VIETNAM NATIONAL UNIVERSITY HCMC
UNIVERSITY OF INFORMATION TECHNOLOGY

—————o0o—————

FINAL REPORT

MACHINE LEARNING FOR DETECTING ABNORMAL TRANSACTION

Course: **E - Commerce**

Instructor: **Master DO DUY THANH**

Student: **PHAM NGUYET QUYNH** <21522537>

NGUYEN THI THUY VY <21520534>

NGUYEN THI CAM TU <21520513>

Class: **IS334.N21.TMCL**

HO CHI MINH CITY, 07/2023

Acknowledgement

We would like to express our sincere gratitude to Mr. Do Duy Thanh, the lecturer of the E-commerce course at the University of Information Technology, for equipping us with the necessary skills and knowledge to successfully complete this project. However, due to our limited expertise in the field, we acknowledge that there may be some shortcomings in our presentation and evaluation of the topic. We sincerely welcome feedback and evaluation from the faculty members to further improve our work.

Sincerely thank you!

Ho Chi Minh City, July 12, 2023.

Pham Nguyet Quynh
Nguyen Thi Thuy Vy
Nguyen Thi Cam Tu

Table of contents:

JUDGE BY THE INSTRUCTOR

<footer>3</footer>

1 ABSRACT

Using advanced spam technologies to commit internet banking fraud involves unauthorized shifting and withdrawal of money from the user's account balance. The concept of fraud often brings credit card fraud to mind, especially given the current scenario. Credit card fraud has significantly increased recently, thanks to the rapid growth of credit card purchases. To detect, identify, and prevent such undesirable activities, fraud detection involves monitoring users' purchase behavior. The goal of this project is to employ various machine learning approaches such as Logistic Regression, Random Forest, FCNN, and ANN to predict genuine and fraudulent transactions based on the transaction amount. The model that exhibits higher accuracy and precision will be considered the best fit for this system.

2 INTRODUCTION

Fraud is a pervasive and costly issue in sectors such as finance, insurance, and e-commerce. Its impact on businesses and organizations includes financial losses, reputational damage, and legal consequences. According to the PwC economic crime survey of 2022, a significant 46% of organizations surveyed reported experiencing fraud or economic crime within a 24-month period.

Detecting and preventing fraudulent activities has become a crucial task for organizations across various industries. The implications of fraudulent behavior, such as financial losses, reputational harm, and legal liabilities, emphasize the need for robust fraud detection measures.

In sectors like finance, insurance, and e-commerce, the stakes are particularly high due to the substantial financial transactions involved. Businesses must be proactive in identifying and addressing fraudulent behavior to protect their assets and maintain customer trust.

The PwC survey findings highlight the pervasive nature of fraud and the urgency to address it. Organizations need effective fraud detection systems and strategies to safeguard their operations and mitigate potential risks. By leveraging advanced technologies and data analysis techniques, businesses can enhance their fraud detection capabilities and prevent financial losses.

In this project, we aim to contribute to the ongoing efforts in fraud detection by developing a machine learning model using the Credit Card Fraud dataset. By utilizing these advanced techniques, we seek to accurately differ-

entiate between genuine and fraudulent transactions, reducing false positives and improving detection accuracy.

The significance of this project lies in its alignment with the challenges faced by organizations across finance, insurance, and e-commerce sectors. By developing effective fraud detection systems, businesses can protect their financial interests, mitigate risks, and maintain the trust of their stakeholders.

3 CHALLENGES IN CREDIT CARD FRAUD DETECTION

Credit card fraud detection faces several challenges that need to be addressed for effective prevention and accurate identification of fraudulent transactions. These challenges include the dynamic nature of fraudulent behavior profiles, the presence of skewed or imbalanced datasets, the selection of optimal features and variables, the choice of appropriate performance evaluation metrics, the risk of class bias and classification errors, and the impact of sampling techniques on accuracy. Overcoming these challenges requires the application of machine learning-based approaches that adapt to the dynamic nature of fraud, consider the specific challenges of imbalanced datasets, and continually enhance fraud prevention accuracy based on individual cardholder activity statistics.

4 CREDIT CARD FRAUD DETECTION PROCESS

The detection of fraud in credit card transactions is a classification problem. It involves building a model using historical credit card transactions, including both genuine and fraudulent ones, to distinguish between legitimate and fraudulent transactions. This constructed model is then utilized to classify new transactions as either legitimate or fraudulent.

To train high-quality machine learning models for credit card fraud detection, a significant amount of internal historical data is required. The quality of the model's training process heavily depends on the availability of sufficient previous data that indicates both fraudulent and normal transactions. Data augmentation techniques, such as dimensionality reduction, are commonly employed to address the class imbalance issue when there is an unequal number of data samples for each class.

There are two types of Machine Learning based methods for detecting credit card fraud:

- Supervised: Such as Decision Tree, Logistic Regression, K-Nearest Neighbors, SVM, Random ForestTree, XGBoost.

- Unsupervised: Such as Principal Component Analysis, K-means clustering and Neural Networks.

