Group Presenter:
1. **Nguyễn Thị Thuý Vy**
2. **Phạm Nguyệt Quỳnh**
3. **Nguyễn Thị Cẩm Tú**

MACHINE LEARNING

FOR DETECTING ABNORMAL TRANSACTION

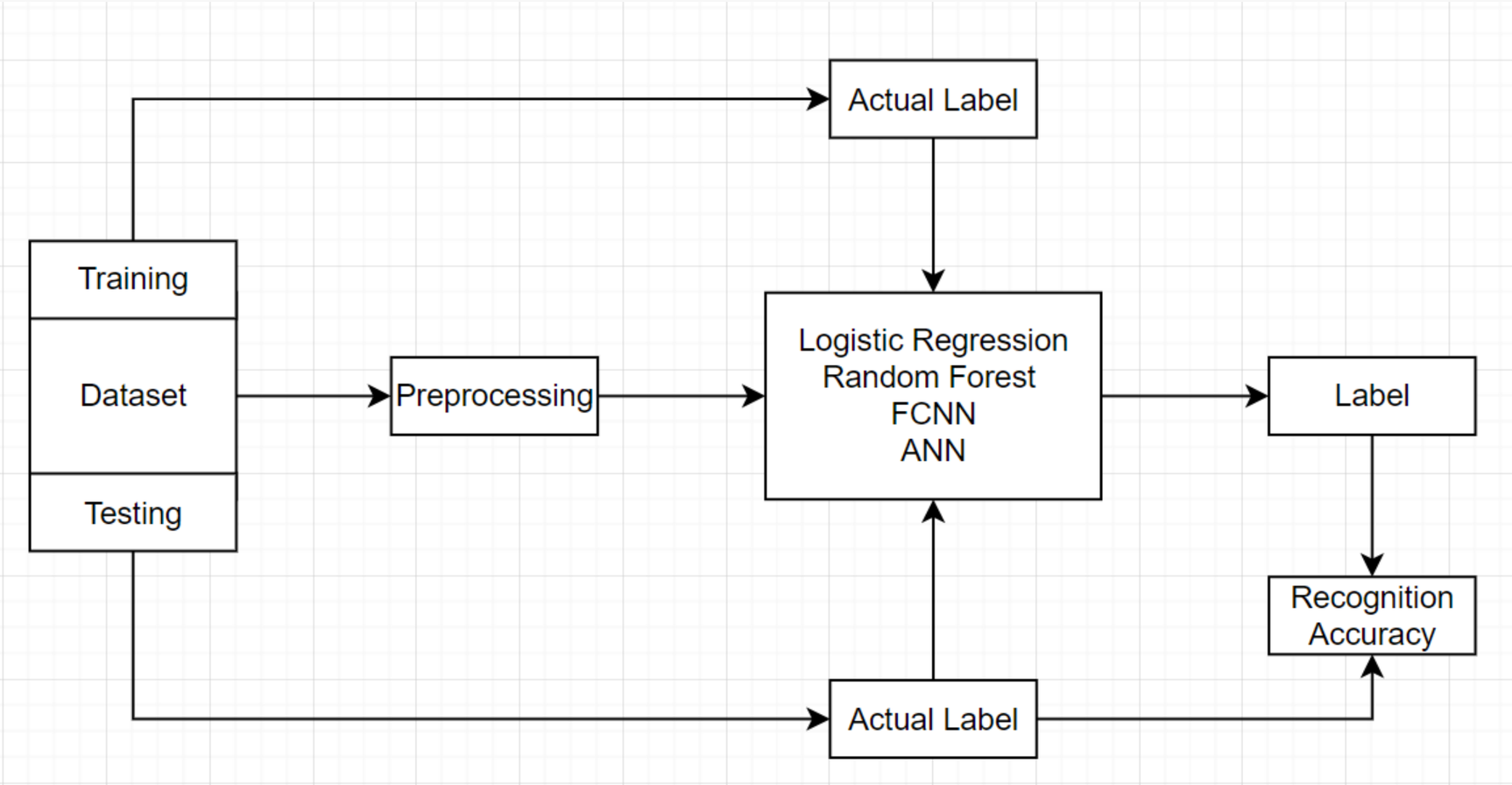
Content

MACHINE LEARNING FOR DETECTING ABNORMAL TRANSACTION

ABSTRACT

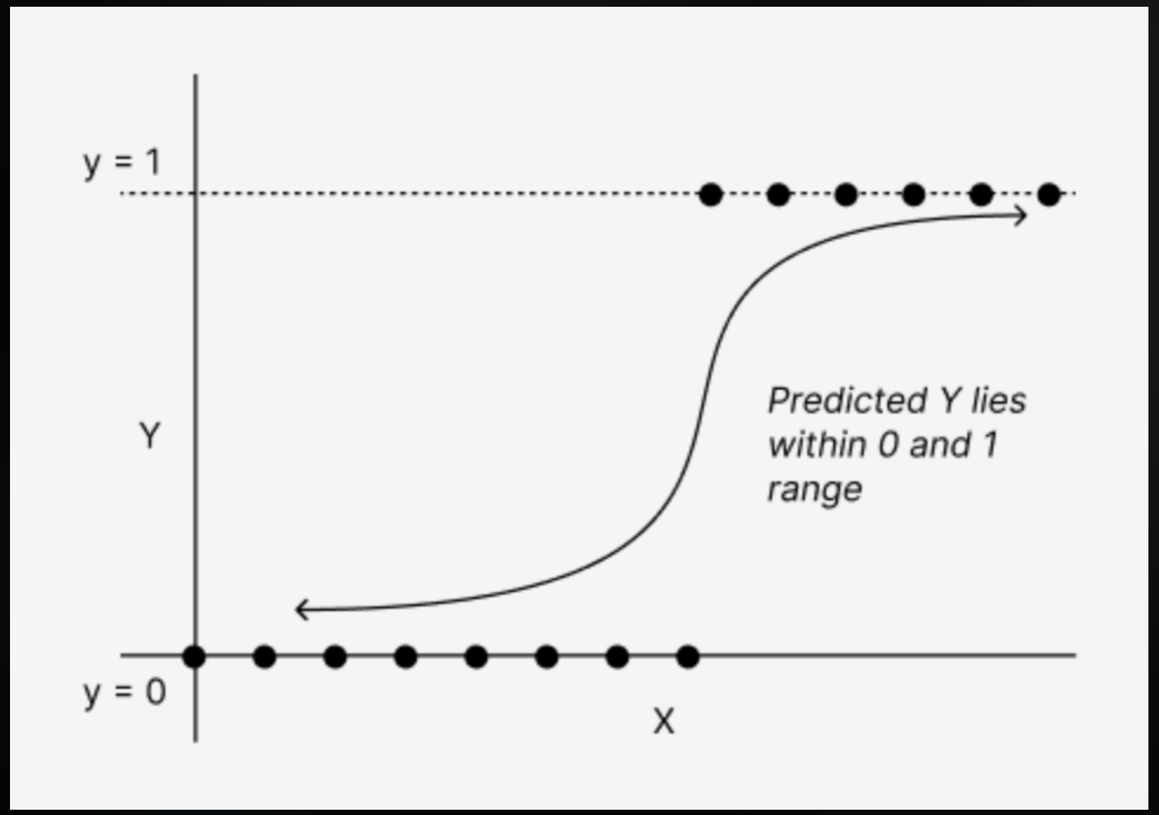
With the rise of credit card fraud and the use of advanced spam technologies, it is crucial to detect and prevent unauthorized transactions. This project aims to employ machine learning techniques like Logistic Regression, Random Forest, FCNN, and ANN to predict genuine and fraudulent transactions based on transaction amounts. The goal is to identify the most accurate and precise model for effective fraud detection and prevention in credit card transactions.

