# FRAUD DETECTION : TERM PROJECT

Information available in audio.

**PRESENTED BY**
BEENA AHMED 25240
BEENISH AHMED 25147

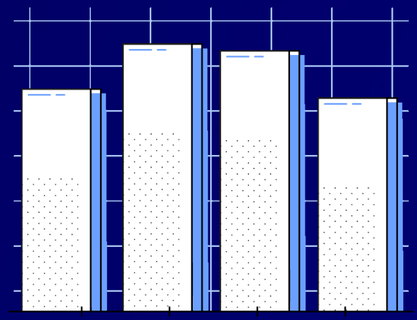
# AGENDA

# GOAL

To train a model that can predict whether a transaction is fraudulent or not based on various features in the dataset such as age, gender, card type, transaction amount, etc.

# ALGORITHMS IMPLEMENTED

- Logistic Regression
- SVM
- K-nearest-NeighborsClassifier
- Decision Tree
- Random Forest
- GradientBoostingClassifier



Project Implemented in PyCharm

# CODE IMPLEMENTATION

- The provided code performs fraud detection using various machine learning models.
- It starts by loading a dataset and preprocessing it by encoding categorical variables and standardizing continuous variables.
- The dataset is then split into training and testing sets. The code implements several classification models including Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Decision Tree, Random Forest, and Gradient Boosting.
- Model performance metrics such as accuracy, recall, precision, F1-score, and balanced accuracy are calculated and displayed.
- Additionally, there is a GUI application that allows users to input their own data for fraud detection.

# Code Snippet Explanation (Decision Tree)

- First, necessary libraries and modules required for the code have been imported.
- DecisionTreeClassifier object has been initialized.
- Decision Tree classifier has been fitted on the training data (X_train and y_train).
- Predict the target variable for the test data (X_test) using the trained Decision Tree classifier and store the predictions in 'y_pred4'.
- Calculate the Accuracy, Precision, Balanced Accuracy, F-1 Score of the Decision Tree classifier by comparing the predicted values ('y_pred4') with the actual values ('y_test')
- Similar approach has been taken to implement other algorithms.

```
print('Decision Tree')
print()
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
y_pred4 = dt.predict(X_test)
accuracy_dt = accuracy_score(y_test, y_pred4)
recall_dt = recall_score(y_test, y_pred4)
print("Recall score: {:.3f}".format(recall))
precision_dt = precision_score(y_test, y_pred4)
f1_dt = f1_score(y_test, y_pred4)
balanced_accuracy_dt = balanced_accuracy_score(y_test, y_pred4)
print('acurracy', accuracy_dt)
print("Precision:", precision_dt)
print("F1-score:", f1_dt)
print("Balanced accuracy:", balanced_accuracy_dt)
print()
print('random forest')
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred5 = rf.predict(X_test)
accuracy_rf = accuracy_score(y_test, y_pred5)
recall_rf = recall_score(y_test, y_pred5)
print("Recall score: {:.3f}".format(recall_rf))
precision_rf = precision_score(y_test, y_pred5)
f1_rf = f1_score(y_test, y_pred5)
balanced_accuracy_rf = balanced_accuracy_score(y_test, y_pred5)
print('acurracy', accuracy_rf)
print("Precision:", precision_rf)
print("F1-score:", f1_rf)
```


# DATASET EXPLANATION

Dataset represents attributes such as age, gender, card type, balance before the transaction, new balance, etc to make the predictions.

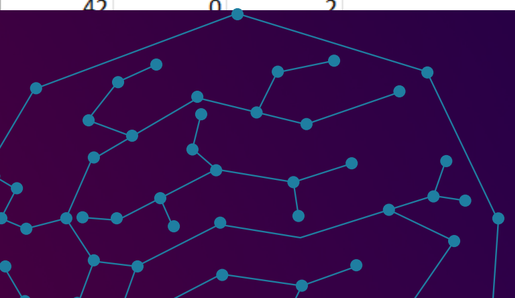
Label indicates whether the transaction is fraudulent or not.(1 fraud,0 no fraud)

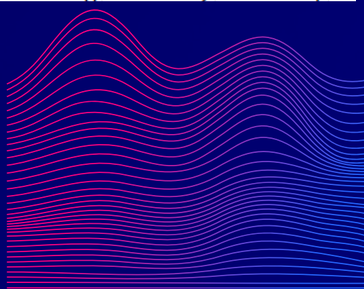
Numerical values are used in the dataset to provide advantages in data analysis and modeling as well as Ease of computation and Machine learning compatibility.