The transaction data is analyzed to extract relevant features that align with business requirements. The data is then split into two sets: one for training (usually 70-80% of the data) and another for testing (usually 20-30% of the data). During the training phase, the models are fed with the data to learn from the historical patterns and predict the likelihood of fraudulent transactions. The ultimate goal is to achieve 100% detection of fraudulent transactions while minimizing the misclassification of non-fraud transactions.

- True Positives: Fraud cases predicted as fraud.

- True Negatives: Non-fraud cases predicted as non-fraud.

- False Positives: Non-fraud cases predicted as fraud.

- False Negatives: Fraud cases predicted as non-fraud.

The figure shows the architectural techniques for representing the overall framework.

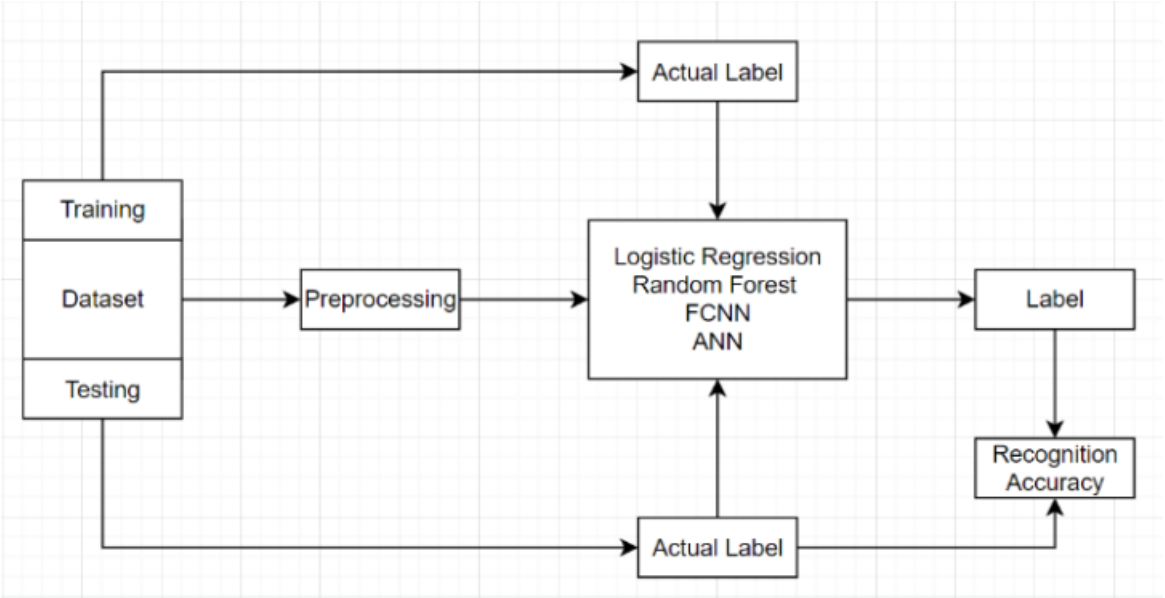


Fig 4.1 Architectural of credit card fraud detection.

5 ALGORITHMS USED FOR CREDIT CARD FRAUD DETECTION

In the context of credit card fraud detection, there are many algorithms that can be used. However, for this project, the selected algorithms are Logistic Regression, Random Forest Classifier, FCNN (Full Connected Neural Network), and ANN (Artificial Neural Network).

5.1 Logistic Regression:

Logistic regression is a supervisor learning classification approach that predicts the probability of a target variable. It is commonly used to categorize data 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.

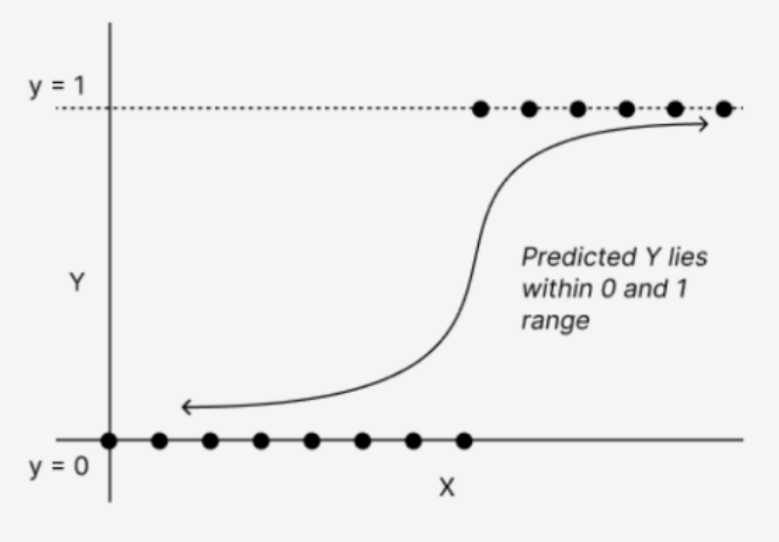


Fig 5.1 Description of the Logistic Regression Algorithm.

