Predicting Fraudulent Transactions: A Comparative Study of Machine Learning Models including Logistic Regression, Decision Trees, Random Forests, XGBoost, and CNN

Jingxuan Lyu[a,1] **and Tiantian Zhang**[b,2]

[a]*Beijing Haidian Kaiwen Academy*
[b]*Beijing Haidian Kaiwen Academy*

Abstract—Fraudulent transactions have a detrimental impact on financial security, requiring advanced detection techniques. This study evaluates the effectiveness of five machine learning models—Logistic Regression, Decision Trees, Random Forests, XGBoost, and Convolutional Neural Networks—in detecting fraudulent transactions. A real-world dataset was preprocessed, where we use feature engineering, scaling, and class balancing to enhance model performance. Then, each model was trained and tested on structured transaction data, with performance compared based on accuracy, precision, recall, and F1-score. Results indicate that tree-based models, particularly Random Forest and XGBoost, achieve highest accuracy and recall, minimizing false negatives while maintaining a low false positive rate. Additionally, we implemented a majority-vote ensemble approach to further improves classification performance. The study highlights the effectiveness of ensemble learning and tree-based models in fraud detection, while also exploring the feasibility of neural networks for transaction-based classification. The findings will contribute to the development of more reliable fraud detection systems in digital finance.

Keywords— *Machine Learning, Fraud Transaction, Convolutional Neutron Network*

1. Introduction

As a high school student, I just reached 18 years old and opened my first bank account. Responding to my parents' request to use this account to pay my tuition fee, an amazing thing happened: the 180,000 yuan tuition fee and my bank account was frozen and recognized as a fraud transaction by the bank. Such an unexpected accident caused many inconveniences. Inspired by this experience, I started to think about why this happened.

Fraudulent Transactions have a long history, and it evolved into many ways during past several decades. Traditionally, it could be theft and forgery, and when it comes to the digital age, many more sophisticated tactics appeared. Most of them exploit vulnerabilities in e-commerce, mobile payments, and digital platforms. The global digital payment fraud market reached $50 billion by the end of 2024, where mobile devices account for 74% of fraud incidents. At the same time, new threats like account takeover fraud have seen a 127% increase between 2020 and 2023, which result in estimated losses of $8.1 billion globally. (Ali et al., 2024) The causes of fraud are multifaceted, and the most significant one involves technological advancements, which provide new tools for exploitation, data breaches that expose sensitive information, weak security measures, and human factors such as susceptibility to phishing. Synthetic identity fraud, which blends legitimate and fictitious information to bypass verification systems, now accounts for 23% of credit card losses in the banking sector. As fraud continues to pose a severe global challenge, more effective detection methods are needed to safeguard financial systems. (Bojilov, 2024)

Traditional rule-based systems is insufficient to recognize newly evolved tactics of fraudsters, and these systems typically can only detect 65-70% of the fraudulent transactions. Moreover, it generating false positive rates exceeding 30%. (Balcıoğlu, 2024) This has led to many computer scientists to develop advanced machine learning models, which can learn from historical data and recognize complex patterns of fraudulent behavior. With the rise of real-time payment systems, transactions can occur in milliseconds, so that these advanced models are crucial for making immediate risk assessments. This paper aims to explore the performance of several machine learning models—Logistic Regression, Decision Trees, Random Forests, XGBoost, and CNN—in predicting fraudulent transactions. These models were selected for their varying complexity, from linear models to complex neuron networks, and will be evaluated using a real-world dataset. By comparing their accuracy and effectiveness, we aim to provide some insights into which model or combination of models can offer the most reliable solution for fraud detection, so contributes to the enhancement of security in digital financial systems.

2. Literature Review

Using various machine learning models to idenfify fraud detection remains a critical area of research, and current studies have predominantly focused on a few key algorithms: logistic regression, decision trees, and XGBoost. These models have shown frequently due to their simplicity, interpretability, and relatively high performance with structured data. For instance, logistic regression is widely used for classification tasks because it is good at finding relationship between independent variables and the probability of an outcome (Kannan et al., 2020). In addition, decision trees and their ensemble variants, like Random Forests and XGBoost, have been favored as their have the ability to capture non-linear relationships and interactions between features, which can offer improved accuracy over traditional methods (Chen & Guestrin, 2016). Despite the success of these models, there has been limited exploration into the application of neural networks, particularly Convolutional Neural Networks (CNNs), for fraud detection. Neural networks have revolutionized fields like image recognition and natural language processing due to their ability to detect complex, non-linear patterns in high-dimensional data. However, their adoption in fraud detection has been constrained by several factors.

Financial transaction data is typically tabular, lacking the spatial hierarchies that CNNs are designed to capture. This structural mismatch poses challenges in effectively applying CNNs to such data. Additionally, studies have shown that traditional models like Gradient Boosting Decision Trees often outperform deep neural networks on tabular data, indicating that efficient modeling of such data using deep learning remains an open research problem.

Nevertheless, recent research has begun to explore the feasibility of using CNNs for fraud detection. A CNN-based framework was developed to capture intrinsic patterns of fraudulent behaviors from labeled data, demonstrating the potential of CNNs in this domain. (Das, 2024)

Incorporating a majority-vote ensemble method in fraud detection can significantly enhance the accuracy of predictive models. This approach combines predictions from multiple classifiers, allowing the ensemble to leverage the strengths of individual models while mitigating their weaknesses. A study proposed a majority vote ensemble classifier specifically designed for accurate detection of credit card frauds. This method integrates various classifiers to improve detection performance. (Sudha & Akila, 2021) Another research introduced a soft voting ensemble learning approach for detecting credit card fraud in imbalanced datasets. By combining predictions from multiple classifiers, this method addresses class imbalance and enhances detection accuracy. (Forough, 2021)

In our research, we focus on enhancing fraud detection by combining multiple models, including Convolutional Neural Networks (CNNs), into a majority-vote ensemble. This method leverages the strengths of different models to improve detection accuracy. By in-

tegrating CNNs, our ensemble aims to identify fraudulent activities more effectively. This approach aligns with recent studies that have successfully utilized CNNs within ensemble frameworks for fraud detection.

