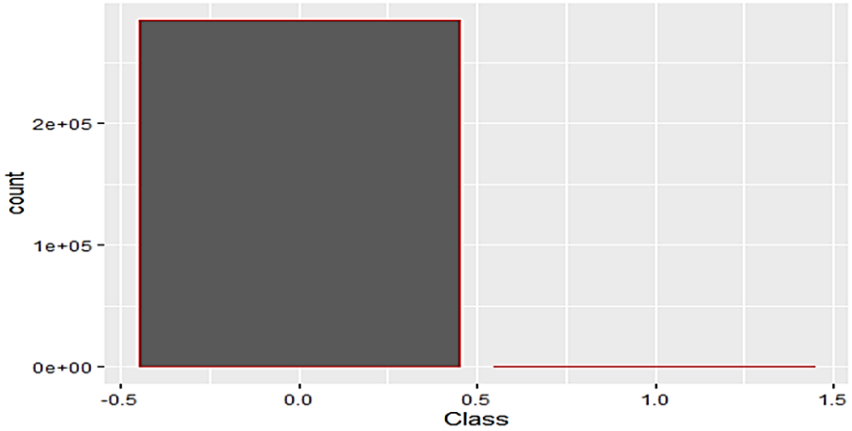

Fig. 2: Frequency distribution of "Class" Variable

Looking at Fig 3.below, , we realize that all the fraudulent transactions were made for less than around 2500, which is far less than that of the true transactions thereby confirming the imbalance nature of the dataset.

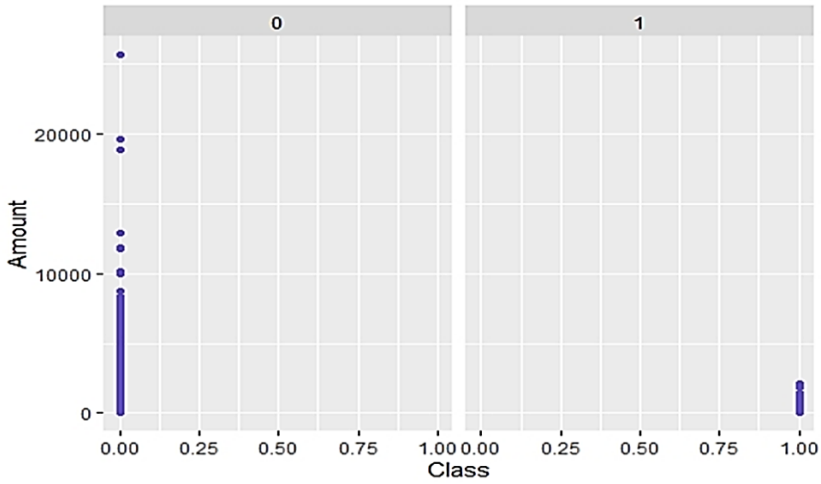

Fig. 3: Amount Vs Class Fraud chart

➢ *Checking Correlation Between the Variables*

Looking at Fig4. below, we realized that there is extremely very little or no correlation among the variables except for V2 which has some negative correlation with amount. We can nonetheless ignore it since it will not affect the outcome of our analysis in any significant way.

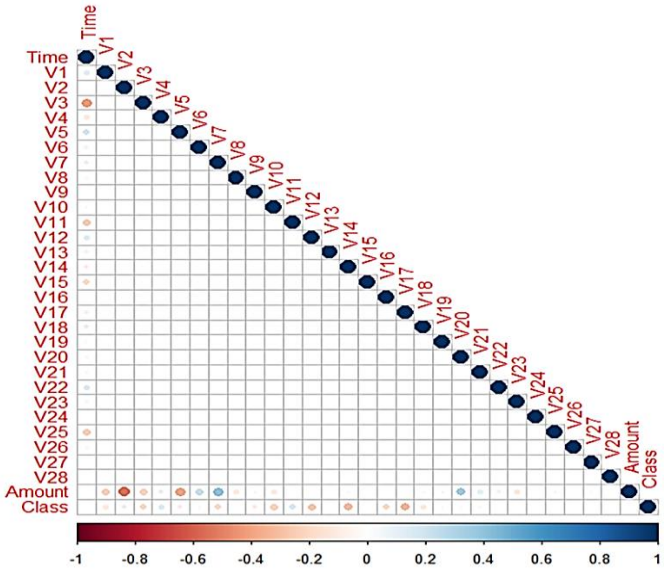

Fig. 4: Correlation Between Variables

ISSN No:-2456-2165

➢ *Data Processing*

Data processing is an important step in machine learning because the quality of the data used to train a model can significantly impact the accuracy and performance of the resulting model. The following data processing steps were carried out:

➢ *Data Cleaning and Feature Selection:*

The raw data did not contain any missing values, hence, there was no need for any data imputation. The Time Variable was irrelevant or unimportant to the outcome we are trying to predict and since it did not have a significant impact on the outcome, we took it out.

➢ *Data Normalization:*

Since the range and distribution of the data could impact the performance of our machine learning algorithms, we Normalized the data to ensure that the model is not biased towards any feature due to differences in scale especially because the data was highly unbalanced. This process was carried out after splitting the dataset into training and testing set.

B. Modeling With Down-Sampling Technique

➢ *Data Partitioning With Down-Sampling Technique*

The dataset was split into training (190450 for Non-fraud Cases and 326 for Fraud Cases) and testing sets with a ratio of 2:1. After splitting the dataset, we balanced the training data using the Down Sampling Technique. By downsampling the data, you are creating a new subset of the original training data where the positive class (i.e., the minority class) is represented more frequently relative to the negative class (i.e., the majority class). This can help address class imbalance issues that can arise in predictive modeling tasks, where one class is significantly more prevalent than the other. After balancing the training data (326 for Non-fraud Cases and 326 for Fraud Cases, we could proceed to building the regularized GLMs.

Top 6

	s0
2	0.1820028
4	0.1732274
5	0.1394861
8	0.1483054
11	0.1540031
13	0.1547350

Bottom 6

	s0
284789	0.1238469
284793	0.1290248
284797	0.2041513
284802	0.1805562
284803	0.1084127
284804	0.1419076

➢ *Model Fitting/Data Modeling*

According to Altman and Marco (1994) and Flitman (1997), an increasing number of statistical models have been applied to data mining tasks, including regression analysis, multiple discriminant analysis, logistic regression, Probit method, and others (Hanagandi, Dhar, & Buescher, 1996). In the context of the credit card dataset, Regulatizd GLMS (Ridge, Lasso and Elasticnet models) can be used to identify the features that are most relevant for detecting fraudulent transactions while also reducing the effects of multicollinearity. These models work by adding a penalty term to the ordinary least squares regression (OLS) objective function, which shrinks the regression coefficients towards zero. The Ridge regression adds the L2 norm of the coefficients as a penalty term, Lasso regression adds the L1 norm of the coefficients as a penalty term, and Elasticnet regression adds a combination of L1 and L2 norm of the coefficients as a penalty term. By adding these penalty terms, these models can reduce the coefficients of some features to zero, effectively eliminating them from the model and thus addressing the issue of multicollinearity.

We performed cross-validation on the model using the cv.glmnet() function from the glmnet package. We then used **the coef(CV, CV$lambda.min)** R snippet to retrieve the coefficient estimates for the Ridge model fit, with the optimal lambda value selected through cross-validation, allowing us to see which variables are most strongly associated with the response variable in the final model. Lastly, we then used the R code snippet **coef(CV, CV$lambda.1se)** to retrieve the coefficient estimates for the model fit with the lambda value selected through cross-validation that is one standard error away from the optimal lambda value, allowing us to see which variables are most strongly associated with the response variable in a more parsimonious model.

➢ *Ridge Regression Model Down-Sampling Technique*

The following Ridge model predictions were obtained.

ISSN No:-2456-2165

- Ridge Regression Model Evaluation Down-Sampling Technique

✓ Miscalculation Rate for Ridge Model for Down-Sampling Technique

The mean of the miscalculation rate is at 0.005966118. A miscalculation rate of 0.005966118 indicates that our model has 0. 5966118% incorrect predictions which means it has 99.4033882% correct predictions hence the model is predicting accurately.

✓ Confidence Interval for The Area Under the Curve(AUC) for Ridge for Down-Sampling Technique

The model has confidence 0.95 of predicting correctly with a confidence interval between 0.966 and 0.992. at an accuracy of 97.9%.

✓ ROC Curve of the best fit Model for Ridge for Down-Sampling Technique

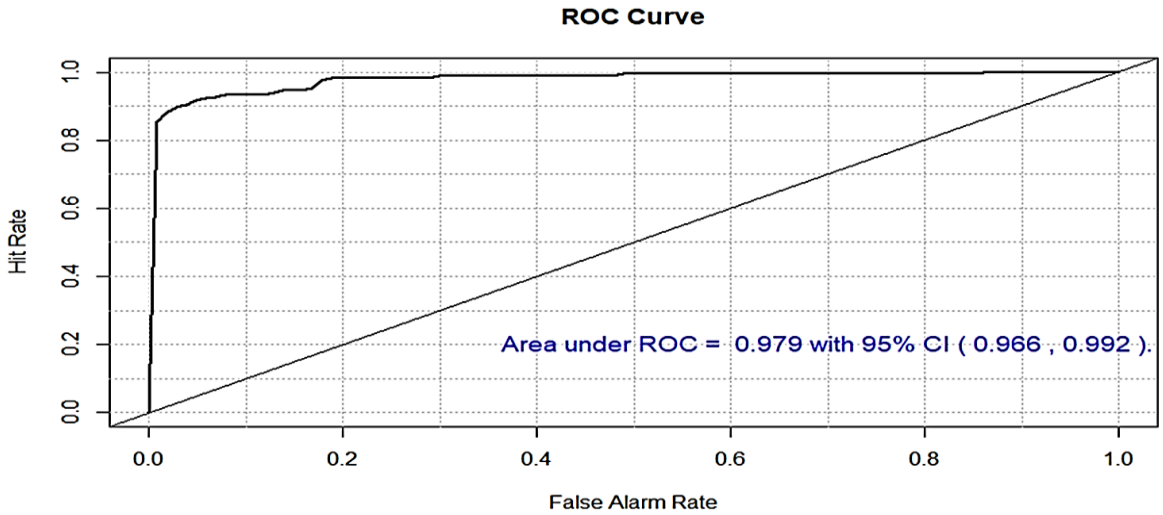

Fig. 5: ROC Curve of the best fit Model for Ridge (Down-Sampling Technique).