CREDIT CARD FRAUD DETECTION PROCESS



ALGORITHMS

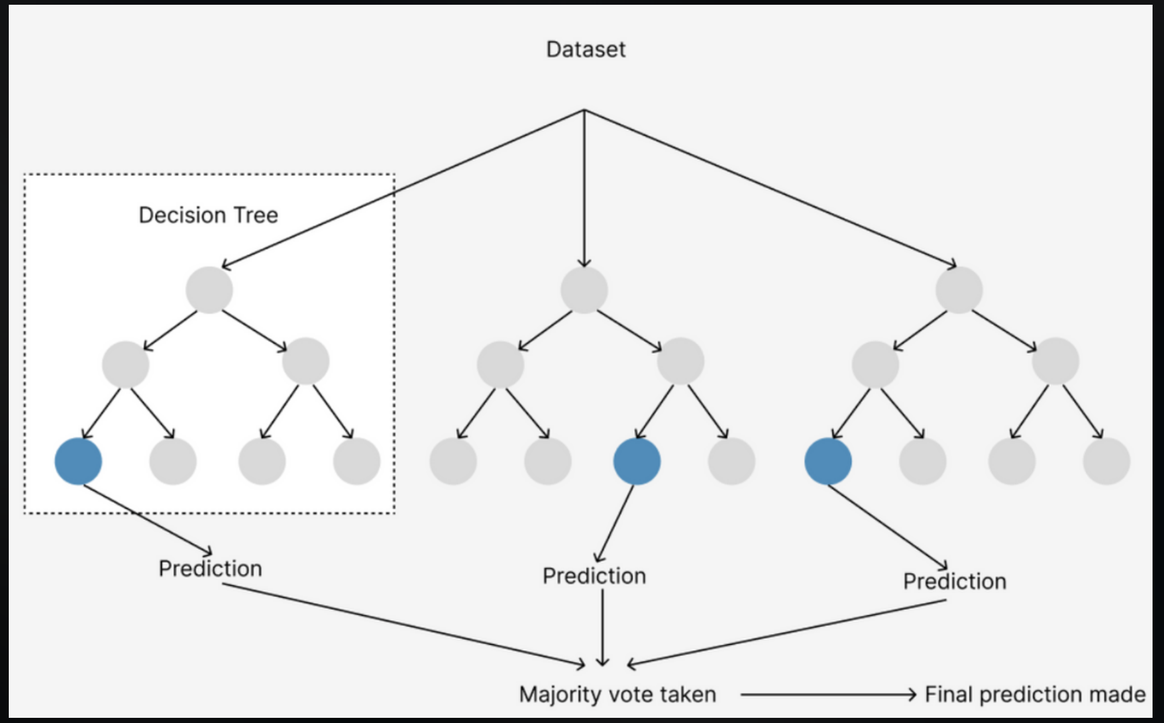
Logistic Regression



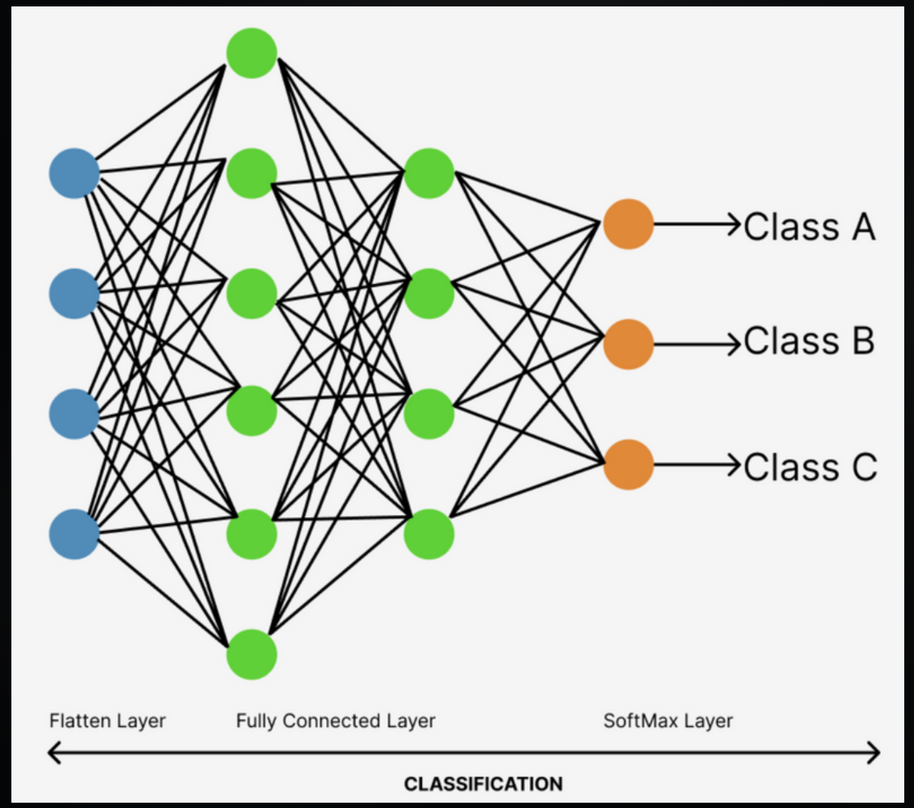
- Logistic regression is a supervised learning classification approach.
- It predicts the probability of a target variable based on a set of if-then statements represented in a decision tree structure.
- The algorithm splits data into subsets by evaluating significant attributes and maximizing information gain.
- It measures the impurity of result classes to determine the best splits.
- Logistic regression is known for its transparency and straightforward application through a series of questions down the decision tree.
- It is particularly useful for classifying data and predicting probabilities in various applications.

Random Forest Classifier

- Random Forest is an ensemble learning technique that combines multiple decision trees for improved classification or regression performance.
- It aggregates predictions from multiple trees to enhance accuracy and smooth out errors.
- Random Forest is particularly effective in handling unbalanced and missing data.
- It may struggle to make predictions beyond the range of the training data in regression tasks.
- Random Forest has the potential to overfit noisy datasets.
- It is essential to evaluate the performance of Random Forest based on its performance on the specific dataset being used.