5.2 Random Forest Classifier:

Random Forest is a powerful ensemble learning technique that combines multiple decision trees to enhance classification or regression performance. By aggregating the predictions from multiple trees, it helps to smooth out errors and improve the overall accuracy of the model. Random Forest is known for its ability to handle unbalanced and missing data efficiently. However,

it may struggle to make predictions beyond the range of the training data in regression tasks and can overfit noisy datasets. Ultimately, the performance of Random Forest, like any algorithm, should be assessed based on its performance on the specific dataset at hand.

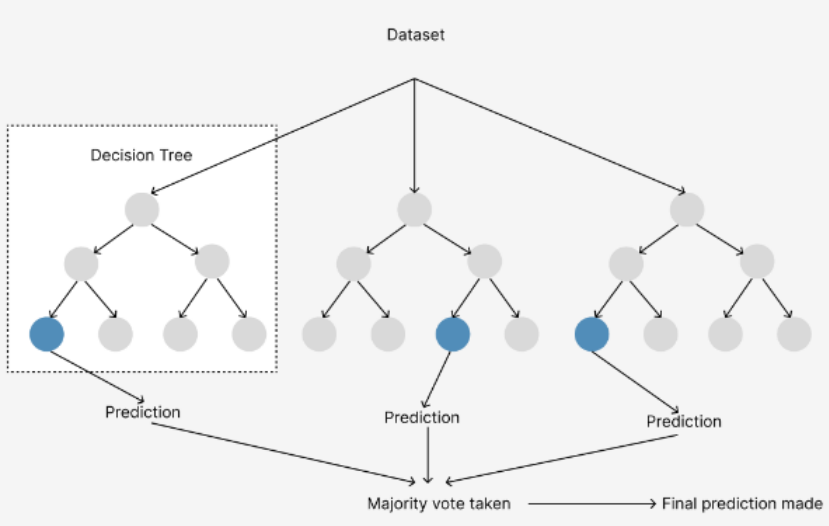


Fig 5.2 Description of the Random Forest Classifier Algorithm.

5.3 Fully Connected Neural Network:

Fully Connected Neural Networks (FCNNs) are widely employed in credit card fraud detection. They consist of interconnected layers of neurons that can capture intricate patterns and relationships within transaction data. FCNNs learn from labeled data to identify fraudulent behavior and make predictions about the likelihood of new transactions being fraudulent. Their ability to uncover complex patterns enhances the accuracy of fraud detection systems.

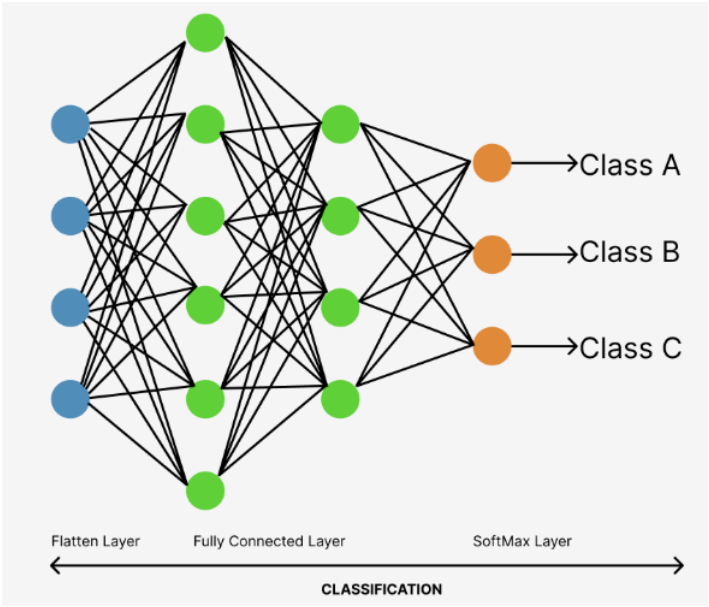


Fig 5.3 Description of the FCNN Algorithm.

5.4 Artificial Neural Network:

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

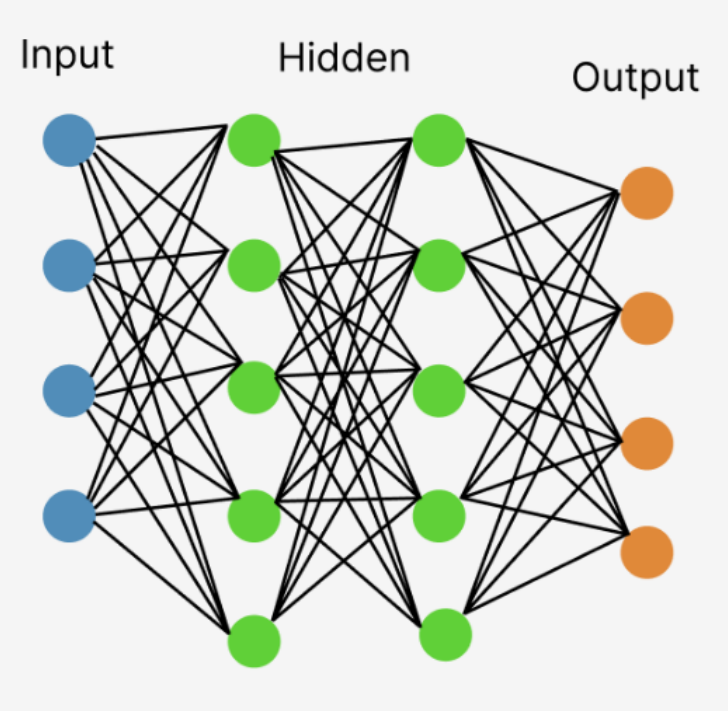


Fig 5.4 Description of the ANN Algorithm.

