ML Midterm: Credit Card Fraud Detection

1. Introduction/Background

Literature Review

According to the FTC, credit card fraud was the most reported type of identity theft with over 400,000 reports [1]. That equates to more than $10 billion lost in 2023 alone in fraud-related incidents [1]. With digital transactions rising, card fraud losses have topped at over a 10% increase since 2021, the largest increase since 2018 [2]. Machine learning algorithms offer a promising solution, and with the vast amount of transaction data and user behavior available, these approaches can improve accuracy, provide faster processing, and adapt to new fraud patterns [3].

Dataset Description

This dataset [4] is a collection of credit card transactions made by European cardholders over two days in September 2013. There are 284,807 transactions, of which 492 (0.172%) are fraudulent. Contains only numerical data after PCA transformation.

2. Problem Definition

Problem Identification

Detecting and preventing fraudulent activities in real-time to minimize financial losses and protect individuals, businesses, and organizations.

Motivation for a Solution

Rule based methods are lacking due to high number of factors that influence each other. A machine learning solution will need to be implemented to solve this and reduce financial losses.

3. Methods

Data Preprocessing Methods

SMOTE - Synthetic Minority Over-sampling Technique (Balancing Classes)

Our dataset was highly imbalanced, with 99.83% of transactions being Non-Fraud. To address this, we plan to use SMOTE to generate synthetic samples for the minority class (Fraud). We'll also compare results with ADASYN, a similar technique, using quantitative validation metrics. Below is the code which bases off the Logistic Regression we also implemented and we see an improvement in scores. Below is the code.

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score, f1_score, precision_score, recall_score, confusion_matrix, roc_curve
from imblearn.over_sampling import SMOTE
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset
df = pd.read_csv('creditcard.csv')

# define features and target
X = df.drop(columns=['Class', 'Time'])
y = df['Class']
```

```
# split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)

# Apply SMOTE to balance the training set
smote = SMOTE(random_state=42)
X_train_sm, y_train_sm = smote.fit_resample(X_train, y_train)

# hyperparameter tuning for Logistic Regression
param_grid = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'penalty': ['l1', 'l2'],
    'solver': ['liblinear']
}

# GridSearchCV with Logistic Regression
grid_search = GridSearchCV(
    LogisticRegression(max_iter=2000, random_state=42, class_weight='balanced'),
    param_grid,
    cv=5,
    scoring='roc_auc'
)
grid_search.fit(X_train_sm, y_train_sm)

# Best model from grid search
best_model = grid_search.best_estimator_

# Predict on the test set with the best model
y_pred = best_model.predict(X_test)
y_proba = best_model.predict_proba(X_test)[:, 1]

# Evaluation metrics
roc_auc = roc_auc_score(y_test, y_proba)
f1 = f1_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()

# ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_proba)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'ROC Curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()

# Precision-Recall Curve
precision_vals, recall_vals, _ = precision_recall_curve(y_test, y_proba)
plt.figure(figsize=(8, 6))
plt.plot(recall_vals, precision_vals, label=f'F1 Score = {f1:.2f}')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend()
plt.show()
```

```
# Evaluation metrics
print(f"ROC-AUC Score: {roc_auc:.2f}")
print(f"F1 Score: {f1:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
```

Power Transformer (Treating skewness)

We identified skewed features in our dataset and will apply the Yeo-Johnson power transformer to normalize the distributions using sklearn.

```
from sklearn.preprocessing import PowerTransformer
import pandas as pd

# Specify the columns to apply the transformer (excluding Time, Amount, and Class for example)
columns_to_transform = [col for col in data.columns if col not in ['Time', 'Amount', 'Class']]

# Initialize and fit the PowerTransformer
pt = PowerTransformer(method='yeo-johnson', standardize=True)
data[columns_to_transform] = pt.fit_transform(data[columns_to_transform])

print("Power Transformation applied to skewed features.")
```

Winsorization Method (Outlier Detection/Extraction)

Ou ere minimal in our dataset, but we will use numpy to apply Winsorization based on the Interquartile Range (IQR) to remove extreme values beyond the 1st and 99th percentiles.

