**Sri Lanka Institute of Information Technology**



**Fundamentals of Data Mining – IT3051**

B.Sc. (Hons) in Information Technology

Credit Card Fraud Detection System

Mini Project – Final Report

Group Number: G39

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Name with Initials** | **Registration Number** | **Contact**  **Number** | **Email Address** |
| 1. | Zainab M.Z | IT21070594 | 0770708910 | it21070594@my.sliit.lk |
| 2. | Rashida M.S.F | IT21013850 | 0778330116 | it21013850@my.sliit.lk |
| 3. | Nuha M.N | IT21004568 | 0771713211 | it21004568@my.sliit.lk |
| 4. | Bishirhafi F.S.M.T | IT21006166 | 0741312573 | it21006166@my.sliit.lk |

LINKS:

GITHUB LINK: [**github repository link**](https://github.com/IT21070594/FDM-Project)

INTERFACE LINK: [**App Link**](https://credit-guard-pro.streamlit.app/)

VIDEO PRESENTATION:

DECLARATION

We hereby declare that the assignment submitted is original except for source material explicitly acknowledged. All the members of the group have read and checked that all parts of the piece of work, irrespective of whether they are contributed by individual members or all members as a group.

ABSTRACTION

In today's world, organizations gather and store a large amount of data, including credit card transactions. Data mining is a powerful tool that helps these organizations uncover valuable insights and patterns within this data.

Imagine you work for a credit card company, and you want to detect fraudulent transactions before they cause financial losses. Data mining and machine learning can help you achieve this goal.

This project involves a step-by-step process to analyze the data and find patterns that can indicate whether a transaction is fraudulent or not. It's like being a detective and looking for clues in the data to catch fraudulent activities.

Many businesses use these data mining techniques to enhance security and protect their customers from financial harm. So, in simpler terms, data mining in the context of credit card fraud detection is about using data to identify and prevent fraudulent transactions, just like a detective solving a case.

**CONTENTS**

[1. PROBLEM DEFINITION 4](#_Toc148751422)

[2. TARGET AND BUSINESS GOALS 5](#_Toc148751423)

[3. DATA MINING FUNCTIONALITY 6](#_Toc148751424)

[4. DATA SELECTION 7](#_Toc148751425)

[4.1 INTRODUCTION TO THE DATASET 7](#_Toc148751426)

[5. DATA PREPROCESSING 9](#_Toc148751427)

[5.1 DATA EXPLORATION AND UNDERSTANDING 10](#_Toc148751428)

[5.2 DATA CLEANING 15](#_Toc148751429)

[6. IMPLEMENTATION OF THE MODELS 21](#_Toc148751430)

[6.1 RANDOM FOREST MODEL 22](#_Toc148751431)

[6.2 LOGISTIC REGRESSION MODEL 27](#_Toc148751432)

[6.3 NAÏVE BAYES MODEL 32](#_Toc148751433)

[6.4 SUPPORT VECTOR MACHINE MODEL 36](#_Toc148751434)

[6.5 K-NEAREST NEIGHBOR (KNN) MODEL 41](#_Toc148751435)

[7. EVALUATION OF THE MODELS 46](#_Toc148751436)

[7.1 CONFUSION MATRICES 46](#_Toc148751437)

[KNN (K - Nearest Neighbors) 49](#_Toc148751438)

[7.2 CLASSIFICATION REPORTS FOR EACH OF THE MODEL: 49](#_Toc148751439)

[7.3 F1 SCORE, PRECISION AND RECALL 51](#_Toc148751440)

[8. FRONTEND DEVELOPMENT 52](#_Toc148751441)

[8.1 TECHNOLOGY STACK 52](#_Toc148751442)

[8.2 FRONTEND INTERFACE 52](#_Toc148751443)

[9. ASSESSMENT OF THE PROJECT 53](#_Toc148751444)

[9.1 PROBLEMS ASSOCIATED 53](#_Toc148751445)

[9.2 IMPROVEMENTS 53](#_Toc148751446)

[10. WORK DISTRIBUTION 55](#_Toc148751447)

# **PROBLEM DEFINITION**

In recent years, there has been a noticeable increase in the number of credit card fraud cases, as more and more fraudulent transactions occur. These fraudulent activities pose a significant challenge for financial institutions, especially with the rise of online transactions. It has been reported that "almost 20% of online credit card transactions are fraudulent," according to industry reports. This represents a substantial portion of all transactions, impacting both financial losses and the reputation of the financial institution.

Customers typically have the freedom to use their credit cards for various transactions, including online purchases. However, this freedom has also led to increased risks of fraudulent activities, as fraudsters can exploit vulnerabilities in the system. This not only affects financial losses but also damages the reputation of the institution and can lead to higher costs for security measures.

To address this problem, it's essential to analyze the patterns and behaviors associated with fraudulent transactions. Data mining and machine learning techniques can play a crucial role in identifying these patterns and predicting the likelihood of a transaction being fraudulent.

Customers often make transactions for a variety of goods and services, including online purchases and in-store transactions. When fraudulent transactions occur, it can lead to financial losses for both the cardholder and the financial institution. Additionally, fraudulent activities can disrupt the normal operation of the financial system, causing inconvenience and financial stress for customers.

By applying data mining and machine learning technologies, we can create classification models to determine if a transaction is likely to be fraudulent or not. These models will analyze transaction data thoroughly, identifying factors that contribute to fraudulent activities. Out of the various models tested, the most accurate one will be recommended to improve the security and profitability of the financial institution.

Therefore, it is crucial to leverage insights from data to understand the key parameters that affect fraudulent transactions. This will help in identifying the exact reasons behind these activities, allowing for more effective fraud prevention measures and minimizing financial losses for both customers and the institution.

# **2.** **TARGET AND BUSINESS GOALS**

In the context of credit card fraud detection, the primary business goal is to minimize financial losses due to fraudulent transactions. This overarching goal includes several sub-goals outlined below:

1. Minimize Financial Losses: The main objective is to reduce the financial impact caused by fraudulent credit card transactions. This includes preventing unauthorized charges and refunds for fraudulent activities.

2. Optimize Resource Allocation: To achieve this goal, it's essential to efficiently allocate resources to combat fraud. This involves investing in fraud detection technologies and personnel while avoiding unnecessary expenditures.

3. Early Fraud Detection: The system should be capable of identifying potential fraudulent transactions as early as possible. Early detection can prevent or limit the financial damage caused by fraudsters.

4. Prevent Last-Minute Frauds: One of the sub-goals is to minimize last-minute fraudulent activities. Detecting and preventing fraud before transactions are completed is crucial to reducing financial risks.

5. Communication with Clients: Establishing communication channels with clients and intermediary parties can help inform them about potential fraud concerns. This proactive approach can lead to early intervention and mitigation.

6. Minimize False Positives: While detecting fraud is crucial, it's also essential to minimize false positives. False alarms can inconvenience legitimate cardholders and impact their trust in the financial institution.

7. Enhance Customer Experience: Balancing fraud prevention with positive customer experience is vital. Ensuring that customers' preferences and security concerns are met can lead to higher customer satisfaction and loyalty.

8. Reduce Fraud Records: Ultimately, the goal is to reduce the number of fraudulent transactions recorded for a particular credit card company. This includes minimizing the frequency and severity of fraudulent incidents.

# **DATA MINING FUNCTIONALITY**

In response to the challenges outlined earlier, a practical solution has been implemented, consisting of a web application integrated with a machine learning model. This solution aims to address the problem of predicting whether a credit card transaction is fraudulent or not. The primary objective is to proactively identify potentially fraudulent transactions, allowing for early intervention and minimizing financial losses.

In this project, the main data mining functionality that will be explored is classification. Classification is “the procedure of discovering a model that represents and distinguishes data classes or concepts, for the objective of being able to use the model to predict the class of objects whose class label is anonymous.

Key elements of this solution include:

1.Scalable Web Application: A user-friendly and scalable web application has been developed. This application provides an interface for users to input transaction data and receive predictions regarding the likelihood of a transaction being fraudulent.

2. Machine Learning Model: The heart of this solution is a machine learning model. Initially, a training dataset is used to train the model. During this training phase, the model learns to recognize patterns and characteristics associated with fraudulent transactions.

3. Testing and Evaluation: After training, the model is tested using a separate testing dataset. This evaluation phase assesses the model's accuracy in predicting fraudulent transactions. The model's performance is measured, and various metrics, such as accuracy, precision, recall, and F1 score, are considered to determine its effectiveness.

4. Model Selection: Among the models tested, the one with the highest accuracy and overall performance is selected as the final model for making predictions.

5. Predictive Functionality: With the chosen model, the web application can predict, in real-time, whether a given credit card transaction is likely to be fraudulent. Users can input transaction details, and the model will provide a prediction based on its learned patterns.

By implementing this solution, the credit card company aims to enhance its fraud detection capabilities, reduce financial losses caused by fraudulent transactions, and provide a more secure and reliable service to its customers. Data mining, particularly classification, plays a critical role in achieving these objectives by enabling the automated identification of potentially fraudulent activities.

# **DATA SELECTION**

## **INTRODUCTION TO THE DATASET**

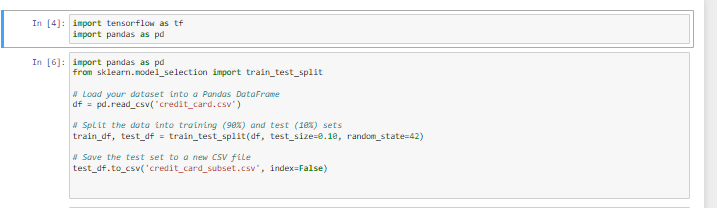
This dataset was obtained from a data science site called DataCamp, which had obtained and put together a dataset from different information obtained from various banks, it was partially cleaned and adapted by them.

The dataset is comprised of purchase transactions between several merchants and customers. The dataset contains information about credit card transactions, including various attributes such as customer details, merchant information, transaction amount, location of purchase, and whether each transaction was fraudulent or not.

The motivation behind this task is to protect the credit card company's customers from fraud and enhance their confidence in using the company's credit cards. By accurately identifying fraudulent transactions, the company can prevent financial losses and maintain a reputation for safety and security.

The main objective is to create a predictive model that can accurately identify fraudulent transactions. The emphasis is on being cautious, meaning that the model should prioritize correctly flagging transactions as fraudulent, even if it occasionally raises false alarms.

The dataset consists of 339,607 records and 15 columns, since the dataset was too huge and it took very long to run the models, a subset containing 10% (33,960 records) of the data was derived from the main dataset using stratified sampling technique, so that the classes will be equally distributed.



The columns of the initial dataset and the respective data types are shown below.

|  |  |
| --- | --- |
| **COLUMN NAME** | **DATA TYPE** |
| is\_fraud | int64 |
| trans\_date\_trans\_time | object |
| merchant | object |
| category | object |
| amt | float64 |
| city | object |
| state | object |
| late | float64 |
| long | float64 |
| city\_pop | int64 |
| job | object |
| dob | object |
| trans\_num | object |
| merch\_lat | float64 |
| merch\_long | float64 |

Dataset link: [Dataset link](https://drive.google.com/drive/folders/1MZWEmRZFHMvglwNqYTrl797eb3holptB)

# **DATA PREPROCESSING**

As a part of the Knowledge Discovery in Databases process, data pre-processing plays a significant role. Therefore, exceptional care was paid to complete this process in a detailed manner. The data preprocessing process was carried out in four main sections: **Data Exploration and Understanding, Data Cleaning, Feature Engineering, Formatting data, Feature Selection and Handling Class Imbalance using oversampling.**

Initially, the necessary imports were called:



Once the libraries were imported, the dataset was loaded as shown below.



