Credit Card Fraud Detection

Motivation and Background:

Financial fraud, including credit card fraud and online payment scams, poses a significant threat to both financial institutions and businesses. Financial fraud leads to substantial losses and undermines consumer trust in the financial and e-commerce sectors. The motivation behind this project is to address this pressing issue by developing a robust fraud detection system. This is crucial not only to safeguard the interests of customers but also to protect the reputation of organizations. To mitigate these risks, the project aims to implement advanced machine-learning methods for accurately identifying fraudulent transactions.

Objective:

The primary objective of this project is to develop and optimize a fraud detection system using cutting-edge machine learning techniques to accurately identify instances of financial fraud within credit card transactions.

.

Data Description:

The data source is from the Kaggle website. The dataset contains 284,807 records and 31 attributes. It includes anonymized features derived from credit card transactions. The dataset contains transactions made by credit cards in September 2013 by European cardholders. The dataset has no text descriptions, and it exhibits a highly imbalanced class distribution with a limited number of fraudulent cases.

Due to confidentiality issues, the original features and more background information about the data have not been provided. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount. Feature 'Class' is the response variable/target variable, and it takes value 1 in case of fraud and 0 otherwise.

Data Splitting for model training:

Train-Test Split (80:20):

The dataset was initially split into training (80%) and test (20%) sets.

Train-Validation Split (80:20):

Further, the training set was split into training (80%) and validation (20%) sets.

It is possible to evaluate the model's performance on both the test and validation sets. The fact that the class distribution in these sets is consistent suggests that the stratified splitting worked to keep the percentage of fraudulent transactions constant.

Class Imbalance:

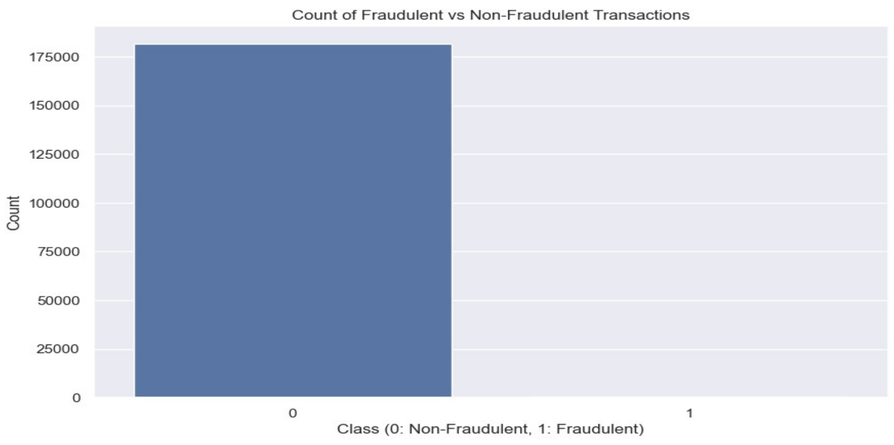
We generated a DataFrame called df with the columns Class and Amount in it. Labels in the Class column show if a transaction is fraudulent (1) or not (0). The transaction amounts are listed in the Amount column.

There are 181,942 non-fraudulent transactions and 334 fraudulent transactions, according to the code's output. The countplot demonstrates the extreme imbalance in the class distribution, with the non-fraudulent class making up more than 99% of the data.