```
import numpy as np

# Function to apply Winsorization
def winsorize_iqr(df, columns, lower_percentile=0.01, upper_percentile=0.99):
    for col in columns:
        Q1 = df[col].quantile(lower_percentile)
        Q3 = df[col].quantile(upper_percentile)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        # Cap the values at the bounds
        df[col] = np.where(df[col] < lower_bound, lower_bound, df[col])
        df[col] = np.where(df[col] > upper_bound, upper_bound, df[col])
    return df

# Apply Winsorization to the desired columns
columns_to_winsorize = [col for col in data.columns if col not in ['Time', 'Amount', 'Class']]
data = winsorize_iqr(data, columns_to_winsorize)

print("Winsorization applied to outlier-prone features.")
```

ML Algorithms/Models

Logistic Regression (Supervised)

We used logistic regression to classify whether a transaction is fraudulent based on variables like transaction amount, location, and time. In the dataset, V1 to V28 are the features. These are principal components derived from PCA. These features represent complex patterns in the original transaction data, transformed to maintain confidentiality. Logistic regression is good for detecting credit card fraud because it provides a clear probability-based classification for binary outcomes (fraud vs. non-fraud) and does good with large, structured datasets like this one.

Library: LogisticRegression from scikit-learn.

Below is our Logistic Regression model. It is implemented in Python using scikit-learn. It includes hyperparameter tuning, evaluation metrics, and performance visualization:

Download Logistic Regression Notebook

Download Logistic Regression Notebook

Click the link above to download the Logistic Regression implementation notebook.