The higher AUC -ROC, the better the performance of the model at distinguishing between positive and the negative classes so when the AUC is between o.5 and 1, that is, 0.5<AUC<1, then there is a high chance that then model can distinguish between the positive class values from the negative class values. Since our AUC value is 0.979, we have a 95% confidence Interval between 0.966 and 0.992 that our model can differentiate between is FTP(False Positive Rate) and the TPR(True Positive rate).

✓ Recall and Precision Score for Ridge Model for Down-Sampling Technique

We obtained a precision of 0.994311 which means 99.4311% of our prediction is relevant. We obtained a recall of the recall of 0.9997108 shows that our model has accuracy of 99.9997108 in correctly classifying the total relevant results.

✓ Recall And Precision Curves for Ridge Model for Down-Sampling Technique

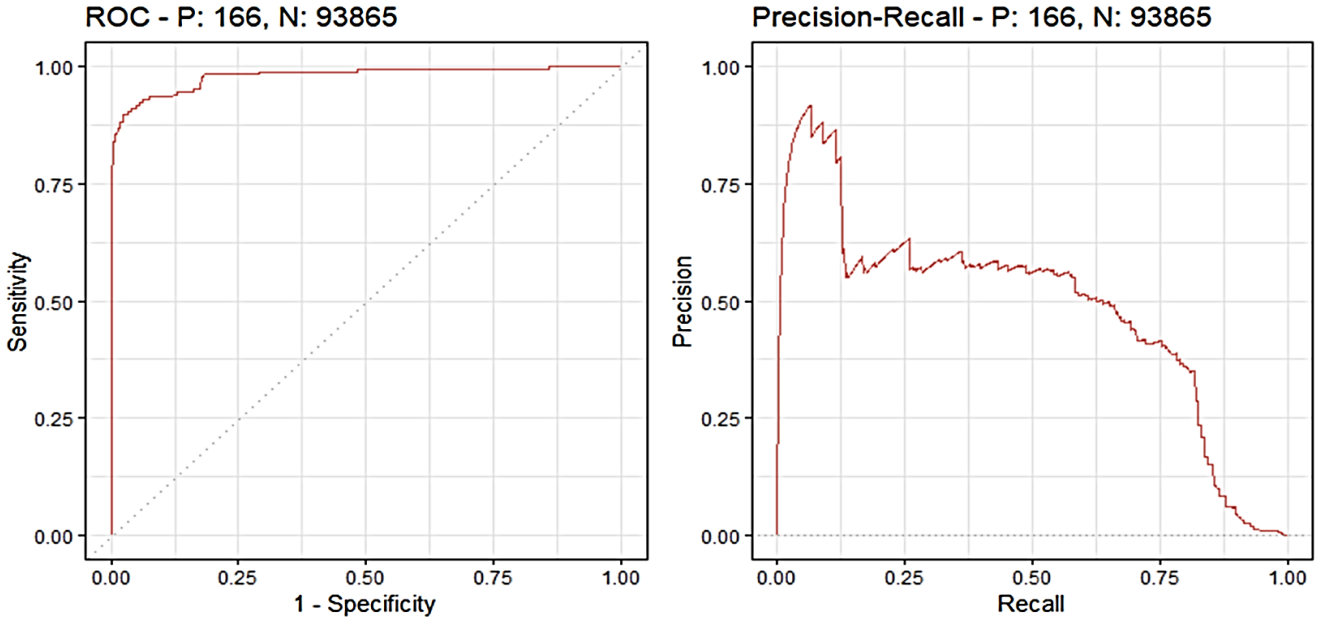

Fig. 6: Recall And Precision Curves for Ridge Model (Down-Sampling Technique)

ISSN No:-2456-2165

These curves give the shape we would expect. At thresholds with low recall, the precision is correspondingly high, and at very high recall, the precision begins to drop. Looking at the Precision-Recall Curve, we notice the curve gets precision up to about 81.

Top 6

	s0
2	**0.14**
4	**0.09**
5	**0.12**
8	**0.12**
11	**0.09**
13	**0.10**

- Elasticnet Model Evaluation Down-Sampling Technique

✓ Miscalculation Rate for Elasticnet Model for Down-Sampling Technique

The mean of the miscalculation rate is at 0.01261286. A miscalculation rate of 0.01261286 indicates that our model has 1.261286 % incorrect predictions which means it has 98.738714% correct predictions hence the model is predicting accurately.

➢ *Elasticnet Regression Model for Down-Sampling Technique*

The following Elasticnet Model predictions were obtained.

Bottom 6

	s0
284789	**0.11**
284793	**0.07**
284797	**0.17**
284802	**0.13**
284803	**0.01**
284804	**0.09**

✓ Confidence Interval for The Area Under the Curve(AUC) foElasticnet for Down-Sampling Technique

The model has confidence 0.95 of predicting correctly with a confidence interval between 0.880 and 0.982 at an accuracy of 93.% .

✓ ROC Curve of the best fit Model for Elasticnet for Down-Sampling Technique

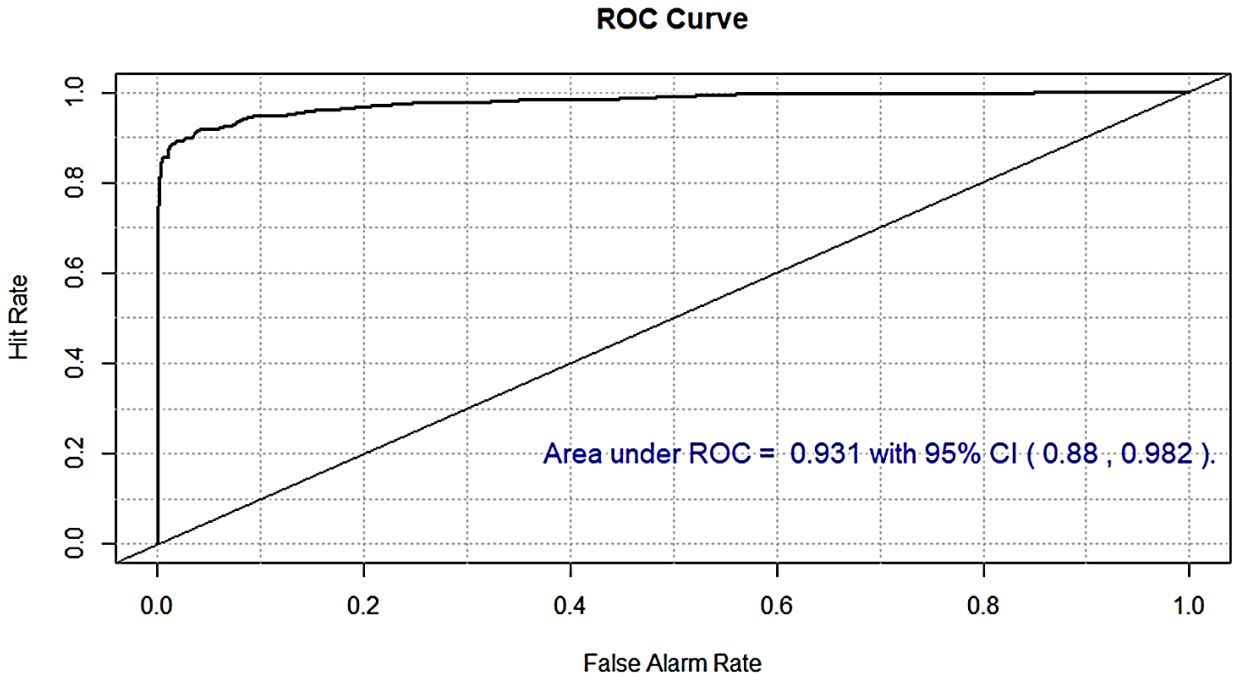

Fig. 7: ROC Curve of the best fit Model for Elasticnet (Down-Sampling Technique)

The higher AUC -ROC, the better the performance of the model at distinguishing between positive and the negative classes so when the AUC is between o.5 and 1, that is, 0.5<AUC<1, then there is a high chance that the model can distinguish between the positive class values from the negative class values. Since our AUC value is 0.931, we have a 95% accuracy and a confidence Interval between 0.88 and 0.982 that our model can differentiate between is FTP(False Positive Rate) and the TPR(True Positive rate)

✓ Recall and Precision Score for Elasticnet Model for Down-Sampling Technique

We obtained a precision of 0.9875886 which means 98.75886% of our prediction is relevant. We obtained a recall of the recall of 0.9997735 which shows that our model has accuracy of 99.97735% in correctly classifying the total relevant results.

✓ Recall and Precision Curves for Elasticnet Model for Down-Sampling Technique

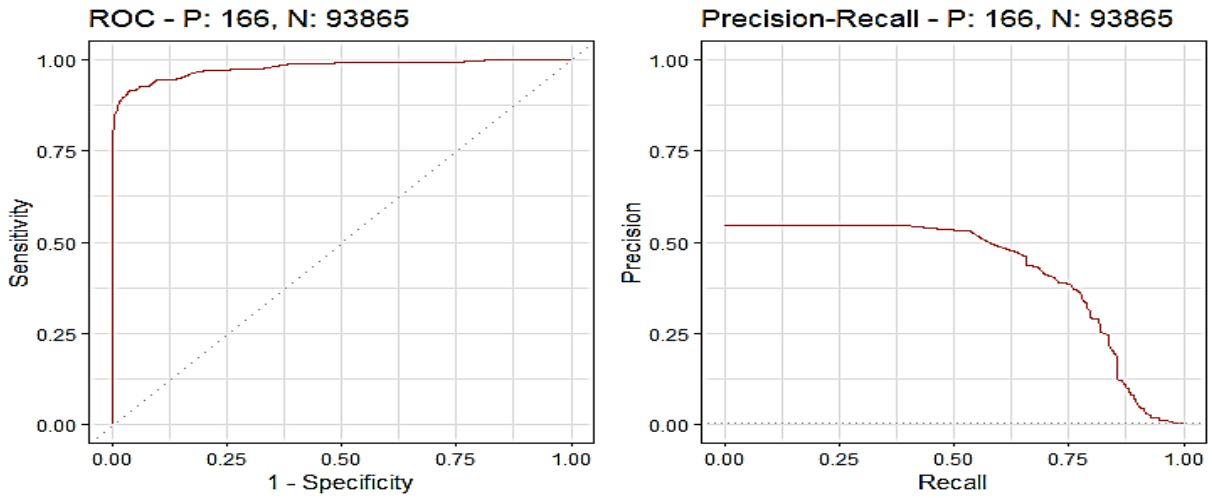

Fig. 8: Recall and Precision Curves for Elasticnet Model (Down-Sampling Technique)

These curves give the shape we would expect. At thresholds with low recall, the precision is correspondingly high though at a constant rate, and at very high recall, the precision begins to drop. Looking at the Precision-Recall Curve, we notice the curve gets precision up to about 78.

