CPSC 597

Fraudulent Transaction Detection System

Yathartha Patankar
885189803

Project Advisor : Prof. Kanika Sood

Project Reviewer : Prof. Bin Cong

Introduction

FTDS, or Fraudulent Transaction Detection System, is an advanced solution leveraging machine learning algorithms to detect and prevent fraudulent transactions within financial systems. By analyzing patterns, anomalies, and various data points, FTDS efficiently identifies suspicious activities, mitigating financial risks and ensuring the integrity of transactions. This system is designed to provide real-time insights, enabling swift action to combat fraudulent behavior and safeguard financial assets.

Problem Statement

- Develop an FTDS capable of detecting anomalies and adapting to emerging fraud patterns using machine learning, ensuring efficient and accurate fraud detection.

- Develop an FTDS capable of handling imbalanced datasets and utilizing autoencoders for anomaly detection, ensuring robust fraud detection capabilities.

Objectives:

1. To tackle the imbalance of dataset.
2. Implement Autoencoder for enhance fraud detection.
3. Create an application which uses the trained model to analyse transactions.

Technology Used

Basic Architecture

Implementing Autoencoders

Incorporating the 3 layer autoencoder for improved and precise fraud detection.

Data Ingestion

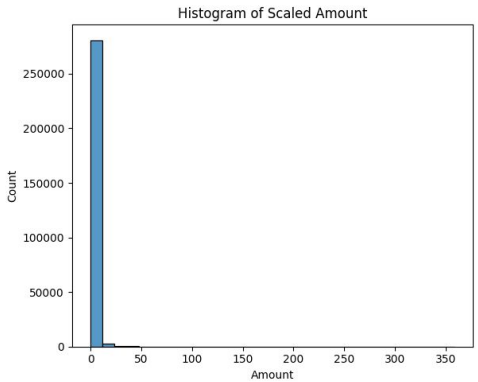
Includes importing the dataset from the source

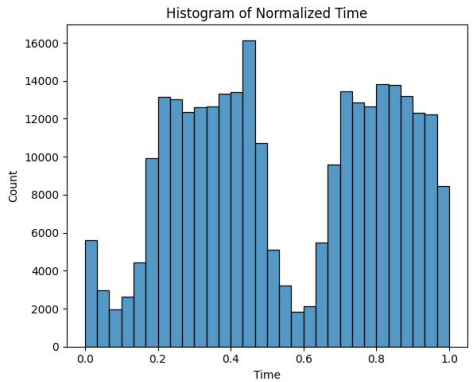
Balancing the Classes

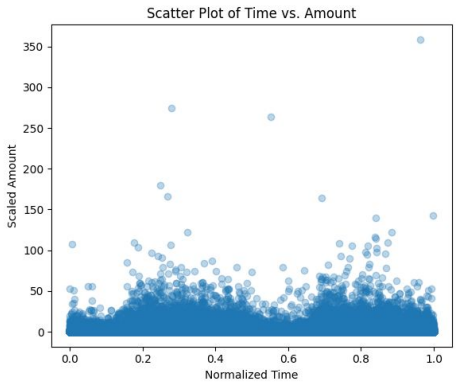
This steps involves tackling the class imbalance through SMOTE.