Decision Tree (Supervised)

We plan to use features like transaction amount and location at each node to make decisions. For instance, transactions above a certain amount may be flagged as more likely to be fraudulent.

Library: DecisionTreeClassifier from scikit-learn.

Download Decision Tree Notebook

Download Decision Tree Notebook

Click the link above to download the Decision Tree implementation notebook.

XGBoost (Supervised)

Fraud detection is imbalanced, with way more legitimate transactions than fraudulent ones. XGBoost will give more weight to fraudulent transactions, making sure the model focuses on the harder-to-detect fraud cases.

Library: XGBClassifier class from xgboost.

Download XGBoost Notebook

Download XGBoost Notebook

Click the link above to download the XGBoost implementation notebook.

4. Potential Results and Discussion

Quantiative Metrics

ROC-AUC Score

Good for distinguishing between positive and negative classes. Also, our dataset is imbalanced. It ranges from 0 to 1. Higher ROC-AUC score is better.

F1 Score

Based on the above precision and recall values, we calculate F1 Score (from 0 to 1) and gives us a harmonic average of both values. Higher F1 score is better.

Precision

Precision is our measure of accuracy. Higher precision score is better.

Recall

This is our measure of completeness. Higher recall score is better.

Project Goals

We wish to evaluate our three models and choose the one with the highest quantitative metrics and oversampling techniques. Using non-computationally intensive models helps substainably reduce carbon imprint when training and our models only store parameters so there is no PII at risk in regard to data privacy.

Expected Results