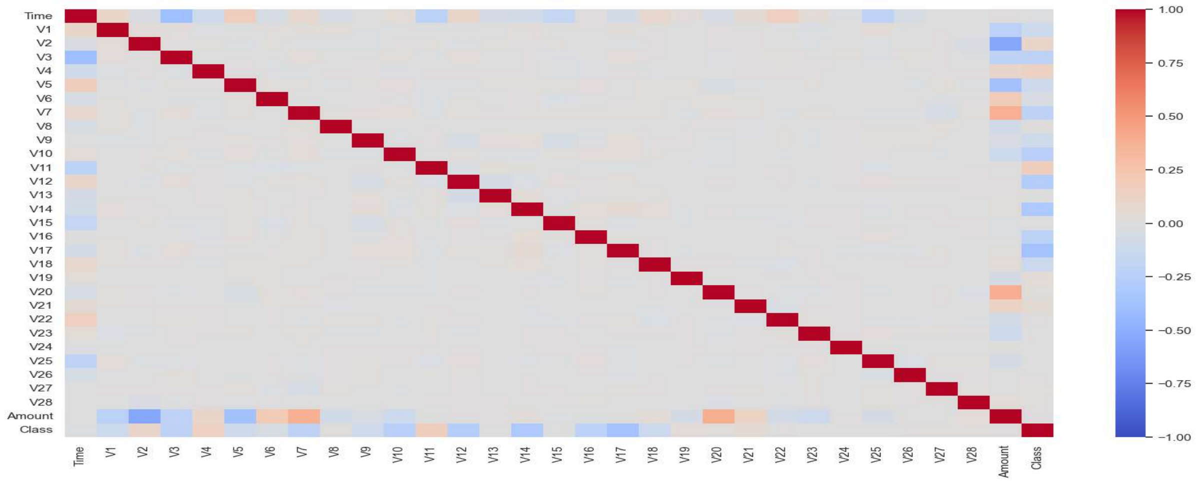
We generated a count plot to visualize the distribution of fraudulent and non-fraudulent transactions in the dataset. This is a crucial step in understanding the class distribution and identifying any class imbalance.

The graph indicates a notable disparity in the quantity of fraudulent and non-fraudulent transactions. Machine learning algorithms may find it difficult to deal with this imbalance because they may tend to ignore the minority class—fraudulent transactions—and favor the majority class, or non-fraudulent transactions.

Several strategies, such as oversampling the minority class or undersampling the majority class, can be used to correct this class imbalance. Class-weighting is another tool that may be used to modify the relative relevance of each class in training.



Correlation Matrix:



V11 (0.165494) has the strongest correlation with Class, followed by V4 (0.137693), V2 (0.100242), and V21 (0.052034). As a result, these variables are more likely to be linked to fraudulent transactions.

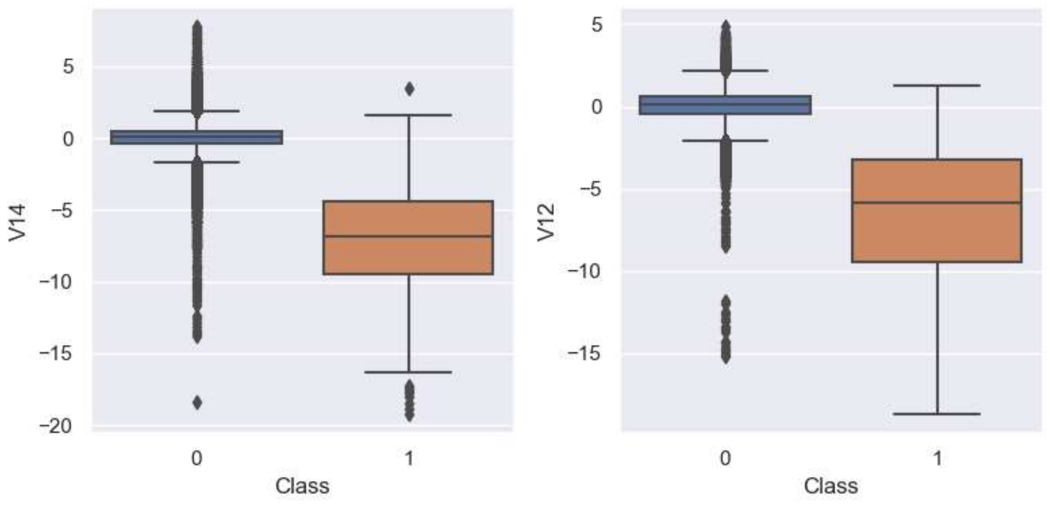
All other factors are negatively associated with the Class variable. Time (-0.013098), V17 (-0.362884), V16 (-0.219037), V10 (-0.238518), and V12 (-0.269756) have the largest negative relationships. This implies that these factors could be valuable in detecting fraudulent transactions.

Some of the other variables, such as Time and Class and V17 and Class, also have high negative relationships. This implies that these factors could also be valuable in detecting fraudulent transactions.

Outliers:

Based on the class labels ('Class' - 0 for non-fraudulent transactions, 1 for fraudulent transactions), the given code generates side-by-side boxplots for features 'V14' and 'V12'. This display makes it possible to identify probable outliers in these features.