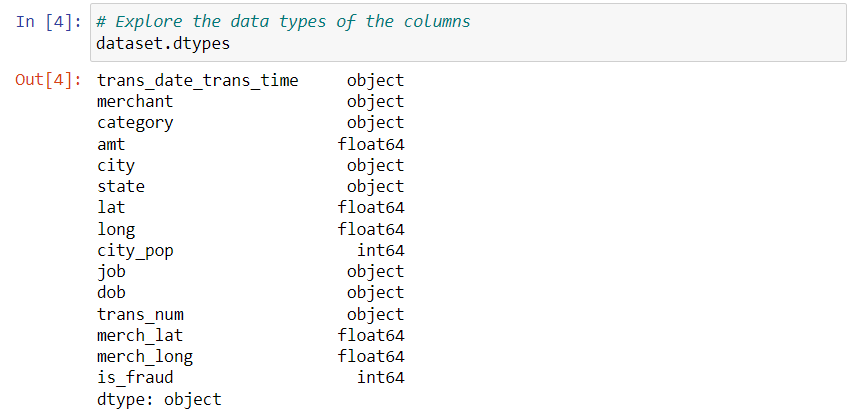
The process of implementation of the preprocessing is explored in detail in the following sections.

## **DATA EXPLORATION AND UNDERSTANDING**

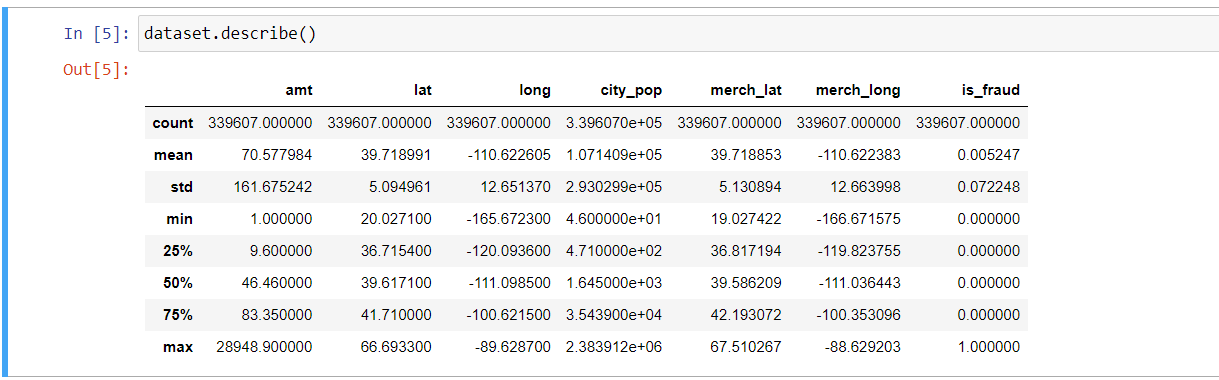
Initially, before any changes were made to the dataset, it was necessary to observe what the dataset looks like. This was done as shown:



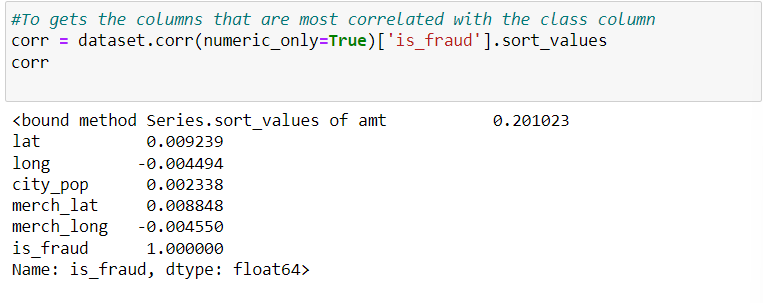
The following figure shows an overview of the columns that are available in the dataset:



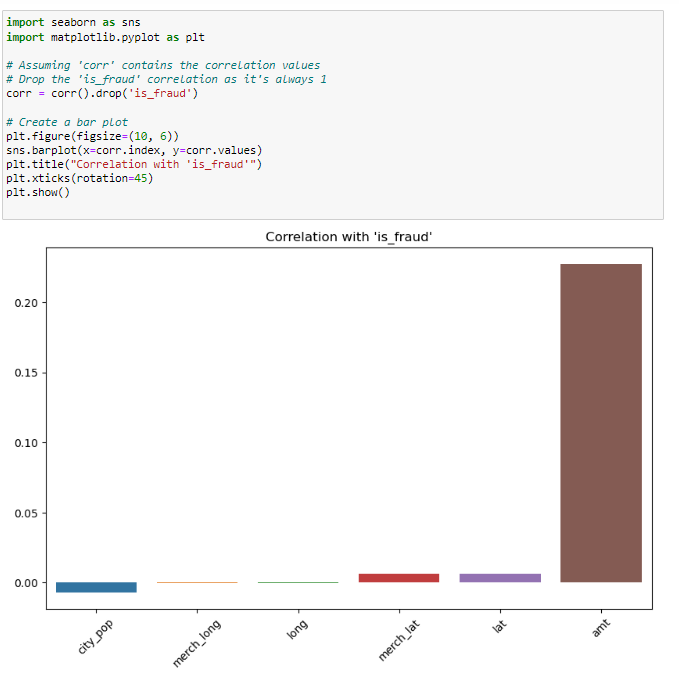
The following was carried out to describe the values in each of the columns:



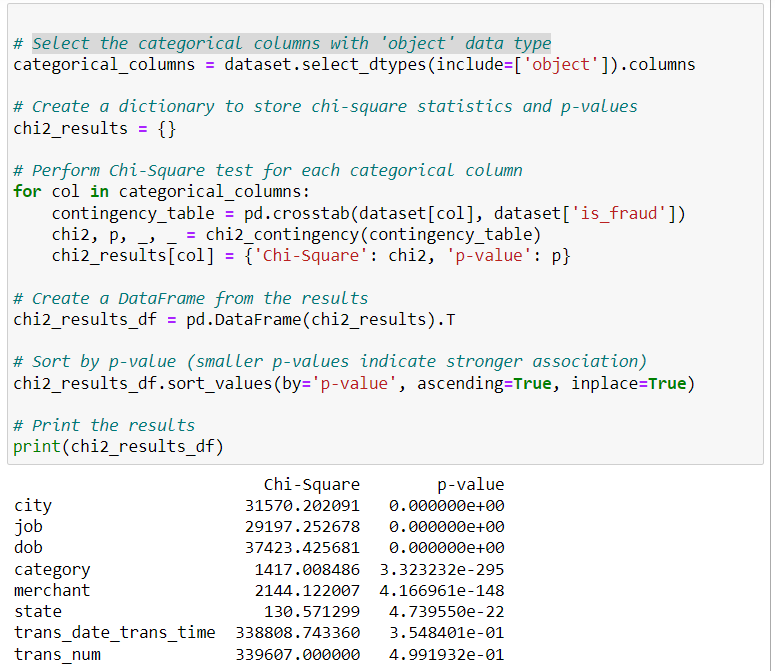
To check the numerical columns that are the most correlated with the class column, “is\_fraud”, the correlation values respective to the class attribute were found, so that the significancy level of each column can be measured. The 'amt' feature appears to have a positive correlation with 'is\_fraud,' suggesting that higher transaction amounts might be associated with a higher likelihood of fraud. The numerical location features and 'city\_pop' feature have a very weak correlation with 'is\_fraud,' suggesting they may not be strong predictors.



A bar chart was plotted to compare the correlation of the numeric columns to the class column:



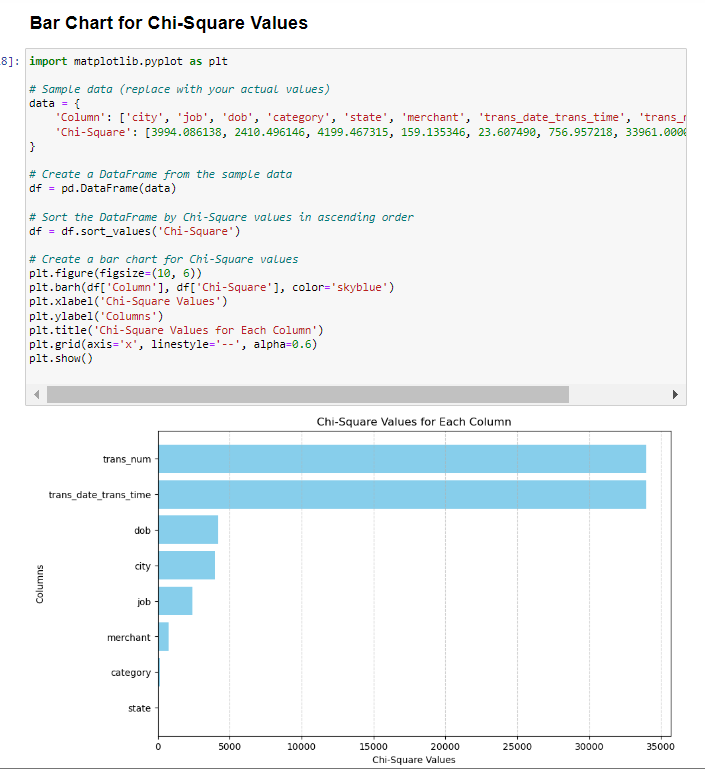
To check the correlation between categorical columns (with the data type 'object') (city, job, dob, category, merchant, state, trans\_date\_trans\_time, and trans\_num) and a binary target variable 'is\_fraud’ in the dataset, we calculated the point-biserial correlation coefficient for each categorical column against the 'class' column.





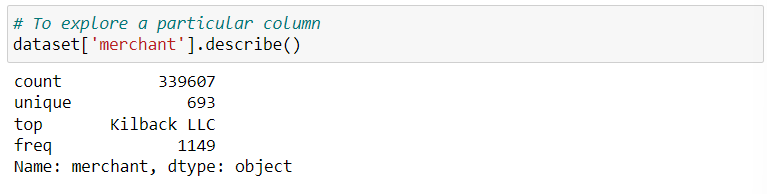
A bar chart was plotted for –log(p-value) for each column as above.

A bar chart was plotted to compare the Chi Square values for each column:



Columns like 'city,' 'job,' 'dob,' 'category,' 'state,' and 'merchant' have exceptionally low p-values close to zero, indicating a strong association with the target variable ('isFraud'). The 'trans\_date\_trans\_time' column has a p-value of approximately 0.354, indicating that it may not be strongly associated with the target variable. The 'trans\_num' column has a p-value of approximately 0.499, indicating no significant association with the target variable.

To explore a particular column and determine the number of unique values, and categories in the categorical columns the following was implemented:



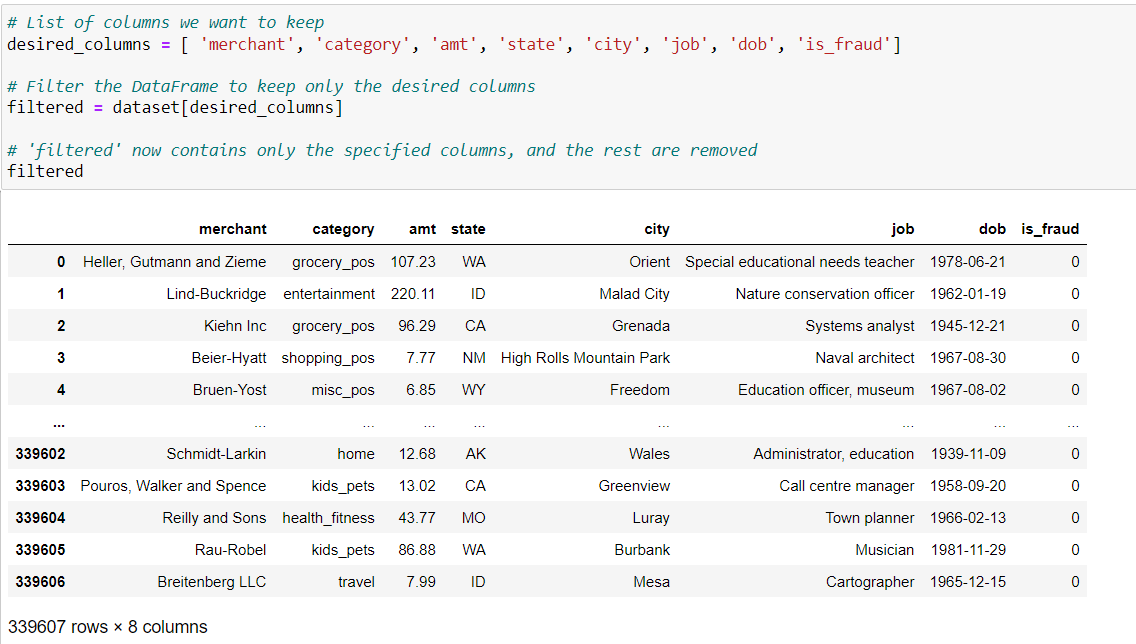
Doing this helped make decisions like the type of encoding process and choosing best features and other data preprocessing decisions.