We expect a score of above 90% for all four quantitative metrics in being able to detect fraud vs non-fraud.

Logistic Regression Results

Below are the outputs from our Logistic Regression model. This includes the confusion matrix, ROC curve, precision-recall curve, and quantiative metrics.

Quantiative Metrics

ROC-AUC Score: 0.96

F1 Score: 0.71

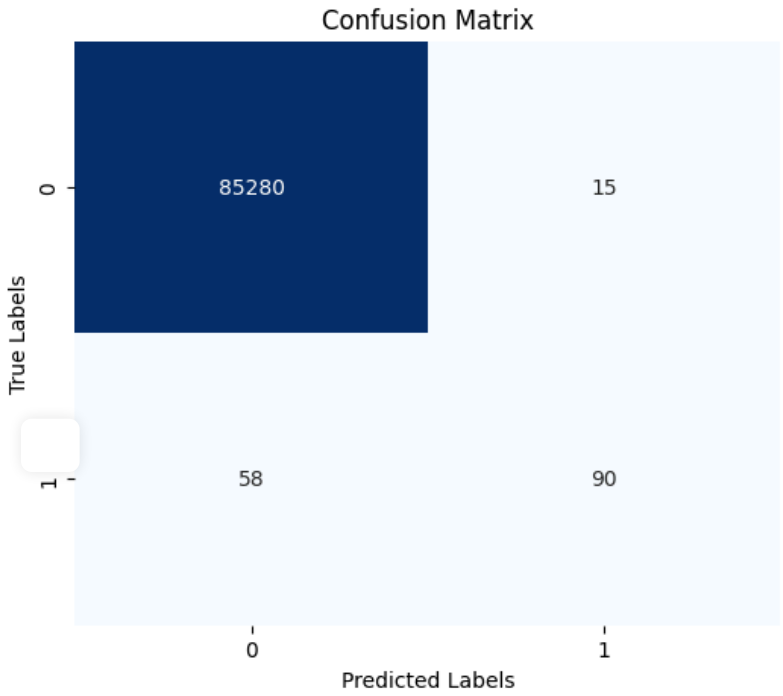
Precision: 0.86

Recall: 0.61

The ROC-AUC score of 0.96 means that our model is good at distinguishing between fraudulent and non-fraudulent transactions. The precision score of 0.86 means that most of the flagged fraud cases are actually fraud, reducing false positives. However, the recall score of 0.61 means that there are still some fraudulent cases that the model misses.

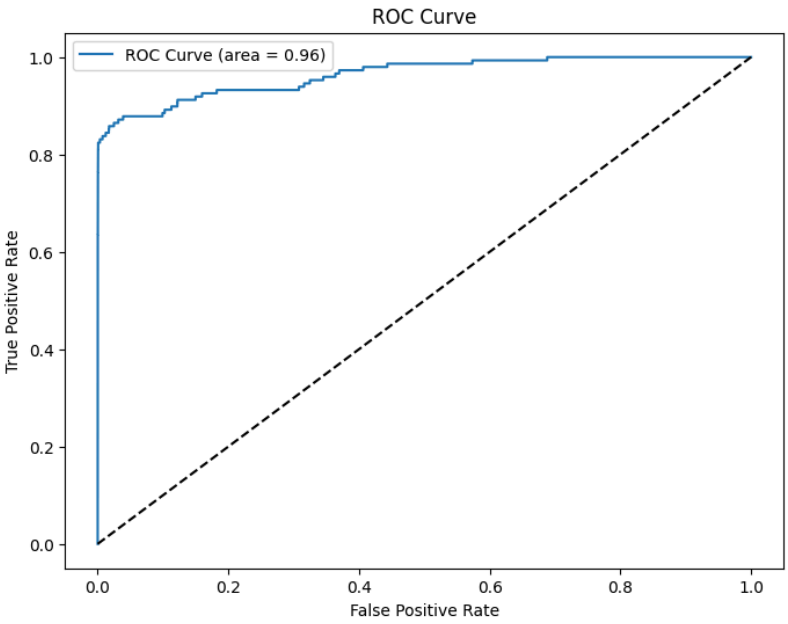
As we implement more ML models and preprocessing techniques, we expect further improvements in these metrics.

Confusion Matrix



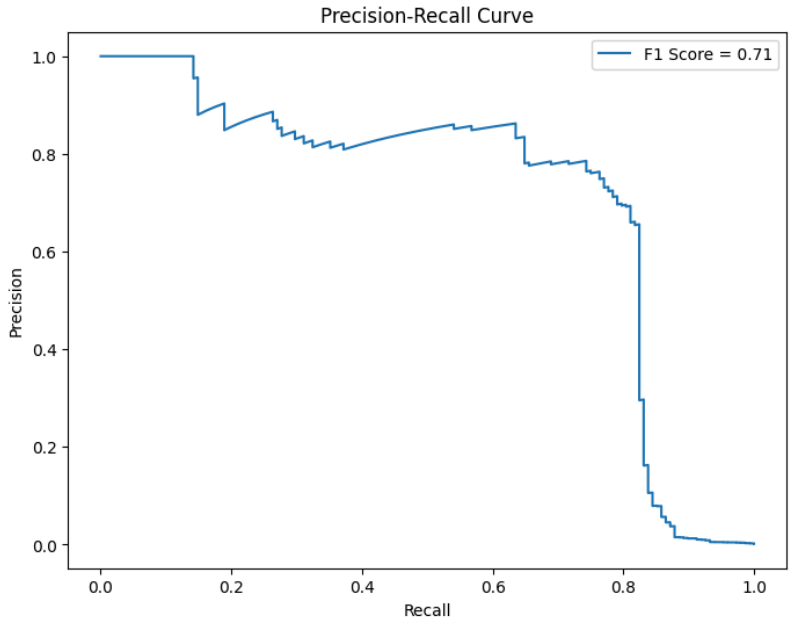
The confusion matrix shows true vs. predicted values, showing the model's performance in correctly identifying fraudulent and non-fraudulent transactions.

ROC Curve



The ROC curve shows the model's ability to distinguish between classes, with the area under the curve (AUC) indicating overall performance.

Precision-Recall Curve



The precision-recall curve shows the balance between precision and recall, helping assess the model's performance on our imbalanced dataset.

Decision Tree Results

Bel the outputs from our Decision-Tree model which includes the confusion matrix, ROC curve, precision-recall curve, and quantiative metrics.

Quantiative Metrics

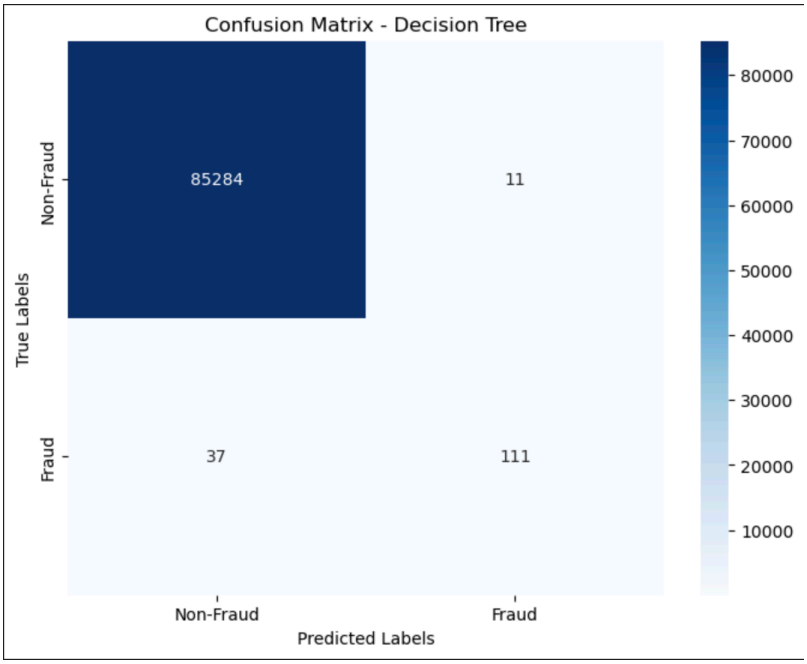
ROC-AUC Score: 0.88

F1 Score: 0.82

Precision: 0.90

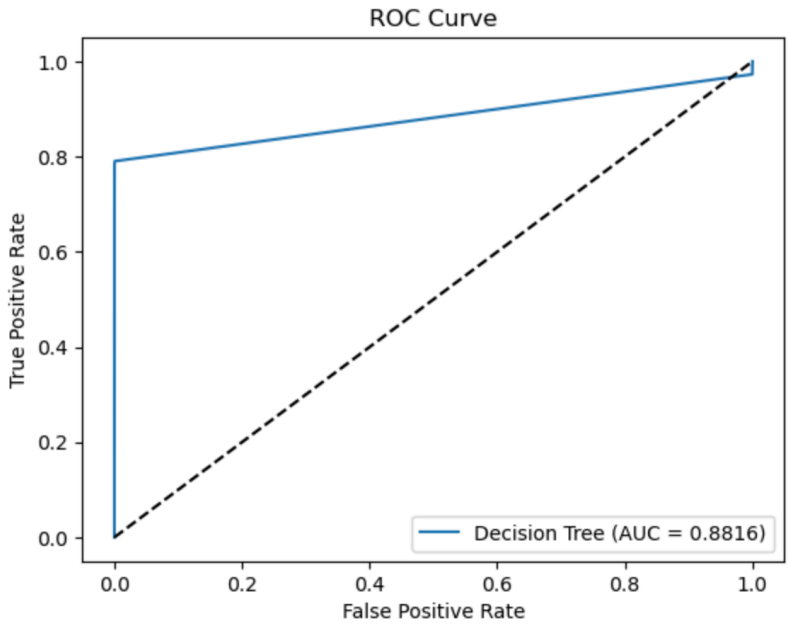
Recall: 0.75

