CMPE 255 - Data Mining

CREDIT CARD FRAUD DETECTION - GROUP 10

Armaghan Abtabhi 016565965 **Kunjal Shah 016698747**

Dasaradh Gutta 015949258 **Priyanka Sharma 016037125**

1. Introduction

1.1 Motivation

The number of Americans using credit cards is at an all-time high. Globally, there are currently close to 3 billion credit cards in use. The risk of fraud increases along with the use of cards. Despite the use of modern security measures, identity thieves continue to find ever-more creative techniques to deceive consumers, businesses, and organizations. So it necessitates the development of sophisticated algorithms that are capable of identifying subtle patterns and abnormalities in huge and complicated datasets. Credit card fraud detection is a difficult and fascinating subject in the field of machine learning. The potential to have a real-world impact while also enhancing the state-of-the-art in machine learning is what led to the selection of the topic of credit card fraud detection.

1.2 Objectives

The objective is building a ML model which will determine the credit card fraud transaction is fraudulent or genuine.The dataset contains numerical features(V1 to V28) obtained through PCA transformation, it also contains time and transaction amount.The objective is to manage the imbalanced dataset, choose appropriate characteristics using statistical tests, and compare various machine learning algorithms to find the one that performs the best in terms of reliably detecting fraudulent transactions while minimizing false positives.Data visualization will be used in the notebook to gain insights into the dataset, statistical tests will be used to select features, data balancing methods like SMOTE will be used, and various ML models, including logistic regression, SVM, random forests, decision trees, and KNN, will be used to model the dataset. The findings will also be presented for easier understanding. The ultimate objective is to create a model that can precisely identify credit card fraud, which can aid in preventing monetary losses and enhancing the security of credit card transactions.

2. System Design/Implementation

In this section, we will cover the algorithms that were chosen, the technology and tools that were utilized, the design of the system, and various scenarios in which the system can be used.

2.1 Selected Algorithms

In this project, we classified whether a credit card transaction is fraudulent or genuine. This is a **binary classification** problem with highly unbalanced data. The project consists of these parts:

2.1.1- Dataset Information/ Visualization:

The dataset provided consists of credit card transactions made by European cardholders in September 2013. The dataset spans a duration of two days and contains a total of 284,807 transactions. Among these

transactions, there are 492 instances of fraudulent activity, indicating a highly imbalanced dataset where fraud cases represent only 0.172% of the total transactions.

The data shows a significant imbalance with the majority of the transactions categorized as "No Fraud." Therefore, a classification model trained on this data will tend to favor the majority class and may not accurately predict the presence of fraud. Consequently, it is essential to balance the data to ensure the model is accurate.
Dataset Link: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

2.1.2- Models:

Logistic Regression:

A statistical technique called logistic regression is applied to binary classification issues with the aim of estimating the likelihood of an event occurring based on a set of input factors. Using a logistic function to calculate the likelihood that the dependent variable will take a specific value, it simulates the link between one or more independent factors and a categorical dependent variable. Using a decision threshold, the logistic regression model maps a probability value to a binary event as its output. The model is frequently used in a variety of industries, including marketing, healthcare, and finance, to forecast consumer behavior, diagnose diseases, and determine credit risk, respectively.

Support Vector Classifier:

The supervised learning algorithm known as Support Vector Classifier (SVC), often referred to as Support Vector Machine (SVM), is a tool for binary and multi-class classification issues. It operates by locating a hyperplane in the feature space that optimizes the distance between the two classes. The margin is the separation between the nearest data points from both classes and the hyperplane. Finding the best hyperplane to divide the classes in a high-dimensional space known as the kernel space is the basic goal of SVM. In order to accomplish this, the initial input features are mapped to a higher-dimensional space where the data can be separated more easily. In this higher-dimensional space, SVM determines the hyperplane that optimizes the margin.

Decision Tree Classifier:

A supervised learning algorithm for classification issues is a decisiontreclassifier. Each internal node serves as a characteristic or attribute, and each branch, based on the value of the feature, serves as a decision ruleThe leaves of the tree represent the class labels. The algorithm works by recursively partitioning the data into smaller subsets based on the values of the features. The partitioning is done in a way that maximizes the information gain or minimizes the entropy at each node. This means that at each step, the algorithm selects the feature that provides the most significant reduction in the impurity of the subsets.