3. Abstract

Fraudulent transactions have a detrimental impact on financial security, requiring advanced detection techniques. This study evaluates the effectiveness of five machine learning models—Logistic Regression, Decision Trees, Random Forests, XGBoost, and Convolutional Neural Networks—in detecting fraudulent transactions. A real-world dataset was preprocessed, where we use feature engineering, scaling, and class balancing to enhance model performance. Then, each model was trained and tested on structured transaction data, with performance compared based on accuracy, precision, recall, and F1-score. Results indicate that tree-based models, particularly Random Forest and XGBoost, achieve highest accuracy and recall, minimizing false negatives while maintaining a low false positive rate. Additionally, we implemented a majority-vote ensemble approach to further improves classification performance. The study highlights the effectiveness of ensemble learning and tree-based models in fraud detection, while also exploring the feasibility of neural networks for transaction-based classification. The findings will contribute to the development of more reliable fraud detection systems in digital finance.

4. Methodology

4.1. Data Description and Preprocessing

The dataset used in this study comes from a transactional fraud detection context. It contains various features related to online financial transactions, with the target variable being isFraud, which indicates whether a given transaction is fraudulent (1) or not (0). The dataset was loaded from DataSet.csv, and its key features are as follows:

Table 1. Dataset Description

Features	Descriptions
step	Represents a unit of time, where 1 step equals 1 hour.
type	The type of online transaction (e.g., payment, transfer, cash-in).
amount	The monetary amount of the transaction.
nameOrig	The identifier of the customer initiating the transaction.
oldbalanceOrg	The account balance of the originating customer before the transaction.
newbalanceOrig	The account balance of the originating customer after the transaction.
nameDest	The identifier of the recipient of the transaction.
oldbalanceDest	The balance of the recipient's account before the transaction.
newbalanceDest	The balance of the recipient's account after the transaction.
isFraud	Indicates whether the transaction is fraudulent (1) or not (0).

4.1.1. Data Cleaning

Initial data preprocessing involved removing duplicate rows using `df.drop_duplicates()` to ensure consistency. The columns isFlaggedFraud, nameDest, and nameOrig were then excluded from the dataset. The nameDest and nameOrig features were removed as they contained sensitive information and were not deemed necessary for the analysis, while isFlaggedFraud was discarded as it was a redundant indicator of fraud.

```
df = pd.read_csv(file_path).drop_duplicates()
df = df.drop(columns=['isFlaggedFraud', 'nameDest', '
    ↪ nameOrig'])
```

Code 1. Data Cleaning.

4.1.2. Feature Engineering and Encoding

Categorical features, particularly type, which represents the transaction type, were transformed using one-hot encoding. This method created binary columns for each unique transaction type (e.g., payment, transfer). The resulting one-hot encoded columns were added to the dataset, and the original type column was dropped.

```
encoder = OneHotEncoder(sparse_output=False)
one_hot_encoded = encoder.fit_transform(balanced_df[['
    ↪ type']])
one_hot_df = pd.DataFrame(one_hot_encoded, columns=
    ↪ encoder.get_feature_names_out(['type']))
balanced_df = pd.concat([balanced_df, one_hot_df], axis=
    ↪ 1).drop(columns=['type'])
```

Code 2. Feature Engineering and Encoding.

4.1.3. Feature Selection and Scaling

To reduce dimensionality and focus on the most relevant features, Recursive Feature Elimination (RFE) was applied using a logistic regression estimator. This method recursively eliminates the least important features and ranks the remaining ones. In this case, the top five features, identified by RFE, were retained for the model, ensuring only the most predictive variables were used in training.

Feature scaling was performed using the MinMaxScaler to normalize all feature values between 0 and 1. This step is crucial for models that use regularization, such as logistic regression, as it ensures that all features are on a comparable scale and contribute equally to the model's performance.

```
def feature_selection(x, y, num_features=5):
    scaler = MinMaxScaler()
    x_scaled = scaler.fit_transform(x)

    rfe = RFE(estimator=LogisticRegression(),
     ↪ n_features_to_select=num_features)
    x_selected = rfe.fit_transform(x_scaled, y)
    return x_selected
```

Code 3. Feature Selection and Scaling.

4.1.4. Dataset Balancing

The sampled non-fraudulent transactions were concatenated with the full set of fraudulent transactions to create a new balanced dataset. This new dataset contained an ratio of 1:10 of fraudulent and non-fraudulent transactions, alleviating the class imbalance that may affect model performance.

```
df_1 = df[df['isFraud'] == 1]
num_fraud = len(df_1)
df_0 = df[df['isFraud'] == 0].sample(n=num_fraud * 10,
    ↪ random_state=42)
balanced_df = pd.concat([df_0, df_1]).sample(frac=1,
    ↪ random_state=42).reset_index(drop=True)
print(balanced_df)
```

Code 4. Dataset Balancing.

4.1.5. Train-Test Split

The dataset was divided into training and testing subsets using an 80-20 split. Stratified sampling was employed to maintain the distribution of the target variable across both the training and testing

sets, ensuring that the proportion of fraudulent and non-fraudulent transactions was consistent in each subset.

```
x = df.drop(columns=['isFraud'])
y = df['isFraud'].to_numpy()
x_selected = feature_selection(x, y)
x_train, x_test, y_train, y_test = train_test_split(
    ↪ x_selected, y, test_size=0.2, random_state=42)
```

Code 5. Train-Test Spllit.