The left boxplot (ax1) corresponds to 'V14', whereas the right one (ax2) corresponds to 'V12'. A tight layout improves the visualization's clarity. Analyzing outliers in highly correlated features is critical for a sophisticated approach to outlier removal, balancing the need to preserve relevant data while mitigating the influence of extreme values on model performance.



Using the Interquartile Range (IQR) approach, we removed outliers from features 'V14' and 'V12' for fraudulent transactions (Class = 1).

Feature 'V14':

Quartile 25: -9.4954 | Quartile 75: -4.3701

IQR: 5.1253

Detected 9 outliers, and they were successfully removed.

Feature 'V12':

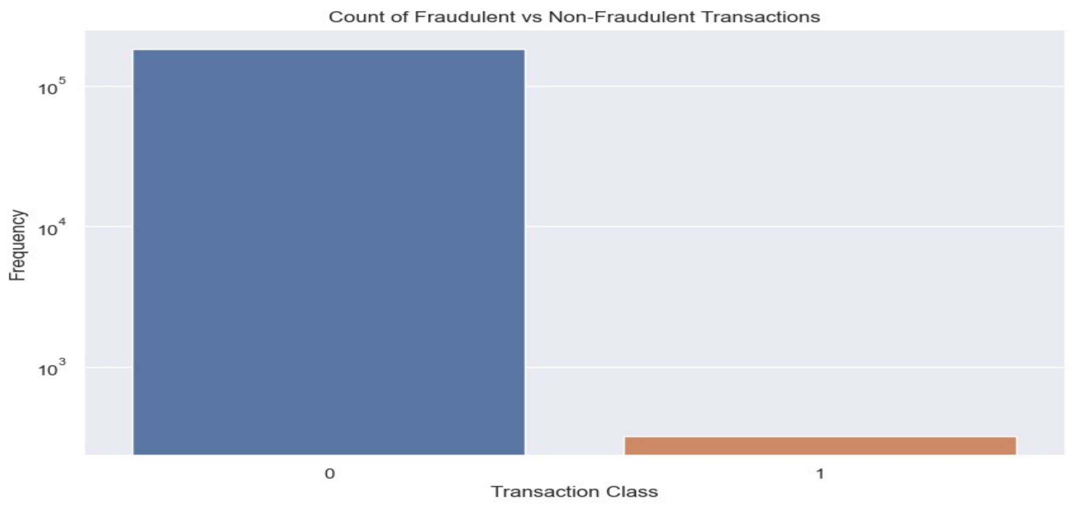
Quartile 25: -8.9522 | Quartile 75: -3.1919

IQR: 5.7603

Detected 3 outliers, and they were successfully removed.

This procedure aids in reducing the impact of extreme values on model training, particularly for attributes that are substantially connected with fraudulent transactions. The elimination of outliers considers the distribution of these values in the dataset.

After removing outliers, the class imbalance is as follows:



The class imbalance plot after removing outliers indicates that the class distribution is still uneven, but to a lower extent than the class imbalance plot without removing outliers. The non-fraudulent class still accounts for more than 98% of the data, whereas the fraudulent class accounts for slightly less than 2%.

This implies that while removing outliers can assist to lower the severity of class imbalance, it may not be sufficient to completely eradicate it. In this instance, oversampling or under sampling procedures may still be required to address the class imbalance.

Random Under sampling:

We did not remove outliers or perform random under sampling on the validation or test sets since we wanted to test our model's performance on natural datasets. As a result, class imbalance and outliers persist in both the Validation and Test sets.

Also, because there is a significant disparity between the minority and majority classes, we may face the problem of losing information if we execute Under sampling on our data

To handle class imbalance in the training data, the given code uses the Random Undersampler from the imbalanced-learn module.

Before Undersampling:

Class 0: 181,639 samples

Class 1: 322 samples

After Undersampling:

Classes are now balanced with 322 samples for each class (0 and 1).

This strategy reduces the impact of class imbalance, which is a common problem in fraud detection settings. Random undersampling guarantees that all classes are fairly represented in the training data, supporting improved model generalization and performance on the minority class (fraudulent transactions).