➢ *Lasso Regression for Down-Sampling Technique*

The following Elasticnet Model predictions were obtained.

Top 6	
	s0
2	0.14
4	0.11
5	0.21
8	0.19
11	0.11
13	0.12

Bottom 6	
	s0
284789	0.15
284793	0.09
284797	0.20
284802	0.14
284803	0.01
284804	0.12

- Lasso Model Evaluation for Down-Sampling Technique

✓ Miscalculation Rate for Elasticnet Model for Down-Sampling Technique

The mean of the miscalculation rate is at 0.009273537. A miscalculation rate of 0.009273537 indicates that our model has 0.9273537% incorrect predictions which means it has 99.0726463% correct predictions hence the model is predicting accurately.

✓ Confidence Interval for The Area Under the Curve(AUC) for Lasso for Down-Sampling Technique

The model has confidence 0.95 of predicting correctly with a confidence interval between 0.880 and 0.982 at an accuracy of 93.1%.

✓ ROC Curve of the best fit Model for Lasso for Down-Sampling Technique

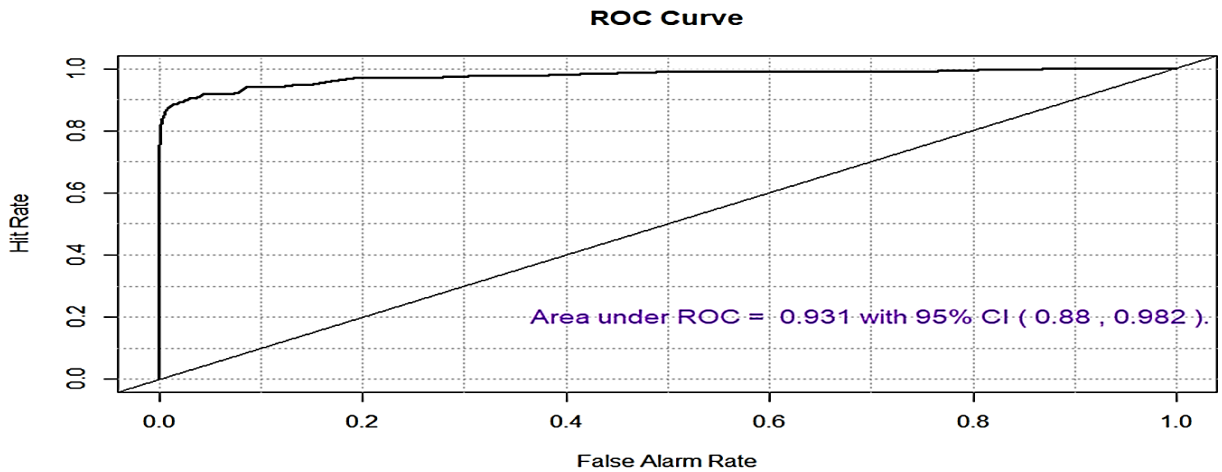

Fig. 9: ROC Curve of the best fit Model for Lasso (Down-Sampling Technique)

ISSN No:-2456-2165

The higher AUC -ROC, the better the performance of the model at distinguishing between positive and the negative classes so when the AUC is between 0.5 and 1, that is, 0.5<AUC<1, then there is a high chance that the model can distinguish between the positive class values from the negative class values. Since our AUC value is 0.931, we have a 95% confidence Interval between 0.88 and 0.982 that our model can differentiate between is FTP(False Positive Rate) and the TPR(True Positive rate)

✓ Recall and Precision Score for Lasso Model for Down-Sampling Technique

We obtained a precision of 0.9909338 which means 99.09338 % of our prediction is relevant. We obtained a recall of the recall of 0.9997743 which shows that our model has accuracy of 99.97743% in correctly classifying the total relevant results.

✓ Recall and Precision Curves for Lasso Model for Down-Sampling Technique

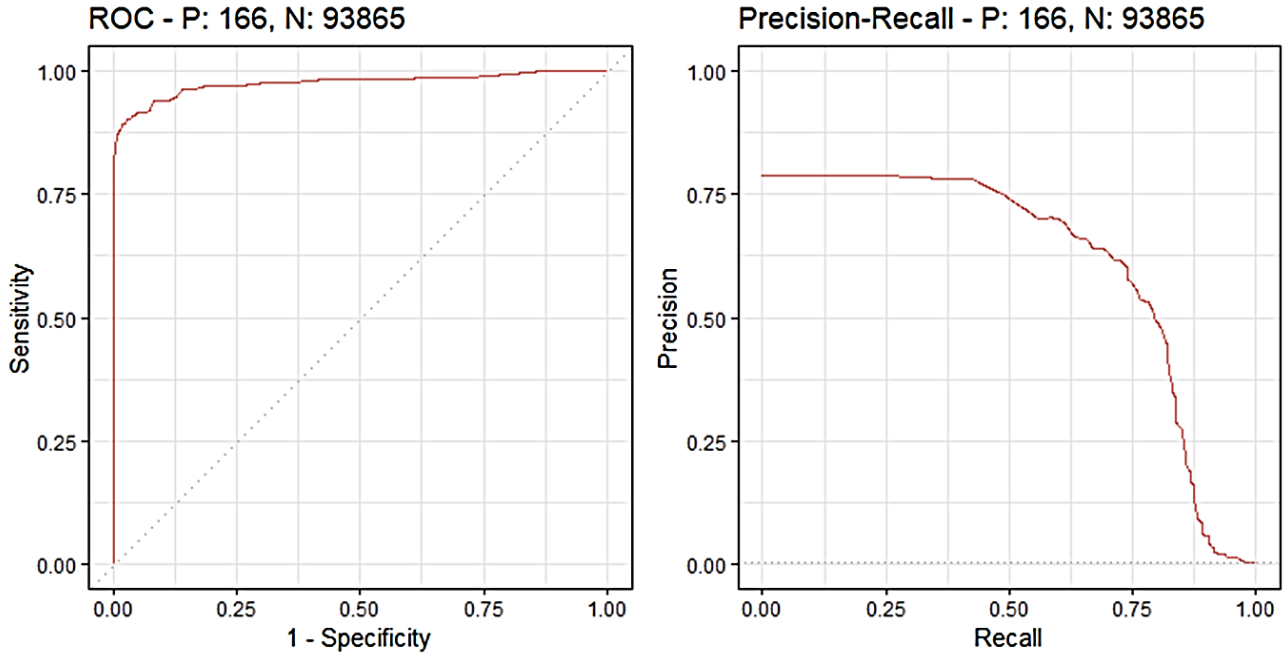

Fig. 10: Recall and Precision Curves for Lasso Model (Down-Sampling Technique)

These curves give the shape we would expect. A threshold with low recall, the precision is correspondingly high though at a constant rate, and at very high recall, the precision begins to drop. Looking at the Precision-Recall Curve, we notice the curve gets precision up to about 88.

C. Modelling with Up-Sample Technique

➢ *Ridge Regression*

The following Ridge Regression Model predictions were obtained.

Top 6	
	s0
2	**0.15150102**
4	**0.12316558**
5	**0.11707270**
8	**0.08736508**
11	**0.12797648**
13	**0.11691316**

Bottom 6	
	s0
284789	**0.09827895**
284793	**0.09921419**
284797	**0.17479334**
284802	**0.15405885**
284803	**0.06276707**
284804	**0.10295522**

- Ridge Regression Model Evaluation for Up-Sample Technique
- Miscalculation Rate for Ridge Model for Up-Sample Technique

The mean of the miscalculation rate is at 0.006171316. A miscalculation rate of 0. 006171316 indicates that our model has 0. 6171316% incorrect predictions which means it has 99.3828684% correct predictions hence the model is predicting accurately.

- Confidence Interval for The Area Under the Curve(AUC) for Ridge for Up-Sample Technique

The model has confidence 0.95 of predicting correctly with a confidence interval between 97.00 and 99.40. at an accuracy of 98.2%.