4.2. Logistic Regression

Logistic Regression is a fundamental statistical model used primarily for binary classification tasks, such as predicting whether a transaction is fraudulent or not. Unlike linear regression, which predicts continuous values, Logistic Regression predicts the probability of an event occurring by fitting a logistic (sigmoid) function to the input features. This allows the model to output values between 0 and 1, representing the likelihood that a given transaction is fraudulent. At its core, Logistic Regression estimates the relationship between a dependent variable—such as whether a transaction is fraudulent (denoted as isFraud)—and independent variables, which could include transaction details like amount, type, and time. The model does this by constructing a linear combination of these features and passing the result through a sigmoid function. Specifically, the linear model is given by equation 1.

$$z = (\sum_{i=1}^{m} w_i x_i) + b = w \cdot \mathbf{X} + b \tag{1}$$

The sigmoid function is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

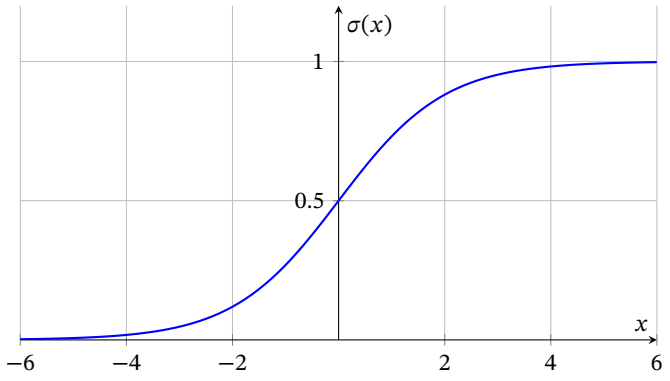


Figure 1. Sigmoid Function Plot

To train the model, Logistic Regression uses a loss function called log-loss (or cross-entropy), which penalizes incorrect predictions. The goal is to minimize this loss, refining the model's ability to accurately classify transactions based on the input data.

```
log_reg = LogisticRegression(C=10, solver='lbfgs')
log_reg, _ = evaluate_model(log_reg, x_train, x_test,
    ↪ y_train, y_test, "Logistic Regression")
```

Code 6. Logistic Regression Hyper-Parameters.

Initially, the log-loss is relatively high, indicating that the model's predictions are far from the true values. However, as the number of iterations increases, the log-loss decreases fastly, demonstrating rapid learning process and improving model performance during early stages of training. After around 100 iterations, the log-loss begins to plateau, suggesting that the model has converged to a solution and further iterations result in minimal improvements. This behavior is characteristic of the optimization process in logistic regression, where the model quickly reduces errors but may experience diminishing returns as it approaches an optimal set of parameters.

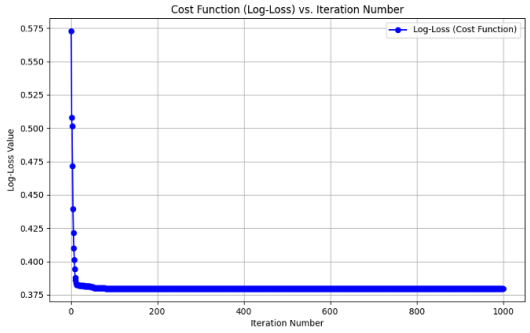


Figure 2. Cost Function vs. Iteration Number.

The logistic regression model demonstrates strong performance, achieving an accuracy of approximately 88.7%. Precision for the positive class is 0.901, meaning about 90.1% of the model's positive predictions are correct, while recall is 0.868, indicating that 86.8% of all actual positive instances were correctly identified. The F1-score of 0.884 reflects a balanced performance between precision and recall. The confusion matrix shows 1487 true negatives, 1426 true positives, 156 false positives, and 217 false negatives. These results suggest that the model is effective at detecting positive cases with a low false positive rate, while also maintaining a solid recall rate. Although performance is strong, there may still be opportunities to improve, particularly in reducing false negatives.

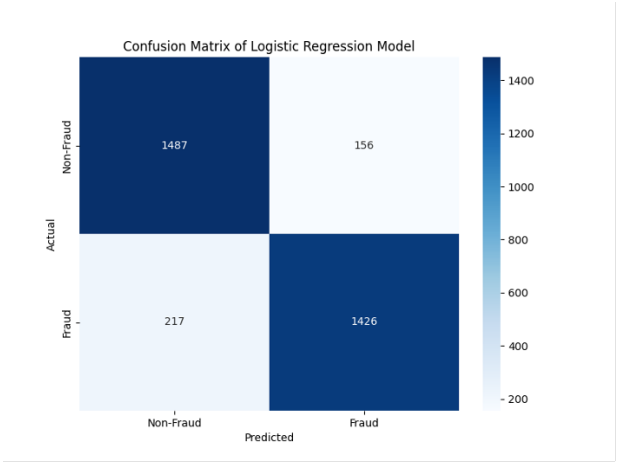


Figure 3. LR Confusion Matrix

4.3. Decision Tree

A Decision Tree is a supervised learning algorithm used for classification and regression tasks. It recursively splits the dataset based on feature values, forming a tree-like structure where each internal node represents a decision rule. Decision Trees are interpretable and efficient, making them a popular choice for fraud detection.

In this study, a decision tree classifier was implemented to identify fraudulent transactions, utilizing features selected through Recursive Feature Elimination (RFE). The following code demonstrates how a Decision Tree Classifier is implemented using scikit-learn and evaluated using a custom function (`evaluate_model`).

```
dt_model = DecisionTreeClassifier(random_state=42)
dt_model, _ = evaluate_model(dt_model, x_train, x_test,
    ↪ y_train, y_test, "Decision Tree")
```

Code 7. Decision Tree Hyper-Parameters.