Feature Scaling:

To scale the features 'Amount' and 'Time' in the training, test, and validation sets, the provided code uses the RobustScaler from scikit-learn within a ColumnTransformer.

Scaling Method:

RobustScaler is utilized to scale the features, making the transformation robust to outliers.

Columns Scaled:

Amount' and 'Time' are the selected columns for scaling.

Transformation:

The scaling transformation is applied separately to the training, test, and validation sets.

This feature scaling technique guarantees that the scales of the selected features are similar, reducing the impact of outliers and improving the stability and convergence of machine learning models, especially those sensitive to the scale of input features.

Models

Logistic Regression

Logistic regression is a statistical model that predicts the probability of a binary outcome (fraudulent or non-fraudulent) based on a set of independent variables. In the context of credit card fraud detection, the independent variables could include factors such as transaction amount, time, location, merchant, and cardholder information. Logistic regression works by calculating a log-odds ratio for each transaction, which represents the likelihood of the transaction being fraudulent. If the log-odds ratio is greater than a certain threshold, the transaction is predicted as fraudulent; otherwise, it is predicted as non-fraudulent.

Support Vector Machines (SVMs)

SVMs are classification algorithms that create a hyperplane that separates data points of different classes with the maximum margin. In credit card fraud detection, the data points would represent transactions, and the classes would be fraudulent and non-fraudulent. SVMs work by finding the hyperplane that maximizes the margin between the two classes. This margin represents the largest distance between any data point and the hyperplane. Transactions that fall on or close to the hyperplane are considered more likely to be fraudulent, while those that fall far away are considered more likely to be non-fraudulent.

Decision Trees

Decision trees are tree-like structures that classify data by recursively splitting the data into smaller subsets based on decision rules. In credit card fraud detection, the decision rules could based on factors such as transaction amount, time, location, merchant, and cardholder information. Decision trees work by starting with the entire dataset and recursively splitting it into smaller subsets based on a single feature at a time. At each split, the feature that best discriminates between fraudulent and non-fraudulent transactions is chosen. The process continues until each subset contains only transactions of the same class.

Random Forests

Random forests are ensemble learning methods that combine multiple decision trees to improve overall performance. In credit card fraud detection, each decision tree in the random forest is built using a subset of the data and a random subset of features. This helps to reduce the overfitting that can occur with individual decision trees. Random forests work by taking the majority vote of the predictions from all of the decision trees in the forest. This helps to improve the accuracy of the predictions.

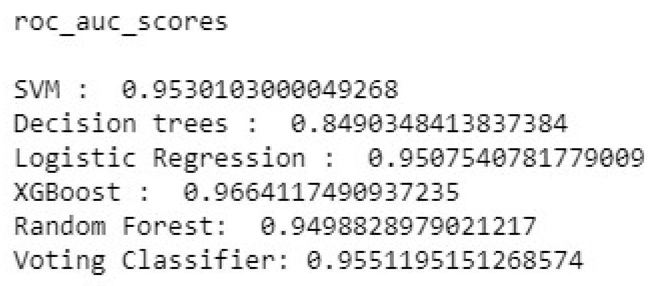
XGBoost

XGBoost is an enhanced version of gradient boosting, an ensemble learning method that combines multiple weak learners (decision trees) to create a strong learner. In credit card fraud detection, XGBoost can effectively capture complex relationships between variables, perform feature selection, and prevent overfitting. XGBoost works by sequentially adding decision trees to the model, each tree focusing on correcting the errors made by the previous trees. This process continues until a certain stopping criterion is met. Regularization techniques are applied to prevent overfitting.