Confusion Matrix



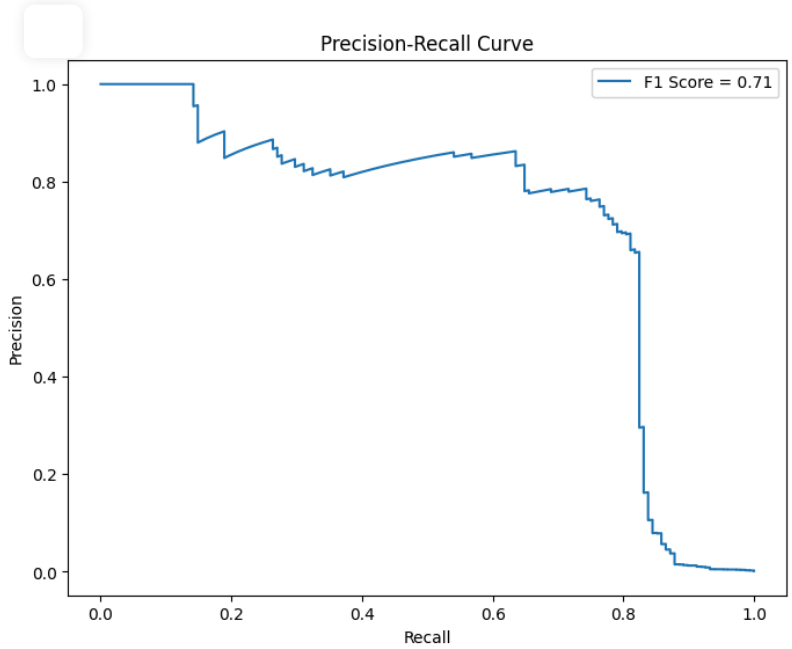
The confusion matrix shows true vs. predicted values, showing the model's performance in correctly identifying fraudulent and non-fraudulent transactions.

ROC Curve



The ROC curve shows the model's ability to distinguish between classes, with the area under the curve (AUC) indicating overall performance.

Precision-Recall Curve



The precision-recall curve shows the balance between precision and recall, helping assess the model's performance on our imbalanced dataset.

XGBoost

Below are the outputs from our XGBoost model which includes the confusion matrix, ROC curve, precision-recall curve, and quantiative metrics.

Quantiative Metrics

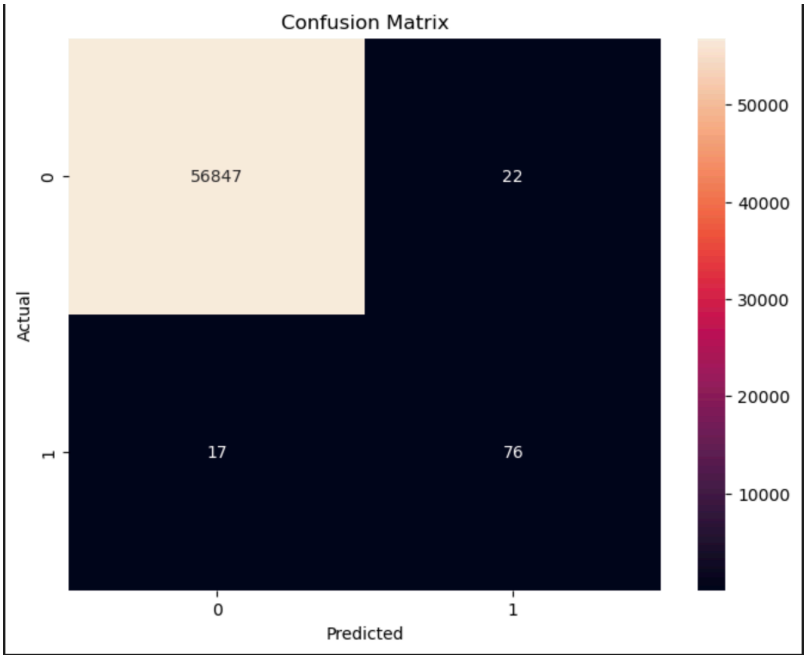
ROC-AUC Score: 0.97

F1 Score: 0.78

Precision: 0.77

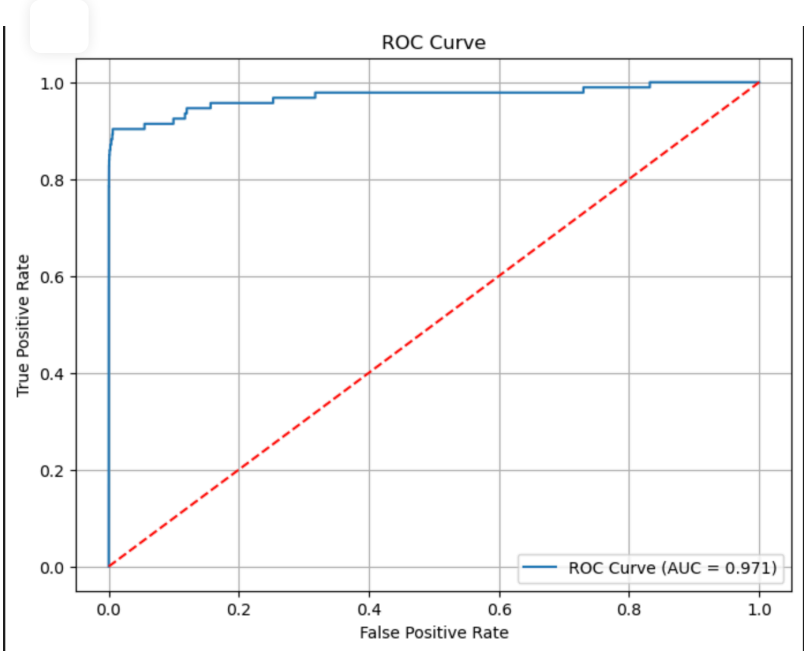
Recall: 0.82

Confusion Matrix



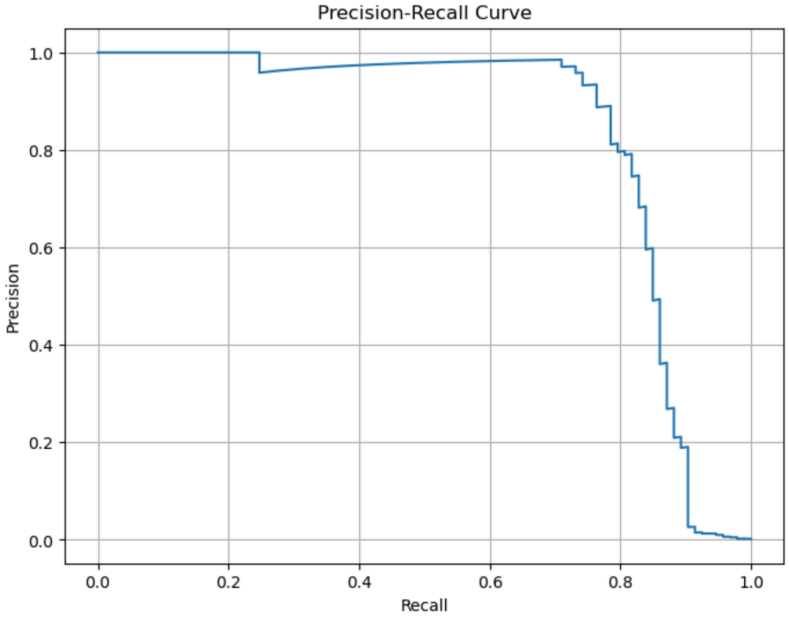
The confusion matrix shows true vs. predicted values, showing the model's performance in correctly identifying fraudulent and non-fraudulent transactions.

ROC Curve



The ROC curve shows the model's ability to distinguish between classes, with the area under the curve (AUC) indicating overall performance.

Precision-Recall Curve



The ROC curve shows the model's ability to distinguish between classes, with the area under the curve (AUC) indicating overall performance.

Comparison of Models

Our comprehensive analysis of credit card fraud detection models, we implemented and evaluated three distinct approaches through both robust visualization techniques and quantitative metrics. Each model revealed compelling insights through confusion matrices, ROC curves, and precision-recall curves, painting a detailed picture of their fraud detection capabilities.