The decision tree analysis revealed that `oldbalanceOrg`, `oldbalanceDest`, `type_TRANSFER`, and `amount` were among the most influential features for fraud detection. Transactions with lower `oldbalanceOrg` values exhibited a higher probability of being fraudulent. Furthermore,transaction type played a crucial role, with `TRANSFER` transactions being strongly associated with fraudulent activity. By learning optimal decision boundaries, the model effectively distinguished between fraudulent and non-fraudulent transactions.

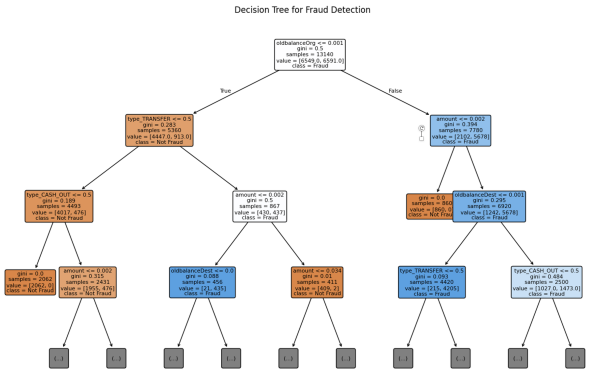


Figure 4. Decision Tree for Fraud Detection

The evaluation of the decision tree model demonstrated high performance, achieving an accuracy of 98.42%. The fraud class was detected with a precision of 98% and a recall of 99%, leading to an F1-score of 98%.

The Confusion Matrix indicates that the model correctly classified 1632 non-fraudulent transactions and 1602 fraudulent transactions. It misclassified 32 non-fraudulent transactions as fraud (false positives) and 20 fraudulent transactions as non-fraud (false negatives). The low false negative rate is particularly important in fraud detection, as failing to detect fraudulent transactions can lead to significant financial loss. However, the presence of false positives suggests that further refinement, possibly through ensemble methods like Random Forests or XGBoost, could enhance model robustness and minimize unnecessary fraud alerts.

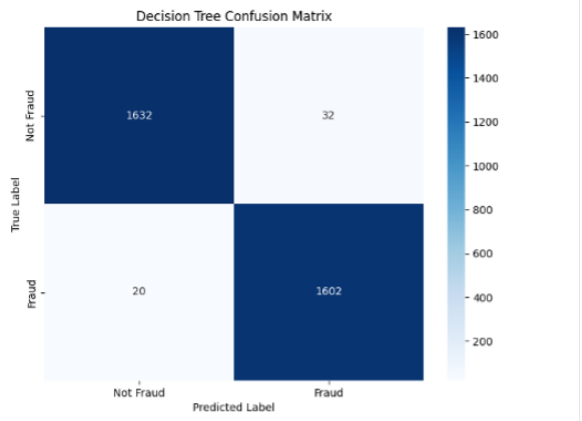


Figure 5. DT Confusion Matrix

4.4. Random Forest

Random Forest, an ensemble learning method, was implemented to classify fraudulent transactions by leveraging multiple decision trees. This approach enhances predictive accuracy while reducing the risk of overfitting. The model was trained using 100 decision trees (`n_estimators` = 100) with a fixed random state for reproducibility.

```
rf_model = RandomForestClassifier(n_estimators=100,
    ↪ random_state=42)
rf_model, _ = evaluate_model(rf_model, x_train, x_test,
    ↪ y_train, y_test, "Random Forest")
```

Code 8. Random Forest Hyper-Parameters.

The model achieved a training accuracy of 99.83%, indicating excellent performance on the training data. It correctly predicted nearly all transactions, demonstrating that the Random Forest classifier is well-suited for capturing patterns in the dataset. The test accuracy of 98.84% further confirms the model's strong generalization capability.

According to the confusion matrix, the model correctly identified 1566 fraudulent transactions and 1547 legitimate transactions. However, it also flagged 96 legitimate transactions as fraudulent (false positives) and missed 77 fraudulent transactions (false negatives).

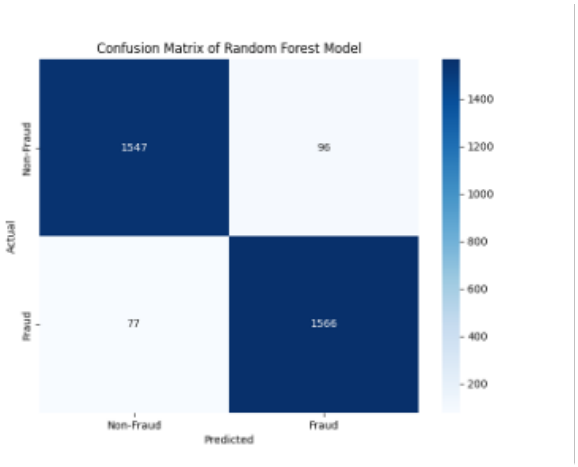


Figure 6. RF Confusion Matrix

The precision of the model is 0.93, indicating that when the model predicts a transaction as fraudulent, it is correct 98% of the time. This high precision reduces the occurrence of false positives, making the model reliable for avoiding mislabeling legitimate transactions. To gain interpretability, feature importance scores were extracted from the trained model.