Random Forest Classifier:

A type of ensemble learning algorithm used for classification issues is the RandomForestClassifier. Based on the concept of decision trees, it produces the class which reflects the mean of the classes (classification) or mean prediction (regression) of the individual trees by constructing a large number of decision trees during training. The risk of overfitting decreases as each decision tree in the forest is built using a random subset of the features and a random portion of the training data. The feature that maximizes the information gain or the Gini impurity is chosen at each node of each tree to find the ideal split

K-Nearest Neighbors:

K-Nearest Neighbors (KNN) is a supervised ML model used in regression and classification applications such as the one in question. The model works in identifying ‘k’ nearest points in the feature space to a given query point. The class or value of the query is then determined based on the average value of the KNN. The distance metric used to measure the proximity between data points can vary, but the most commonly used metric is Euclidean distance.

2.2 Tools and Technologies

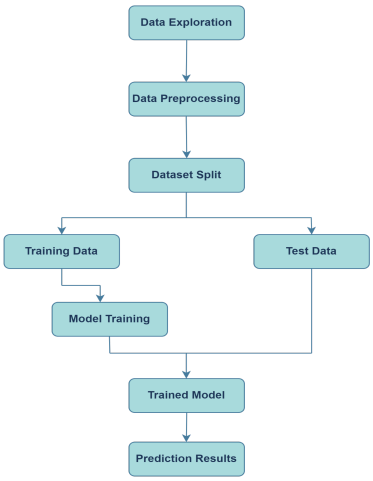
The tools and technologies used in the project are listed as follows:

1. Google Colab for shared code development
2. Sk-learn for building and evaluating machine learning models.
3. pandas for data manipulation and analysis.
4. numpy for numerical computing and array manipulation.
5. matplotlib for creating visualizations.
6. seaborn for creating statistical visualizations.

The specific modules imported from these libraries include various classifiers such as KNeighborsClassifier, RandomForestClassifier, DecisionTreeClassifier, SVC, and LogisticRegression. For evaluating the performance of the models these imported modules are used: confusion_matrix, roc_auc_score, plot_roc_curve, precision_recall_curve, and classification_report. Additionally, the imported modules for performing feature selection such as SelectKBest and f_classif, as well as modules for performing hyperparameter tuning such as GridSearchCV and RepeatedStratifiedKFold are used. Finally, we used modules for splitting the data into training and testing sets.

2.3 Process Flow

The process flow is illustrated in the figure on the right.



3. Methodology

3.1. Data Preprocessing

The data pre-processing steps are essential for building a robust classification model, especially when the dataset is highly unbalanced as in our case. Here are the key data pre-processing steps used in the project:

3.1.1. Data Balancing:

Since our dataset is highly unbalanced, with the majority of transactions being labeled as "No Fraud," data balancing becomes crucial to avoid bias in our model's predictions. We suggest using a combination of undersampling and oversampling techniques for data balancing.

- **Undersampling:** This step involves trimming down the majority class samples(No Fraud cases) to reduce their dominance in the dataset. We determine the undersampling ratio based on the

desired minority class samples. In this case, we use a sampling strategy of 0.1, which corresponds to achieving a ratio of 492 minority to majority class samples.

- **Oversampling:** After undersampling, we increase minority class samples (Fraud cases) to further balance the dataset. The oversampling ratio is determined based on the desired minority samples. In this case, we use a sampling strategy of 0.5, resulting in a ratio of 2460 minority to majority class samples.
- **Final Class Samples:** After applying both undersampling and oversampling, our dataset consists of 4920 samples for both the majority class (No Fraud cases) and minority class (Fraud cases).

To implement data balancing, we chose to utilize the ***imbalanced-learn*** library. By installing this library using the pip install imbalanced-learn command, we were able to access its functionalities for undersampling and oversampling.