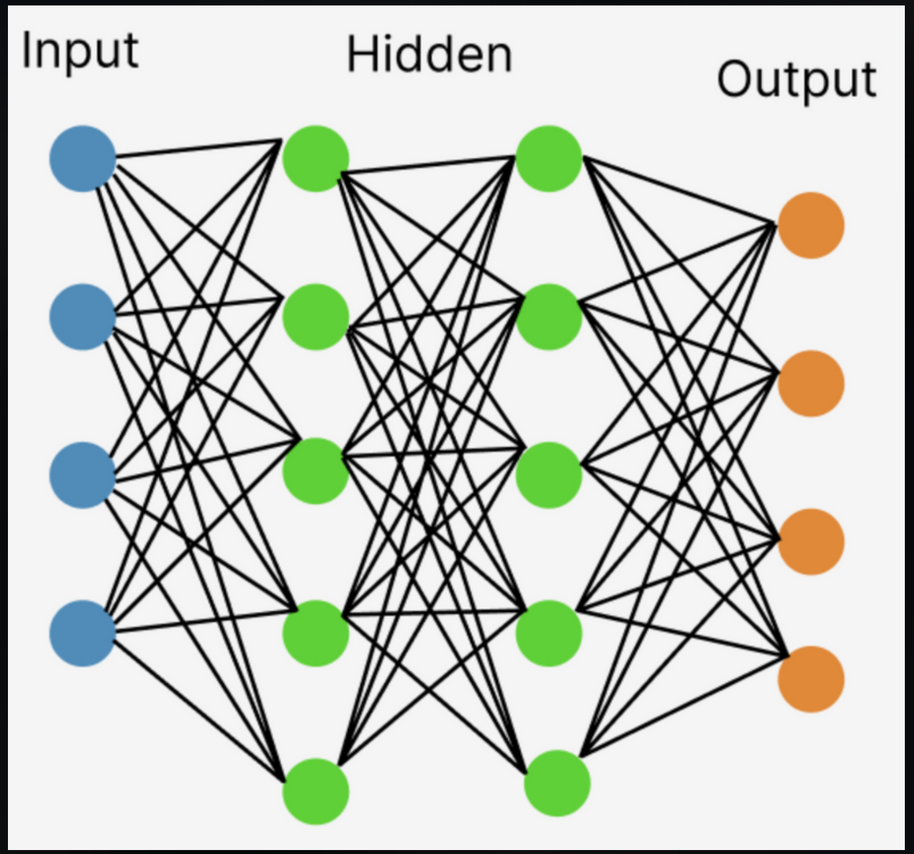
Fully Connected Neural Network



- FCNNs are extensively used in credit card fraud detection.
- They consist of interconnected layers of neurons that analyze transaction data.
- FCNNs learn from labeled data to identify fraudulent behavior.
- They can make predictions about the likelihood of new transactions being fraudulent.
- FCNNs excel at capturing intricate patterns and relationships in transaction data.
- Their ability to uncover complex patterns enhances the accuracy of fraud detection systems.

Artificial Neural Network

- Artificial Neural Networks (ANNs) are used in credit card fraud detection.
- ANNs mimic the human brain and learn patterns to identify fraudulent transactions.
- They analyze transaction data and optimize fraud detection through training.
- ANNs excel at capturing complex relationships in the data.
- They can adapt to changing fraud patterns, enhancing the effectiveness of fraud detection systems.



COMPARISON OF MODELS BASED ON ADVANTAGES AND LIMITATIONS

METHODS	PROS	CONS
Logistic Regression	Simplicity and interpretability	Assumes linear relationships between features and target variable
	Efficient computation	Limited ability to capture complex interactions and non-linear relationships
	Works well with linearly separable data	Prone to underfitting if data is not well represented by a linear model

COMPARISON OF MODELS BASED ON ADVANTAGES AND LIMITATIONS

METHOS	PROS	CONS
Random Forest Classifier	Ensemble of decision trees improves classification performance	Lack of interpretability compared to individual decision trees
	Handles high-dimensional data and complex relationships	Longer training time for large datasets
	Robust to outliers and noise	Can overfit noisy datasets if not properly tuned
		Limited ability to extrapolate beyond the range of training data

COMPARISON OF MODELS BASED ON ADVANTAGES AND LIMITATIONS

METHODS	PROS	CONS
FCNN	Ability to capture complex patterns and relationships in data	Requires a large amount of training data to avoid overfitting
	Adaptability to changing fraud patterns	Longer training time, especially for deep networks
	Can handle high-dimensional and non-linear data	Prone to black-box behavior and lack of interpretability
		Sensitive to hyperparameter tuning

COMPARISON OF MODELS BASED ON ADVANTAGES AND LIMITATIONS

METHODS	**PROS**	**CONS**
ANN	Flexibility to model complex relationships and non-linear patterns	Requires substantial computational resources
	Robust feature learning capabilities	Longer training time, especially for deep architectures
	Can handle high-dimensional data	Prone to overfitting if not properly regularized
	Adaptability to evolving fraud patterns	Interpretability can be challenging due to network complexity

EXPLORATORY DATA ANALYSIS

DATASET

In our dataset, there are **284,807 transactions over a 48-hour** period out of which only 0.172% are fraud and the rest are genuine.

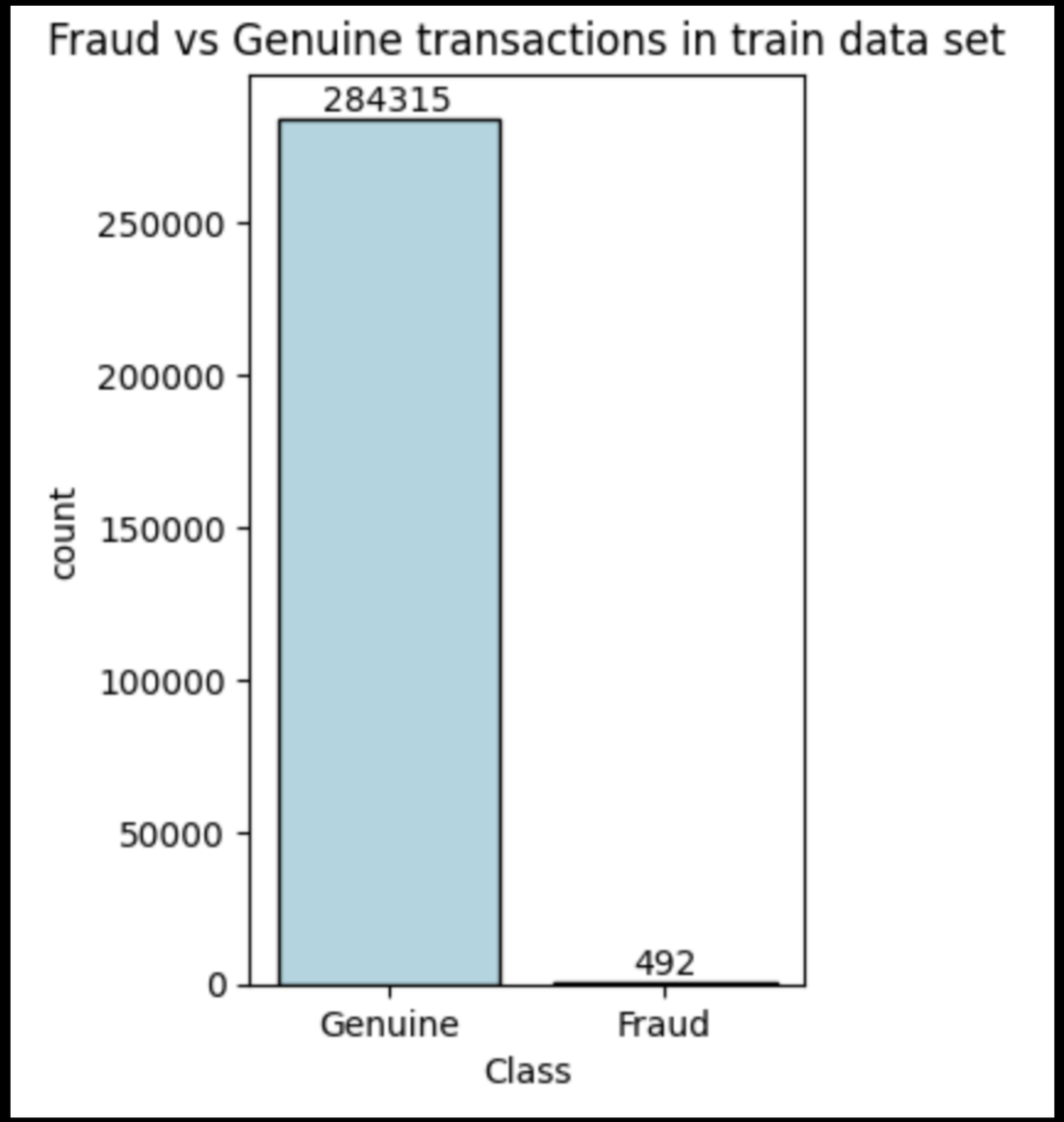
PAPER

Streaming active learning strategies for real-life credit card fraud detection: assessment and visualization