The following table includes the number of unique categories in each of the categorical columns:

|  |  |
| --- | --- |
| Categorical Column | Number of categories |
| Merchant | 693 |
| Category | 14 |
| City | 176 |
| State | 13 |
| Job | 163 |
| trans\_date\_trans\_time | 338504 |
| dob | 187 |

## **5.2** **DATA CLEANING**

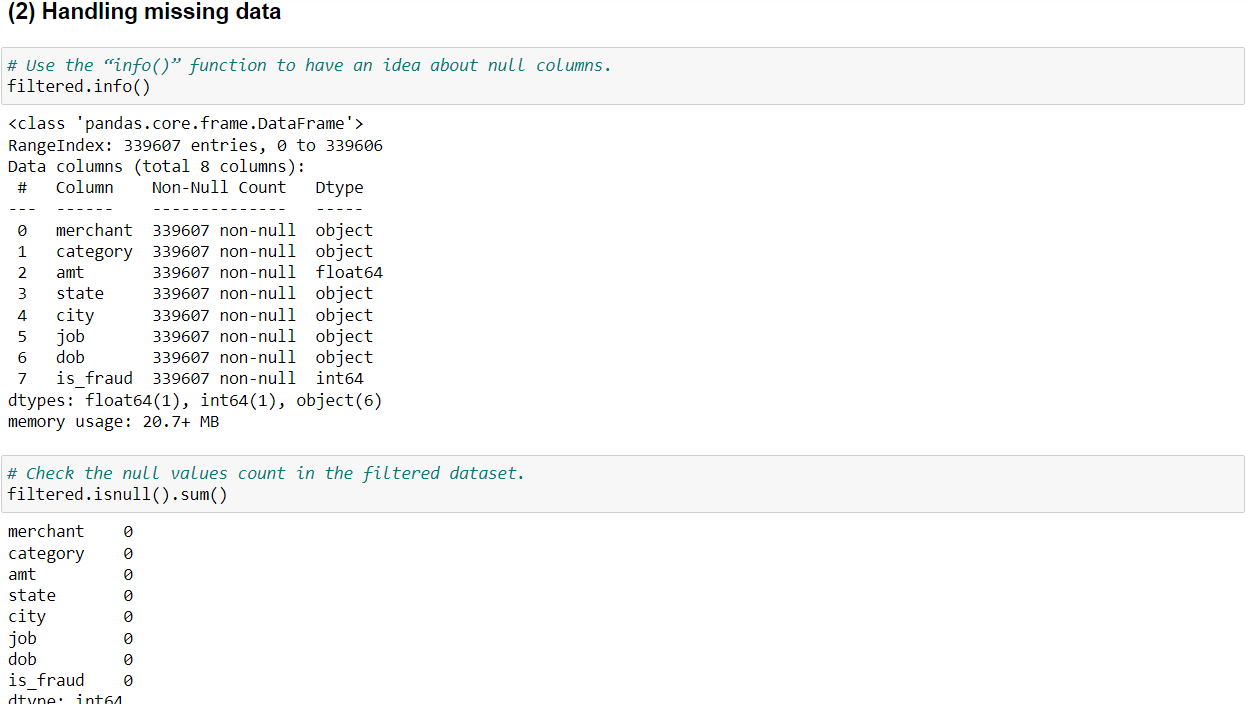
The process of data cleansing was begun with the first step being removing irrelevant columns as shown below:



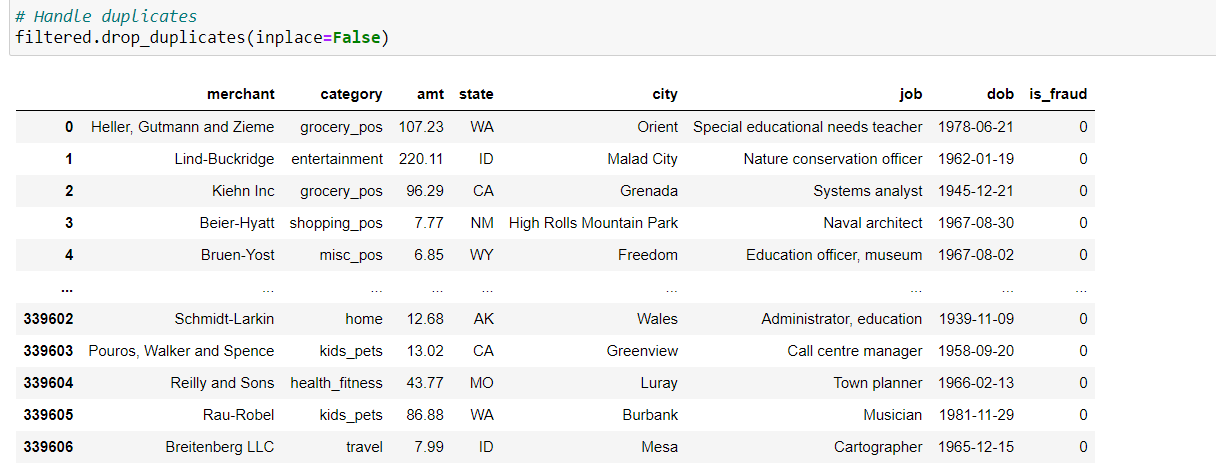
The columns lat, long, merch\_lat, merch\_long and city\_population were dropped because when exploring the data set it was found that these were weakly correlated with the ‘is\_fraud’ class column, suggesting that they were not strong predictors.

The columns ‘trans\_date\_trans\_time’ and ‘trans\_num’ were dropped because they had significantly high p values, which indicated no significant association with the target variable. (this was since they were unique values each time and this meant they did not help predict the class column value in any way).

As the next step of data cleansing, it was taken as a necessary step to remove any tuples containing null values, while exploring for null values it was found that none of the columns contained any null values, this was because the dataset on the website was initially cleansed.



Once this was completed, it was checked to see if there were any duplicate rows. And duplicate rows if present was meant to be removed as a cleansing mechanism.

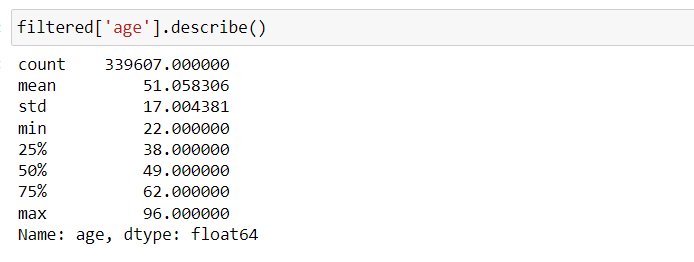
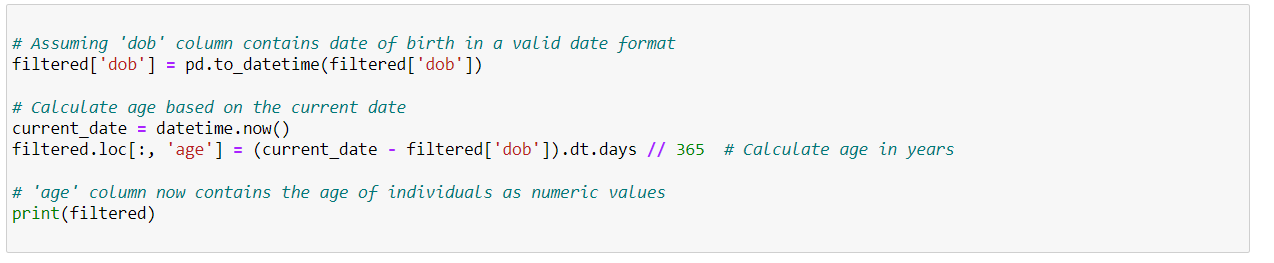


5.3 FEATURE ENGINEERING

Feature engineering is the process of selecting, creating, or transforming features (attributes or variables) in a dataset to improve the performance of machine learning models. It involves identifying relevant information within the data, extracting meaningful patterns, and representing them in a way that enhances the model's ability to make accurate predictions or classifications.

While running the models it was found that the dob column was in object form, therefore, to convert it to numerical/ binary data type we as the first step of feature engineering we converted the dob column to age column by calculating the age of the individuals based on their birthdates. We calculated the age by subtracting the birthdate from the current date (datetime.now()) and then converting the result to years using the astype('<m8[Y]') method.

Now, the 'age' column will contain the age of individuals as numeric values in years.



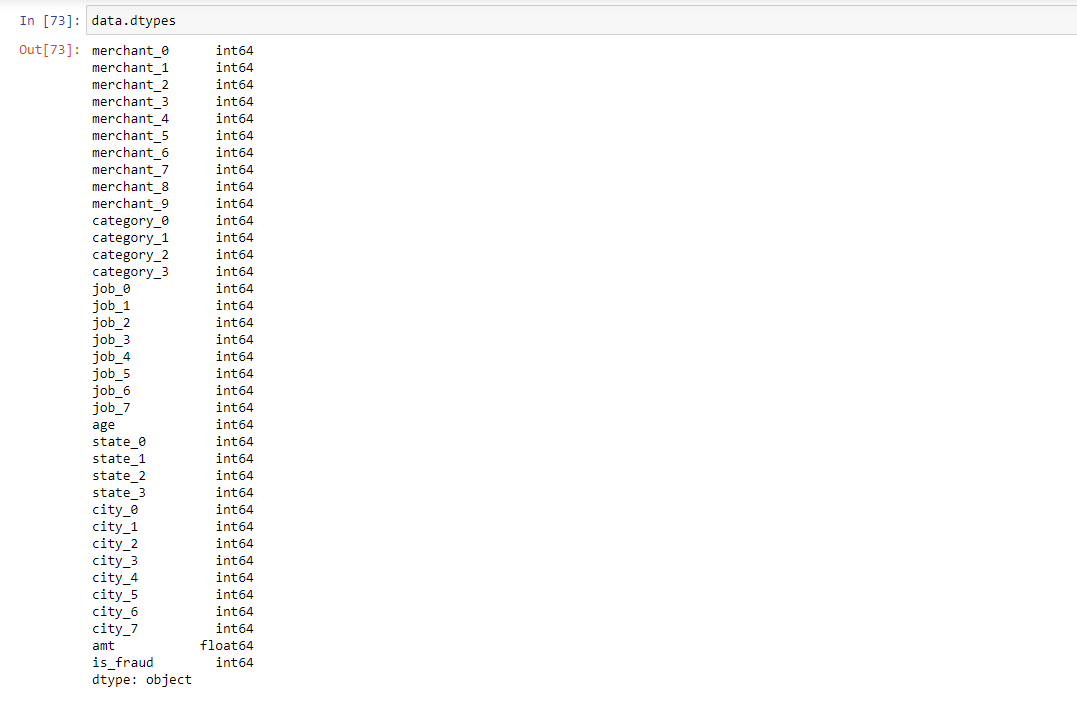
Once we completed this we dropped the dob column.