- ROC Curve of the best fit Model for Ridge for Up-Sample Technique

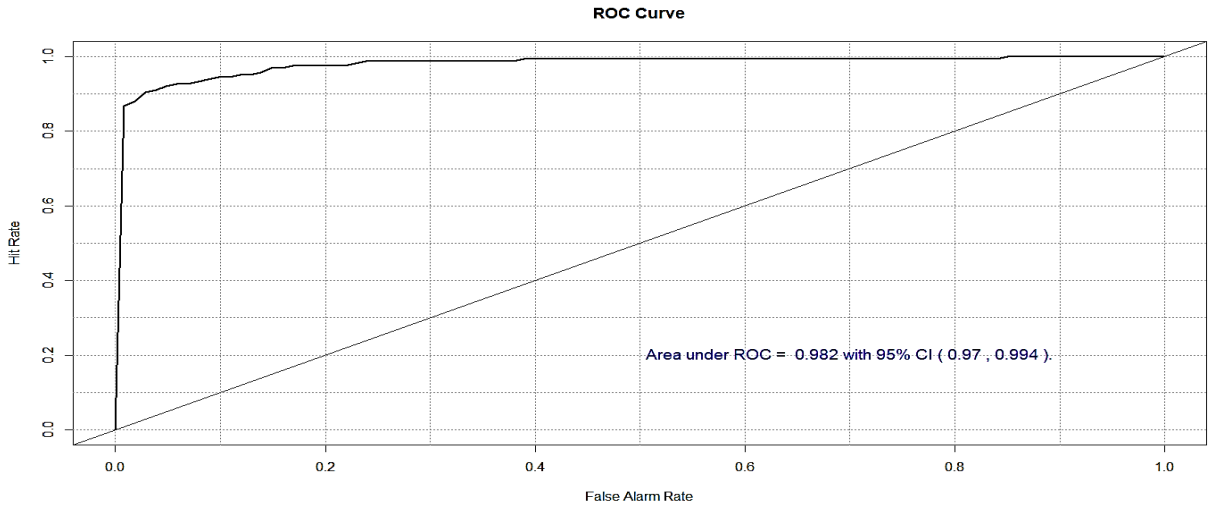

Fig. 11: ROC Curve of the best fit Model for Ridge Model (Up-Sampling Technique)

The higher AUC -ROC, the better the performance of the model at distinguishing between positive and the negative classes so when the AUC is between 0.5 and 1, that is, 0.5<AUC<1, then there is a high chance that then model can distinguish between the positive class values from the negative class values. Since our AUC value is 0.982, we have a 95% confidence Interval between 0.970 and 0.994 that our model can differentiate between is FTP(False Positive Rate) and the TPR(True Positive rate).

- Recall and Precision Score for Ridge Model for Up-Sample Technique

We obtained a precision of 0.999732 which means 99.9732% of our prediction is relevant. We obtained a recall of the recall of 0.993512 shows that our model has accuracy of 99.3512% in correctly classifying the total relevant results.

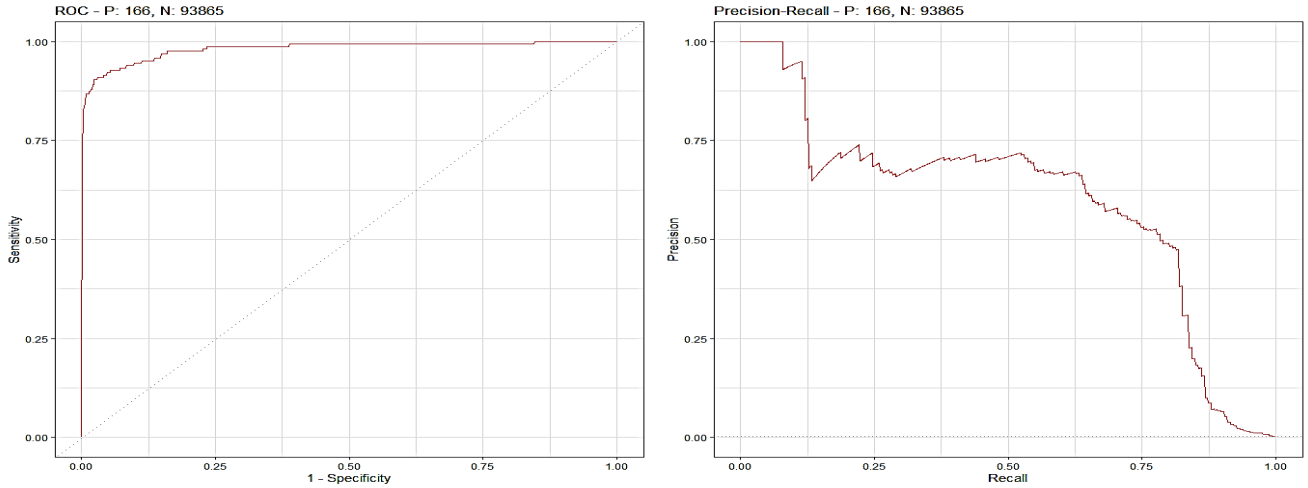

Fig. 12: Recall And Precision Curves for Ridge Model (Up-Sampling Technique)

These curves give the shape we would expect. At thresholds with low recall, the precision is correspondingly high though at a constant rate, and at very high recall, the precision begins to drop. Looking at the Precision-Recall Curve, we notice the curve gets precision up to about 83.

➢ *Elasticnet Regression Model for Up-Sampling Technique*

The following Elasticnet Regression Model predictions were obtained.

Top 6

	s0
2	0.03
4	0.01
5	0.05
8	0.06
11	0.02
13	0.01

om 6

	s0
284789	0.02
284793	0.02
284797	0.09
284802	0.05
284803	0.00
284804	0.03

ISSN No:-2456-2165

- Miscalculation Rate for Elasticnet Model for Up-Sampling Technique

The mean of the miscalculation rate is at 0.02283435. A miscalculation rate of 0.02283435 indicates that our model has 2.283435% incorrect predictions which means it has 97.716565% correct predictions hence the model is predicting accurately.

- Confidence Interval for The Area Under the Curve(AUC) for Elasticnet Model for Up-Sampling Technique

The model has confidence 0.95 of predicting correctly with a confidence interval between 0.8958664 and 0.9853755 at an accuracy of 94.0621%.

- ROC Curve of the best fit Model for Elasticnet for Up-Sampling Technique

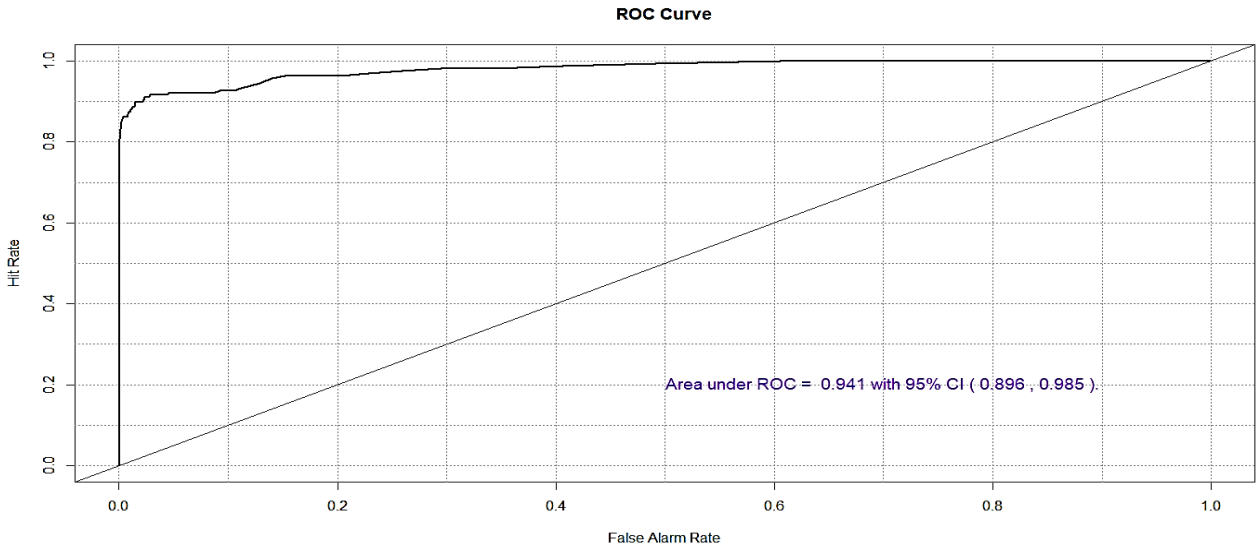

Fig.13: ROC Curve of the best fit Model for Elasticnet (Up-Sampling Technique)

The higher AUC -ROC, the better the performance of the model at distinguishing between positive and the negative classes so when the AUC is between 0.5 and 1, that is, 0.5<AUC<1, then there is a high chance that the model can distinguish between the positive class values from the negative class values. Since our AUC value is 0.941, we have a 95% accuracy and a confidence Interval between 0.896 and 0.985 that our model can differentiate between is FTP(False Positive Rate) and the TPR(True Positive rate)

- Recall and Precision Score for Elasticnet Model for Up-Sapling Technique

We obtained a precision of 0.9998257 which means 99.98257% of our prediction is relevant. We obtained a recall of 0.9776274 which shows that our model has accuracy of 97.76274 in correctly classifying the total relevant results.

- Recall and Precision Curves for Elasticnet Model for Up-Sapling Technique

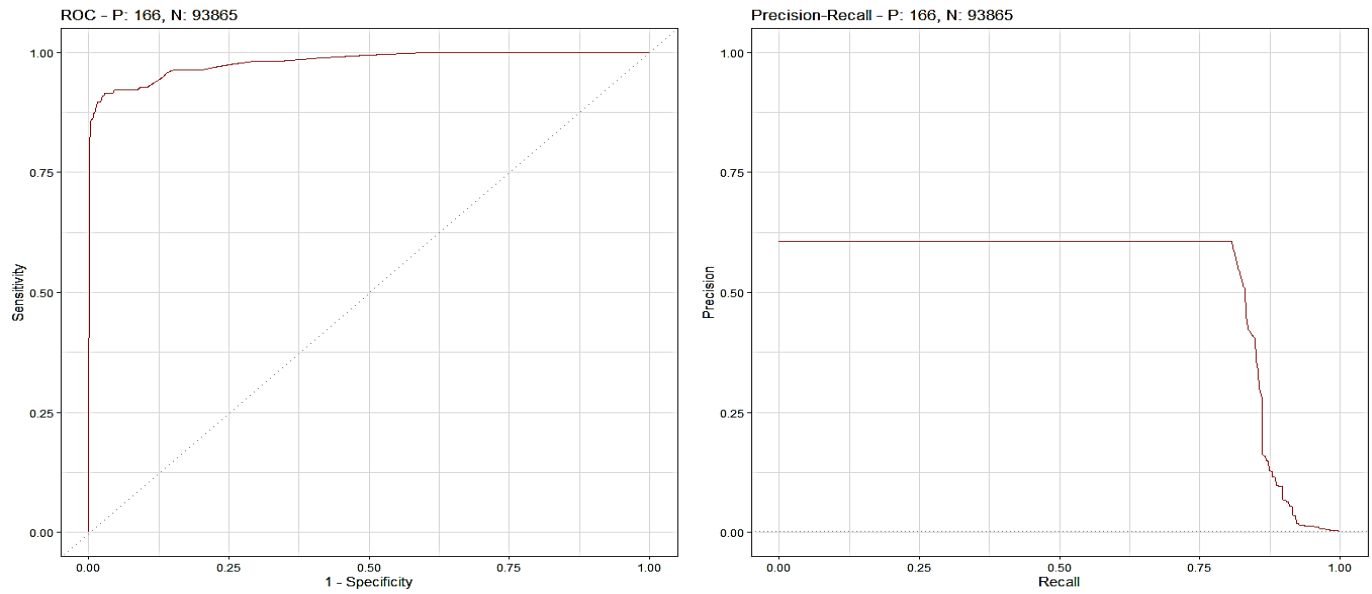

Fig.14: Recall and Precision Curves for Elasticnet Model (Up-Sampling Technique)

ISSN No:-2456-2165

Just like the other curves, these curves give the shape we would expect. At thresholds with low recall, the precision is correspondingly high though at a constant rate, and at very high recall, the precision begins to drop. Looking at the Precision-Recall Curve, we notice the curve gets precision up to about 83.