5.5 Comparison of models based on advantages and limitation:

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

Method	Advantages	Disadvantages
Random Forest Classifier	+Ensemble of decision trees improves classification performance. +Handles high-dimensional data and complex relationships. +Robust to outliers and noise.	+Lack of interpretability compared to individual decision trees. +Longer training time for large datasets. +Can overfit noisy datasets if not properly tuned. +Limited ability to extrapolate beyond the range of training data.
FCNN	+Ability to capture complex patterns and relationships in data. +Adaptability to changing fraud patterns. +Can handle high-dimensional and non-linear data.	+Requires a large amount of training data to avoid overfitting. +Longer training time, especially for deep networks. +Prone to black-box behavior and lack of interpretability. +Sensitive to hyperparameter tuning.
ANN	+Flexibility to model complex relationships and non-linear patterns. +Robust feature learning capabilities. +Can handle high-dimensional data. +Adaptability to evolving fraud patterns.	+Requires substantial computational resources. +Longer training time, especially for deep architectures. +Prone to overfitting if not properly regularized. +Interpretability can be challenging due to network complexity.

6 METRICS USED FOR EVALUATION

6.1 Confusion Matrix:

The Confusion Matrix summarizes the overall performance of a model in credit card fraud detection, providing four types of outcomes: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

6.2 Accuracy:

Accuracy is the percentage of correctly predicted test data.

$$\text{Accuracy} = \frac{\text{Number of Correct predictions}}{\text{Total number of predictions made}}$$

6.3 Recall:

Recall is the ratio of True Positives to the sum of True Positives and False Negatives. The F1 Score aims to strike a balance between recall and precision, indicating the accuracy and robustness of the classifier.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

6.4 Precision:

Precision is a metric that measures the proportion of correctly predicted positive cases out of the total cases predicted as positive. It highlights the model's accuracy in identifying positive cases and is particularly important when the cost of false positives is high, such as in credit card fraud detection. Higher precision indicates fewer false positives and a more reliable classification model.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

6.5 F1 Score:

The F1 Score combines accuracy and recall to evaluate the classifier's performance. Precision is the ratio of True Positives to the sum of True Positives and False Positives.

$$F\text{-}1 = \frac{2 * \text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}}$$

7 EXPLORATORY DATA ANALYSIS

The project utilizes a dataset comprising the transaction history of credit cardholders in Europe. The dataset was obtained from September 2013 and consists of 284,807 rows and 31 columns. Among these, 492 cases of credit card fraud, accounting for 0.2% of the complete dataset, were detected. Genuine transactions make up 99.8% of the data.

The dataset is imbalanced and contains only numeric values, which are the outcomes of the PCA transformation. The 'Time' and 'Amount' columns are not transformed. The 'Class' column serves as the response variable, indicating 0 for genuine or non-fraudulent transactions and 1 for fraudulent transactions. Additionally, there are no null values in the entire dataset, indicating its cleanliness. Table 1 provides further details about the column types.

Table 1

Column	Names Non-Null	Count Dtype
Time	284807	float64
	non-null	
V1 to V28	284807	float64
	non-null	
Amount	284807	float64
	non-null	
Class	284807	int64
	non-null	

The data is divided into a training set and a test set, with 70% of the data allocated to the training set and 30% to the test set. The model is constructed

using six different machine learning approaches. The approach that achieves the highest accuracy, recall score, and precision score will be considered the best fit for this problem.

For each input variable, a histogram is constructed to show its distribution.

8 MODEL CREATION AND TRAINING

8.1 Logistic Regression:

The data is loaded initially with the assistance of the pandas collection. The dataset is split in the next step into values of x and y and sizes of both the values are written. To carry out the procedure of training and testing,

the method named train_test_split() is used. After distributing the data into train and test sets, LogisticRegression() algorithm is implemented.

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

Fig 8.1.1 Logistics Regression() implementation.

The objective function objective is defined for Optuna. It optimizes hyperparameters C (regularization parameter) and max_iter (maximum number of iterations) to create a Logistic Regression model. The model is trained on the training data, and its accuracy is evaluated on the test set.

```
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
```

Fig 8.1.2 Logistics Regression model 's objective function.

An Optuna study is created, and the hyperparameter optimization process is executed for a specified number of trials (3 in this case). The best hyperparameters found by Optuna are printed.

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

study.optimize(objective, n_trials=3)

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

Fig 8.1.3 Optuna study.