Credit card fraud detection: a realistic modeling and a novel learning strategy

Reproducible machine Learning for Credit Card Fraud Detection - Practical Handbook

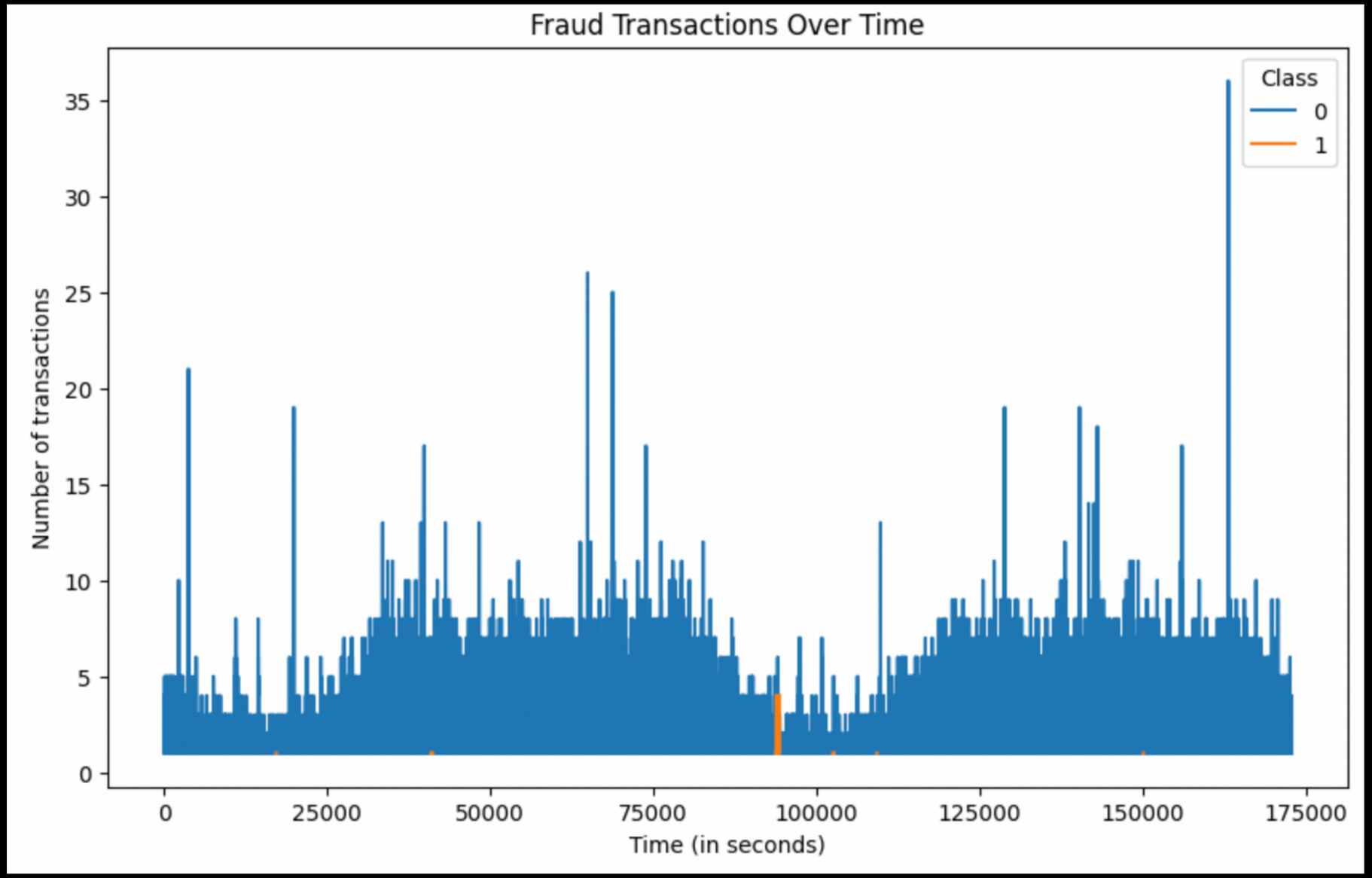
EDA

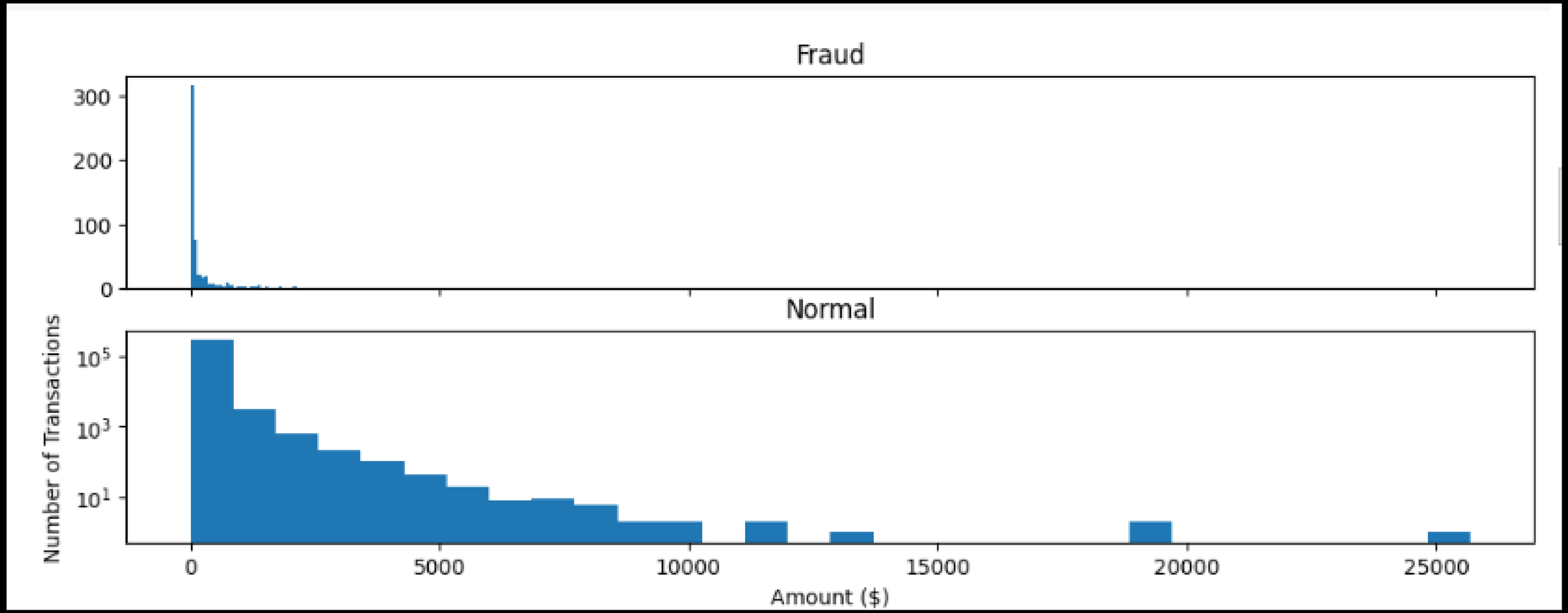



```
df.isnull().sum().sum()
```

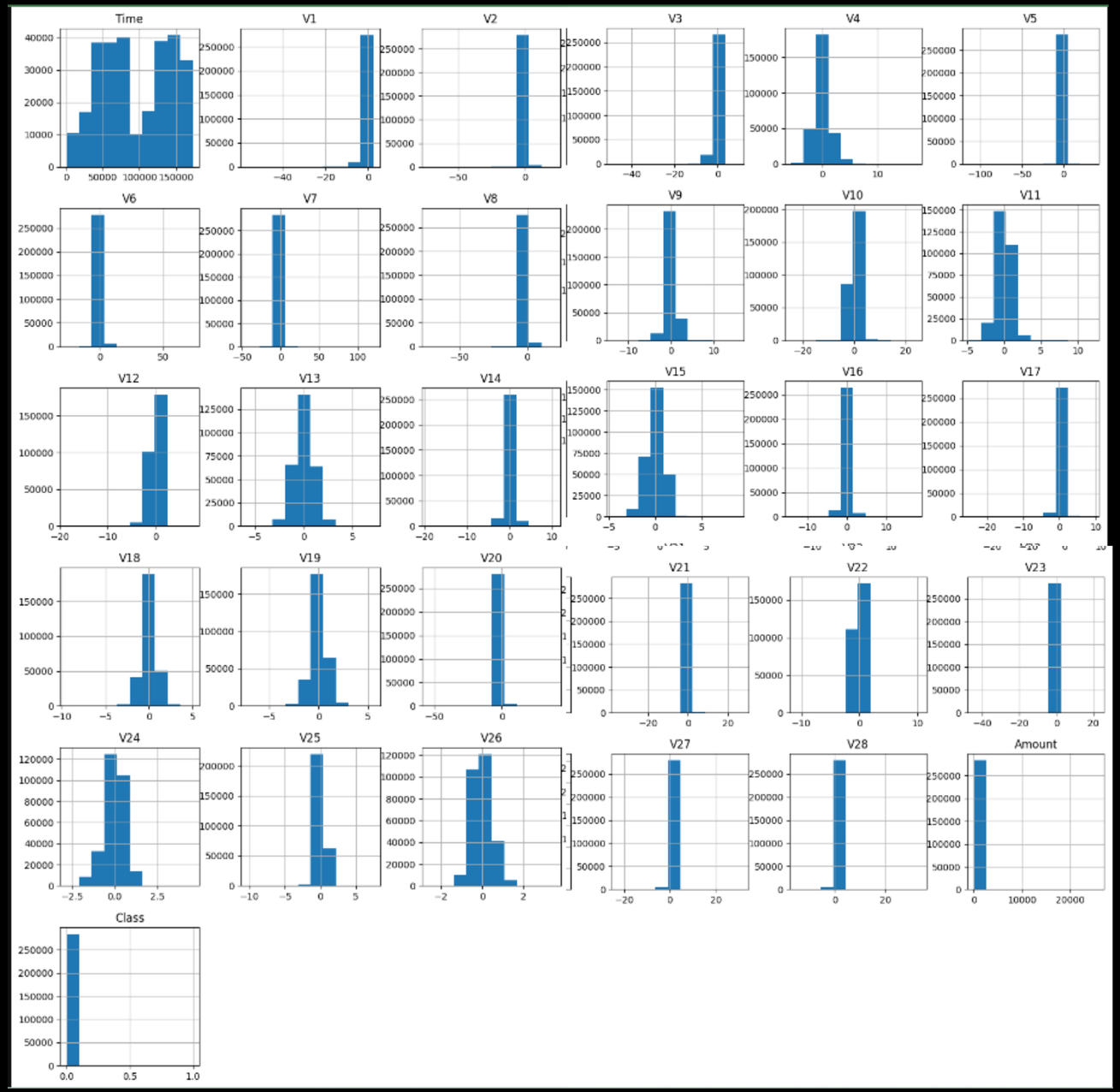
```
0
```

EDA


Fraud Transactions Over Time
Class
0
1
Number of transactions
0
5
10
15
20
25
30
35
0
25000
50000
75000
100000
125000
150000
175000
Time (in seconds)


Fraud
300
200
100
0
Normal
Number of Transactions
10^5
10^3
10^1
0
5000
10000
15000
20000
25000
Amount ($)