➢ *Lasso Regression for Up-Sampling Technique*
The following Lasso Model predictions were obtained.

Top 6	
	s0
2	0.02
4	0.01
5	0.05
8	0.06
11	0.01
13	0.01

Bottom 6	
	s0
284789	0.02
284793	0.02
284797	0.09
284802	0.05
284803	0.00
284804	0.03

➢ *Lasso Model Evaluation for Up-Sampling Technique*

- Miscalculation Rate for Elasticnet Model for Up-Sampling Technique

The mean of the miscalculation rate is at 0.009273537. A miscalculation rate of 0.009273537 indicates that our model has 0.9273537% incorrect predictions which means it has 99.0726463% correct predictions hence the model is predicting accurately.

- Confidence Interval for The Area Under the Curve(AUC) for Lasso for Up-Sampling Technique

The model has confidence 0.95 of predicting correctly with a confidence interval between 0.880 and 0.982 at an accuracy of 93.1%.

- ROC Curve of the best fit Model for Lasso for Up-Sampling Technique

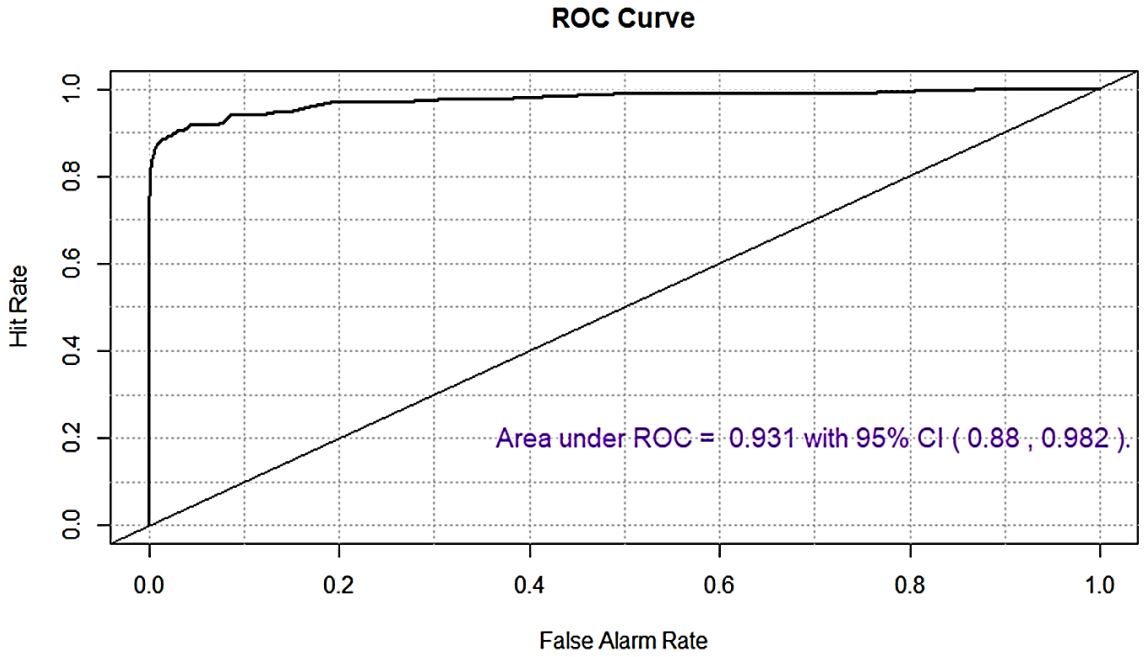

Fig. 15: ROC Curve of the best fit Model for Lasso (Up-Sampling Technique)

The higher AUC -ROC, the better the performance of the model at distinguishing between positive and the negative classes so when the AUC is between 0.5 and 1, that is, 0.5<AUC<1, then there is a high chance that the model can distinguish between the positive class values from the negative class values. Since our AUC value is 0.931, we have a 95% confidence Interval between 0.88 and 0.982 that our model can differentiate between is FTP(False Positive Rate) and the TPR(True Positive rate)

- Recall and Precision Score for Lasso Model for Up-Sampling Technique

We obtained a precision of 0.9909338 which means 99.09338 % of our prediction is relevant. We obtained a recall of the recall of 0.9997743 which shows that our model has accuracy of 99.97743% in correctly classifying the total relevant results.

- Recall and Precision Curves for Lasso Model for Up-Sampling Technique

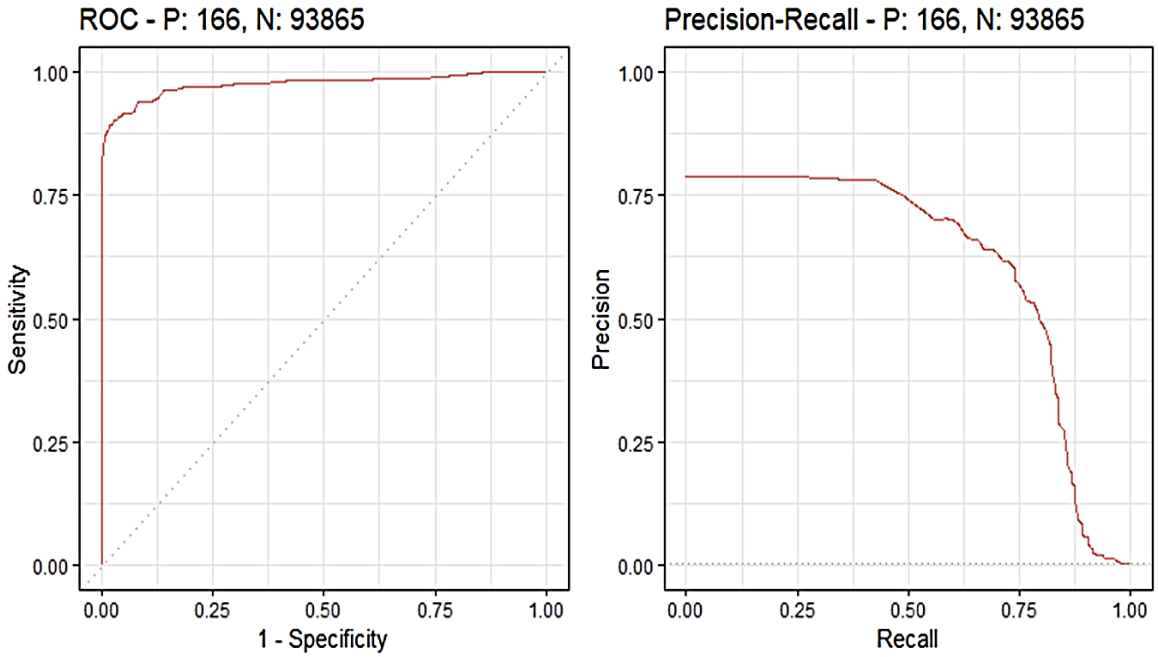

Fig. 16: Recall and Precision Curves for Lasso Model (Up-Sampling Technique)

These curves give the shape we would expect. A threshold with low recall, the precision is correspondingly high though at a constant rate, and at very high recall, the precision begins to drop. Looking at the Precision-Recall Curve, we notice the curve gets precision up to about 88.

V. CONCLUSION

From the outputs obtained from modeling while using the Down-Sampling Technique of the Ride Regression, Elasticnet Regression and Lasso Regression, we realize that all the three models are accurate in credit card fraud transaction detection. Amongst the three models, we notice that Ridge Regression is the best with an accuracy of 98%, a confidence interval between 97% and 99%, with 95% confidence. Lasso Regression is the next best model with an accuracy of 93.2%, a confidence interval between 88% and 98%, with 95% confidence, just slightly above Elasticnet which comes third with an accuracy of 93.1%, a confidence interval between 87% and 98%, with 95% confidence. Nonetheless we will say Lasso and Elasticnet both have equal accuracy while Ridge is the best.

When we look at the outputs obtained from modeling while using the Up-Sampling Technique of the Ride Regression, Elasticnet Regression and Lasso Regression, we also realize that all the three models are accurate in credit card fraud transaction detection though with very slight differences where we notice that Ridge Regression is still the best with an accuracy of 98.2%, a confidence interval between 97.00 and 99.40 with a 95% confidence. Elasticnet Regression is the next best with an accuracy of 94.0621%, a confidence interval between 0.8958664 and 0.9853755 with a 95% confidence unlike Lasso which was the next best in the Down-Sampling Technique. Lasso Regression is the third best with an accuracy of 93.1%, a confidence interval between 0.880 and 0.982 with a 0.95 confidence in predicting, unlike Elasticnet which was the third in the Down-Sampling Technique.

RECOMMENDATION

We recommend that the final models are deployed to make predictions on new data and their performance being monitored regularly.