3.1.2. Feature Selection:

The dataset contains a large number of features, which makes it difficult to comprehend. Therefore, we will create a correlation map that focuses only on the relationship between each feature and the target variable. We will create two models based on the features selected from the correlation plot and the ANOVA score plot as described below:

- **Correlation Matrix**: Initially, we compute a correlation matrix to understand relationships between features and target variables. We plot a heatmap to visualize the correlations. We find that the features V3, V4, V7, V10, V11, V12, V14, V16, and V17 have significant correlations (both positive and negative) with the target variable (Class).
- **ANOVA Test**: We also employ the ANOVA (Analysis of Variance) test as a feature selection technique. This test calculates the ANOVA scores for each feature by evaluating the relationship between the feature and the target variable. A higher ANOVA score indicates a stronger relationship. We sort the features based on their ANOVA scores and select only the features with scores greater than 50 for our model.

3.1.3. Dataset Creation:

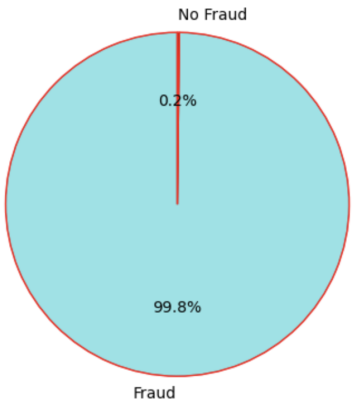
We created two separate datasets for modeling based on the two feature selection methods:

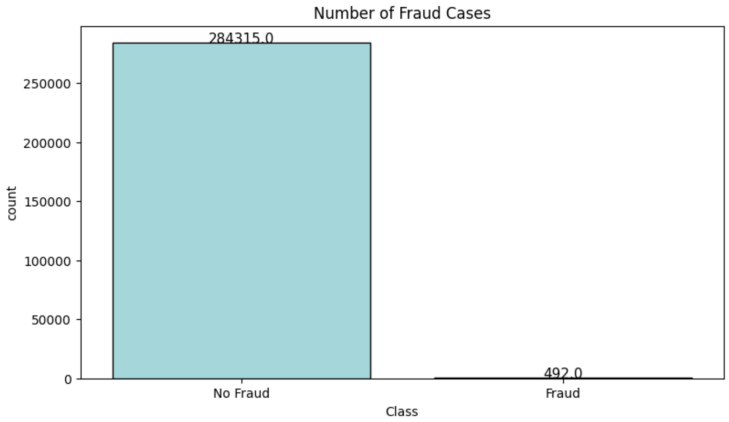
- **Dataset for Model based on Correlation Plot (df1):** This dataset includes 9 features and the target variable Class.
- **Dataset for Model based on ANOVA Score (df2):** This dataset includes the selected features from the ANOVA test, determined by ANOVA scores greater than 50.

3.1.4. Model Evaluation:

Since we have balanced our dataset using a combination of undersampling and oversampling, the traditional evaluation metric of accuracy may not provide an accurate representation of our model's performance. Instead, tested and used the confusion matrix, ROC-AUC graph, and ROC-AUC score for model evaluation. These metrics provide insights into our model's performance in detecting both fraud and non-fraud cases.

3.2. Data Visualization





3.3. Model Evaluation Techniques

We consider cross-validation, ROC-AUC scores, ROC curve visualization, the confusion matrix, and a detailed classification report to obtain a comprehensive understanding of our model's performance. The techniques are described as follows:

Cross-Validation Score:

We use the "***cross_val_score"*** function from scikit-learn for obtaining cross-validation score. It performs cross-validation by splitting our training data into multiple folds and trains model on the different combinations. The score represents the average performance of our model across all the folds. In this code, we use the "***roc_auc***" scoring metric, which calculates the area under the ROC curve. We calculate the mean of the cross-validation scores and display it as a percentage.

ROC-AUC Score and Plot:

We used ***roc_auc_score*** function from SK-Learn to obtain the ROC-AUC score. This score measures the performance of our classification model by computing the area. It depicts the true positive rate (sensitivity) against the false positive rate (1 - specificity). The ROC-AUC score provides an overall accuracy of the model to distinguish between negative and positive cases. We display this score as a percentage.

Additionally, we employ the plot_roc_curve function to plot the ROC curve for our classifier. This curve visually represents the trade-off between the sensitivity and 1-specificity. The plot helps us assess performance of the model and its ability in balancing between true positives and false positives.