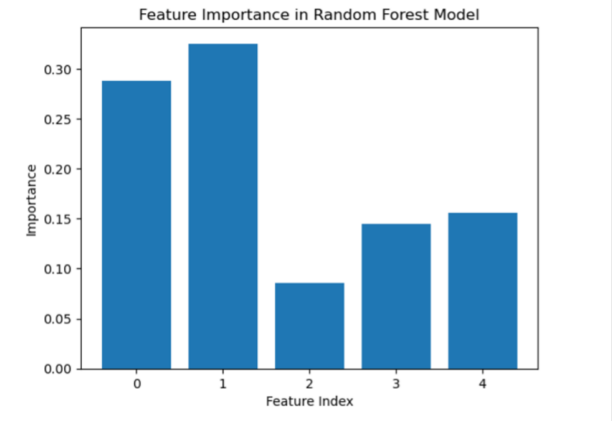


Figure 7. Important Features

4.5. XGBoost

XGBoost (Extreme Gradient Boosting) is a highly efficient and scalable machine learning algorithm that has become one of the most popular tools for solving predictive modeling problems, particularly in structured data tasks. XGBoost is an implementation of gradient boosting that focuses on improving the performance and speed of the traditional gradient boosting framework. It builds an ensemble of decision trees in a sequential manner, where each tree corrects the errors of the previous one, allowing the model to capture complex patterns in the data. The following code demonstrates how to implement an XGBoost Classifier using the xgboost library and evaluate it using a custom function.

```
xgb_model = xgb.XGBClassifier(eval_metric='logloss')
xgb_model, _ = evaluate_model(xgb_model, x_train, x_test,
    ↪ y_train, y_test, "XGBoost")
```

Code 9. XGBoost Hyper-Parameters.

The performance of the XGBoost model in predicting fraudulent transactions is evaluated through several key metrics, namely accuracy, log loss, and the confusion matrix. The model achieved an accuracy of 99.07%, demonstrating that it correctly predicted the class for the vast majority of instances. Given that the data has been preprocessed to balance the number of fraudulent and normal transactions, accuracy provides a meaningful indicator of the model's performance without the risk of being skewed by class imbalance.

Log loss is also implemented in XGBoost. Unlike accuracy, which simply measures the proportion of correct predictions, log loss takes into account the uncertainty of the predictions. The log loss value of 0.1035 indicates that the model's predicted probabilities are relatively close to the actual outcomes. A lower log loss value means that the model is making confident and accurate predictions, with its estimated probabilities aligning well with the true labels.

The confusion matrix provides deeper insight into the model's ability to correctly classify both classes. The model identified 1,569 true negatives (legitimate transactions) and 1,588 true positives (fraudulent transactions), with only 74 false positives (legitimate transactions incorrectly flagged as fraud) and 55 false negatives (fraudulent transactions missed). This suggests that the model is performing well in detecting fraudulent transactions, with a low rate of false negatives.

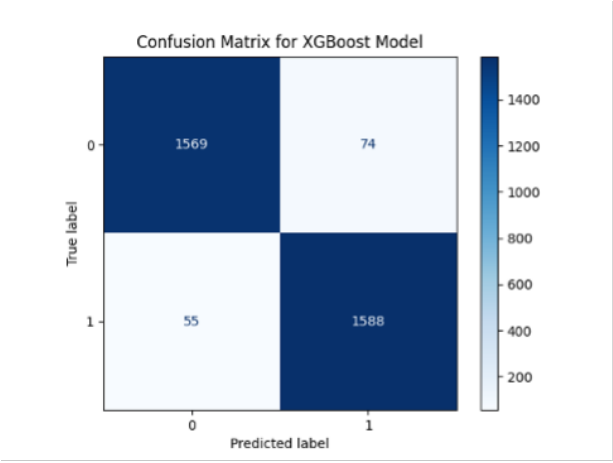


Figure 8. XGB Confusion Matrix

4.6. Convolutional Neural Network

While CNNs are traditionally used for image processing, they can be effectively applied to tabular datasets by reshaping the input into a grid-like structure. This study implemented a 1D CNN model to leverage spatial relationships among features and detect fraudulent transactions.

The dataset was preprocessed to ensure compatibility with the CNN model. First, the selected numerical features were normalized using MinMaxScaler to bring them into a uniform range between 0 and 1. Since CNNs typically process structured data in multi-dimensional formats, the input features were reshaped into a three-dimensional tensor, with the number of samples, time steps (set to one for tabular data), and feature dimensions. This transformation allowed the convolutional layers to extract local patterns across transaction features effectively.

```
X_train_reshaped = x_train.reshape(-1, 5, 1)
X_test_reshaped = x_test.reshape(-1, 5, 1)
```

Code 10. Reshaping for CNN.

The CNN architecture consisted of an input layer followed by one-dimensional convolutional layers with ReLU (Rectified Linear Unit) activation functions. The convolutional layers applied multiple filters to detect relevant feature interactions, followed by max-pooling layers to reduce dimensionality and prevent overfitting. A flattening layer converted the pooled feature maps into a dense representation, which was then processed by fully connected layers. The final output layer used a sigmoid activation function to produce a probability score, indicating the likelihood of a transaction being fraudulent.

```
model = keras.Sequential([
    layers.Conv1D(filters=64, kernel_size=2, activation='
    ↪ relu', input_shape=(5, 1)),
    layers.Dropout(0.3),
    layers.Conv1D(filters=32, kernel_size=2, activation='
    ↪ relu'),
    layers.Dropout(0.3),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(1, activation='sigmoid')
])
```

Code 11. CNN Structure.

The model was trained using the binary-cross-entropy loss function and optimized using the Adam optimizer. To enhance generalization, dropout regularization was applied to the fully connected layers, mitigating overfitting risks. The dataset was split into training and validation sets using an 80-20 ratio, with early stopping implemented to halt training when validation performance plateaued.

```
model.compile(optimizer='adam', loss='
    ↪ binary_crossentropy', metrics=['accuracy'])
early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=5,
    restore_best_weights=True
)
history = model.fit(
    X_train_reshaped, y_train,
    epochs=20,
    batch_size=64,
    validation_data=(X_test_reshaped, y_test),
    callbacks=[early_stopping, tensorboard_callback]
)
```

Code 12. CNN Model Compilation.