The next step of feature engineering we did was encoding the categorical columns, we had to choose between different encoding methods such as one-hot encoding, label encoding, binary encoding, and ordinal encoding. We decided to use binary encoding after eliminating the rest of the methods due to the following reasons,

* + One hot encoding- Each category in each categorical column was converted to a separate column and the presence of that particular category was represented by one for that column, in the case of our dataset, the merchant column had 693 categories, job had 163 categories and city had 176, using one hot encoding for all categorical variables gave 1065 columns once the encoding was done, this affected the dimensionality of the dataset significantly, hence we eliminated this method.
  + Label encoding and ordinal encoding – this gave a unique integer value for each unique category, but using this would have resulted in the column gaining an ordinal feature which would have an impact on the classification model we choose to do hence this was eliminated as well.

As a solution we decided to go with binary encoding which is a combination of both label encoding and one hot encoding, but it maintained the dimensionality and reduced the number of columns to 37 and reduced the ordinal relationship in the column as well.

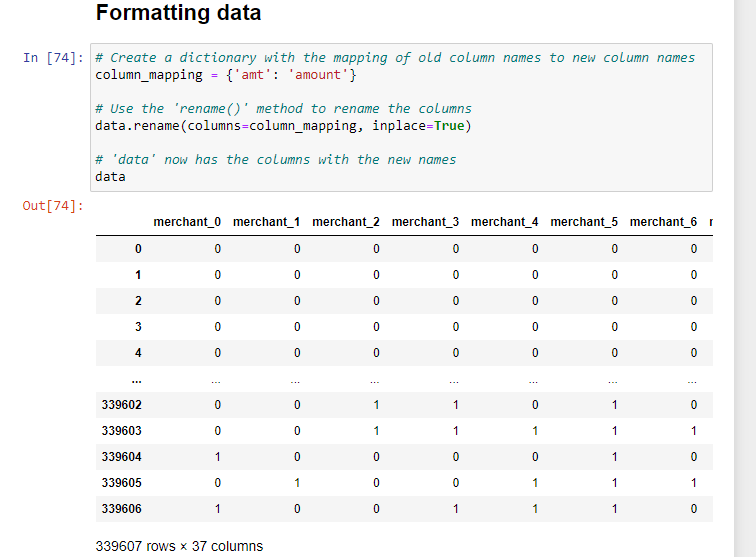




At the end of feature engineering step, all columns are with the correct data types and correct formats.

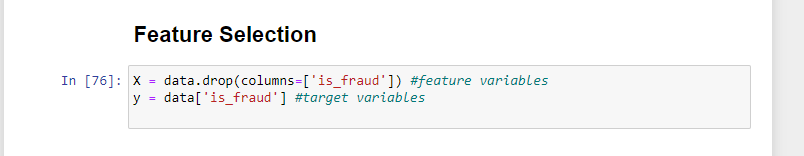
5.4 FORMATTING DATA

We formatted the column names and gave appropriate names to the column, column amt was converted to amount.

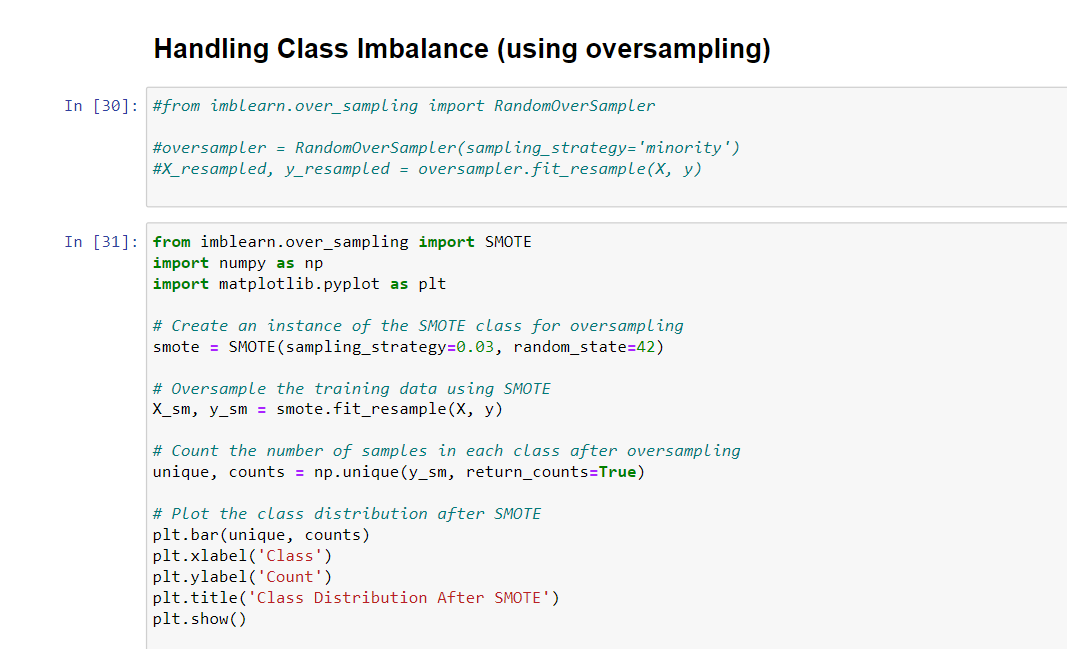


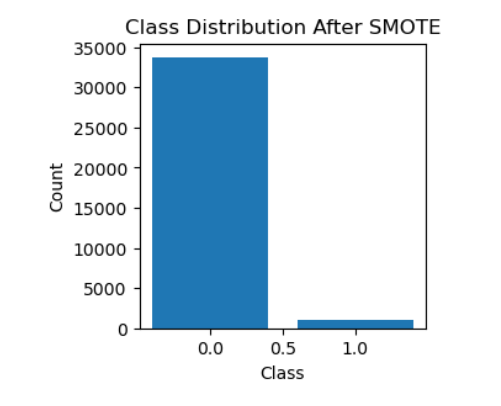
5.5 FEATURE SELECTION

Defined the feature columns and the class column.



5.6 HANDLING CLASS IMBALANCE USING OVER SAMPLING (SMOTE)





SMOTE, which stands for Synthetic Minority Over-sampling Technique, is a technique used in machine learning for handling imbalanced datasets, particularly in binary classification problems. When you have a dataset where one class (typically the minority class) is significantly underrepresented compared to the other class (majority class), it can lead to imbalanced learning, and some machine learning models may perform poorly on such data.

SMOTE is designed to address this issue by oversampling the minority class, effectively creating synthetic samples to balance the class distribution.

# IMPLEMENTATION OF THE MODELS

The implementation of the models is the prime motive of this project, and it was mandatory to select the best model for the classification. To choose the most appropriate model, several classification models as listed below, were built as the first step.

* + Random Forest Model
  + Logistic Regression Model
  + Naïve Bayes Model
  + Support Vector Machine Model
  + K-Nearest Neighbors (KNN) Model

The models were implemented using the cross-validation approach.

The basic idea behind cross-validation is to divide the dataset into multiple subsets (or "folds"), where a portion of the data is used for training the model, and the rest is used for testing its performance. Here's how it works:

Data Splitting: The dataset is divided into "k" equal-sized folds, where "k" is a user-defined number (e.g., 5 or 10). Each fold serves as a validation set, and the remaining folds are used for training. For example, in 5-fold cross-validation, the data is divided into five parts, and the process is repeated five times.

Model Training and Testing: In each iteration of the cross-validation process, the model is trained on a combination of "k-1" folds and tested on the remaining fold. This training and testing process is repeated "k" times, with each of the "k" folds serving as the test set exactly once.

Performance Metrics: For each fold, you calculate evaluation metrics (e.g., accuracy, precision, recall, F1-score) to assess the model's performance. After all iterations, you have "k" sets of evaluation metrics, one for each fold.

**ADVANTAGES OF CROSS-VALIDATION:**

Maximizes Data Utilization: Every data point is used for both training and validation in one of the "k" iterations, allowing you to make the most of your available data.

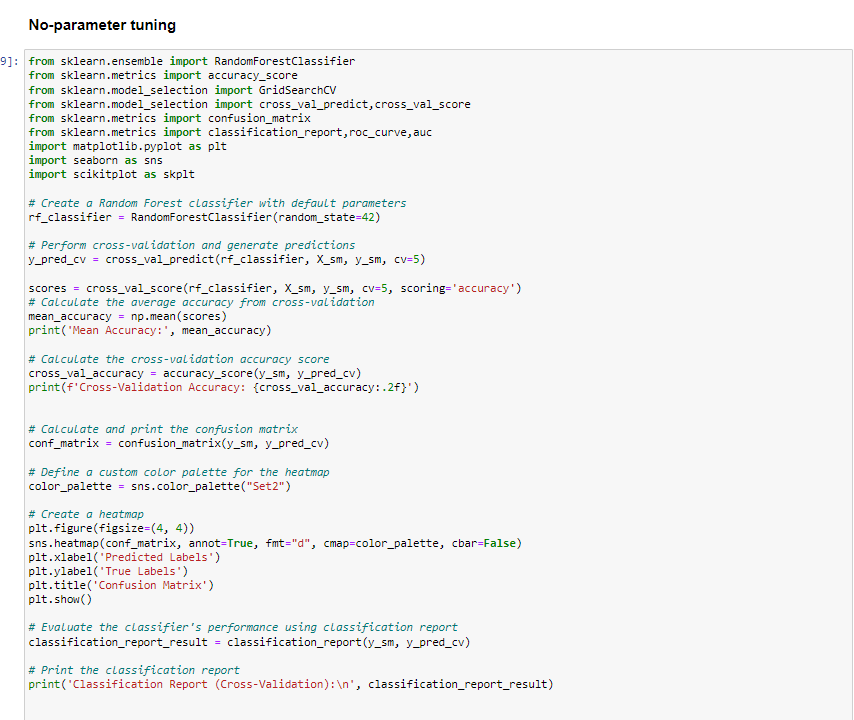
Robust Evaluation: Provides a more robust performance estimate as it is based on multiple rounds of training and testing.

Less Dependent on Data Split: The performance estimate is less dependent on a single random data split, reducing the risk of overfitting or underfitting.

6.1 RANDOM FOREST MODEL

Random forest is a supervised learning algorithm, and it can be considered as a good indicator of feature importance. It has two variations where one is used for classification problems and the other is used for regression problems. It is one of the most flexible and easy algorithms to use. It creates decision trees on the given data samples, gets the prediction from each tree and then selects the best solution by means of voting.

This algorithm combines multiple decision trees, resulting in a forest of trees. In the random forest classifier, the higher the number of trees in the forest, the higher accuracy can be obtained. The implementation of this model will be explored in the following.



Initially, the necessary imports such as RandomForestClassifier, accuracy\_score, GridSearchCV, confusion\_matrix and classification report were successfully called as shown:

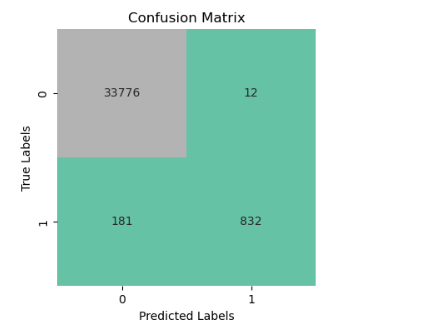
As a startup, the classifier is first instantiated without tuning any parameters, I created a Random Forest classifier (`rf\_classifier`) with default parameters.

This code uses cross\_val\_predict to perform 5-fold cross-validation on the dataset (as specified by cv=5). X\_sm is the feature matrix, and y\_sm is the target variable. During cross-validation, the model is trained and tested on different subsets of the data. The predicted labels (y\_pred\_cv) for each data point are collected.

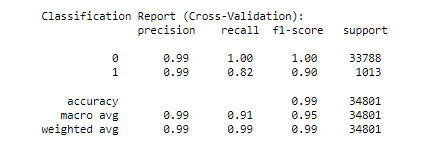
After the cross-validation is completed, this code calculates the accuracy for each fold using cross\_val\_score. It then computes the mean accuracy across all folds. This gives you an idea of the model's overall performance.