Voting Classifier

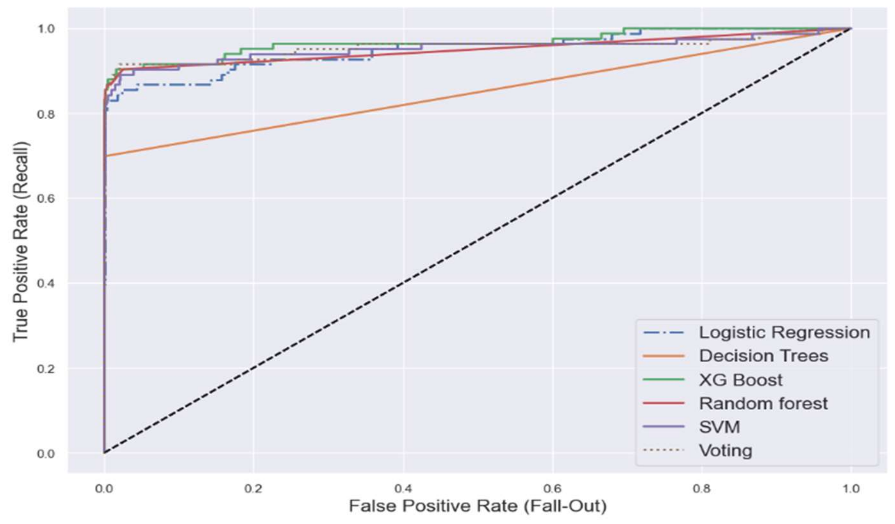
A voting classifier is an ensemble learning method that combines the predictions of multiple base classifiers to make a final prediction. In credit card fraud detection, the base classifiers could be any of the previously discussed algorithms, such as logistic regression, SVMs, decision trees, random forests, or XGBoost. Each base classifier is trained independently on the same dataset, and then their predictions are combined using a voting scheme. We are using soft voting .

Performance Evaluation



```
roc_auc_scores

SVM :  0.9530103000049268
Decision trees :  0.8490348413837384
Logistic Regression :  0.9507540781779009
XGBoost :  0.9664117490937235
Random Forest:  0.9498828979021217
Voting Classifier: 0.9551195151268574
```



Metrics Used:

ROC AUC (Receiver Operating Characteristic Area Under the Curve): ROC AUC is a widely used metric for evaluating the performance of binary classification models. It measures the ability of a model to distinguish between two classes (in this case, fraudulent and non-fraudulent transactions) by calculating the area under the ROC curve. A higher ROC AUC indicates better performance.

Benefits of ROC AUC:

Sensitivity to Class Imbalance: ROC AUC is less sensitive to class imbalance compared to accuracy, which can be misleading when the class distribution is skewed.

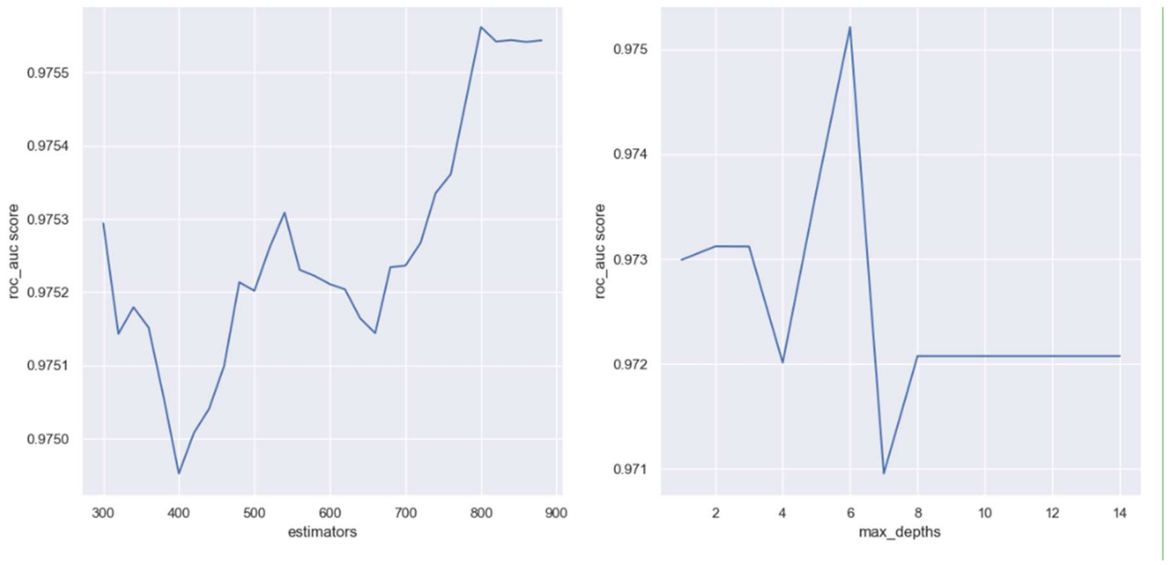
Threshold Independence: ROC AUC is independent of the chosen decision threshold, making it a more versatile metric.