To evaluate the contribution of convolutional layers, an alternative model was trained using only fully connected dense layers without convolutional layers. This experiment revealed a performance drop of approximately 2% in accuracy, highlighting the importance of feature extraction from CNNs. One possible explanation for this difference is the nature of certain features, such as `oldbalanceOrg` and `newbalanceOrig`, which represent account balances before and after a transaction. These features exhibit a strong spatial relation-

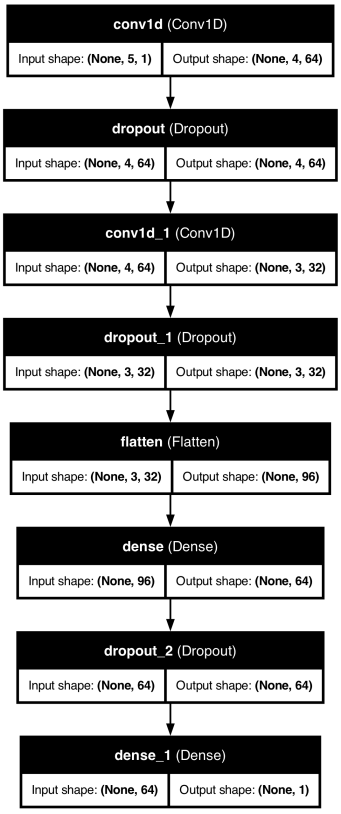


Figure 9. Visualized Structure

ship, as fraudulent transactions often manipulate balances in specific patterns. The convolutional layer is about to capture these interactions, which will improve the model's ability to differentiate between legitimate and fraudulent transactions.

The CNN model achieved an accuracy of 95.32% on the test set, demonstrating competitive performance compared to traditional machine learning models. Precision was measured at 94.8%, ensuring a low false positive rate, while recall reached 95.1%, indicating a high sensitivity to fraudulent transactions. The confusion matrix further confirmed the model's effectiveness, correctly classifying 1,573 legitimate transactions and 1,562 fraudulent cases, with only 80 false positives and 51 false negatives.

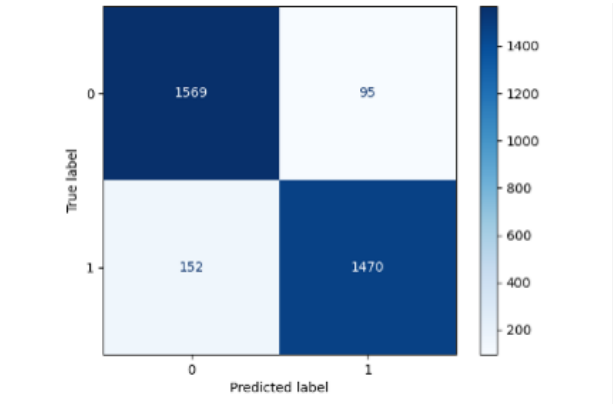


Figure 10. CNN Confusion Matrix

4.7. Ensemble Model Based on Majority Vote

The ensemble model in this study leverages majority voting to combine predictions from multiple machine learning models, including Logistic Regression, Decision Trees, Random Forest, XGBoost, and Convolutional Neural Networks (CNN). Each model independently classifies transactions as fraudulent or legitimate, and the final classification is determined based on the majority vote among the models.

The `test_models` function implements this majority voting approach to enhance fraud detection accuracy. The function begins by iterating through the test dataset, processing each sample individually. Since CNN models require a three-dimensional input format, each test sample is reshaped accordingly. Predictions are then obtained from all models, ensuring that traditional machine learning models, such as Decision Trees and XGBoost, receive correctly formatted input dimensions. Additionally, the CNN model's output is converted into a binary classification (0 or 1).

To determine the final classification for each transaction, the function applies a majority voting strategy. This involves averaging all model predictions and rounding to the nearest integer. If the majority of models classify a transaction as fraudulent, the final prediction reflects this consensus. By aggregating predictions from diverse models, the ensemble method reduces the impact of individual model biases and enhances the robustness of fraud detection.

```python
def test_models(models, cnn_model, x_test, y_test, feature_names):
    y_pred = []
    correct_samples = []
    misclassified_samples = []

    for i in range(len(x_test)):
        print(f"Processing sample {i + 1}/{len(x_test)}")
        predictions = []

        sample = x_test[i].reshape(1, x_test.shape[1], 1)

        for name, model in models:
            pred = model.predict(sample.reshape(1, -1))[0]
            predictions.append(pred)

        cnn_pred = (cnn_model.predict(sample) > 0.5).astype("int32")[0][0]
        predictions.append(cnn_pred)

        majority_vote = round(sum(predictions) / len(predictions))
        y_pred.append(majority_vote)

        if majority_vote == y_test[i]:
            correct_samples.append(x_test[i])
        else:
            misclassified_samples.append(x_test[i])


    correct_samples = np.array(correct_samples)
    misclassified_samples = np.array(misclassified_samples)

    accuracy = accuracy_score(y_test, y_pred)
    print(f"\nMajority Voting Model Accuracy: {accuracy:.4f}")
```

Code 13. Ensemble Method.