A confusion matrix is generated using the true labels (y\_sm) and the predicted labels (y\_pred\_cv). The confusion matrix summarizes the model's performance by showing the number of true positives, true negatives, false positives, and false negatives.



A classification report is generated using classification\_report. This report provides various performance metrics, such as precision, recall, F1-score, and support, for each class in the classification problem.



**IN ORDER TO IMPROVE THE ACCURACY VALUES, PRECISION, RECALL AND F1-SCORE VALUES HYPERPARAMETERS WERE DEFINED AND THE MODEL WAS TUNED TO HAVE A BETTER PERFORMANCE**



param\_grid, defines a grid of hyperparameters for Random Forest. The grid includes various values for hyperparameters like the number of estimators, maximum depth, minimum samples for splitting, minimum samples per leaf, maximum features to consider, and class weights.

Grid search is performed using the provided dataset (X\_sm for features and y\_sm for target labels). It searches through the hyperparameter grid to find the best combination of hyperparameters that maximize the specified scoring metric (accuracy).

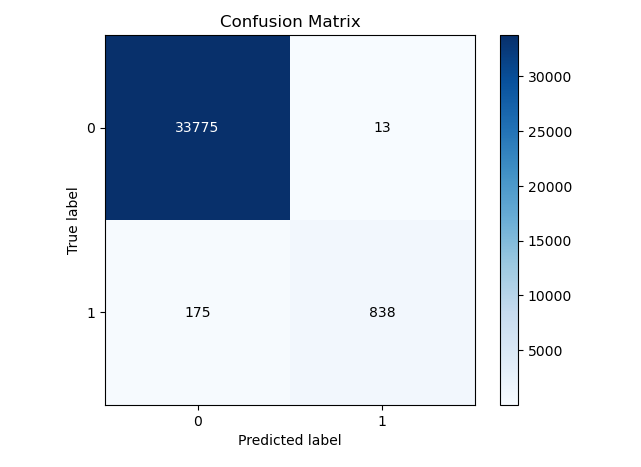
The best Random Forest classifier is instantiated with the best hyperparameters found during the grid search.

Cross-validation is performed using the best classifier. The model is trained and evaluated using 5-fold cross-validation, and the predicted labels (y\_pred\_cv) are collected.

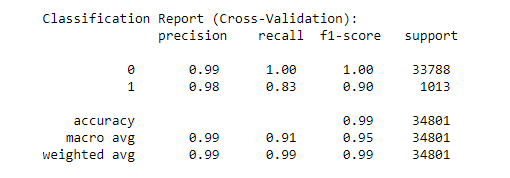
After the cross-validation is completed, this code calculates the accuracy for each fold using cross\_val\_score. It then computes the mean accuracy across all folds. This gives you an idea of the model's overall performance.



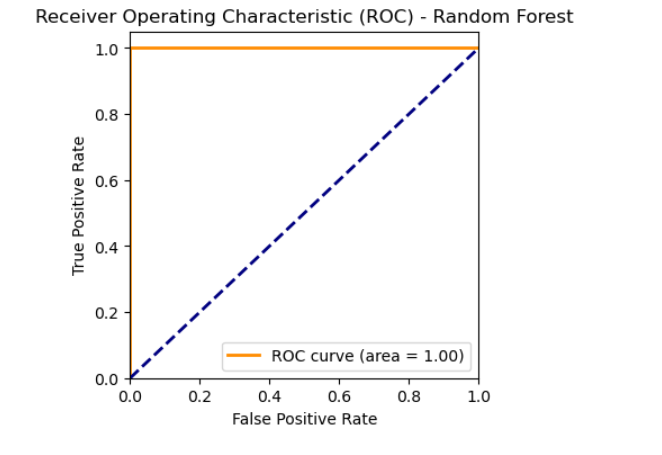
A confusion matrix is generated using the true labels (y\_sm) and the predicted labels (y\_pred\_cv). The confusion matrix summarizes the model's performance by showing the number of true positives, true negatives, false positives, and false negatives.



A classification report is generated using classification\_report. This report provides various performance metrics, such as precision, recall, F1-score, and support, for each class in the classification problem.



The ROC curve for random forest model was plotted:



6.2 LOGISTIC REGRESSION MODEL

Logistic Regression is a statistical model used for binary classification tasks, where the main motive is to predict one of two possible classes, typically represented as 0 (negative class) and 1 (positive class). In our scenario we trained the model using historical data to identify potentially fraudulent transactions, where 0 is not fraudulent and 1 means fraudulent transactions.

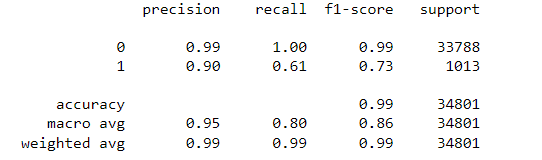
It explains the relationship between a dependent variable and one or more nominal, ordinal, interval or ratio-level independent variables. The main reasons for using this model were the ease of implementation, interpretation and it can also be considered as very efficient to train. Moreover, it is an easily interpretable classification technique that gives the probability of an event occurring, and not just the predicted classification. Moreover, since this dataset has only two values 0 and 1 in the target variable, this model can be recommended.



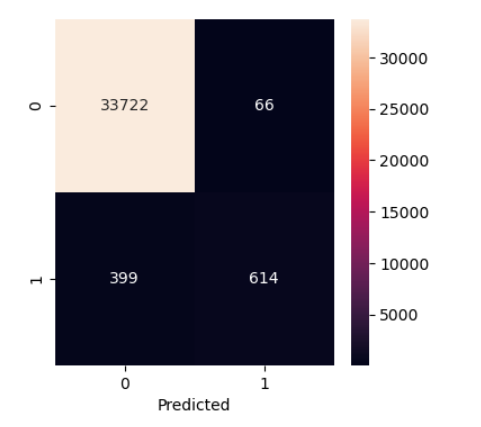
A Logistic Regression model is initialized without specifying any hyperparameters. Next, the Logistic Regression model is trained on the resampled training data (X\_sm for features and y\_sm for target labels). The cross\_val\_predict function generates cross-validated predictions. In this case, 5-fold cross-validation is used, meaning the dataset is split into 5 subsets, and the model is trained and evaluated 5 times. The code calculates accuracy for each fold of cross-validation and computes the mean accuracy across all folds. This provides an overall measure of the model's performance.



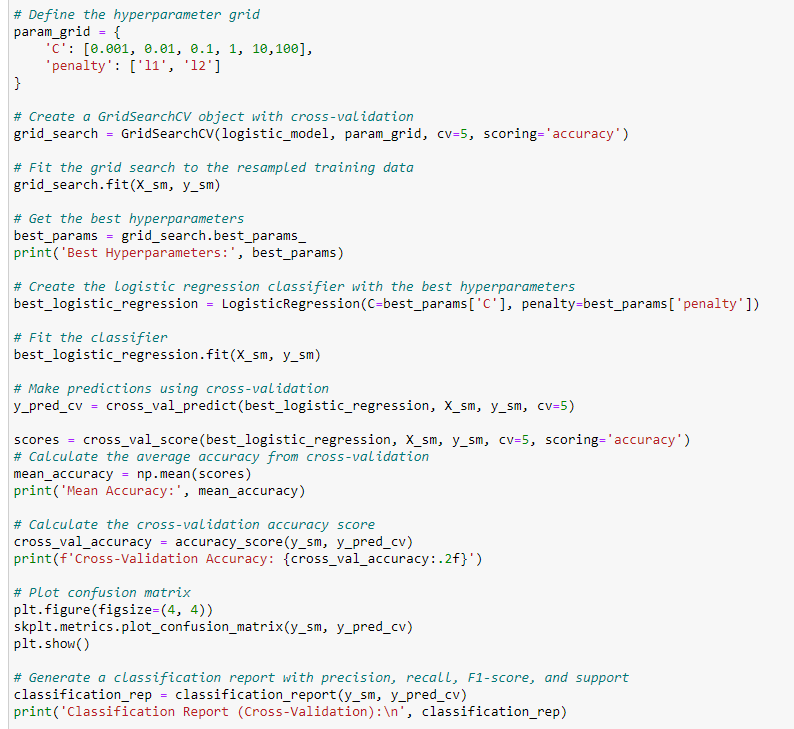
Next it generates a classification report for the cross-validated predictions. This report provides metrics such as precision, recall, F1-score, and support for each class, offering a detailed assessment of the model's classification performance.



A confusion matrix is generated to evaluate the model's performance, showing how many samples were correctly and incorrectly classified. The confusion matrix is displayed as a heatmap for better visualization.



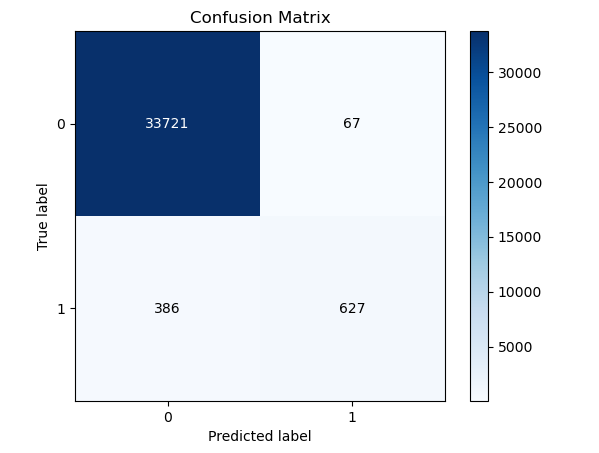
**IN ORDER TO IMPROVE THE ACCURACY VALUES, PRECISION, RECALL AND F1-SCORE VALUES HYPERPARAMETERS WERE DEFINED AND THE MODEL WAS TUNED TO HAVE A BETTER PERFORMANCE**



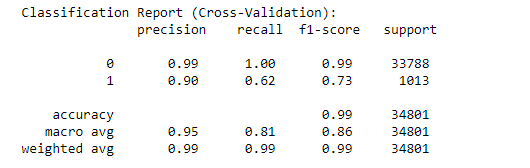
A grid of hyperparameters is defined. It includes values for the regularization parameter C (controls inverse regularization strength) and the penalty type penalty (L1 or L2 regularization). The grid search is executed with the resampled training data (X\_sm for features and y\_sm for target labels). The goal is to find the best combination of hyperparameters. After the grid search is complete, the best hyperparameters are obtained. A new Logistic Regression model is instantiated using the best hyperparameters. The C and penalty values are set to the best values determined by the grid search. Cross-validation is performed using the Logistic Regression model with the best hyperparameters. The cross\_val\_predict function generates cross-validated predictions, and 5-fold cross-validation is used. The code calculates accuracy for each fold of cross-validation and computes the mean accuracy across all folds. This provides an overall measure of the model's performance.



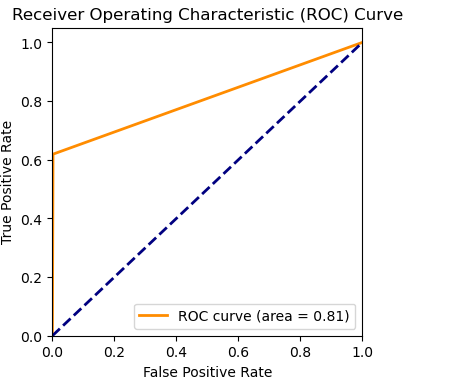
A confusion matrix is generated to evaluate the model's performance, showing how many samples were correctly and incorrectly classified. The confusion matrix is displayed as a heatmap for better visualization.



Next it generates a classification report for the cross-validated predictions. This report provides metrics such as precision, recall, F1-score, and support for each class, offering a detailed assessment of the model's classification performance.