Confusion Matrix:

We use ***confusion_matrix*** function from SKLearn to calculate confusion matrix. This matrix summarizes performance of our model by displaying counts of true positive, false positive, true negative, and false negative predictions. The confusion matrix allows us to assess our model's performance in terms of correctly and incorrectly classified instances. We utilize the values in the confusion matrix to generate a heatmap using the seaborn library, which provides a visual representation of the matrix.

Classification Report:

To generate a comprehensive report of our model's performance, we utilize the ***classification_report*** function from scikit-learn. This report includes metrics such as F1-score, and support for each class(fraud and non-fraud). The F1-score, mean of recall and precision, is responsible in providing us with a measure

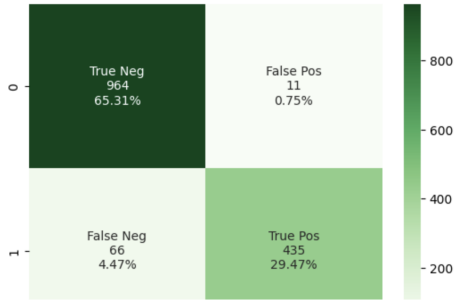
of our algorithm's accuracy in identifying the correct class. The classification report helps us gain insights into the performance of our model for each class.

3.4. Models

3.5.1 Logistic Regression:

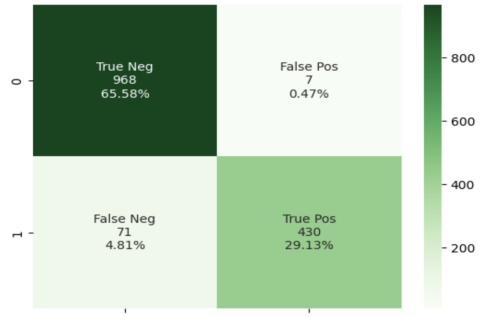
Model based on Correlation Plot:
Cross Validation Score: 98.31%
ROC_AUC Score: 92.85%

	precision	recall	f1-score	support
0	0.94	0.99	0.96	975
1	0.98	0.87	0.92	501
accuracy			0.95	1476
macro avg	0.96	0.93	0.94	1476
weighted avg	0.95	0.95	0.95	1476



Model based on ANOVA Score:
Cross Validation Score: 98.37%
ROC_AUC Score: 92.56%

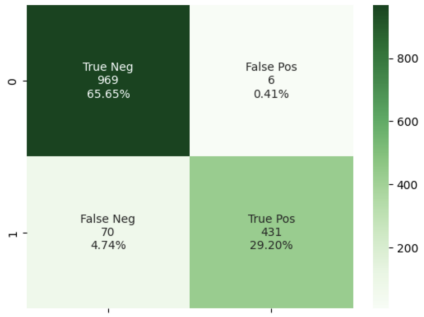
	precision	recall	f1-score	support
0	0.93	0.99	0.96	975
1	0.98	0.86	0.92	501
accuracy			0.95	1476
macro avg	0.96	0.93	0.94	1476
weighted avg	0.95	0.95	0.95	1476



3.5.2 Support Vector Classifier:

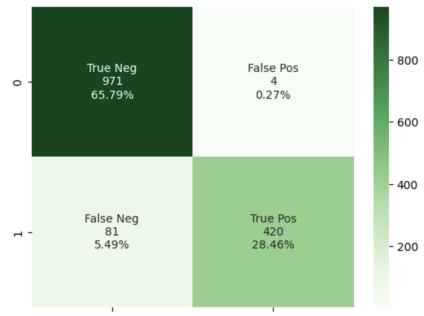
Model based on Correlation Plot:
Cross Validation Score: 98.32%
ROC_AUC Score: 92.71%

	precision	recall	f1-score	support
0	0.93	0.99	0.96	975
1	0.99	0.86	0.92	501
accuracy			0.95	1476
macro avg	0.96	0.93	0.94	1476
weighted avg	0.95	0.95	0.95	1476