Subsequently, the best_model is created using the best hyperparameters, and it is trained on the training data. Predictions are made on the test set using this model.

```
# 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)
```

Fig 8.1.4 Train Logistics Regression using Optuna

The code then calculates various evaluation metrics, including accuracy, precision, recall, and F1 score, using appropriate functions from sklearn.metrics:

```
# Calculate evaluation metrics
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)

# Calculate and display the confusion matrix using seaborn heatmap
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()

# Print the evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 score:", f1)
```

Fig 8.1.5 Calculate evaluation metrics.

This approach gives an accuracy of 99.893%. And lastly, the confusion matrix is created for this approach utilizing confusion_matrix() methodology. It is shown as below:

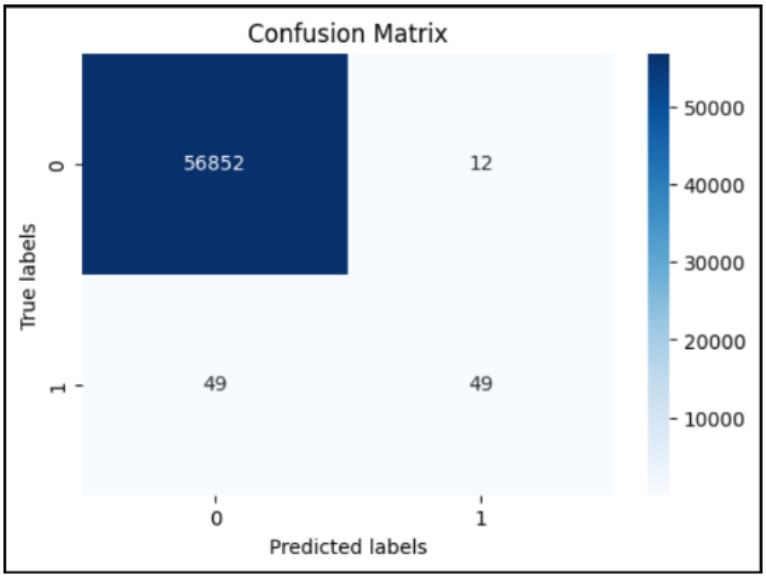


Fig 8.1.6 Logistics Regression 's Confusion Matrix

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

Fig 8.1.7 Logistics Regression's evaluation metrics scores

8.2 Random Forest Classifier:

The data is imported utilizing help of the pandas library.The method named train_test_split() for the procedure of training and testing.

```
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)
```

Fig 8.2.1 Random Forest() implementation.

Define the objective function for the Optuna study. This function is used to optimize the hyperparameters for the Random Forest model. The hyperparameters being optimized are n_estimators, max_depth, min_samples_split, and min_samples_leaf. The function returns the accuracy of the model on the test set.

```python
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
```

Fig 8.2.2 Random Forest model 's objective function.

Create an Optuna study and perform the hyperparameter optimization:

```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)
```

Fig 8.2.3 Optuna study.

After that, we train the Random Forest model using the best hyperparameters.

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

Fig 8.2.4 Train Logistics Regression using Optuna

This approach gives an accuracy of 99.953%. And lastly, the confusion matrix is created for this approach utilizing confusion_matrix() methodology. It is shown as below:

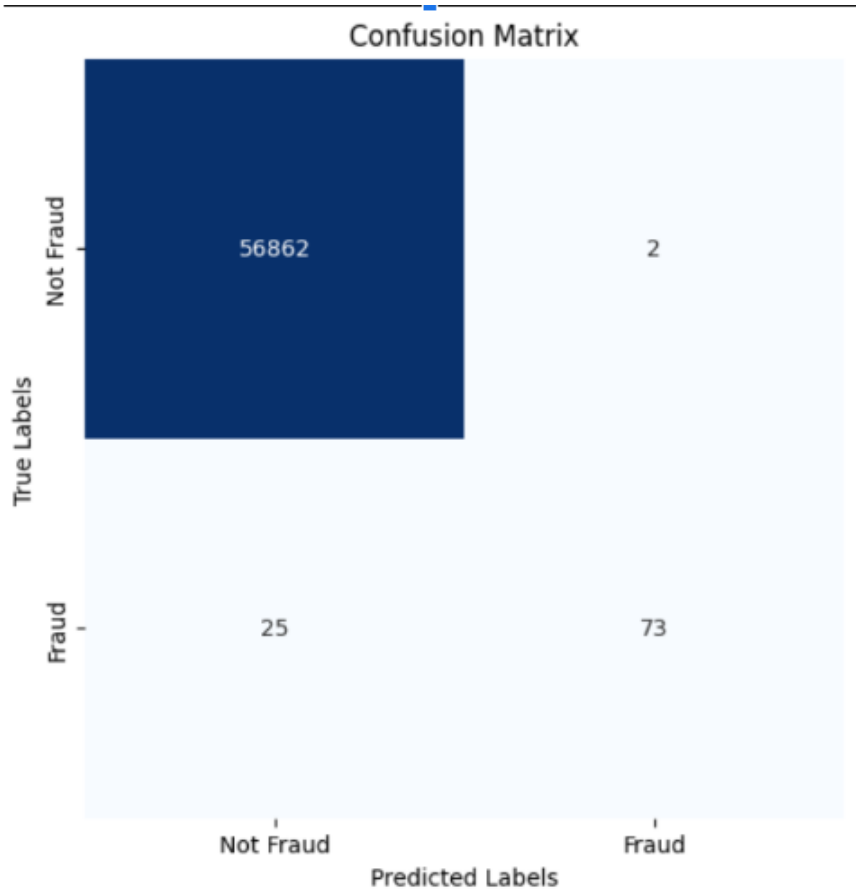


Fig 8.2.5 Random Forest 's Confusion Matrix

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

Fig 8.2.6 Random Forest's evaluation metrics scores

8.3 Fully Connected Neural Network:

The code splits the data into features (X) and the target variable (y) using train_test_split from scikit-learn. It then applies Min-Max scaling to standardize the features between 0 and 1.

```
# Preparing the data
X = df.drop(['Class'], axis=1)
y = df['Class']

scaler = MinMaxScaler()
X = scaler.fit_transform(X)
```

Fig 8.3.1 Preprocessing data.

```
# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Fig 8.3.2 Split the dataset

The objective function is defined for hyperparameter optimization using Optuna. It takes a trial object as input, which represents a single evaluation of the objective function with a set of hyperparameters. The function defines the search space for the hyperparameters, creates a Fully Connected Neural Network (FCNN) model with the suggested hyperparameters, and trains the model on the training data. The accuracy is calculated as the evaluation metric and returned to Optuna for optimization:

```
# 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
```

Fig 8.3.3 Objective function.

The code sets up the Optuna study and runs the hyperparameter optimization using the study.optimize() function. After the optimization process, the best hyperparameters are obtained from study.best_params and printed:

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

# Printing the Best Hyperparameters
print('Optimal Value:', study.best_value)
print('Optimal Hyperparameters:', study.best_params)
```

```
[I 2023-07-17 00:49:53,290] A new study created in memory with name: no-name-8c57e67a-d0f3-41
1781/1781 [==============================] - 4s 2ms/step
[I 2023-07-17 00:55:30,884] Trial 0 finished with value: 0.9993504441557529 and parameters: {
1781/1781 [==============================] - 2s 1ms/step
[I 2023-07-17 01:07:26,978] Trial 1 finished with value: 0.999385555282469 and parameters: {'
1781/1781 [==============================] - 3s 1ms/step
[I 2023-07-17 01:13:54,774] Trial 2 finished with value: 0.999385555282469 and parameters: {'
 Optimal Value: 0.999385555282469
Optimal Hyperparameters: {'n_units_1': 61, 'n_units_2': 49, 'epochs': 88, 'batch_size': 83}
```

Fig 8.3.4 Optuna study.

The best hyperparameters obtained from the optimization are used to create a new Fully Connected Neural Network (FCNN) model (model) with the best architecture. Then, the model is compiled with the adam optimizer and trained on the entire training data with the fit() method:

```
#  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'))

#  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)
```

```
<keras.callbacks.History at 0x798168252080>
```

```
# Predict labels for the test set using the best model
y_pred_proba = model.predict(X_test)
y_pred = np.where(y_pred_proba > 0.5, 1, 0)
```

```
1781/1781 [==============================] - 3s 1ms/step
```

Fig 8.3.5 Creating FCNN with Optuna.

```
# Calculate the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Print the confusion matrix in a pretty way using seaborn
plt.figure(figsize=(6, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False, square=True,
            xticklabels=['Not Fraud', 'Fraud'], yticklabels=['Not Fraud', 'Fraud'])
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

Fig 8.3.6 Predict labels for the test set using FCNN with Optuna.

```
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)
```

Fig 8.3.7 Calculate evaluation metrics.

This approach gives an accuracy of 99.939%. And lastly, the confusion matrix is created for this approach utilizing confusion_matrix() methodology. It is shown as below:

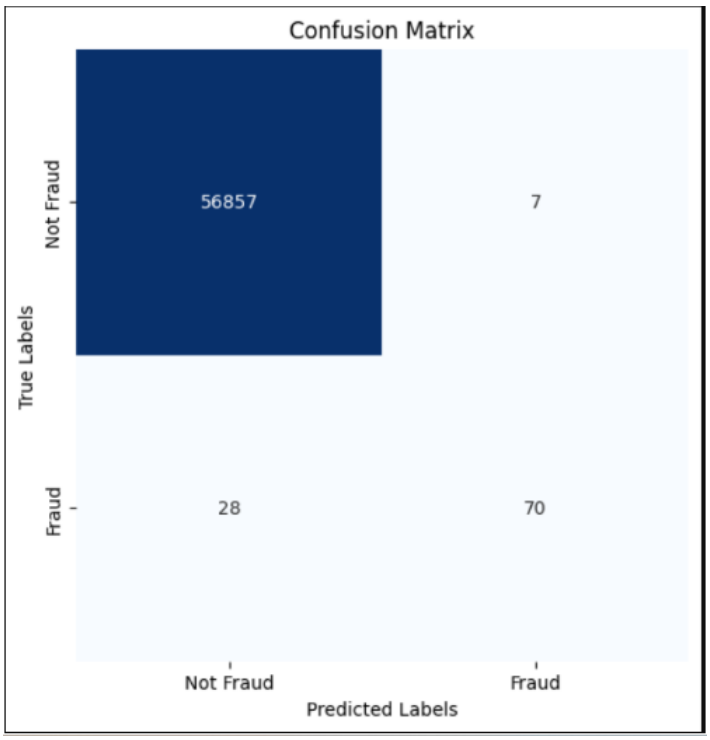


Fig 8.3.8 FCNN 's Confusion Matrix