The ROC curve for logistic regression model was plotted:



## **NAÏVE BAYES MODEL**

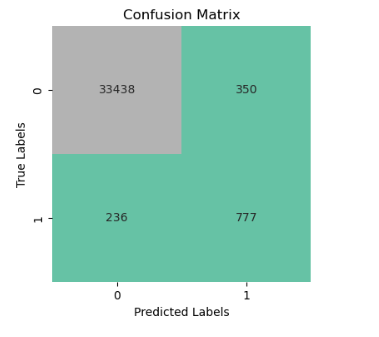
The Naïve Bayes model is a classification technique which is based on the Bayes theorem. Gaussian Naïve Bayes is a variant of Naïve Bayes which supports continuous values, and it also has an assumption that each class is normally distributed. Furthermore, it is also assumed that the features used are independent of each other. This model was chosen for the classification in this context as it is fast in processing and effective in predicting the class of the test dataset. Moreover, the model is effective in predicting outcomes for a few samples in training, when compared to other models. Therefore, it is easy to obtain the estimated probability for a prediction.



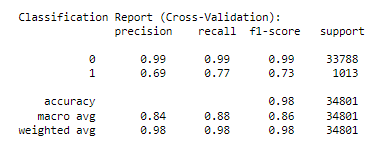
A Gaussian Naive Bayes classifier is created. This classifier assumes that the features are normally distributed. Cross-validation is performed using the Gaussian Naive Bayes classifier. The cross\_val\_predict function generates cross-validated predictions, and 5-fold cross-validation is used. The code calculates accuracy for each fold of cross-validation and computes the mean accuracy across all folds.



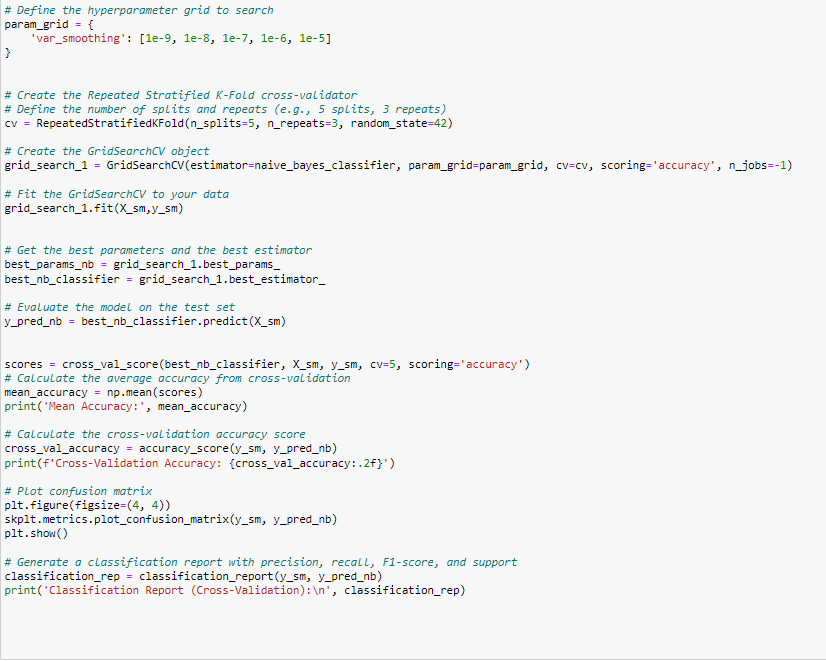
A confusion matrix is generated to evaluate the model's performance, showing how many samples were correctly and incorrectly classified. The confusion matrix is displayed as a heatmap for better visualization.



Next it generates a classification report for the cross-validated predictions. This report provides metrics such as precision, recall, F1-score, and support for each class, offering a detailed assessment of the model's classification performance.



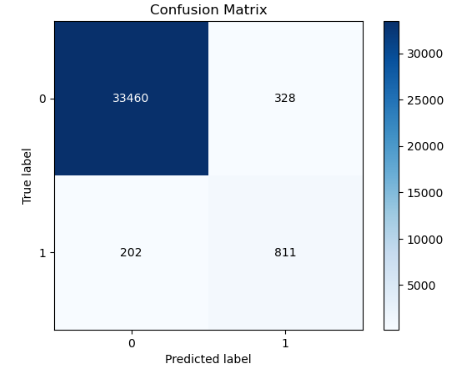
**IN ORDER TO IMPROVE THE ACCURACY VALUES, PRECISION, RECALL AND F1-SCORE VALUES HYPERPARAMETERS WERE DEFINED AND THE MODEL WAS TUNED TO HAVE A BETTER PERFORMANCE**



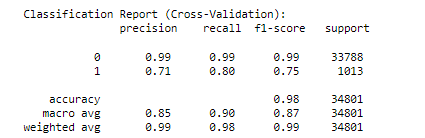
A grid of hyperparameters to search is defined. In this case, it includes different values of var\_smoothing, which is a smoothing parameter in Gaussian Naive Bayes. A Repeated Stratified K-Fold cross-validator is set up. It defines the number of splits (5) and repeats (3) for cross-validation. This is used to evaluate the model's performance with different data splits. A GridSearchCV object is created. It uses the param\_grid defined earlier, the cross-validator cv, and aims to optimize accuracy. The -1 in n\_jobs indicates that all available CPU cores will be used for parallel processing. The GridSearchCV is fitted to the data, which means it performs a search for the best hyperparameters for the Gaussian Naive Bayes model. After the grid search, the best hyperparameters and the best estimator (model) are retrieved. The code calculates accuracy for each fold of cross-validation and computes the mean accuracy across all folds.



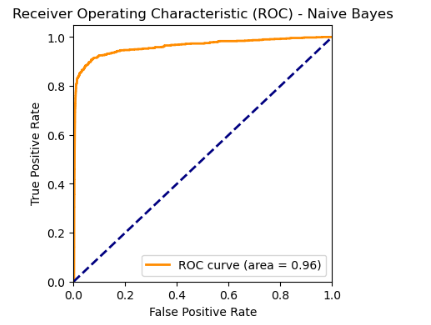
A confusion matrix is generated to evaluate the model's performance, showing how many samples were correctly and incorrectly classified. The confusion matrix is displayed as a heatmap for better visualization.



Next it generates a classification report for the cross-validated predictions. This report provides metrics such as precision, recall, F1-score, and support for each class, offering a detailed assessment of the model's classification performance.



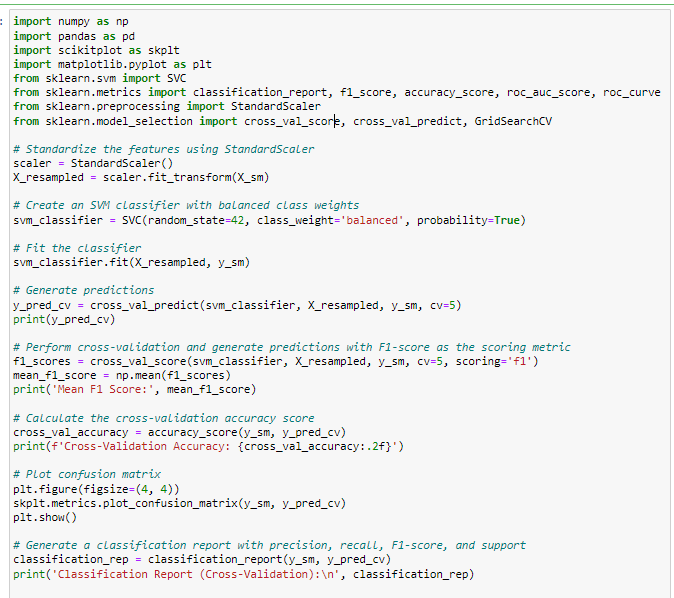
The ROC curve for logistic regression model was plotted:



## **SUPPORT VECTOR MACHINE MODEL**

The Support Vector Classifier (SVC), also known as the Support Vector Machine (SVM) classifier, is a powerful and versatile machine learning model widely used for classification tasks. It is employed to predict one of two possible classes, typically labeled as 0 (negative class) and 1 (positive class), making it well-suited for binary classification scenarios.

SVC is a model that goes beyond linear classification methods and can handle complex decision boundaries in high-dimensional spaces. It is based on the concept of finding a hyperplane that best separates the data into two classes while maximizing the margin between the classes. This hyperplane is referred to as the "support vector."

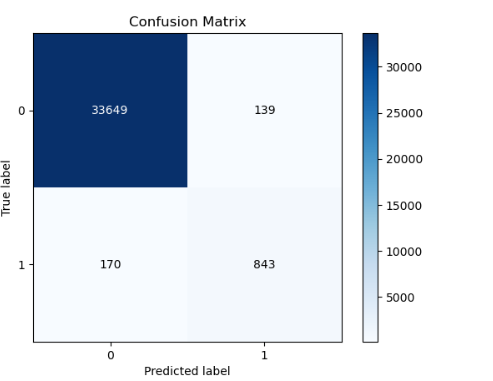


The features in the dataset (X\_sm) are standardized using StandardScaler. Standardization scales the features to have a mean of 0 and a standard deviation of 1, which can improve the performance of some machine learning algorithms. An SVM classifier is created with the following settings:

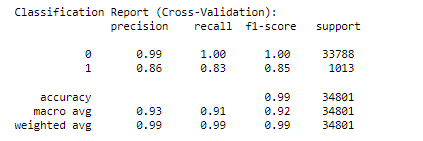
random\_state=42: A random seed for reproducibility.class\_weight='balanced': It assigns weights to classes inversely proportional to their frequencies. This helps in handling imbalanced datasets.probability=True: It enables probability estimates, which is useful for ROC curve analysis later. Cross-validation is performed using the SVM classifier with 5 folds, and predictions are generated. y\_pred\_cv contains the predicted labels for each data point. F1-scores are calculated using cross-validation. The F1-score is a measure of a model's accuracy, balancing precision and recall. The mean F1-score is calculated to provide an overall performance metric.



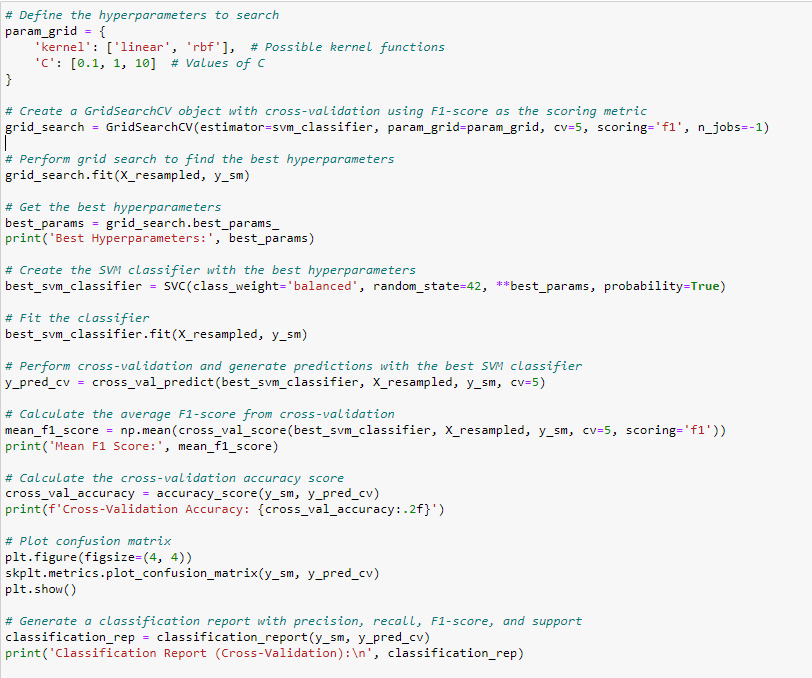
A confusion matrix is generated and visualized using scikit-plot. The confusion matrix helps evaluate the model's performance in terms of true positives, true negatives, false positives, and false negatives.



Next it generates a classification report for the cross-validated predictions. This report provides metrics such as precision, recall, F1-score, and support for each class, offering a detailed assessment of the model's classification performance.