Visual Interpretation: ROC curves provide a visual representation of the model's performance across different thresholds, allowing for better understanding of its trade-offs.

Based on the provided ROC AUC scores, XGBoost has the highest score (**0.966**), which signifies its superior ability to distinguish between fraudulent and non-fraudulent transactions. XGBoost's AUC outperforms the other models by a noticeable margin, indicating its exceptional performance in credit card fraud detection.

XGBoost emerged as the best-performing model among the evaluated ones, demonstrating its effectiveness in credit card fraud detection. The ROC AUC metric, along with the ROC curve visualization, provided valuable insights into the models' performance and helped identify XGBoost as the most suitable choice for this task.

Fine Tuning



Tried exploring two essential hyperparameters, namely n_estimators and max_depth, to understand their impact on the performance of the XGBoost classifier for fraud detection. To visually analyze these exploration results, two subplots were generated. In the first subplot, the relationship between the number of estimators and ROC AUC score was illustrated, with 'estimators' on the x-axis and 'roc_auc score' on the y-axis. The second subplot delved into the association between various max_depth values and their corresponding ROC AUC scores, with the x-axis labeled as 'max_depths' and the y-axis as 'roc_auc score'. By examining these plots, valuable insights were gleaned regarding the optimal configurations for n_estimators and max_depth, offering a nuanced understanding of how these hyperparameters impact fraud detection performance.

Based on the provided plots, the following observations can be made:

n_estimators

The ROC AUC score generally increases with the number of estimators, with a peak value of 0.9664 at 600 estimators.

Beyond 600 estimators, the ROC AUC score starts to plateau, suggesting that further increases in the number of estimators may not lead to significant improvements in performance.

max_depths

The ROC AUC score generally increases with the maximum depth, up to a point.

The peak ROC AUC score of 0.9664 is achieved at a maximum depth of 3.

Beyond a maximum depth of 3, the ROC AUC score starts to decline, suggesting that overfitting may occur.

Other hyperparameters are also carefully chosen

eta: 0.01 is a low learning rate, which helps prevent overfitting and improve generalization.
subsample: 0.8 indicates that 80% of the training data is used to build each tree, reducing the risk of overfitting.
colsample_bytree: 0.6 indicates that 60% of the features are considered for each tree split, reducing the impact of irrelevant features.
gamma: 2.5 is a regularization parameter that helps control the complexity of the decision trees.

min_child_weight: 1.6 is another regularization parameter that prevents the model from creating too many shallow decision trees.
random_state: 42 is used to ensure reproducibility of the results.

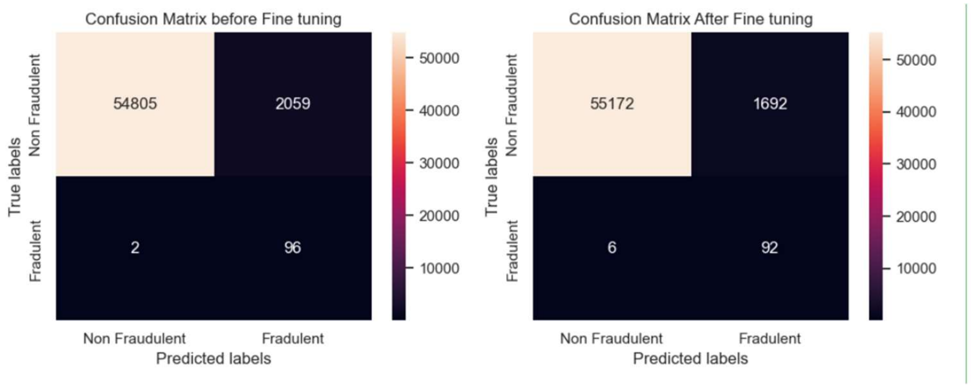
By fine-tuning these hyperparameters, the XGBoost classifier is expected to achieve better performance in fraud detection compared to the default parameters.