```
Accuracy: 0.999385555282469
Precision: 0.9090909090909091
Recall: 0.7142857142857143
F1 score: 0.8
```

Fig 8.3.9 FCNN's evaluation metrics scores

8.4 Artificial Neural Network:

The code splits the data into features (X) and the target variable (y) using train_test_split from scikit-learn. It also applies Min-Max scaling to the data using MinMaxScaler to ensure that all features are within the same range:

```
# Preparing the data
X_train, X_test, y_train, y_test = train_test_split(
    data.drop('Class', axis=1),data['Class'], test_size=0.2, random_state=42)

scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Fig 8.4.1 ANN's data preparation.

The objective function is defined for hyperparameter optimization using Optuna. It takes a trial object as input, which represents a single evaluation of the objective function with a set of hyperparameters. The function defines the search space for the hyperparameters, creates an ANN model with the

suggested hyperparameters, and trains the model on the training data. The accuracy is calculated as the evaluation metric and returned to Optuna for optimization.

```
# Defining the objected 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
```

Fig 8.4.2 Objected Funtion.

The code sets up the Optuna study and runs the hyperparameter optimization using the study.optimize() function:

```
# Hyperparameter Optimization using Optuna
study = optuna.create_study(direction='maximize')


study.optimize(objective, n_trials=3)


print(' Giá trị tối ưu:', study.best_value)
print('Siêu tham số tối ưu:', study.best_params)
```

```
[I 2023-07-12 07:40:11,544] A new study created in memory with name: no-name-90740eb2-7c52-
1781/1781 [==============================] - 2s 1ms/step
[I 2023-07-12 07:46:13,127] Trial 0 finished with value: 0.9994557775359011 and parameters:
1781/1781 [==============================] - 3s 2ms/step
[I 2023-07-12 07:50:21,712] Trial 1 finished with value: 0.999420666409185 and parameters:
1781/1781 [==============================] - 3s 2ms/step
[I 2023-07-12 07:55:38,093] Trial 2 finished with value: 0.9994557775359011 and parameters:
 Giá trị tối ưu: 0.9994557775359011
Siêu tham số tối ưu: {'n_units_1': 90, 'n_units_2': 55, 'epochs': 86, 'batch_size': 111}
```

Fig 8.4.3 Optuna study.

After the optimization process, the best hyperparameters are obtained from study.best_params. A new ANN model (best_model) is created with the best hyperparameters, and it is trained on the entire training data:

```
# Creating and Training the 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)
```

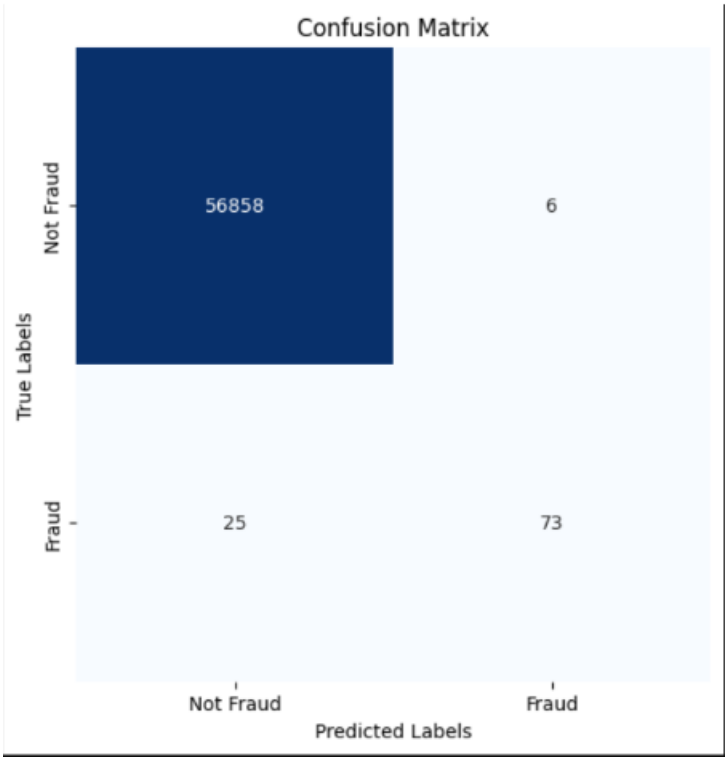
```
<keras.callbacks.History at 0x7f7d2ff8e020>
```

Fig 8.3.4 Creating ANN with Optuna.

```
# Evaluating the Best Model
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)
# Calculate the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Print the confusion matrix
plt.figure(figsize=(6, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False, square=True,
            xticklabels=['Not Fraud', 'Fraud'], yticklabels=['Not Fraud', 'Fraud'])
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()

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)
```

Fig 8.4.5 Calculate evaluation metrics.