**IN ORDER TO IMPROVE THE ACCURACY VALUES, PRECISION, RECALL AND F1-SCORE VALUES HYPERPARAMETERS WERE DEFINED AND THE MODEL WAS TUNED TO HAVE A BETTER PERFORMANCE**



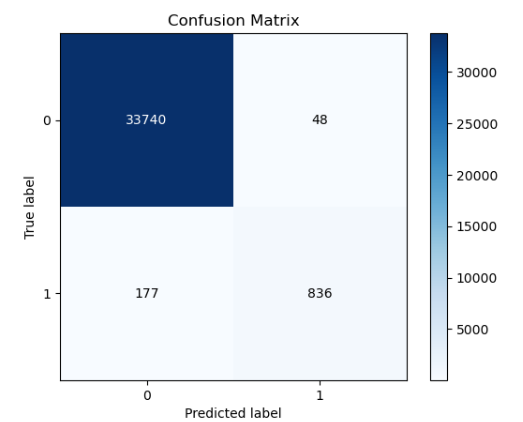
A grid of hyperparameters to search is defined using a Python dictionary. This grid includes two hyperparameters:

'kernel': The kernel function to be used in the Support Vector Machine (SVM). It can be either 'linear' or 'rbf'.

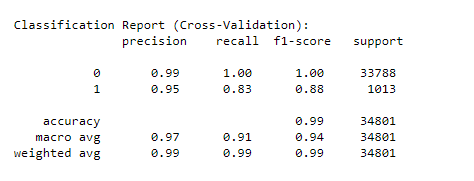
'C': The regularization parameter of the SVM. It takes values of 0.1, 1, and 10. A GridSearchCV object is created. This object will perform a grid search to find the best combination of hyperparameters. The best SVM classifier is fitted to the resampled dataset using the best hyperparameters. Cross-validation is performed with the best SVM classifier, and predictions are generated. The F1-score is used as the evaluation metric. The mean F1-score is calculated by taking the average of F1-scores from the cross-validation runs. The F1-score is a measure of accuracy that balances precision and recall.



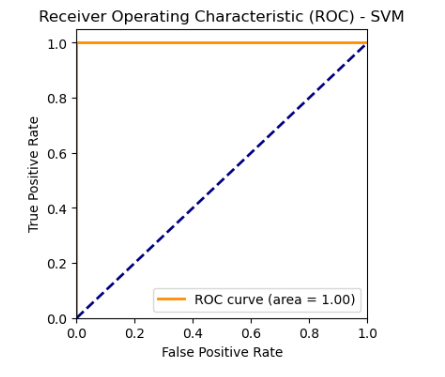
A confusion matrix is generated and visualized using scikit-plot. The confusion matrix helps evaluate the model's performance in terms of true positives, true negatives, false positives, and false negatives.



Next it generates a classification report for the cross-validated predictions. This report provides metrics such as precision, recall, F1-score, and support for each class, offering a detailed assessment of the model's classification performance.



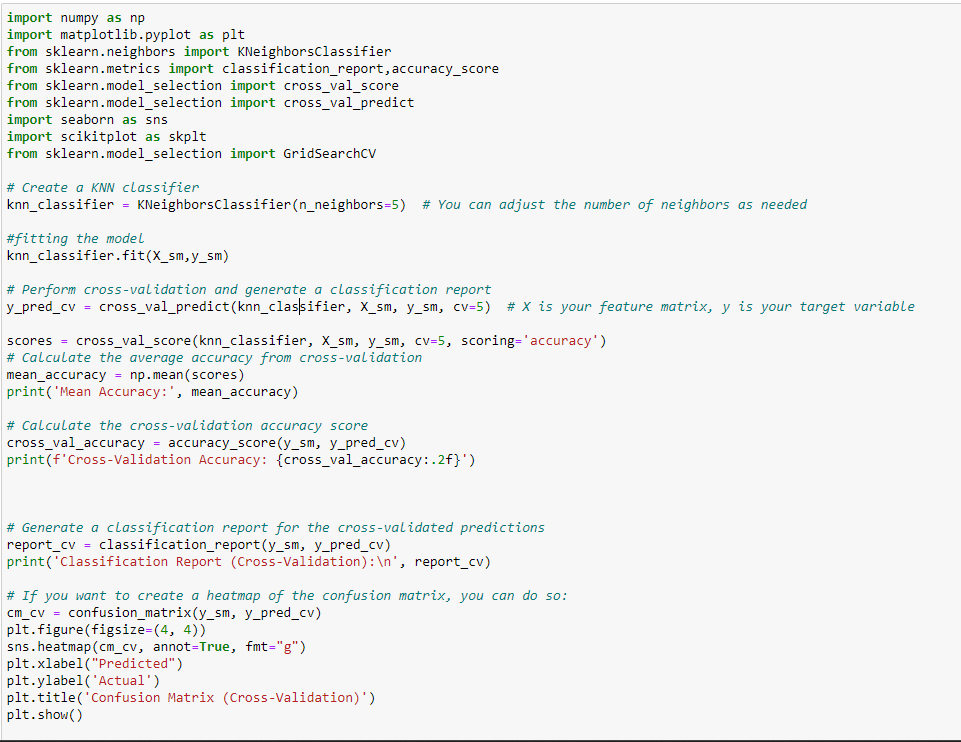
The ROC curve for support vector machine model was plotted:



## **K-NEAREST NEIGHBOR (KNN) MODEL**

K-Nearest Neighbors (KNN) is a simple, yet powerful machine learning algorithm used for both classification and regression tasks. It's a type of instance-based learning, also known as lazy learning, where the model doesn't learn an explicit representation during training. Instead, it memorizes the entire training data set and makes predictions based on the similarity between new data points and those in the training set.

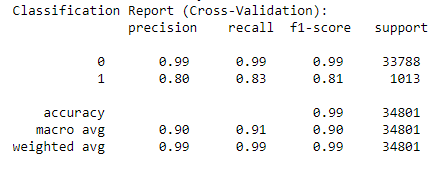
KNN operates on the principle that objects or data points that are similar are more likely to be related or share common characteristics. It calculates the "closeness" between data points using a distance metric, most commonly the Euclidean distance. The algorithm takes a parameter 'K,' which represents the number of nearest neighbors to consider when making predictions. For classification tasks, KNN assigns a class label to a data point based on the majority class among its nearest neighbors. In regression tasks, KNN computes an average or weighted average of the target values of the K nearest neighbors. KNN can use various distance metrics, including Euclidean, Manhattan, Minkowski, and others, to determine the similarity between data points. KNN is often used as a baseline model in machine learning tasks, and it can be a good choice when you don't want to assume a specific data distribution or when the relationships between data points are complex and not easily represented by parametric models.



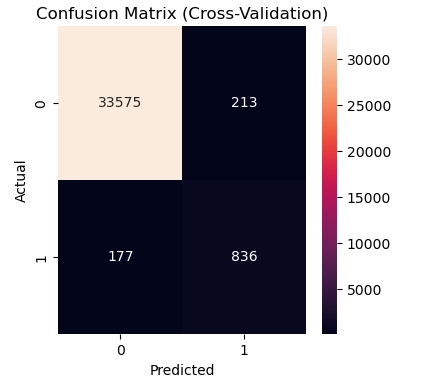
A KNN classifier is created with 5 neighbors. The number of neighbors can be adjusted as needed based on your problem. The number of neighbors determines how many neighboring data points are considered when making predictions. Cross-validation is performed with the KNN classifier to generate predictions. The cv=5 parameter specifies 5-fold cross-validation. Predictions are made on the resampled data (X\_sm and y\_sm). Cross-validation is performed again to calculate the accuracy scores for each fold. The mean accuracy is then computed by taking the average of these accuracy scores. It provides an estimate of the model's performance.



Next it generates a classification report for the cross-validated predictions. This report provides metrics such as precision, recall, F1-score, and support for each class, offering a detailed assessment of the model's classification performance.



A confusion matrix is generated and visualized using scikit-plot. The confusion matrix helps evaluate the model's performance in terms of true positives, true negatives, false positives, and false negatives.



**IN ORDER TO IMPROVE THE ACCURACY VALUES, PRECISION, RECALL AND F1-SCORE VALUES HYPERPARAMETERS WERE DEFINED AND THE MODEL WAS TUNED TO HAVE A BETTER PERFORMANCE.**



A dictionary called param\_grid is defined, which specifies the hyperparameters to be tuned and their potential values. The hyperparameters being optimized for a K-Nearest Neighbors (KNN) classifier are:

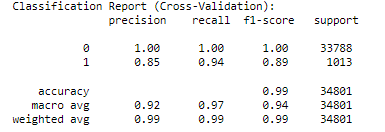
'n\_neighbors': The number of neighbors to consider, with values 3, 5, 7, and 9.

'weights': The weight function used for prediction, with options 'uniform' (all neighbors have equal weight) and 'distance' (weights by inverse of distance).

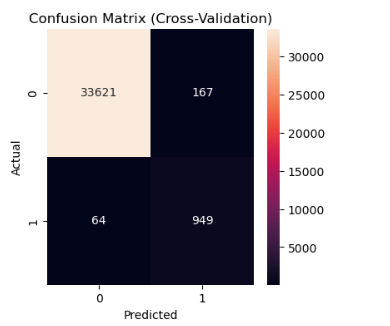
'p': The power parameter for the Minkowski distance metric, with values 1 (Manhattan distance) and 2 (Euclidean distance). The best hyperparameters found by the grid search are obtained and printed and creates a new KNN classifier instance (best\_knn\_classifier) using the best hyperparameters found by the grid search. Cross-validation is performed with the best KNN classifier. Predictions are generated for each fold and stored in y\_pred\_cv.cross-validation accuracy is calculated by evaluating the model's accuracy on each fold. The mean accuracy is calculated from these individual accuracies and printed:



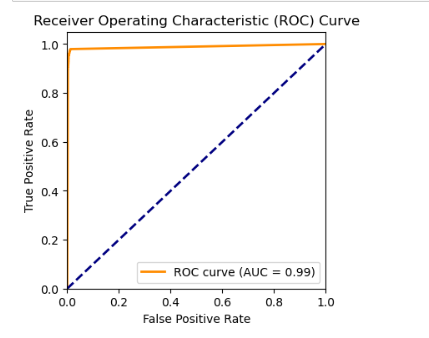
A classification report is generated for the cross-validated predictions. This report includes precision, recall, F1-score, and support for each class in the dataset. The report is printed to provide insights into the classifier's performance.



A confusion matrix is generated and visualized using scikit-plot. The confusion matrix helps evaluate the model's performance in terms of true positives, true negatives, false positives, and false negatives.



The ROC curve for KNN model was plotted:



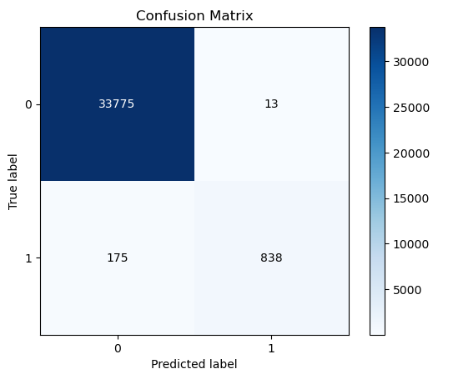
# **EVALUATION OF THE MODELS**

As discussed in the above, the final models which were created were evaluated to find the model which is most appropriate for the classification. In order to do this, the confusion matrices were observed and analyzed while other measures of accuracy, such as the F1 score was also compared.

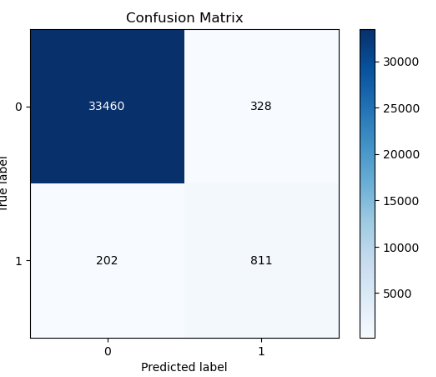
## **CONFUSION MATRICES**

The confusion matrix is a table which visualizes and summarizes the performance of a classification algorithm.