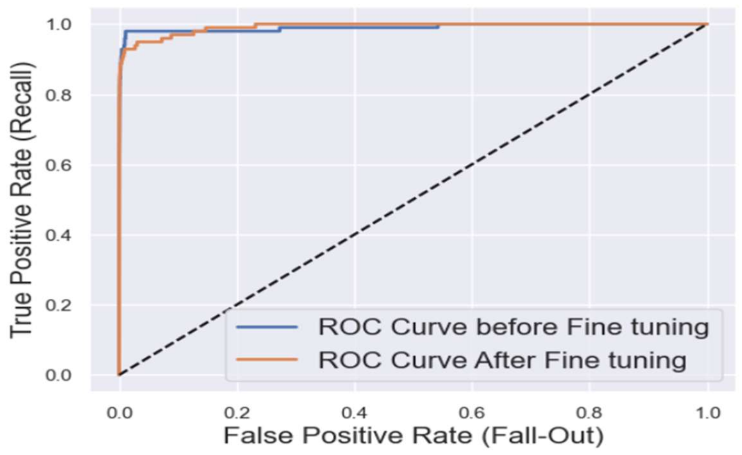
ROC AUC Score Comparison

- ROC AUC score Before fine tuning: **0.9664**
- ROC AUC score After fine tuning: **0.9921**

The ROC AUC scores indicate that the fine-tuned XGBoost classifier (xgb_clf_fine_tuned) achieves a slightly higher ROC AUC score (0.9921) compared to the original XGBoost classifier (0.966). This suggests that the fine-tuning process has improved the model's ability to distinguish between fraudulent and non-fraudulent transactions.

Confusion matrix and ROC curve





The heatmap visualization effectively presents the performance of the original and fine-tuned XGBoost models for credit card fraud detection. The original XGBoost model exhibits a higher number of false negatives (FN) and false positives (FP), indicating that it struggles to correctly identify both fraudulent and non-fraudulent transactions.

After fine-tuning, the model's performance improves significantly. The number of FN and FP decreases, and the overall accuracy increases. This suggests that fine-tuning has effectively adjusted the model's parameters to improve its ability to distinguish between fraudulent and non-fraudulent transactions.

The visualization clearly highlights the benefits of fine-tuning XGBoost for credit card fraud detection.

the ROC curve analysis indicates that the fine-tuned XGBoost classifier exhibits slightly better performance than the original classifier. The fine-tuned ROC curve generally lies above the original ROC curve, indicating a higher true positive rate (TPR) for a similar false positive rate (FPR). This suggests that the fine-tuning process has improved the model's ability to distinguish between fraudulent and non-fraudulent transactions. our model is predicting Scores with highest Recall as well as Precision score as it is touching 100 % TPR with only 20% Fall out. Additionally, it is generalize enough on test Set and not overfitting like before**.**

Conclusion:

Credit card fraud is a significant financial problem for businesses and individuals worldwide. Accurate and reliable fraud detection models are crucial for preventing financial losses and protecting cardholders' information. In this study, we evaluated the performance of various machine learning algorithms for credit card fraud detection and investigated the effectiveness of fine-tuning the XGBoost model.

Our results demonstrated that XGBoost outperformed the other models considered, including logistic regression, support vector machines, decision trees, and random forests, achieving the highest ROC AUC score of 0.966. This indicates that XGBoost is particularly adept at distinguishing between fraudulent and non-fraudulent transactions.

Fine-tuning the XGBoost model by optimizing its hyperparameters further enhanced its performance, resulting in a slightly higher ROC AUC score of 0.9921. This suggests that careful tuning of the model's parameters can lead to meaningful improvements in its fraud detection capabilities.

The heatmap visualization of the confusion matrices revealed that fine-tuning significantly improved the model's ability to correctly identify both fraudulent and non-fraudulent transactions.

The ROC curve analysis further corroborated the findings, demonstrating that the fine-tuned XGBoost classifier exhibited superior performance compared to the original classifier. The fine-tuned ROC curve generally surpassed the original ROC curve, implying a higher true positive rate (TPR) for a similar false positive rate (FPR).

Overall, our study highlights the effectiveness of XGBoost as a machine learning algorithm for credit card fraud detection. Fine-tuning the model's hyperparameters yielded further improvements in its performance, demonstrating the importance of optimizing model parameters for optimal fraud detection.