The trained model can leverage its learned patterns and relationships to predict or respond to new or unseen inputs.
However every model has different accuracy, precision, balanced accuracy and f1 score.

# DATASET EXPLANATION

| age | gender | card_type | balance_before_transaction | newbalance | transaction_type | device_type | Transaction_authorization_code | transaction_city | merchant_location | Channel_used_for_transaction | Country_t | transactio | is_fraudulent |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 52 | 1 | 0 | 125 | 212 | 0 | 1 | 168 | 0 | 1 | 2 | 2 | 3 | 0 |
| 53 | 1 | 0 | 140 | 203 | 1 | 0 | 155 | 1 | 3.1 | 0 | 0 | 3 | 0 |
| 70 | 1 | 0 | 145 | 174 | 0 | 1 | 125 | 1 | 2.6 | 0 | 0 | 3 | 0 |
| 61 | 1 | 0 | 148 | 203 | 0 | 1 | 161 | 0 | 0 | 2 | 1 | 3 | 0 |
| 62 | 0 | 0 | 138 | 294 | 1 | 1 | 106 | 0 | 1.9 | 1 | 3 | 2 | 0 |
| 58 | 0 | 0 | 100 | 248 | 0 | 0 | 122 | 0 | 1 | 1 | 0 | 2 | 1 |
| 58 | 1 | 0 | 114 | 318 | 0 | 2 | 140 | 0 | 4.4 | 0 | 3 | 1 | 0 |
| 55 | 1 | 0 | 160 | 289 | 0 | 0 | 145 | 1 | 0.8 | 1 | 1 | 3 | 0 |
| 46 | 1 | 0 | 120 | 249 | 0 | 0 | 144 | 0 | 0.8 | 2 | 0 | 3 | 0 |
| 54 | 1 | 0 | 122 | 286 | 0 | 0 | 116 | 1 | 3.2 | 1 | 2 | 2 | 0 |
| 71 | 0 | 0 | 112 | 149 | 0 | 1 | 125 | 0 | 1.6 | 1 | 0 | 2 | 1 |
| 43 | 0 | 0 | 132 | 341 | 1 | 0 | 136 | 1 | 3 | 1 | 0 | 3 | 0 |
| 34 | 0 | 1 | 118 | 210 | 0 | 1 | 192 | 0 | 0.7 | 2 | 0 | 2 | 1 |
| 51 | 1 | 0 | 140 | 298 | 0 | 1 | 122 | 1 | 4.2 | 1 | 3 | 3 | 0 |
| 52 | 1 | 0 | 128 | 204 | 1 | 1 | 156 | 1 | 1 | 1 | 0 | 0 | 0 |
| 34 | 0 | 1 | 118 | 210 | 0 | 1 | 192 | 0 | 0.7 | 2 | 0 | 2 | 1 |
| 51 | 0 | 2 | 140 | 308 | 0 | 0 | 142 | 0 | 1.5 | 2 | 1 | 2 | 1 |
| 54 | 1 | 0 | 124 | 266 | 0 | 0 | 109 | 1 | 2.2 | 1 | 1 | 3 | 0 |
| 50 | 0 | 1 | 120 | 244 | 0 | 1 | 162 | 0 | 1.1 | 2 | 0 | 2 | 1 |
| 58 | 1 | 2 | 140 | 211 | 1 | 0 | 165 | 0 | 0 | 2 | 0 | 2 | 1 |
| 60 | 1 | 2 | 140 | 185 | 0 | 0 | 155 | 0 | 3 | 1 | 0 | 2 | 0 |
| 67 | 0 | 0 | 106 | 223 | 0 | 1 | 142 | 0 | 0.3 | 2 | 2 | 2 | 1 |
| 45 | 1 | 0 | 104 | 208 | 0 | 0 | 148 | 1 | 3 | 1 | 0 | 2 | 1 |
| 63 | 0 | 2 | 135 | 252 | 0 | 0 | 172 | 0 | 0 | 2 | 0 | 2 | 1 |
| 42 | 0 | 2 | 120 | 209 | 0 | 1 | 173 | 0 | 0 | 1 | 0 | 2 | 1 |