The approach that achieves the highest accuracy, recall score, precision score and F1_score will be considered the best fit for this problem.


Time
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12
V13
V14
V15
V16
V17
V18
V19
V20
V21
V22
V23
V24
V25
V26
V27
V28
Amount
Class

MODEL CREATION AND TRAINING

Logistic Regression

Preprocessing

```
X = df.drop(['Class'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Create an instance of LogisticRegression

```
model2=LogisticRegression();
model2.fit(X_train, y_train)
```

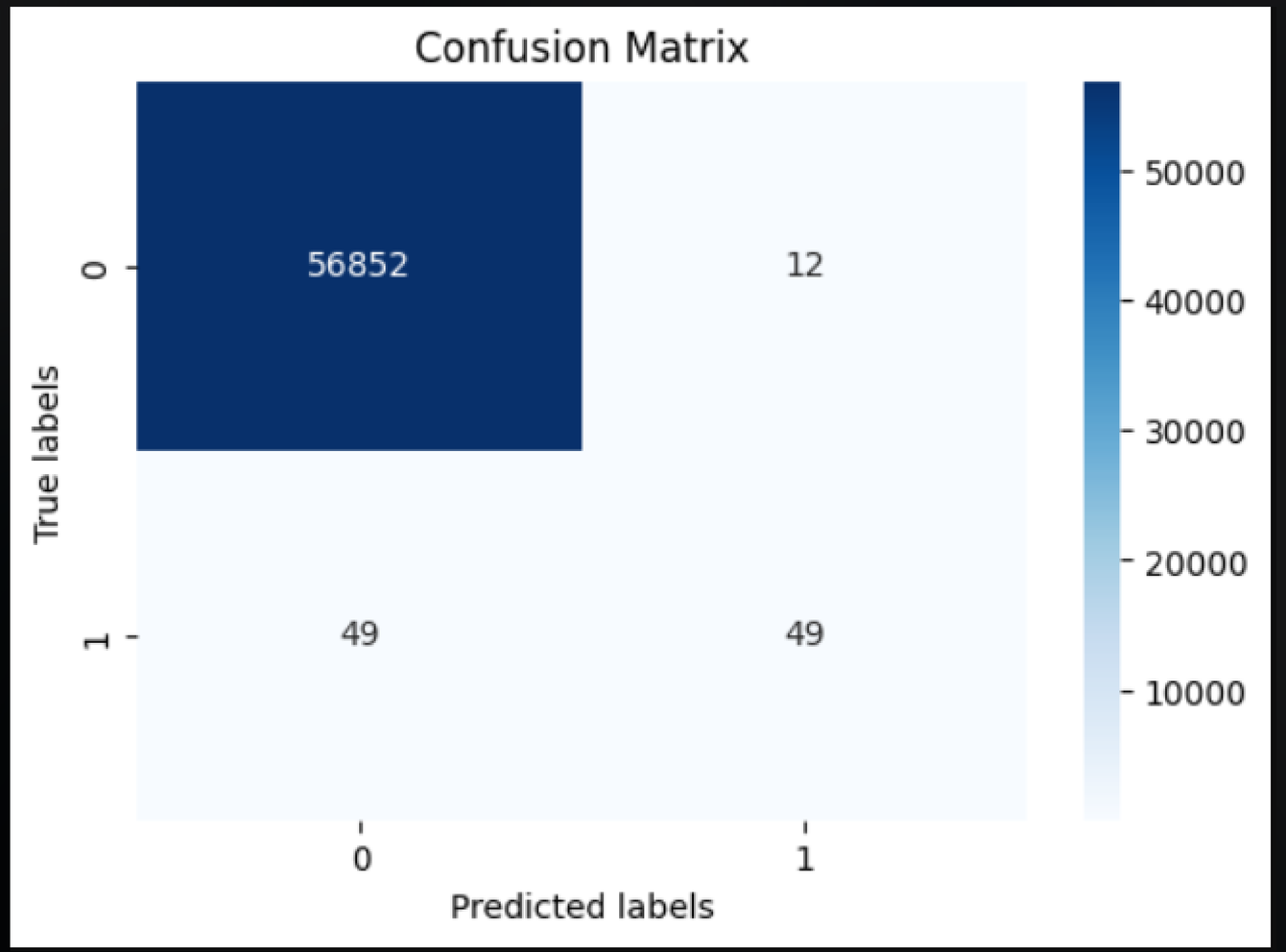
The objective function