REFERENCES

[1.] Acar, A., Aksu, H., & Dogac, A. (2017). A survey of credit card fraud detection techniques: Data and technique-oriented perspective. Journal of Computer and System Sciences, 83(1), 121-136.
[2.] Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. Journal of Network and Computer Applications, 60, 19-31.
[3.] Altman, E. I., Marco, G., & Varetto, F. (1994). Corporate distress diagnosis comparisons using linear discriminant analysis and neural networks. Journal of Banking and Finance, 18(3), 505-529.
[4.] Bentley, P., Kim, J., Jung. G., & Choi, J. (2000). Fuzzy Darwinian Detection of Credit Card Fraud. Proc. of 14th Annual Fall Symposium of the Korean Information Processing Society.
[5.] Credit Cards: Use and Consumer Attitudes, 1970-2000, 86 Fed. Res. Bull. 623 (2000).
[6.] Dal Pozzolo, A., Caelen, O., Johnson, R. A., & Bontempi, G. (2015). Calibrating Probability with Undersampling for Unbalanced Classification. In Symposium on Computational Intelligence and Data Mining (CIDM) (pp. 1-7). IEEE.
[7.] Dinakar, K., & Nair, R. R. (2020). An efficient credit card fraud detection model using machine learning and natural language processing. Procedia Computer Science, 171, 695-703.

ISSN No:-2456-2165

[8.] Dorronsoro, J. R., Ginel, F., Sanchez, J. A., & Cruz, J. M. (1997). Building an online system for fraud detection of credit card operations based on a neural classifier. Proceedings of the International Conference on Artificial Neural Networks (ICANN'97), 683-688.
[9.] Federal Trade Commission. (2020). Consumer Sentinel Network Data Book 2020. Retrieved from https://www.ftc.gov/system/files/documents/reports/consumer-sentinel-network-data-book-2020/csn_data_book_2020.pdf
[10.] Federal Trade Commission. (2021). Consumer Sentinel Network Data Book 2020. Retrieved from https://www.ftc.gov/reports/consumer-sentinel-network-data-book-2020
[11.] Flitman, A. M. (1997). Towards analysing student failures: neural networks compared with regression analysis and multiple discriminant analysis. Computers & Operations Research, 24(4), 367-377.
[12.] Gandomi, A., & Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. International Journal of Information Management, 35(2), 137-144.
[13.] Hanagandi, V., Dhar, S., & Buescher, K. (1996). Application of classification models on credit card fraud detection. Proceedings of the International Conference on Neural Networks, 1996. doi: 10.1109/ICNN.1996.548981
[14.] Nilson Report. (December 2021). Card Fraud Worldwide. Issue 1209.
[15.] The Nilson Report. (2020). Card Fraud Losses Reach $9.47 Billion. Retrieved from https://www.nilsonreport.com/mention/194/card-fraud-losses-reach-9.47-billion/

ISSN No:-2456-2165

APPENDIX

R-CODES USED FOR THE PROJECT

```
setwd("C:/Users/nebcy/Documents/Apsu/Apsu/Data Set STAT5140")
credit_card1 = read.csv("creditcard.CSV")
credit_card<-credit_card1
dim(credit_card)
head(credit_card)

# Basic Data Exploration

install.packages(c('ROCR','ggplot2','corrplot','caTools','class',
                   'randomForest','pROC','imbalance'))
library(ROCR)
library(ggplot2)
library(corrplot)
library(caTools)
library(class)
library(randomForest)
library(pROC)
library(imbalance)
library(rpart)

#Basic Exploratory Data Analysis
#Viewing some initial observations of the dataset

#Summary of the dataset
summary(credit_card)

#Viewing Structure of the dataset
str(credit_card)


sapply(credit_card, FUN=class)

#Checking for missing values
credit_card[!complete.cases(credit_card),]

#Missing values by visualization
install.packages("naniar")
library(naniar)
vis_miss(credit_card, 'warn_large_data' = FALSE)
gg_miss_var(credit_card)


#Viewing the Frequency distribution of "Class" Variable
table(credit_card$Class)

prop.table(table(credit_card$Class))

ggplot(data = credit_card,aes(x=Class))+geom_bar(col="red")

#Plotting the Amount - Fraud chart
ggplot(data = credit_card,aes(y=Amount,x=Class))+geom_point(col="slateblue3") + facet_grid(~Class)

#Checking correlation between the variables
install.packages("corpcor")
library(corpcor)
head(cor2pcor(cov(credit_card)))

library(corrplot)
```

```
credit_card$Class <- as.numeric(credit_card$Class)
corr <- cor(credit_card[],method="pearson")
corrplot(corr,method = "circle", type = "lower")

#Taking out Time Column.
#Since the Time Variab,e is not important for this project, we take it out
credit_card<-credit_card1[,-1]
head(credit_card)

########################## Data partition ##################
set.seed(123)  #maintains consistency of data and makes data set not to randomly change when running it multiple times
n <- nrow(credit_card)
split_data <- sample(x=1:2, size = n, replace=TRUE, prob=c(0.67, 0.33)) #x=1 =training data, x=2 = testing data
train <- credit_card[split_data == 1, ]
test <- credit_card[split_data == 2, ]
y.train <- train$Class
yobs <- test$Class

###################### Balancing data through upsampling
# Load the necessary libraries
install.packages("caret")
library(caret)

# Check the class distribution before upsampling
table(train$Class)


# Check the class of the 'Class' variable
class(train$Class)

# Convert 'Class' to a factor variable if it's not already
train$Class <- as.factor(train$Class)

# Verify that 'Class' is now a factor
class(train$Class)

set.seed(90)  # for reproducibility
down_train_data <- downSample(x = train[, -30], y = train$Class)

# Check the class distribution after upsampling
table(down_train_data$Class)


# Upsample the minority class
set.seed(90)  # for reproducibility
down_train_data <- downSample(x = train[, -30], y = train$Class)

# Check the class distribution after upsampling
table(down_train_data$Class)


#Building A logistic model
install.packages("glmnet")
library(glmnet) #Elastic net
formula0 <- factor(Class)~. #This teslls us num is a categorical variable and this will bring out just 1 and 0
X <- model.matrix (as.formula(formula0), data = down_train_data)[, -1] # trim off the first column and leaving only the predictors

#RIDGE REGRESSION
#Using cross validation to determine the optimal tuning parameter. To chose the best lambda
CV <- cv.glmnet(x=X, y=down_train_data$Class, family="binomial", alpha = 0, # CROSS VALIDATION METHOD
```