ROC-AUC: XGBoost's highest ROC-AUC (0.97) shows it is the most capable at distinguishing between fraud and non-fraud cases across all probability thresholds. Logistic Regression follows closely (0.96), indicating its probabilistic framework works well. Decision Tree lags behind (0.88), which might be due to overfitting or less effective handling of imbalanced data.

F1 Score: Decision Tree has the best F1 Score (0.82), suggesting the best balance between precision and recall. XGBoost (0.78) and Logistic Regression (0.71) are slightly weaker in balancing these metrics.

Precision: Decision Tree's highest precision (0.90) indicates it excels in minimizing false positives, which is important for fraud detection systems. Logistic Regression (0.86) and XGBoost (0.77) are slightly weaker in this regard.

Recall: XGBoost's highest recall (0.82) makes it the best at capturing the maximum number of fraud cases. Decision Tree (0.75) follows, while Logistic Regression (0.61) struggles to capture all fraud cases.

Next Steps

To further enhance our fraud detection system, we can focus on advanced feature engineering, such as time-window aggregations and interaction terms, to better capture temporal and behavioral patterns in the data. Exploring hybrid and ensemble models, like combining XGBoost's high recall with Decision-Tree's interpretability, can help strike a balance between precision and usability. Addressing class imbalance with advanced sampling techniques beyond SMOTE, such as ADASYN or ensemble-based methods, will also be key to improving model performance. Additionally, developing a robust real-time scoring system with adaptive thresholding can optimize the model's performance in live environments while minimizing operational risks.