```python
def objective(trial):
    # Define the hyperparameters to optimize
    C = trial.suggest_loguniform('C', 0.001, 100)
    max_iter = trial.suggest_int('max_iter', 100, 1000, step=100)

    # Create a logistic regression model with the suggested hyperparameters
    model = LogisticRegression(C=C, max_iter=max_iter, random_state=42)

    # Train the model on the training data
    model.fit(X_train, y_train)

    # Predict labels for the test set
    y_pred = model.predict(X_test)

    # Calculate accuracy
    accuracy = accuracy_score(y_test, y_pred)

    return accuracy
```

```
study = optuna.create_study(direction='maximize')

study.optimize(objective, n_trials=3)

best_params = study.best_params
print("Best parameters:", best_params)

# Train the logistic regression model using the best hyperparameters
best_model = LogisticRegression(**best_params, random_state=42)
best_model.fit(X_train, y_train)

# Predict labels for the test set using the best model
y_pred = best_model.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

Accuracy: 0.9989291106351603
Precision: 0.8032786885245902
Recall: 0.5
F1 score: 0.6163522012578616


Confusion Matrix
56852
12
49
49
True labels
0
1
Predicted labels
0
1
50000
40000
30000
20000
10000

Random Forest Classifier

Preprocessing

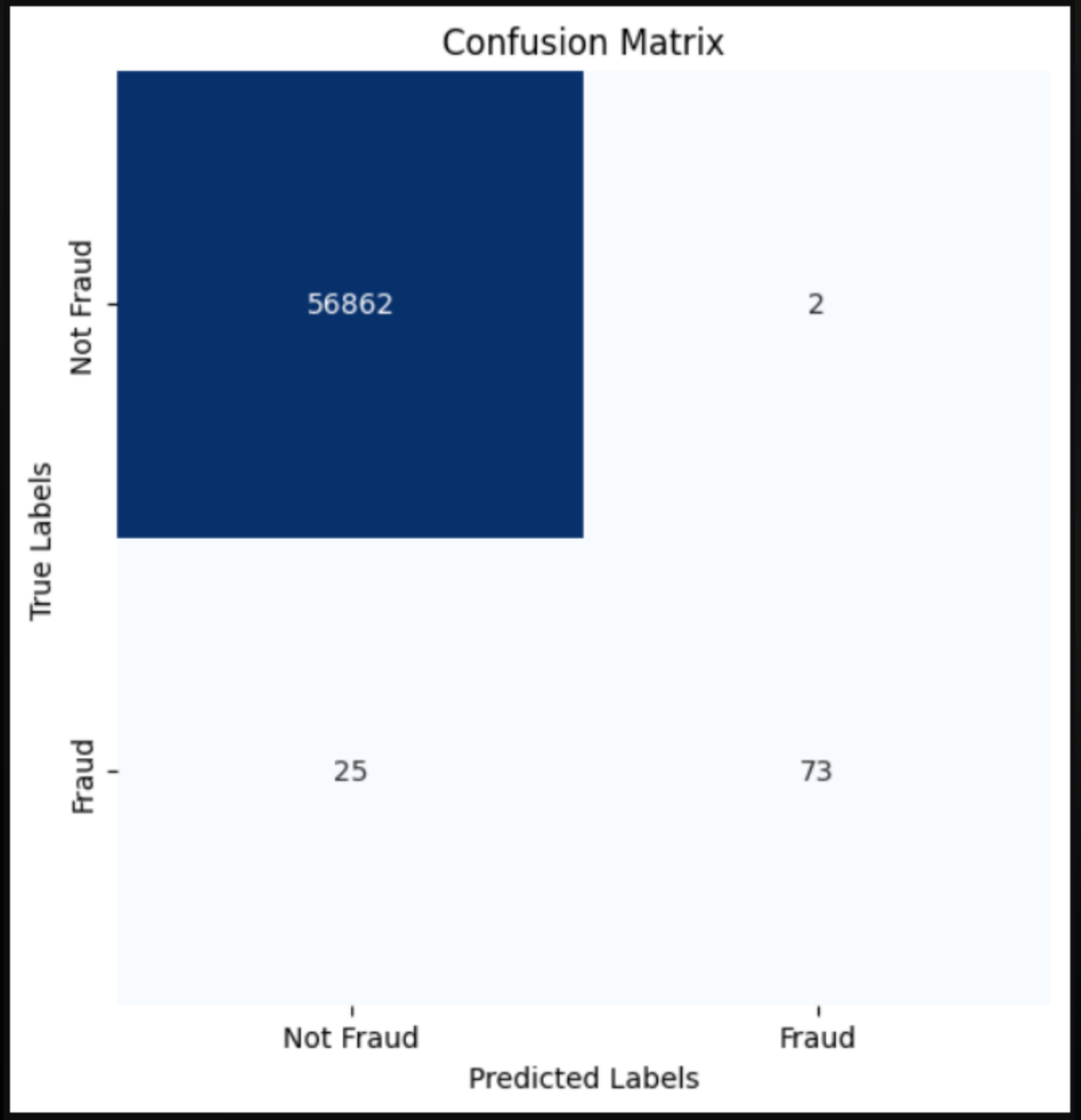
```
X = df.drop(['Class'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Define the objective function for the Optuna study

```
def objective(trial):
    # Define hyperparameters to optimize
    n_estimators = trial.suggest_int('n_estimators', 100, 1000, step=100)
    max_depth = trial.suggest_int('max_depth', 3, 10)
    min_samples_split = trial.suggest_int('min_samples_split', 2, 10)
    min_samples_leaf = trial.suggest_int('min_samples_leaf', 1, 5)

    # Create a Random Forest model with the suggested hyperparameters
    model = RandomForestClassifier(
        n_estimators=n_estimators,
        max_depth=max_depth,
        min_samples_split=min_samples_split,
        min_samples_leaf=min_samples_leaf,
        random_state=42
    )