ISSN No:-2456-2165

```
        nlambda = 200)#we can take out part of the fucntion from standardize =T till maxit=100 so as to let R choose best lamda
         for us.

plot(CV)
coef(CV, CV$lambda.min)
coef(CV, CV$lambda.1se)

b.lambda <- CV$lambda.1se #we can use 1se or min      THIS GIVES US THE BEST LAMBDA
b.lambda

fit.best <- glmnet(x=X, y=down_train_data$Class, family="binomial", alpha = 0,          #BEST FIT WILL NOT HAVE ANY
          DATA DISAPPEARING BECAUSE WE USED RIDGE i.E. LAMBDA = 0
            lambda=b.lambda)
(fit.best$beta)

# Prediction
X.test <- model.matrix (as.formula(formula0), data = test)[, -1]
pred <- predict(fit.best, newx = X.test, type="response") #type = "response" means y is continues
head(pred)
tail(pred)
dim(pred)

##### Finding missclassification rate #making a threshhold
pred1 <- ifelse(pred>0.5, 1, 0)  #if it is bigger than 50%, we say yes
pred1

(miss.rate <- mean(yobs != pred1))

pred<-pred[, 1] #gives only the index (1, 0) and not the column

pred1<-pred1[,1]

#Plotting ROC curve of the fit.best model.
install.packages("cvAUC")
library(cvAUC)
AUC <- ci.cvAUC(predictions = pred, labels = yobs, folds=1:NROW(test), confidence = 0.95)
AUC
(auc.ci <- round(AUC$ci, digits = 3))

install.packages("verification")
library(verification)
mod.glm <- verify(obs = yobs, pred = pred)
roc.plot(mod.glm, plot.thres=NULL)
text(x=0.7, y=0.2, paste("Area under ROC = ", round(AUC$cvAUC, digits = 3), "with 95% CI (",
                  auc.ci[1], ",", auc.ci[2], ").", sep = " "), col="blue", cex =1.2)

#confusion matrix  (we dont need it in this case because the data in imbalance)
#library(caret)
#library(e1071)
#confusionMatrix( as.factor(pred1),as.factor(yobs),positive = "1")

#RECALL AND PRECISION SCORE, WE CAN USE THIS
#precision(as.factor(yobs),as.factor(pred1))
#recall(as.factor(yobs), as.factor(pred1))
# Assuming yobs and pred1 are binary (0/1) vectors or factors
library(caret)

# Calculate precision
precision_score <- posPredValue(as.factor(pred1), as.factor(yobs))

# Calculate recall
```

```
recall_score <- sensitivity(as.factor(pred1), as.factor(yobs))

# Print the precision and recall scores
cat("Precision Score:", precision_score, "\n")
cat("Recall Score:", recall_score, "\n")


#RECALL AND PRECISION CURVES
install.packages("precrec")
library(precrec)
precrec_ridge <- evalmod(scores = pred, labels = yobs)
precrec_ridge
autoplot(precrec_ridge)

##################################

#ELASTICNET REGRESSION
#Using cross validation to determine the optimal tuning parameter. To chose the best lambda
CV2 <- cv.glmnet(x=X, y=down_train_data$Class, family="binomial", alpha = 0.5, # CROSS VALIDATION METHOD
            nlambda = 200)#we can take out part of the fucntion from standardize =T till maxit=100 so as to let R choose best lamda
            for us.

plot(CV2)
coef(CV2, CV2$lambda.min)
coef(CV2, CV2$lambda.1se)

b.lambda2 <- CV2$lambda.1se #we can use 1se or min      THIS GIVES US THE BEST LAMBDA
b.lambda2

fit.best2 <- glmnet(x=X, y=down_train_data$Class, family="binomial", alpha = 0.5,        #BEST FIT WILL SHOW SOME DATA
            DISAPPEARING BECAUSE WE USED LASSO i.E. LAMBDA = 1
              lambda=b.lambda2)
(fit.best2$beta)

# Prediction
X.test <- model.matrix (as.formula(formula0), data = test)[, -1]
pred_elasticnet <- predict(fit.best2, newx = X.test, type="response") #type = "response" means y is continues
(head(pred_elasticnet <- round(pred_elasticnet, digits = 2)))
(tail(pred_elasticnet <- round(pred_elasticnet, digits = 2)))

dim(pred_elasticnet)

##### Finding missclassification rate #making a threshhold
pred1_elasticnet <- ifelse(pred_elasticnet>0.5, 1, 0)  #if it is bigger than 50%, we say yes
pred1_elasticnet

(miss.rate <- mean(yobs != pred1_elasticnet))

pred_elasticnet<-pred_elasticnet[, 1] #gives only the index (1, 0) and not the column

pred1_elasticnet<-pred1_elasticnet[,1]

#Plotting ROC curve of the fit.best model.
library(cvAUC)
AUC2 <- ci.cvAUC(predictions = pred1_elasticnet, labels = yobs, folds=1:NROW(test), confidence = 0.95)
AUC2
(auc.ci_elasticnet <- round(AUC2$ci, digits = 3))

library(verification)
mod.glm_elasticnet <- verify(obs = yobs, pred = pred_elasticnet)
roc.plot(mod.glm_elasticnet, plot.thres=NULL)
```

```
text(x=0.7, y=0.2, paste("Area under ROC = ", round(AUC2$cvAUC, digits = 3), "with 95% CI (",
                         auc.ci_elasticnet[1], ",", auc.ci_elasticnet[2], ").", sep = " "), col="blue", cex =1.2)

#confusion matrix  (we dont need it in this case because the data in imbalance)
#library(caret)
#library(e1071)
#confusionMatrix( as.factor(pred1),as.factor(yobs),positive = "1")

#FOR RECALL AND PRECISION SCORE, WE CAN USE THIS     ....Since our data is imbalance, we use RECALL AND
        PRECISION SCORE to see if it can predict the minority class since it will definitely predict the majority class since it
        has enough data on which to learn so with our output off 0.9917754, we can conclude that the model is prdicting well.
#precision(as.factor(yobs),as.factor(pred1_elasticnet))
#recall(as.factor(yobs), as.factor(pred1_elasticnet))

################
#RECALL AND PRECISION SCORE, WE CAN USE THIS
#precision(as.factor(yobs),as.factor(pred1_elasticnet))
#recall(as.factor(yobs), as.factor(pred1_elasticnet))
# Assuming yobs and pred1_elasticnet are binary (0/1) vectors or factors
library(caret)

# Calculate precision
precision_score <- posPredValue(as.factor(pred1_elasticnet), as.factor(yobs))

# Calculate recall
recall_score <- sensitivity(as.factor(pred1_elasticnet), as.factor(yobs))

# Print the precision and recall scores
cat("Precision Score:", precision_score, "\n")
cat("Recall Score:", recall_score, "\n")

#RECALL AND PRECISION CURVES
#install.packages("precrec")
library(precrec)
precrec_elasticnet <- evalmod(scores = pred_elasticnet, labels = yobs)
precrec_elasticnet
autoplot(precrec_elasticnet)

##############################################
#LASSO REGRESSION
#Using cross validation to determine the optimal tuning parameter. To chose the best lambda
CV1 <- cv.glmnet(x=X, y=down_train_data$Class, family="binomial", alpha = 1, # CROSS VALIDATION METHOD
        nlambda = 200)#we can take out part of the fucntion from standardize =T till maxit=100 so as to let R choose best lamda
        for us.

plot(CV1)
coef(CV1, CV1$lambda.min)
coef(CV1, CV1$lambda.1se)

b.lambda1 <- CV1$lambda.1se #we can use 1se or min      THIS GIVES US THE BEST LAMBDA
b.lambda1

fit.best1 <- glmnet(x=X, y=down_train_data$Class, family="binomial", alpha = 1,          #BEST FIT WILL SHOW SOME DATA
        DISAPPEARING BECAUSE WE USED LASSO i.E. LAMBDA = 1
          lambda=b.lambda1)
(fit.best1$beta)

# Prediction
X.test <- model.matrix (as.formula(formula0), data = test)[, -1]
pred_lasso <- predict(fit.best1, newx = X.test, type="response") #type = "response" means y is continues
(head(pred_lasso <- round(pred_lasso, digits = 2)))
```

```
(tail(pred_lasso <- round(pred_lasso, digits = 2)))

dim(pred_lasso)

##### Finding missclassification rate #making a threshhold
pred1_lasso <- ifelse(pred_lasso>0.5, 1, 0)  #if it is bigger than 50%, we say yes

(miss.rate <- mean(yobs != pred1_lasso))

pred_lasso<-pred_lasso[,1] #gives only the index (1, 0) and not the column

pred1_lasso<-pred1_lasso[,1]

#Plotting ROC curve of the fit.best model.
library(cvAUC)
AUC1 <- ci.cvAUC(predictions = pred1_lasso, labels = yobs, folds=1:NROW(test), confidence = 0.95)
AUC1
(auc.ci_lasso <- round(AUC1$ci, digits = 3))

library(verification)
mod.glm_lasso <- verify(obs = yobs, pred = pred_lasso)
roc.plot(mod.glm_lasso, plot.thres=NULL)
text(x=0.7, y=0.2, paste("Area under ROC = ", round(AUC1$cvAUC, digits = 3), "with 95% CI (",
                         auc.ci_lasso[1], ",", auc.ci_lasso[2], ").", sep = " "), col="purple", cex =1.2)

#confusion matrix  (we dont need it in this case because the data in imbalance)
#library(caret)
#library(e1071)
#confusionMatrix( as.factor(pred1),as.factor(yobs),positive = "1")

#FOR RECALL AND PRECISION SCORE, WE CAN USE THIS
#precision(as.factor(yobs),as.factor(pred1_lasso))
#recall(as.factor(yobs), as.factor(pred1_lasso))

#RECALL AND PRECISION CURVES
#install.packages("precrec")
library(precrec)
precrec_lasso <- evalmod(scores = pred_lasso, labels = yobs)
precrec_lasso
autoplot(precrec_lasso)

###############
# Calculate precision
precision_score <- posPredValue(as.factor(pred1_lasso), as.factor(yobs))

# Calculate recall
recall_score <- sensitivity(as.factor(pred1_lasso), as.factor(yobs))

# Print the precision and recall scores
cat("Precision Score:", precision_score, "\n")
cat("Recall Score:", recall_score, "\n")


#RECALL AND PRECISION CURVES
#install.packages("precrec")
library(precrec)
precrec_lasso <- evalmod(scores = pred1_lasso, labels = yobs)
precrec_lasso
autoplot(precrec_lasso)
```

```
# Clear the console
cat("\014")

# Restart R
q(save = "no")

setwd("C:/Users/nebcy/Documents/Apsu/Apsu/Data Set STAT5140")
credit_card1 = read.csv("creditcard.CSV")
credit_card<-credit_card1
dim(credit_card)
head(credit_card)

# Basic Data Exploration

install.packages(c('ROCR','ggplot2','corrplot','caTools','class',
                   'randomForest','pROC','imbalance'))
library(ROCR)
library(ggplot2)
library(corrplot)
library(caTools)
library(class)
library(randomForest)
library(pROC)
library(imbalance)
library(rpart)

#Basic Exploratory Data Analysis
#Viewing some initial observations of the dataset

#Summary of the dataset
summary(credit_card)

#Viewing Structure of the dataset
str(credit_card)

sapply(credit_card, FUN=class)

#Checking for missing values
credit_card[!complete.cases(credit_card),]

#Missing values by visualization
install.packages("naniar")
library(naniar)
vis_miss(credit_card, 'warn_large_data' = FALSE)
gg_miss_var(credit_card)


#Viewing the Frequency distribution of "Class" Variable
table(credit_card$Class)

prop.table(table(credit_card$Class))

ggplot(data = credit_card,aes(x=Class))+geom_bar(col="red")

#Plotting the Amount - Fraud chart
ggplot(data = credit_card,aes(y=Amount,x=Class))+geom_point(col="slateblue3") + facet_grid(~Class)

#Checking correlation between the variables
install.packages("corpcor")
library(corpcor)
head(cor2pcor(cov(credit_card)))
```

```
library(corrplot)
credit_card$Class <- as.numeric(credit_card$Class)
corr <- cor(credit_card[],method="pearson")
corrplot(corr,method = "circle", type = "lower")

#Taking out Time Column.
#Since the Time Variab,e is not important for this project, we take it out
credit_card<-credit_card1[,-1]
head(credit_card)

######################### Data partition ##################
set.seed(123)  #maintains consistency of data and makes data set not to randomly change when running it multiple times
n <- nrow(credit_card)
split_data <- sample(x=1:2, size = n, replace=TRUE, prob=c(0.67, 0.33)) #x=1 =training data, x=2 = testing data
train <- credit_card[split_data == 1, ]
test <- credit_card[split_data == 2, ]
y.train <- train$Class
yobs <- test$Class

###################### Balancing data through upsampling
# Load the necessary libraries
install.packages("caret")
library(caret)

# Check the class distribution before upsampling
table(train$Class)


# Check the class of the 'Class' variable
class(train$Class)

# Convert 'Class' to a factor variable if it's not already
train$Class <- as.factor(train$Class)

# Verify that 'Class' is now a factor
class(train$Class)

set.seed(90)  # for reproducibility
up_train_data <- upSample(x = train[, -30], y = train$Class)

# Check the class distribution after upsampling
table(up_train_data$Class)


# Upsample the minority class
set.seed(90)  # for reproducibility
up_train_data <- upSample(x = train[, -30], y = train$Class)

# Check the class distribution after upsampling
table(up_train_data$Class)

#Building A logistic model
install.packages("glmnet")
library(glmnet) #Elastic net
formula0 <- factor(Class)~. #This teslls us num is a categorical variable and this will bring out just 1 and 0
X <- model.matrix (as.formula(formula0), data = up_train_data)[, -1] # trim off the first column and leaving only the predictors

#RIDGE REGRESSION
#Using cross validation to determine the optimal tuning parameter. To chose the best lambda
CV <- cv.glmnet(x=X, y=up_train_data$Class, family="binomial", alpha = 0, # CROSS VALIDATION METHOD
```