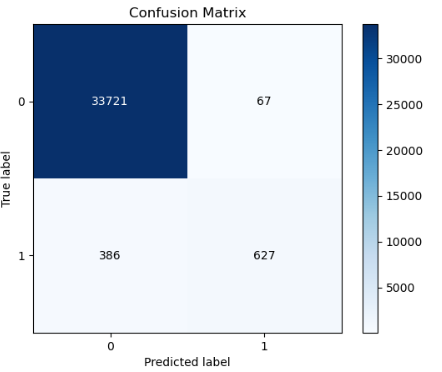
**Random Forest Classier:**



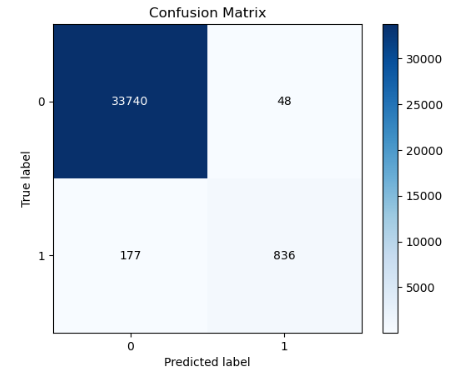
**Naive Bayes Model:**



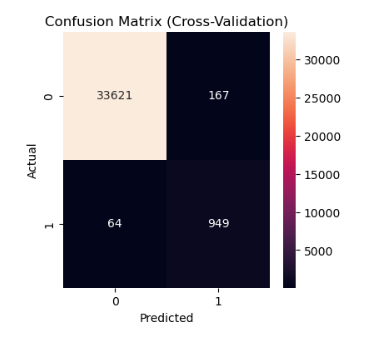
**Logistic Regression Model:**



**Support Vector Machine Model:**

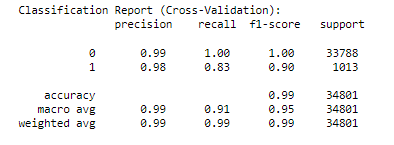


**KNN (K - Nearest Neighbors)**

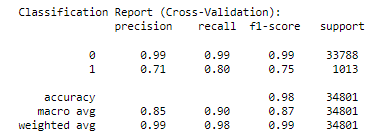


## **7.2 CLASSIFICATION REPORTS FOR EACH OF THE MODEL:**

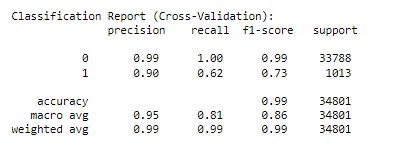
**Random Forest Classier:**



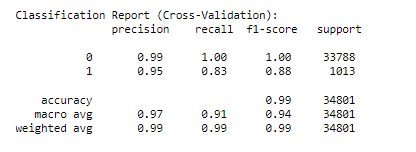
**Naive Bayes Model:**



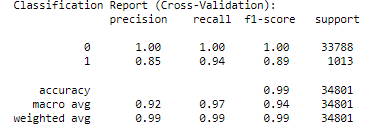
**Logistic Regression Model:**



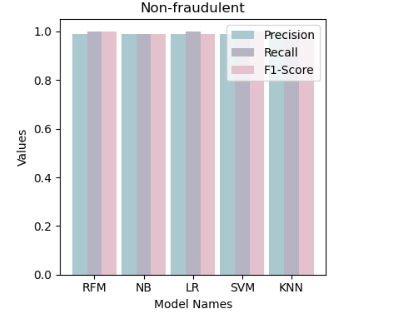
**Support Vector Machine Model:**

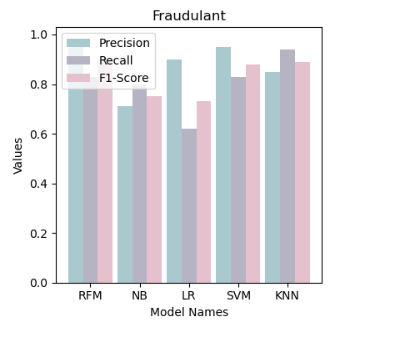


**KNN (K - Nearest Neighbors)**



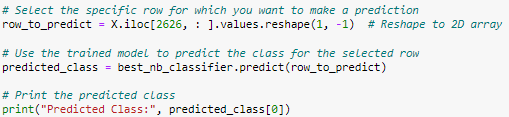
## **7.3 F1 SCORE, PRECISION AND RECALL**





7.4 PREDICTIONS OF THE CLASS COLUMN:

Prediction of the class column was made for each model for a given known record from the dataset, in the following way:



The values predicted by the models were then compared with the true value in the data set for comparison.

**The models that predicted correctly: Random Forest, SVM and KNN**

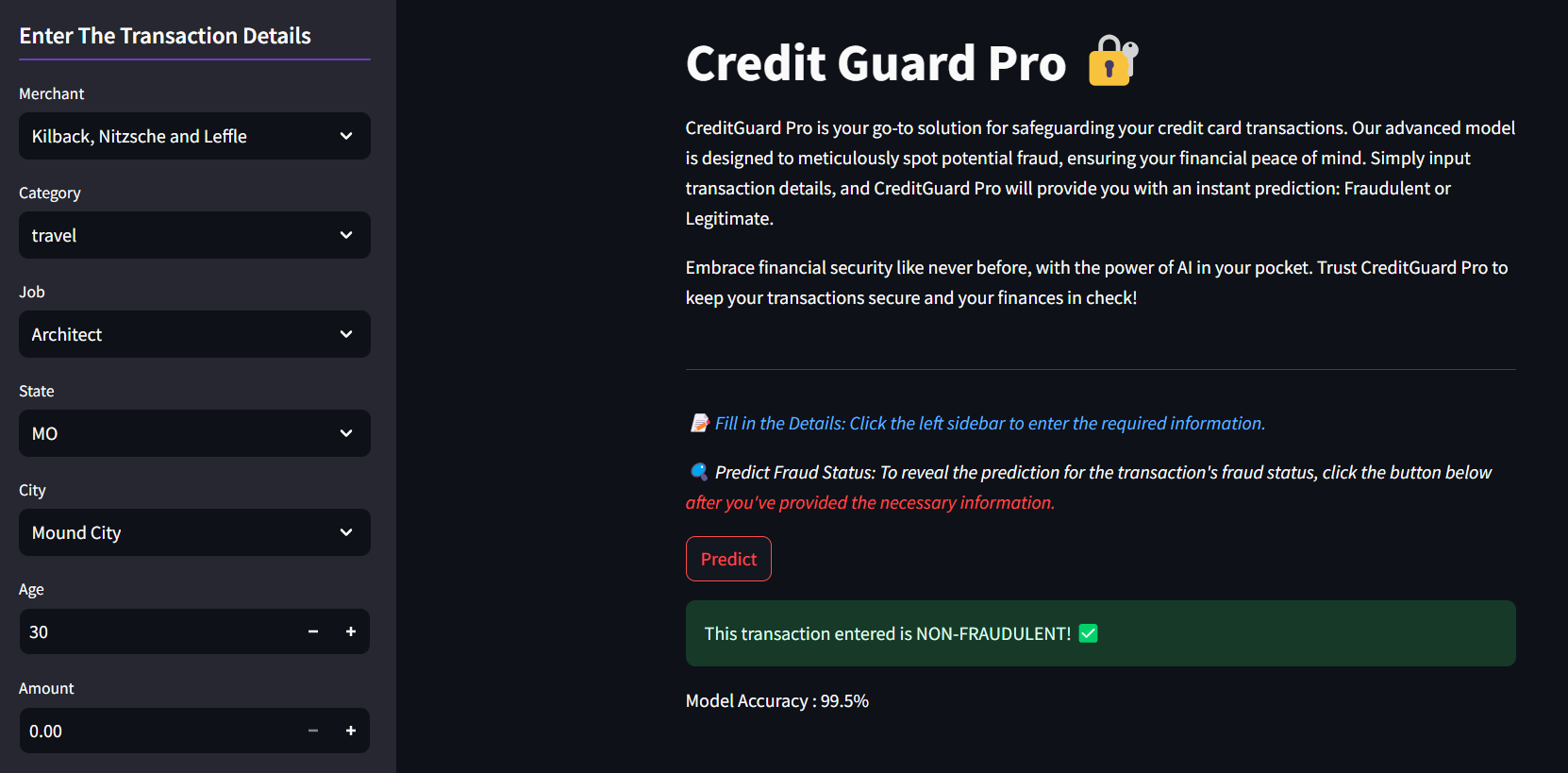
**After comparing the F1 scores, precision and recall scores we have chosen to go with random forest as random has a higher precision, recall and accuracy combination in comparison to the rest of the models.**

# **FRONTEND DEVELOPMENT**

## **TECHNOLOGY STACK**

The technology stack which was used for the development of this project is shown below. For the data preprocessing and the model building, which is basically the backend of the application, Python was used. And it was developed in Jupyter Notebook. For the front-end development, python and streamlit libraries were used. Streamlit was also used to deploy the application.

## **FRONTEND INTERFACE**



# **ASSESSMENT OF THE PROJECT**

The project team was able to deliver a successful working solution with all the objectives and targets defined in the statement of work covered properly.

* + The project was successfully deployed to the cloud and is accessible with the URL provided.
  + The team was able to implement four different classification models and choose the best on based on the accuracy of the models and give their recommendation.’
  + It was ensured to use machine learning and data mining techniques which were simplified through simple and easy-to-use user interfaces.
  + Project team was able to develop a responsive and simple, yet elegant user interface that can not only attract users but can maximize the quality of the user experience.
  + The team was able to ensure that all deliverables were provided in the given time frame, according to the Gantt chart which was produced in the Statement of Work.

## **PROBLEMS ASSOCIATED**

* + The dataset had a lack of strongly correlated features, therefore there was difficult to select the best features.
  + Produced model is only valid for a single location (USA Based).
  + The models took too long to fit and run.
  + Highly imbalanced dataset, had to do several sampling techniques.
  + Resolving issues and errors

## **9.2 IMPROVEMENTS**

* + Collect more data from all over the world.
  + Use computers with better processing power to develop the application.
  + Monitor your memory and CPU usage during model training to identify bottlenecks.
  + Consider using cloud-based services that provide scalable and powerful computing resources. Cloud providers like AWS, Google Cloud, and Azure offer GPU and TPU instances for machine learning tasks.
  + Use batch processing to split your dataset into smaller chunks. This can help prevent running out of memory during model training and evaluation.

# **WORK DISTRIBUTION**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME** | **REGISTRATIO**  **N NUMBER** | **ROLES** | **RESPONSIBILITIES** |
| IT21013850 | Rashida M.S.F | Data mining’  Data Analyzing Scope planning Documentation  Statistical Analyzing | Study the data set and data pre-processing.  Come up with a relevant model to build using python.  Design the UI for the frontend Documentation Parameter Tuning  Model Testing  Frontend and backend integration |
| IT21070594 | Zainab M.Z | Data mining  Data Analyzing Scope planning Documentation  Statistical Analyzing | Study the data set and data pre-processing.  Come up with a relevant model to build using python.  Frontend and backend integration  Parameter Tuning  Model Testing  Design the UI for the frontend Documentation |
| IT21006166 | Bishirhafi F.S.M.T | Data mining  Data Analyzing Scope planning Documentation  Statistical Analyzing | Study the data set and data pre-processing.  Come up with a relevant model to build using python.  Frontend development  Parameter Tuning  Model Testing  Design the UI for the frontend Documentation. |
| IT21004568 | Nuha M.N | Data mining  Data Analyzing Scope planning Documentation  Statistical Analyzing | Study the data set and data pre-processing.  Come up with a relevant model to build using python.  Frontend development  Parameter Tuning  Model Testing  Design the UI for the frontend Documentation. |

------------------------------------------- THE END -------------------------------------------