    # Train the model on the training data
    model.fit(X_train, y_train)

    # Predict labels for the test set
    y_pred = model.predict(X_test)

    # Calculate accuracy
    accuracy = accuracy_score(y_test, y_pred)

    return accuracy
```

```python
study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=3)

# Print the best hyperparameters found by Optuna
best_params = study.best_params
print("Best parameters:", best_params)

# Train the Random Forest model using the best hyperparameters
best_model = RandomForestClassifier(**best_params, random_state=42)
best_model.fit(X_train, y_train)
 y_pred = best_model.predict(X_test)

 accuracy = accuracy_score(y_test, y_pred)
 precision = precision_score(y_test, y_pred)
 recall = recall_score(y_test, y_pred)
 f1 = f1_score(y_test, y_pred)

 print("Accuracy:", accuracy)
 print("Precision:", precision)
 print("Recall:", recall)
 print("F1 score:", f1)
```

Accuracy: 0.9995259997893332
Precision: 0.9733333333333334
Recall: 0.7448979591836735
F1 score: 0.8439306358381503


Confusion Matrix
Not Fraud
Fraud
True Labels
56862
2
25
73
Not Fraud
Fraud
Predicted Labels

Fully Connected Neural Network

Define the objective function for the Optuna study

```
# Defining the objective Function for Optuna
def objective(trial):
    model = Sequential()
    model.add(Dense(trial.suggest_int('n_units_1', 16, 128), input_dim=X_train.shape[1], activation='relu'))
    model.add(Dense(trial.suggest_int('n_units_2', 8, 64), activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

    model.fit(X_train, y_train, epochs=trial.suggest_int('epochs', 10, 100),
              batch_size=trial.suggest_int('batch_size', 32, 128), verbose=0)

    y_pred_proba = model.predict(X_test)
    y_pred = np.where(y_pred_proba > 0.5, 1, 0)
    accuracy = accuracy_score(y_test, y_pred)
    return accuracy
```

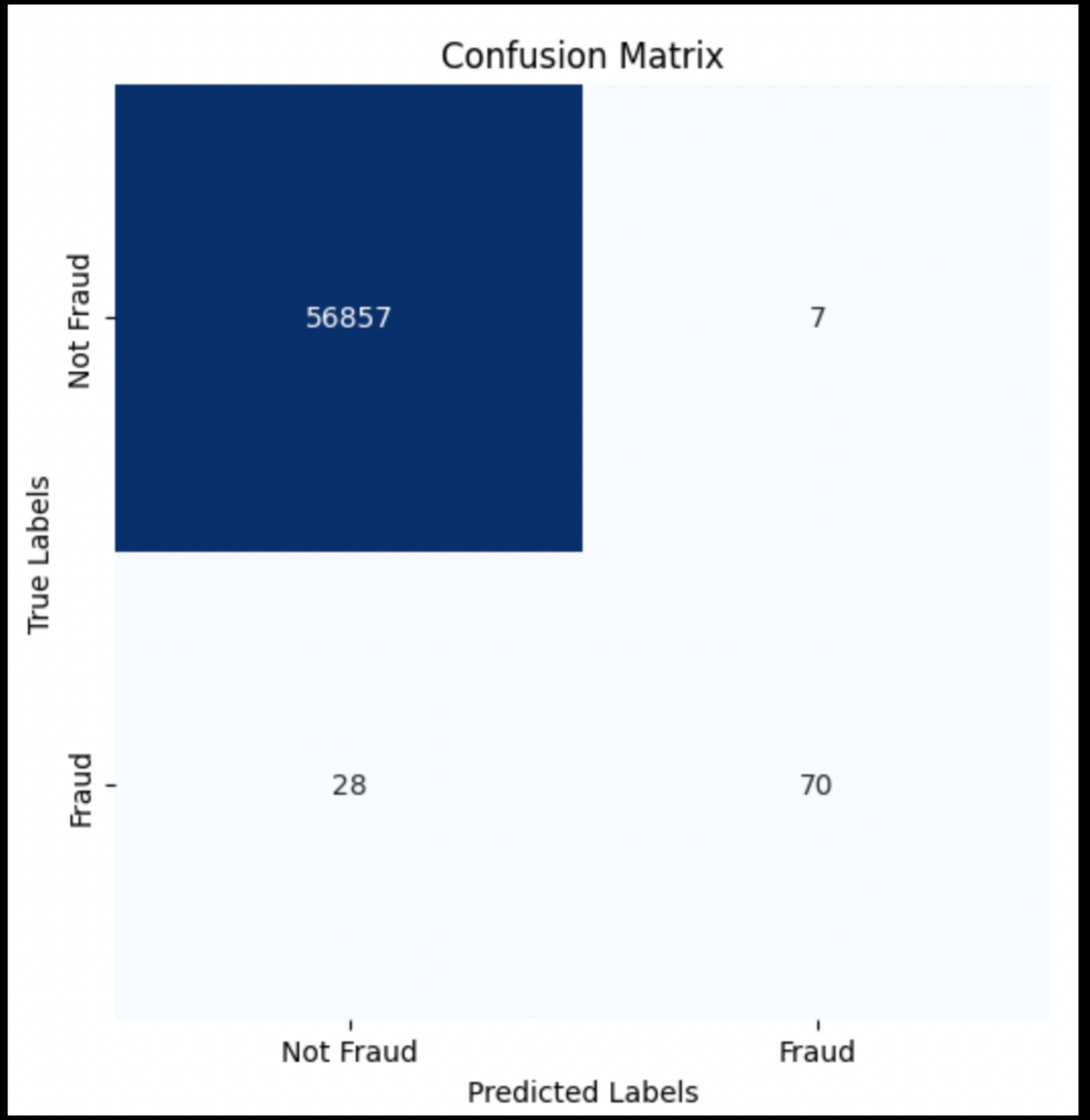
```python
# Hyperparameter Optimization using Optuna
study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=3)
```

```python
#  Creating the Best Model:
best_params=study.best_params
model = Sequential()
model.add(Dense(best_params['n_units_1'], input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(best_params['n_units_2'], activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

```python
#  Fine-tuning
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train,epochs=best_params['epochs'],
          batch_size=best_params['batch_size'], verbose=0)
```

```python
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

Accuracy: 0.999385555282469
Precision: 0.909090909090909
Recall: 0.7142857142857143
F1 score: 0.8


Confusion Matrix
Not Fraud
Fraud
True Labels
56857
7
28
70
Not Fraud
Fraud
Predicted Labels

Artificial Neural Network

Define the objective function for the Optuna study