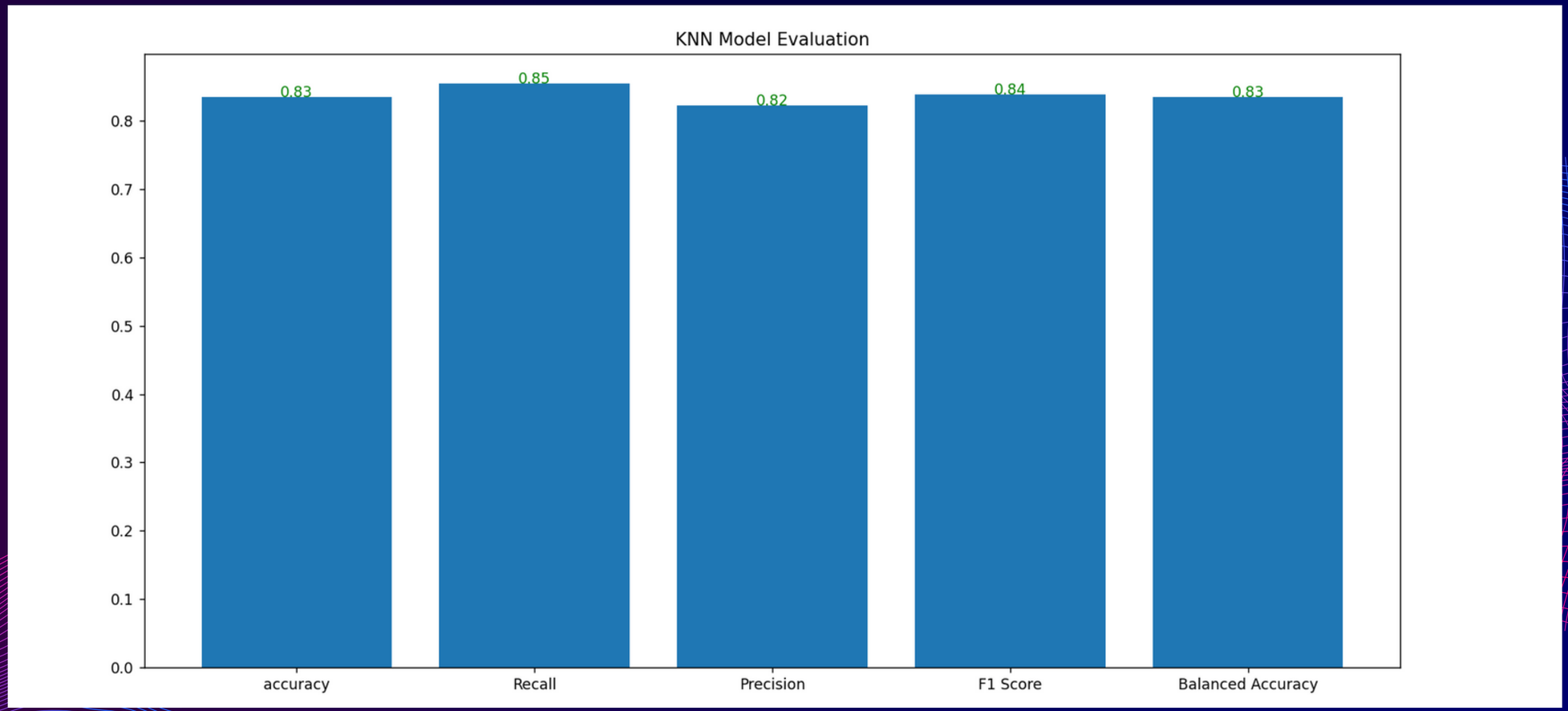




# GRAPHS : KNN

**The KNN algorithm makes predictions based on the principle that similar instances are likely to have similar labels. It classifies a new instance by comparing it to the labeled instances in the training data and selecting the majority class among its K nearest neighbors.**

**When making predictions for a new instance, the algorithm will consider the labels of the 10 nearest neighbors in the training data to determine the majority class.**




Scores

Evaluation Metrics

# RESULTS AND EVALUATION

***Logistic Regression***

recall 0.8932

acurracy 0.819

Precision: 0.779

F1-score: 0.832

Balanced accuracy: 0.819

***GradientBoostingClassifier***

Recall score: 0.862

acurracy 0.8032

Precision: 0.7575

F1-score: 0.8064

Balanced accuracy: 0.8060

***K-NEAREST-NEIGHBORSCLASSIFIER***

RECALL SCORE: 0.845

ACCURACY 0.790

PRECISION: 0.7631

F1-SCORE: 0.801

BALANCED ACCURACY: 0.7899

**The Support Vector Machine (SVM) model is the best because it has the highest overall performance based on accuracy, precision, recall, F1-score, and balanced accuracy.**