Model based on ANOVA Score:
Cross Validation Score: 98.23%
ROC_AUC Score: 91.71%

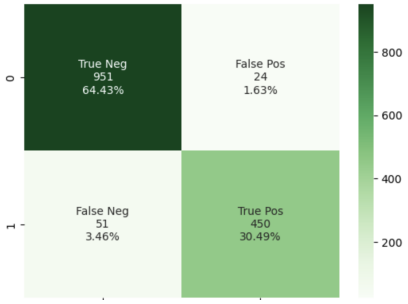
	precision	recall	f1-score	support
0	0.92	1.00	0.96	975
1	0.99	0.84	0.91	501
accuracy			0.94	1476
macro avg	0.96	0.92	0.93	1476
weighted avg	0.95	0.94	0.94	1476



3.5.3 Decision Tree Classifier:

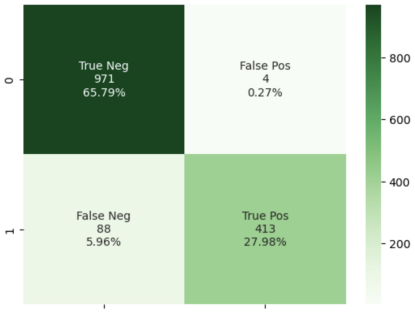
Model based based on Correlation Plot:
Cross Validation Score: 97.23%
ROC_AUC Score: 93.68%

	precision	recall	f1-score	support
0	0.95	0.98	0.96	975
1	0.95	0.90	0.92	501
accuracy			0.95	1476
macro avg	0.95	0.94	0.94	1476
weighted avg	0.95	0.95	0.95	1476



Model based on ANOVA Score:
Cross Validation Score: 96.48%
ROC_AUC Score: 91.01%

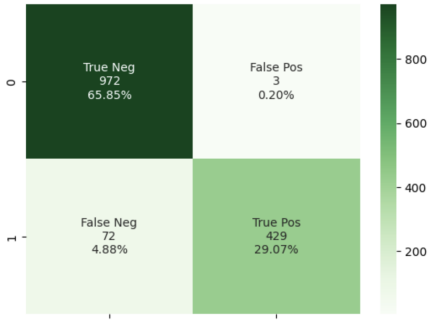
	precision	recall	f1-score	support
0	0.92	1.00	0.95	975
1	0.99	0.82	0.90	501
accuracy			0.94	1476
macro avg	0.95	0.91	0.93	1476
weighted avg	0.94	0.94	0.94	1476



3.5.4 Random Forest Classifier:

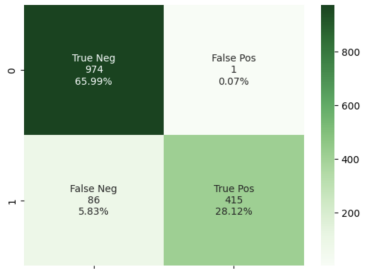
Model based on Correlation Plot:
Cross Validation Score: 98.35%
ROC_AUC Score: 92.66%

	precision	recall	f1-score	support
0	0.93	1.00	0.96	975
1	0.99	0.86	0.92	501
accuracy			0.95	1476
macro avg	0.96	0.93	0.94	1476
weighted avg	0.95	0.95	0.95	1476



Model based on ANOVA Score:
Cross Validation Score: 98.09%
ROC_AUC Score: 91.37%

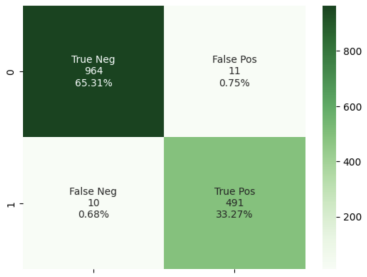
	precision	recall	f1-score	support
0	0.92	1.00	0.96	975
1	1.00	0.83	0.91	501
accuracy			0.94	1476
macro avg	0.96	0.91	0.93	1476
weighted avg	0.95	0.94	0.94	1476