Looking ahead, we should explore the potential of neural network approaches for uncovering more complex patterns, ensuring they remain interpretable and practical. Techniques like SHAP or LIME can enhance model transparency, fostering trust among stakeholders. Establishing performance monitoring and continuous learning frameworks will ensure the system adapts to evolving fraud patterns. Finally, quantifying the trade-offs between false positives and false negatives in terms of business impact will help fine-tune the model's priorities for real-world deployment, balancing accuracy with operational efficiency.

5. References

[1] Federal Trade Commission, "Consumer Sentinel Network Data Book 2023," Federal Trade Commission, Washington, D.C., USA, Rep. 2023.

[2] Upgraded Points, "Credit Card Fraud & ID Theft Statistics," Upgraded Points, LLC, Austin, TX, USA, Rep. 2023.

[3] Chargebacks911, "Credit Card Fraud Statistics," Chargebacks911, Clearwater, FL, USA, Rep. 2023.

[4] ULB, "Credit Card Fraud Detection," Kaggle, 2018. [Online]. Available: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/data. [Accessed: 23-Sep-2024].

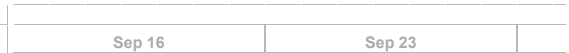
Gantt Chart

Proposal Gantt Chart

GANTT CHART

PROJECT TITLE: ML Project Timeline

TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION
Project Team Composition	All	10/4/2024	10/4/2024	0
Project Proposal				
Introduction & Background	Aman	9/16/2024	9/20/2024	4
Problem Definition	Shreyas	9/16/2024	9/26/2024	10
Methods	Royce & Armin	9/16/2024	9/30/2024	14
Potential Dataset	Aman	9/16/2024	9/19/2024	3
ial Results & Discussion	Meeth	9/16/2024	10/1/2024	15
Creation & Recording	Royce & Shreyas & Armin	10/2/2024	10/4/2024	2
GitHub Page	Shreyas & Royce	9/22/2024	10/2/2024	10