```python
# Defining the objective Function for Optuna
def objective(trial):
    model = Sequential()
    model.add(Dense(trial.suggest_int('n_units_1', 16, 128), input_dim=X_train.shape[1], activation='relu'))
    model.add(Dense(trial.suggest_int('n_units_2', 8, 64), activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

    model.fit(X_train, y_train, epochs=trial.suggest_int('epochs', 10, 100), batch_size=trial.suggest_int('batch_size', 32, 128), verbose=0)

    y_pred_proba = model.predict(X_test)
    y_pred = np.where(y_pred_proba > 0.5, 1, 0)
    accuracy = accuracy_score(y_test, y_pred)
    return accuracy
```

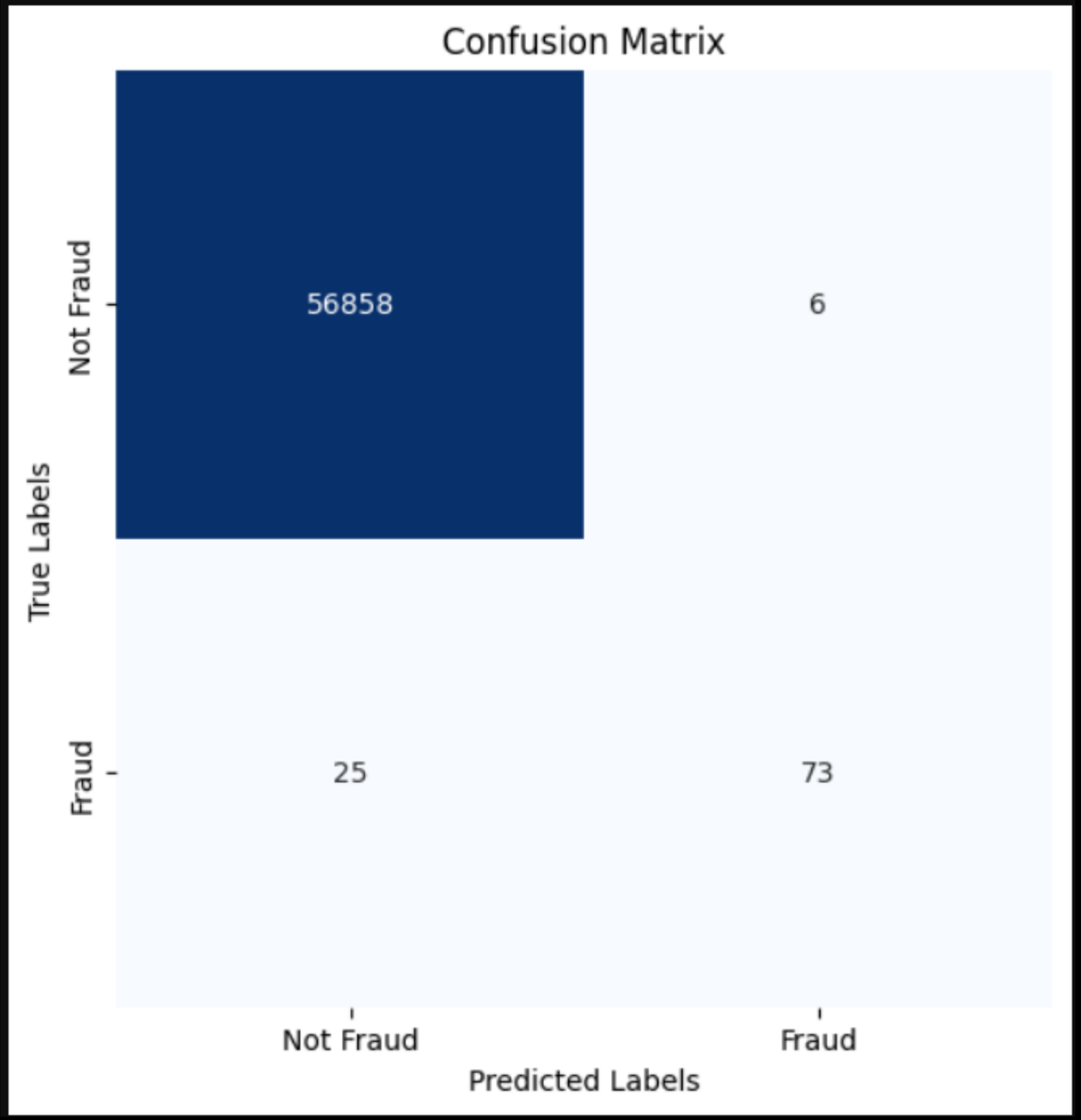
```python
study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=3)
```

```
study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=3)
```

```
# Creating and traing best model
best_params = study.best_params
best_model = Sequential()
best_model.add(Dense(best_params['n_units_1'], input_dim=X_train.shape[1], activation='relu'))
best_model.add(Dense(best_params['n_units_2'], activation='relu'))
best_model.add(Dense(1, activation='sigmoid'))
best_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
best_model.fit(X_train, y_train, epochs=best_params['epochs'], batch_size=best_params['batch_size'], verbose=0)
```

```
# Evaluate
y_pred_proba = best_model.predict(X_test)
y_pred = np.where(y_pred_proba > 0.5, 1, 0)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

Accuracy: 0.9994908886626171
Precision: 0.9259259259259259
Recall: 0.7653061224489796
F1 score: 0.8379888268156425


Confusion Matrix
Not Fraud
Fraud
True Labels
56858
6
25
73
Not Fraud
Fraud
Predicted Labels

Comparison of Machine Learning approaches

Name of Algorithms	TN	TP	Type 2 Error	Type 1 Error	Misclassification Rate	Accuracy	Precision	Recall	F1 Score
Logistic Regression	56852	49	25	12	0.0010708893648397	0.9989291106351603	0.8032786885245902	0.5	0.6163522012578616
Random Forest Classifier	56862	73	25	2	0.0004740002106668	0.9995259997893332	0.9733333333333334	0.7448979591836735	0.8439306358381503
FCNN	56857	70	28	7	0.000614444717531	0.9993855552825469	0.9090909090909091	0.7142857142857143	0.8
ANN	56858	73	25	6	0.0005442224640989	0.9994557775359011	0.9240506329113924	0.7448979591836735	0.8248587570621469

CONCLUSION

The **accuracy** is maximum for **Random Forest Classifier** followed by ANN.
The **misclassification/error rate** is minimum for **Random Forest Classifiers** followed by ANN.
The value of **sensitivity/recall** is maximum for **Random Forest Classifier as well as ANN**.
Precision is maximum for **Random Forest Classifier** followed by ANN.
F1 Score is maximum for **Random Forest Classifier** followed by ANN.
Type 2 Error is minimum for **Random Forest Classifier** followed by ANN.

CONCLUSION

For straightforward classification tasks with clear relationships between input and output data -> machine learning, specifically Random Forest.

For more complex problems -> deep learning, particularly ANN. This is due to the flexibility and adaptability of ANN in handling complex patterns and non-linear relationships within the data.

CONCLUSION

While machine learning algorithms like Random Forest excel in simpler classification scenarios, deep learning techniques such as ANN are better suited to tackle intricate problems that demand more advanced data processing capabilities.

In case of fraudulent transaction classification problem, we will choose **Random Forest** among the tried algorithms to get the most optimal results.

REFERENCES

(1) Credit Card Fraud Detection Using Machine Learning Approach}, by Kanal Bhadresh Soni, Madhuri Chopade, Rahul Vaghela

(2) Credit card transaction data analysis and performance evaluation of machine learning algorithms for credit card fraud detection} by Vivek Ghai and Sandeep Singh Kang

(3) Temporal Convolutional Neural Network for theClassification of Satellite Image Time Series} by Charlotte Pelletier, Geoffrey I. Webb, Senior Fellow, IEEE, and Franc ¸ois Petitjean

(4) Speech and Language Processing: Chapter 5. Logistic Regression} by Daniel Jurafsky and James H. Martin

(5) Random Forests for Regression and Classification

(6) Neural Networks vs. Random Forests – Does it always have to be Deep Learning? by Prof. Dr. Peter Roßbach

Link Youtube Video

Thank You!

FOR LISTENING