3.5.5 K-Nearest Neighbors:

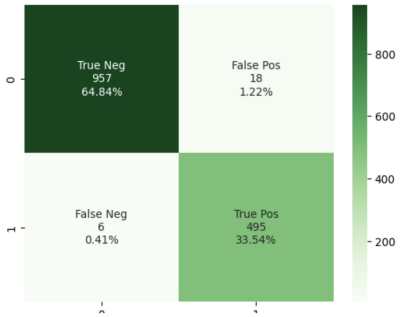
Model based on Correlation Plot:
Cross Validation Score: 99.32%
ROC_AUC Score: 98.44%

	precision	recall	f1-score	support
0	0.99	0.99	0.99	975
1	0.98	0.98	0.98	501
accuracy			0.99	1476
macro avg	0.98	0.98	0.98	1476
weighted avg	0.99	0.99	0.99	1476



Model based on ANOVA Score:
Cross Validation Score: 99.62%
ROC_AUC Score: 98.48%

	precision	recall	f1-score	support
0	0.99	0.98	0.99	975
1	0.96	0.99	0.98	501
accuracy			0.98	1476
macro avg	0.98	0.98	0.98	1476
weighted avg	0.98	0.98	0.98	1476



3.5. Results - Summary of Models

Model	Feature Selection Method	Cross Validation Score	ROC-AUC Score
Logistic Regression	Correlation Plot	98.31%	92.85%
Logistic Regression	ANOVA Score	98.37%	92.56%
Support Vector Classifier	Correlation Plot	98.32%	92.71%
Support Vector Classifier	ANOVA Score	98.23%	91.71%
Decision Tree Classifier	Correlation Plot	97.23%	93.68%
Decision Tree Classifier	ANOVA Score	96.48%	91.01%
Random Forest Classifier	Correlation Plot	98.35%	92.66%
Random Forest Classifier	ANOVA Score	98.09%	91.37%
K-Nearest Neighbors	Correlation Plot	99.32%	98.44%
K-Nearest Neighbors	ANOVA Score	99.62%	98.48%

4. Discussions And Conclusions

4.1. Decisions made

- In order to choose a dataset that satisfies the necessary requirements, we discussed and did research on the topic.
- We made decisions and selected specific actions that would be carried out throughout the project lifespan after the dataset was completed.
- The Kaggle competition introduction said that the JSON column labeled "totals" contains our goal variable, "totalTransactionRevenue," which is located there. To make use of these JSON values, we must flatten them first and then add them to our dataset.
- Finally, we choose among many regression models to forecast the objective.

4.2. Difficulties faced

- Null values: We deleted any columns that have more than 80% empty values because the given dataset has a lot of null values.
- Considering all the variables, including the complexities of the data, the volume of data, the sheer quantity of features, we also had trouble developing algorithms and getting the best results.
- Size of dataset: The training dataset was excessively big, so we immediately ran into problems with cleaning and preprocessing.

4.3. Things that worked

- With all of the regression models, Random Forest, support vector machine, etc. we modified the various parameters to get the highest accuracy.
- The dataset provided us with the opportunity to uncover the problems in dealing with such massive datasets. However, with proper study, we were able to deal with the data successfully.

4.4. Things that didn't work well

We began with logistic regression without data balancing, which returned an accuracy of 80%, but most of the Fraud cases weren't detected and the classification was highly biased towards non fraudulent cases due to the initial imbalance in the dataset. As a result, we experimented with several data balancing techniques and used a combination of undersampling and oversampling of the dataset to obtain accuracy of over 90% while also being able to identify the fraudulent transactions.

4.5. Conclusion

Finally, the credit card fraud detection dataset is a valuable tool for detecting fraudulent transactions and safeguarding financial institutions and customers from financial loss. We found that MLmethods like logistic regression, decision trees, and random forests are successful in detecting fraudulent transactions based on factors like transaction amount, location, and time. However, we acknowledge the continual challenge of keeping up with new fraudster strategies, as well as the necessity for continuous refining and enhancement of fraud detection algorithms. Overall, this initiative has helped to improve fraud detection capabilities and protect consumers from financial fraud.

5. Project Plan

- The project proposal was owned and developed by all team members.
- Data preprocessing, exploratory data analysis, feature selection, and data balancing were performed by Priyanka Sharma and Armaghan Abtahi.
- Model training, testing, and evaluation were handled by Kunjal Shah and Dasaradh Gutta.
- Report documentation was a team effort, with contributions from all team members.
- The final presentation was prepared and delivered by all team members together.

6. Github Link - https://github.com/ArmaghanAb/CMPE255_Project