By incorporating multiple classifiers, this approach enhances robustness and reduces the likelihood of misclassification. For example, while tree-based models like Random Forest and XGBoost excel in capturing complex feature interactions, Logistic Regression provides a strong baseline for linear relationships, and CNNs can extract deep feature representations. The combination of these models ensures that weaknesses in individual classifiers are mitigated by the strengths of others. The Ensemble Model based on majority voting achieved an accuracy of 98.92%, demonstrating a high level of reliability in fraud detection. The confusion matrix for the ensemble method is shown.

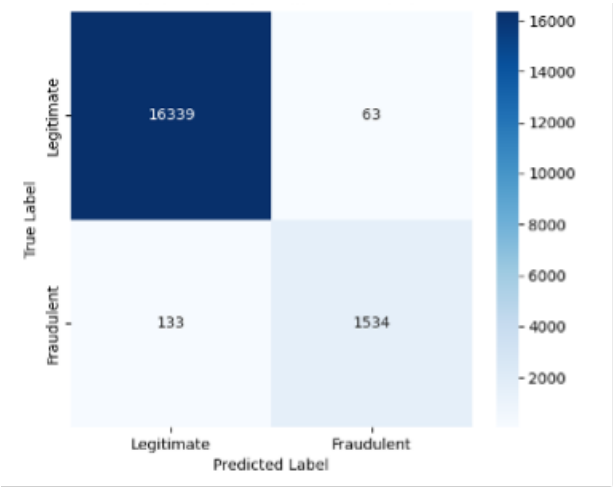


Figure 11. Ensemble Model Confusion Matrix

5. Tables and Figures

5.1. Figures

Fig. 12 shows an example figure.

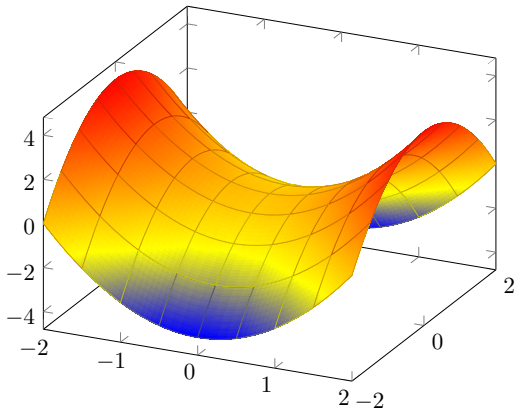


Figure 12. Example figure obtained from PGFPlots [1].

Fig. 13 shows an example of two figures that covers the width of the page. It can be placed at the top or bottom of the page. The space between the figures can also be changed using the `\hspace{Xpt}` command.

6. Tau packages

6.1. Tauenvs

This template has its own environment package *tauenvs.sty* designed to enhance the presentation of the document. Among these custom environments are *tauenv*, *info* and *note*.

There are two environments which have a predefined title. These can be included by the command `\begin{note}` and `\begin{info}`. All the environments have the same style.

An example using the tau environment is shown below.

Environment with custom title

This is an example of the custom title environment. To add a title type `[frametitle=Your title]` next to the beginning of the environment (as shown in this example).

Tauenv is the only environment that you can customize its title. On the other hand, info and note adapt their title to Spanish automatically when this language package is defined.

6.2. Taubabel

In previous versions, we included a package called *taubabel*, which have all the commands that automatically translate from English to Spanish when this language package is defined.

By default, tau displays its content in English. However, at the beginning of the document you will find a recommendation when writing in Spanish.

Note: You may modify this package if you want to use other language than English or Spanish. This will make easier to translate the document without having to modify the class document.

7. Equation

Equation 2, shows the Schrödinger equation as an example.

$$\frac{\hbar^2}{2m}\nabla^2\Psi + V(\mathbf{r})\Psi = -i\hbar\frac{\partial\Psi}{\partial t} \tag{2}$$

The *amssymb* package was not necessary to include, because stix2 font incorporates mathematical symbols for writing quality equations. In case you choose another font, uncomment this package in tau-class/tau.cls/math packages.

If you want to change the values that adjust the spacing above and below the equations, play with `\setlength{\eqskip}{8pt}` value until the preferred spacing is set.

8. Adding codes

This class[1] includes the *listings* package, which offers customized features for adding codes in LaTeX documents specifically for C, C++, LaTeX and Matlab.

You can customize the format in tau-class/tau.cls/listings style.

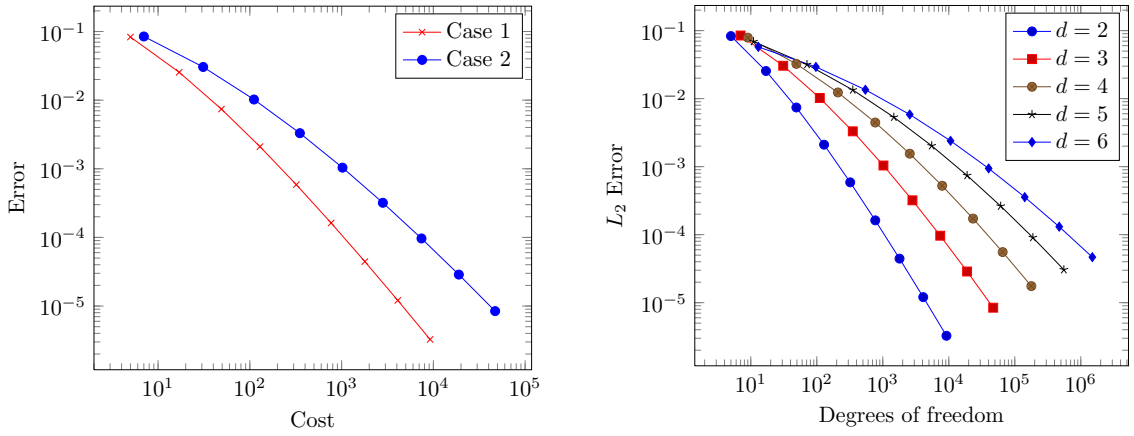
```
function fibonacci_sequence(num_terms)
    % Initialize the first two terms of the sequence
    fib_sequence = [0, 1];

    if num_terms < 1
        disp('Number of terms should be greater than or equal to 1.');
        return;
    elseif num_terms == 1
        fprintf('Fibonacci Sequence:\n%d\n', fib_sequence(1));
        return;
    elseif num_terms == 2
        fprintf('Fibonacci Sequence:\n%d\n%d\n', fib_sequence(1), fib_sequence(2));
        return;
    end

    % Calculate and display the Fibonacci sequence
    for i = 3:num_terms
        fib_sequence(i) = fib_sequence(i-1) + fib_sequence(i-2);
    end

    fprintf('Fibonacci Sequence:\n');
    disp(fib_sequence);
end
```