ISSN No:-2456-2165

```
        nlambda = 200)#we can take out part of the fucntion from standardize =T till maxit=100 so as to let R choose best lamda
         for us.

plot(CV)
coef(CV, CV$lambda.min)
coef(CV, CV$lambda.1se)

b.lambda <- CV$lambda.1se #we can use 1se or min      THIS GIVES US THE BEST LAMBDA
b.lambda

fit.best <- glmnet(x=X, y=up_train_data$Class, family="binomial", alpha = 0,          #BEST FIT WILL NOT HAVE ANY DATA
          DISAPPEARING BECAUSE WE USED RIDGE i.E. LAMBDA = 0
           lambda=b.lambda)
(fit.best$beta)

# Prediction
X.test <- model.matrix (as.formula(formula0), data = test)[, -1]
pred <- predict(fit.best, newx = X.test, type="response") #type = "response" means y is continues
head(pred)
tail(pred)
dim(pred)

##### Finding missclassification rate #making a threshhold
pred1 <- ifelse(pred>0.5, 1, 0)  #if it is bigger than 50%, we say yes
pred1

(miss.rate <- mean(yobs != pred1))

pred<-pred[, 1] #gives only the index (1, 0) and not the column

pred1<-pred1[,1]

#Plotting ROC curve of the fit.best model.
install.packages("cvAUC")
library(cvAUC)
AUC <- ci.cvAUC(predictions = pred, labels = yobs, folds=1:NROW(test), confidence = 0.95)
AUC
(auc.ci <- round(AUC$ci, digits = 3))

install.packages("verification")
library(verification)
mod.glm <- verify(obs = yobs, pred = pred)
roc.plot(mod.glm, plot.thres=NULL)
text(x=0.7, y=0.2, paste("Area under ROC = ", round(AUC$cvAUC, digits = 3), "with 95% CI (",
                  auc.ci[1], ",", auc.ci[2], ").", sep = " "), col="blue", cex =1.2)

#confusion matrix  (we dont need it in this case because the data in imbalance)
#library(caret)
#library(e1071)
#confusionMatrix( as.factor(pred1),as.factor(yobs),positive = "1")

#RECALL AND PRECISION SCORE, WE CAN USE THIS
#precision(as.factor(yobs),as.factor(pred1))
#recall(as.factor(yobs), as.factor(pred1))
# Assuming yobs and pred1 are binary (0/1) vectors or factors
library(caret)

# Calculate precision
precision_score <- posPredValue(as.factor(pred1), as.factor(yobs))

# Calculate recall
```

```
recall_score <- sensitivity(as.factor(pred1), as.factor(yobs))

# Print the precision and recall scores
cat("Precision Score:", precision_score, "\n")
cat("Recall Score:", recall_score, "\n")


#RECALL AND PRECISION CURVES
install.packages("precrec")
library(precrec)
precrec_ridge <- evalmod(scores = pred, labels = yobs)
precrec_ridge
autoplot(precrec_ridge)

##################################

#ELASTICNET REGRESSION
#Using cross validation to determine the optimal tuning parameter. To chose the best lambda
CV2 <- cv.glmnet(x=X, y=up_train_data$Class, family="binomial", alpha = 0.5, # CROSS VALIDATION METHOD
            nlambda = 200)#we can take out part of the fucntion from standardize =T till maxit=100 so as to let R choose best lamda
            for us.

plot(CV2)
coef(CV2, CV2$lambda.min)
coef(CV2, CV2$lambda.1se)

b.lambda2 <- CV2$lambda.1se #we can use 1se or min      THIS GIVES US THE BEST LAMBDA
b.lambda2

fit.best2 <- glmnet(x=X, y=up_train_data$Class, family="binomial", alpha = 0.5,          #BEST FIT WILL SHOW SOME DATA
            DISAPPEARING BECAUSE WE USED LASSO i.E. LAMBDA = 1
              lambda=b.lambda2)
(fit.best2$beta)

# Prediction
X.test <- model.matrix (as.formula(formula0), data = test)[, -1]
pred_elasticnet <- predict(fit.best2, newx = X.test, type="response") #type = "response" means y is continues
(head(pred_elasticnet <- round(pred_elasticnet, digits = 2)))
(tail(pred_elasticnet <- round(pred_elasticnet, digits = 2)))

dim(pred_elasticnet)

##### Finding missclassification rate #making a threshhold
pred1_elasticnet <- ifelse(pred_elasticnet>0.5, 1, 0)  #if it is bigger than 50%, we say yes
pred1_elasticnet

(miss.rate <- mean(yobs != pred1_elasticnet))

pred_elasticnet<-pred_elasticnet[, 1] #gives only the index (1, 0) and not the column

pred1_elasticnet<-pred1_elasticnet[,1]

#Plotting ROC curve of the fit.best model.
library(cvAUC)
AUC2 <- ci.cvAUC(predictions = pred1_elasticnet, labels = yobs, folds=1:NROW(test), confidence = 0.95)
AUC2
(auc.ci_elasticnet <- round(AUC2$ci, digits = 3))

library(verification)
mod.glm_elasticnet <- verify(obs = yobs, pred = pred_elasticnet)
roc.plot(mod.glm_elasticnet, plot.thres=NULL)
```

```
text(x=0.7, y=0.2, paste("Area under ROC = ", round(AUC2$cvAUC, digits = 3), "with 95% CI (",
                         auc.ci_elasticnet[1], ",", auc.ci_elasticnet[2], ").", sep = " "), col="blue", cex =1.2)

#confusion matrix  (we dont need it in this case because the data in imbalance)
#library(caret)
#library(e1071)
#confusionMatrix( as.factor(pred1),as.factor(yobs),positive = "1")

#FOR RECALL AND PRECISION SCORE, WE CAN USE THIS
#precision(as.factor(yobs),as.factor(pred1_elasticnet))
#recall(as.factor(yobs), as.factor(pred1_elasticnet))

################
#RECALL AND PRECISION SCORE, WE CAN USE THIS
#precision(as.factor(yobs),as.factor(pred1_elasticnet))
#recall(as.factor(yobs), as.factor(pred1_elasticnet))
# Assuming yobs and pred1_elasticnet are binary (0/1) vectors or factors
library(caret)

# Calculate precision
precision_score <- posPredValue(as.factor(pred1_elasticnet), as.factor(yobs))

# Calculate recall
recall_score <- sensitivity(as.factor(pred1_elasticnet), as.factor(yobs))

# Print the precision and recall scores
cat("Precision Score:", precision_score, "\n")
cat("Recall Score:", recall_score, "\n")


#RECALL AND PRECISION CURVES
#install.packages("precrec")
library(precrec)
precrec_elasticnet <- evalmod(scores = pred_elasticnet, labels = yobs)
precrec_elasticnet
autoplot(precrec_elasticnet)

############################################
#LASSO REGRESSION
#Using cross validation to determine the optimal tuning parameter. To chose the best lambda
CV1 <- cv.glmnet(x=X, y=up_train_data$Class, family="binomial", alpha = 1, # CROSS VALIDATION METHOD
                 nlambda = 200)#we can take out part of the fucntion from standardize =T till maxit=100 so as to let R choose best lamda
                 for us.

plot(CV1)
coef(CV1, CV1$lambda.min)
coef(CV1, CV1$lambda.1se)

b.lambda1 <- CV1$lambda.1se #we can use 1se or min      THIS GIVES US THE BEST LAMBDA
b.lambda1

fit.best1 <- glmnet(x=X, y=up_train_data$Class, family="binomial", alpha = 1,          #BEST FIT WILL SHOW SOME DATA
                    DISAPPEARING BECAUSE WE USED LASSO i.E. LAMBDA = 1
                      lambda=b.lambda1)
(fit.best1$beta)

# Prediction
X.test <- model.matrix (as.formula(formula0), data = test)[, -1]
pred_lasso <- predict(fit.best1, newx = X.test, type="response") #type = "response" means y is continues
(head(pred_lasso <- round(pred_lasso, digits = 2)))
(tail(pred_lasso <- round(pred_lasso, digits = 2)))
```

```
dim(pred_lasso)

##### Finding missclassification rate #making a threshhold
pred1_lasso <- ifelse(pred_lasso>0.5, 1, 0)  #if it is bigger than 50%, we say yes

(miss.rate <- mean(yobs != pred1_lasso))

pred_lasso<-pred_lasso[,1] #gives only the index (1, 0) and not the column

pred1_lasso<-pred1_lasso[,1]

#Plotting ROC curve of the fit.best model.
library(cvAUC)
AUC1 <- ci.cvAUC(predictions = pred1_lasso, labels = yobs, folds=1:NROW(test), confidence = 0.95)
AUC1
(auc.ci_lasso <- round(AUC1$ci, digits = 3))

library(verification)
mod.glm_lasso <- verify(obs = yobs, pred = pred_lasso)
roc.plot(mod.glm_lasso, plot.thres=NULL)
text(x=0.7, y=0.2, paste("Area under ROC = ", round(AUC1$cvAUC, digits = 3), "with 95% CI (",
                         auc.ci_lasso[1], ",", auc.ci_lasso[2], ").", sep = " "), col="purple", cex =1.2)

#confusion matrix  (we dont need it in this case because the data in imbalance)
#library(caret)
#library(e1071)
#confusionMatrix( as.factor(pred1),as.factor(yobs),positive = "1")

#FOR RECALL AND PRECISION SCORE, WE CAN USE THIS
#precision(as.factor(yobs),as.factor(pred1_lasso))
#recall(as.factor(yobs), as.factor(pred1_lasso))

#RECALL AND PRECISION CURVES
#install.packages("precrec")
library(precrec)
precrec_lasso <- evalmod(scores = pred_lasso, labels = yobs)
precrec_lasso
autoplot(precrec_lasso)

###############

# Calculate precision
precision_score <- posPredValue(as.factor(pred1_lasso), as.factor(yobs))

# Calculate recall
recall_score <- sensitivity(as.factor(pred1_lasso), as.factor(yobs))

# Print the precision and recall scores
cat("Precision Score:", precision_score, "\n")
cat("Recall Score:", recall_score, "\n")

#RECALL AND PRECISION CURVES
#install.packages("precrec")
library(precrec)
precrec_lasso <- evalmod(scores = pred1_lasso, labels = yobs)
precrec_lasso
autoplot(precrec_lasso)
```