***random forest***

Recall score: 0.897

acurracy 0.8360

Precision: 0.787

F1-score: 0.8387

Balanced accuracy: 0.838

***svc***

Recall score: 0.942

accuracy 0.873

Precision: 0.829

F1-score: 0.881

Balanced accuracy: 0.872

***Decision Tree***

Recall score: 0.854

acurracy 0.7540

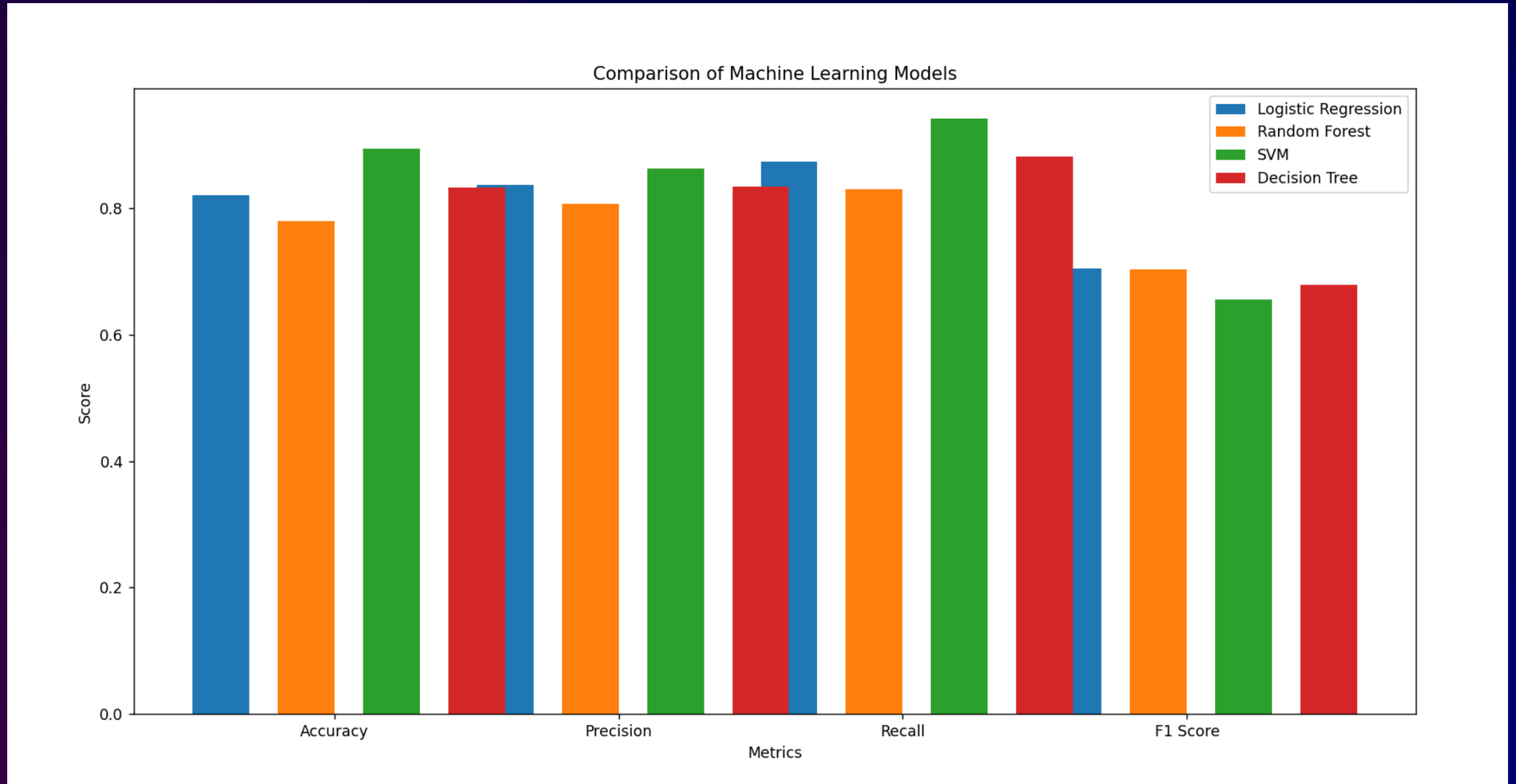
Precision: 0.7692

F1-score: 0.727

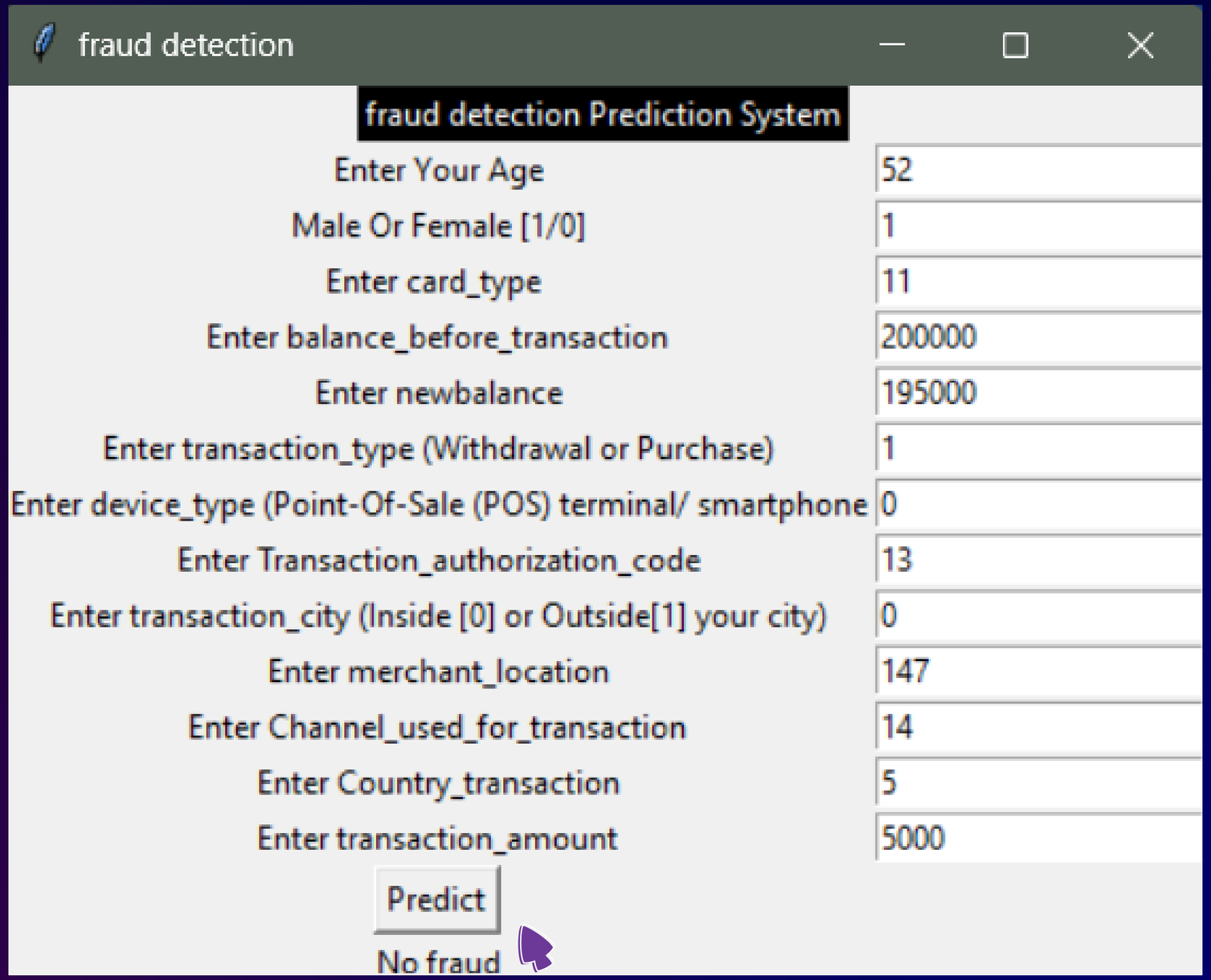
Balanced accuracy: 0.75107

# COMPARISON OF ALL MODELS IMPLEMENTED

THE GRAPH ALLOWS FOR A VISUAL COMPARISON OF HOW WELL EACH MODEL PERFORMS ACROSS DIFFERENT METRICS. IT CAN HELP IDENTIFY WHICH MODELS EXCEL IN SPECIFIC METRICS OR PROVIDE AN OVERALL UNDERSTANDING OF THE COMPARATIVE PERFORMANCE OF THE MODELS.



# OUTPUT: GUI REPRESENTATION



# LIME

The red bars indicate the negative impact of a feature on the prediction, meaning that when the value of that feature increases, it tends to push the prediction towards the negative class (class '0' in this case).

The green bars represent the positive impact of a feature on the prediction. When the value of a feature increases, it tends to push the prediction towards the positive class (class '1' in this case).