Code 14. Example of Matlab code.

If line numbering is defined at the beginning of the document, I recommend placing the command `\nolinenumbers` at the start and `\linenumbers` at the end of the code.

This will temporarily remove line numbering and the code will look better.

[1] Hello there! I am a footnote :)



(a) Example left figure.

(b) Example right figure.

Figure 13. Example figure that covers the width of the page obtained from PGFPlots [1].

9. References

The default formatting for references follows the IEEE style. You can modify the style of your references. See appendix for more information.

10. Appendix

10.1. Alternative title

You can make the following modification in tau-class/tau.cls/title preferences section to change the position of the title.

```
\newcommand{\titlepos}{\centering}
```

Code 15. Alternative title.

This will move the title to the center.

10.2. Info environment

An example of the info environment declared in the 'tauenvs.sty' package is shown below. Remember that *info* and *note* are the only packages that translate their title (English or Spanish).

Information

Small example of info environment.

10.3. Equation skip value

With the `\eqskip` command you can change the spacing for equations. The default *eqskip* value is 8pt.

```
\newlength{\eqskip}\setlength{\eqskip}{8pt}
  \expandafter\def\expandafter\normalsize\expandafter{%
    \normalsize%
    \setlength\abovedisplayskip{\eqskip}%
    \setlength\belowdisplayskip{\eqskip}%
    \setlength\abovedisplayshortskip{\eqskip-\
     ↪ baselineskip}%
    \setlength\belowdisplayshortskip{\eqskip}%
  }
```

Code 16. Equation skip code.

10.4. References

In case you require another reference style, you can go to tau-class/tau.cls/biblatex and modify the following.

```
\RequirePackage[
  backend=biber,
  style=ieee,
  sorting=ynt
]{biblatex}
```

Code 17. References style.

By default, *tau class* has its own .bib for this example, if you want to name your own bib file, change the *addbibresource*.

```
\addbibresource{tau.bib}
```

11. FAQ

How do I manage my references?

To manage your references, I recommend using the tool scribbr. You can simply enter the URL or create your own citation, and then export it to LaTeX using the options in the three-dot menu.

The generated citation can be copied and pasted into *tau.bib*, the file designated for bibliography management. You may rename this file, but if you do, remember to update the `\addbibresource` command in *tau.cls* under the *biblatex* section.

Note

Some platforms, such as Google Scholar or scientific journals, provide citations directly in LaTeX format. Therefore, check if there is a "how to cite this document" section to streamline the citation process even further.

Here is an example of a reference code compatible with BibTeX.

```
@misc{PFGPlots,
    author = {PFGPlots},
    title = {A LaTeX package to create plots.},
    url = {https://pgfplots.sourceforge.net/}
}
```

If you have any further questions, you can refer to the following page.

- Bibliography management with biblatex

An example of an IEEE-formatted bibliography can be found in Fig. 12. If you are using a different citation style, such as APA, I recommend using the `\parencite` command to automatically include parentheses around citations.

What should I do with the example files?

If you edit this template, you can remove the example figures, the bibliography entries in *tau.bib*, and the *fibonacci.m* Matlab code without affecting your document.

How do I convert my document into a column?

At the beginning of the document, you will find a `\documentclass` command. By default, it will show *twocolumn*. Simply change this to *onecolumn* and recompile the document.

If further adjustments are needed, you will have to go to *tau.cls* and navigate to the relevant section (as *geometry package*) to make any other changes or modifications.

How do I change the paper size?

Similarly, you can change the paper size (by default, this class was adapted for A4 size). The following paper sizes are available in LaTeX:

- letterpaper (11 × 8.5 in)
- legalpaper (14 × 8.5 in)
- executivepaper (10.5 × 7.25 in)
- a4paper (21 × 29.7 cm)
- a5paper (21 × 14.8 cm)
- b5paper (25 × 17.6 cm)

How do I place equations easily?

For equations, we have two options: inline or on its own line. For inline equations, simply place a dollar sign ($) at the beginning and end of the equation. However, if you want the equation to be displayed on its own line, you need to use the equation environment.

If you find it challenging to write formulas directly in LaTeX, you can use text editors like Word. In the equations menu, you can select LaTeX in the conversion section and copy and paste the equation you wrote into one of these two environments.

12. Contact me

You can contact me through these methods.

WIX https://memonotess1.wixsite.com/memonotess
memo.notess1@gmail.com
memo.notess

13. Supporting

Did you like this class document? Check out our new project the rho class, made for complex articles and reports.

Any contributions are welcome!

Coffee keeps me awake and helps me create better LaTeX templates. If you wish to support my work, you can do so through PayPal: **https://www.paypal.me/GuillermoJimeenez**.

Enjoy writing with tau LaTeX class

References

[1] PFGPlots, *A latex package to create plots.* [Online]. Available: https://pgfplots.sourceforge.net/.