Midterm Report				
Model 1 (M1) Design & Selection	Person 1,2 & 3	2/17/2024	2/27/2024	10
M1 Data Cleaning	Person 1	2/17/2024	2/27/2024	10
M1 Data Visualization	Person 2	2/17/2024	2/27/2024	10
M1 Feature Reduction	Person 3	2/17/2024	2/27/2024	10
M1 Implementation & Coding	Person 4	2/28/2024	3/17/2024	19
M1 Results Evaluation	All	3/18/2024	3/20/2024	2
Model 2 (M2) Design & Selection	Person 1,2 & 3	2/28/2024	3/6/2024	8
M2 Data Cleaning	Person 1	2/28/2024	3/6/2024	8
M2 Data Visualization	Person 2	2/28/2024	3/6/2024	8
M2 Feature Reduction	Person 3	2/28/2024	3/6/2024	8
M2 Coding & Implementation	Person 5	3/7/2024	3/17/2024	10
M2 Results Evaluation	All	3/18/2024	3/20/2024	2
Midterm Report	All	3/28/2024	3/29/2024	1
Final Report				
Model 3 (M3) Design & Selection	Person 1,2 & 3	3/14/2024	3/20/2024	6
M3 Data Cleaning	Person 1	3/14/2024	3/20/2024	6
M3 Data Visualization	Person 2	3/14/2024	3/20/2024	6
M3 Feature Reduction	Person 3	3/14/2024	3/20/2024	6
M3 Implementation & Coding	Person 4	4/6/2024	4/14/2024	8
M3 Results Evaluation	All	4/15/2024	4/17/2024	2
M1-M3 Comparison	All	4/18/2024	4/26/2024	8
Video Creation & Recording	All	4/18/2024	4/26/2024	8
Final Report	All	4/18/2024	4/23/2024	5



Contributions

Team Member	Contributions till Midterm
Armin Moinian	Researched algorithms/models for credit card fraud detection, Logical Regression code.

Team Member	Contributions till Midterm
Aman Binu Kasim	Literature review, Data pre-processing, Gantt Chart, website formatting.
Meeth Mohan Naik	Quantitative metrics, project goals, expected outcomes, XGBoost code.
Royce B Arockiasamy	Data pre-processing, presentation slides, integration, video recording.
Shreyas Gadagi	Problem definition, presentation slides, decisiont tree code, UI development.