The length of each bar corresponds to the magnitude of the feature's contribution.

The y-axis represents the feature names or labels of the dataset.

The x-axis represents the importance or contribution values of the features.

# LIME

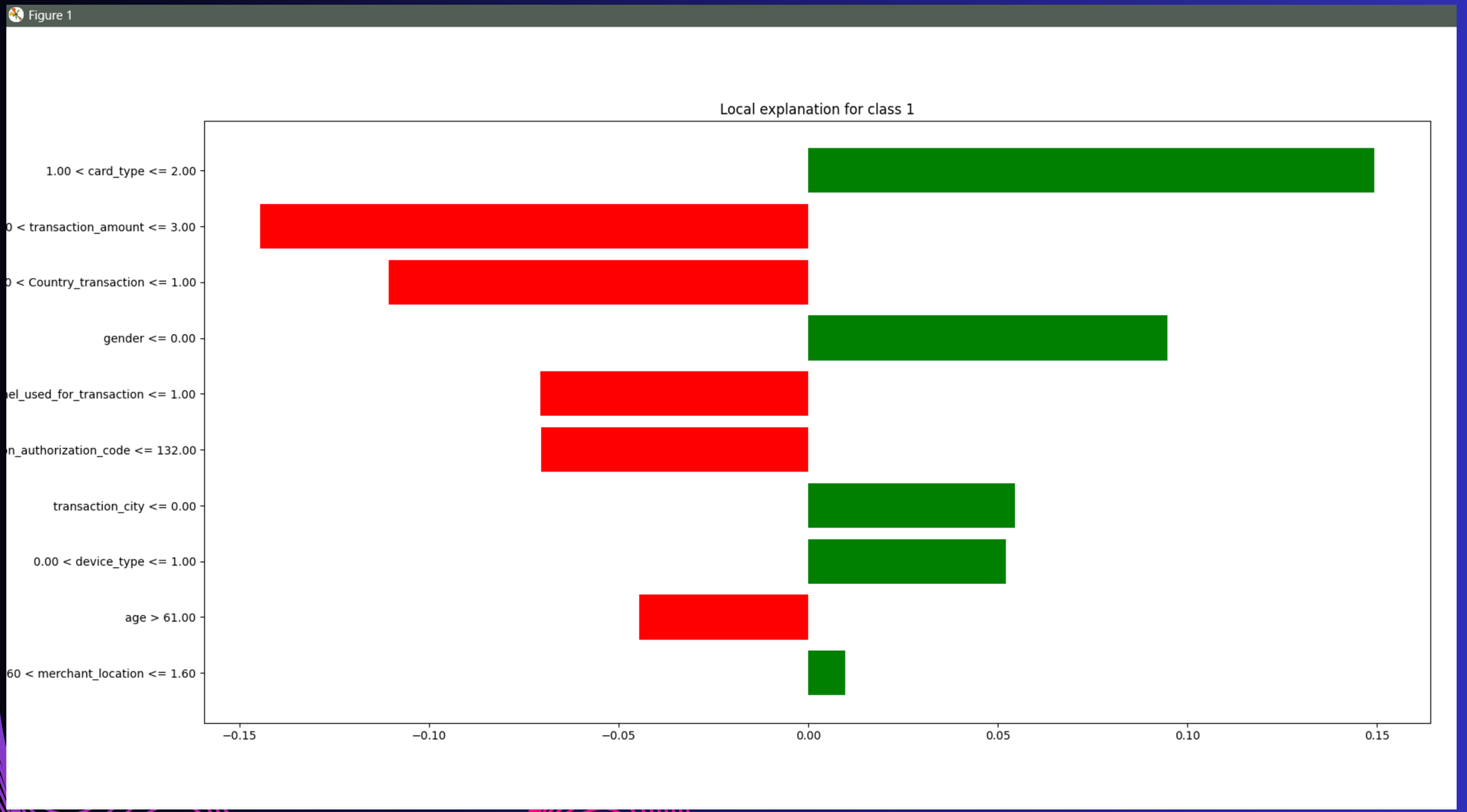


Figure 1
Local explanation for class 1
1.00 < card_type <= 2.00
0 < transaction_amount <= 3.00
0 < Country_transaction <= 1.00
gender <= 0.00
nel_used_for_transaction <= 1.00
on_authorization_code <= 132.00
transaction_city <= 0.00
0.00 < device_type <= 1.00
age > 61.00
60 < merchant_location <= 1.60
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15