This approach gives an accuracy of 99.946%. And lastly, the confusion matrix is created for this approach utilizing confusion_matrix() methodology. It is shown as below:



ANN 's Confusion Matrix

```
Accuracy: 0.9994557775359011
Precision: 0.9240506329113924
Recall: 0.7448979591836735
F1 score: 0.824858757062147
```

ANN's evaluation metrics scores.

8.5 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

Fully Connected Neural Network	56857	70	28	7	0.00061444471 7531	0.99938555528 2469	0.909090909090 9091	0.71428571428 57143	0.8
Artificial Neural Network	56858	73	25	6	0.00054422246 40989	0.99945577753 59011	0.92405063291 13924	0.74489795918 36735	0.82485875706 2147

9 CONCLUSION AND FUTURE SCOPE

The results clearly demonstrate that machine learning is well-suited for credit card fraud prediction.

As per the experimental results it has been observed that 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 (Prediction: Legitimate/Non-Fraud but Actual:Fraudulent), which is a serious one, is minimum for Random Forest Classifier followed by ANN. Based on all the observations Random Forest Classifier comes out to be the best overall algorithm as per the current result analysis.

In traditional two-class classification problems, machine learning algorithms aim to minimize misclassifications by using metrics like Accuracy and F1-score to measure model performance. However, in certain cases, it may be more preferable to accept some misclassifications in exchange for lower overall costs. When the costs of misclassifications vary across samples, a cost-sensitive, example-dependent learning technique can be employed. By incorporating misclassification costs for false negatives (FN) and false positives (FP), we can customize the model to better suit specific cost considerations.

A cost function will be used to evaluate the Machine Learning model's performance with the given data. This function measures the discrepancy between projected and expected values, enabling us to assess the model's pre-

diction accuracy. By minimizing the cost function, we can fine-tune the model and improve its predictive capability.

After experimenting with various algorithms, including both deep learning (such as Artificial Neural Networks - ANN) and machine learning (such as Random Forest), we discovered that for straightforward classification tasks with clear relationships between input and output data, machine learning, specifically Random Forest, tends to be the optimal choice. (this case is a prime example).

On the other hand, for more complex problems that involve higher requirements and intricate data processing, deep learning, particularly ANN, emerges as the most suitable option. This is due to the flexibility and adaptability of ANN in handling complex patterns and non-linear relationships within the data.

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.

10 REFERENCES

References

[1] **Neural Networks vs. Random Forests – Does it always have to be Deep Learning?** by Prof. Dr. Peter Roßbach. Link*: https://blog.frankfurt-school.de/wp-content/uploads/2018/10/Neural-Networks-vs-Random-Forests.pdf*

[2] **Credit Card Fraud Detection Using Machine Learning Approach**, by Kanal Bhadresh Soni, Madhuri Chopade, Rahul Vaghela. Link: *https://www.academia.edu/75231522/Credit_Card_Fraud_Detection_Using_Machine_Learning_Approach?fbclid=IwAR3aaC-UMwW8TPR7B1bN-tlvNMe-EFU0iddG8YNjgGQgNwaD7I0UgUQ7Bvc*

[3] **Credit card transaction data analysis and performance evaluation of machine learning algorithms for credit card fraud detection** by Vivek Ghai and Sandeep Singh Kang. Link:

https://www.academia.edu/90815750/Credit_ card_ transaction_ data _ analysis_ and_ performance_ evaluation_ of_ machine_ learning_ algori thms_ for_ credit_ card_ fraud_ detection?fbclid=IwAR2sSpfkINsdLWNR pclqe8atIQWVwr6rnqGMcLPf9dzqIAAoD5sMwHkRsZo

[4] **Temporal Convolutional Neural Network for theClassification of Satellite Image Time Series** by Charlotte Pelletier, Geoffrey I. Webb, Senior Fellow, IEEE, and Franc¸ois Petitjean. Link: *https://www.researchgate.net/publication/331525817_ Temporal_ Convolu tional_ Neural_ Network_ for_ the_ Classification_ of_ Satellite_ Image_ Time_ Series*

[5] **Speech and Language Processing: Chapter 5. Logistic Regression** by Daniel Jurafsky and James H. Martin. Link: *https://web.stanford.edu/*$\tilde{j}$*urafsky/slp3/5.pdf*

[6] **Random Forests for Regression and Classification** by Adele Cutler. Link: *https://www.usu.edu/math/adele/randomforests/ovronnaz.pdf*