Accomplishments

1. **Data Loading and Preprocessing**

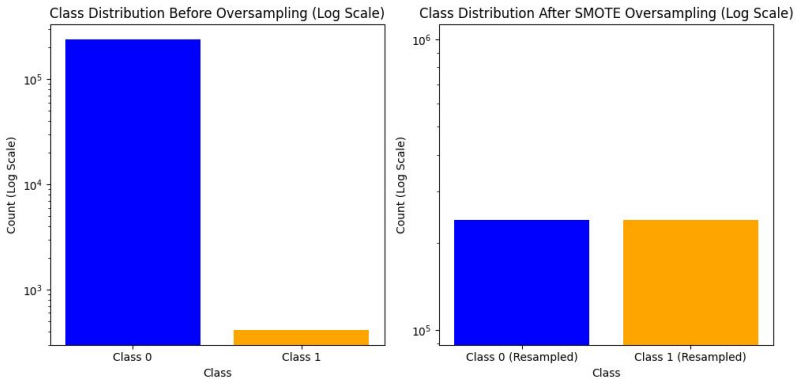


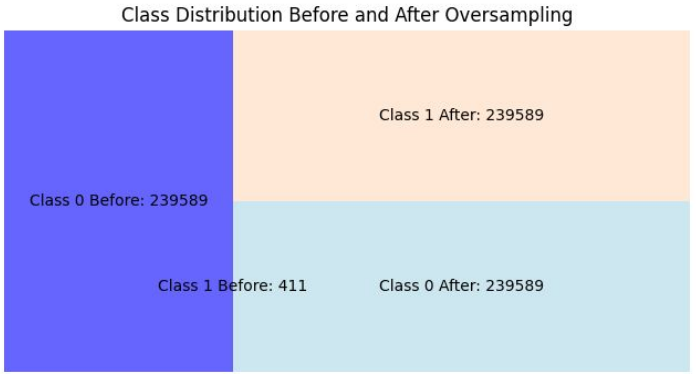




Accomplishments

2. Class Imbalance Mitigation(SMOTE)





Results Comparison

	precision	recall	f1-score	support
Not Fraud	1.00	1.00	1.00	21955
Fraud	0.91	0.64	0.75	45
accuracy			1.00	22000
macro avg	0.95	0.82	0.88	22000
weighted avg	1.00	1.00	1.00	22000

```
[[21952     3]
 [   16    29]]
```

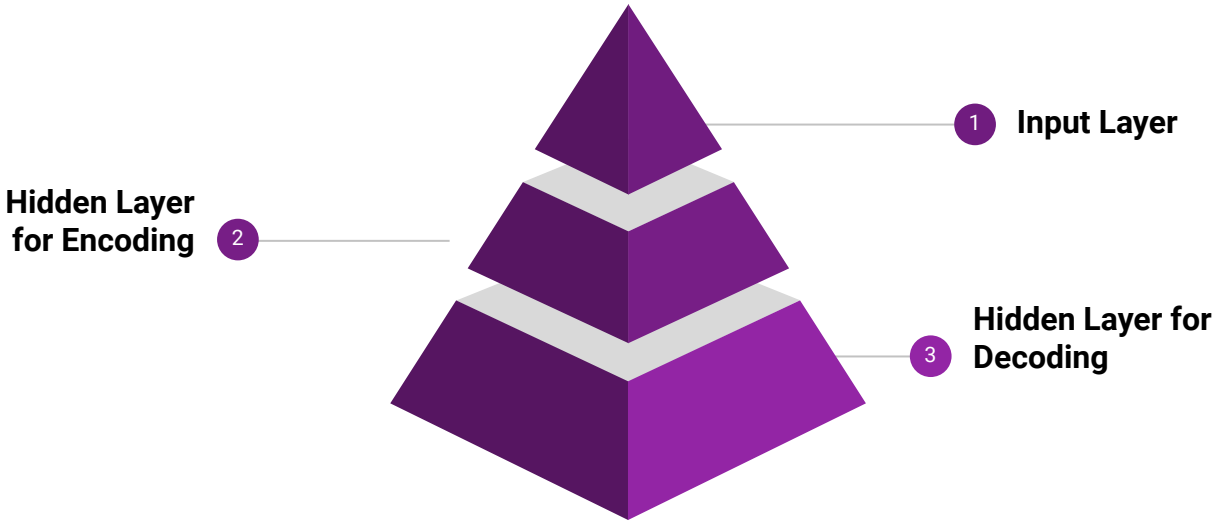
	precision	recall	f1-score	support
Not Fraud	1.00	0.98	0.99	21955
Fraud	0.07	0.91	0.13	45
accuracy			0.98	22000
macro avg	0.54	0.94	0.56	22000
weighted avg	1.00	0.98	0.99	22000

```
Confusion Matrix LR model after SMOTE
[[21412   543]
 [    4    41]]
```

Classification Report (before and after SMOTE)

Accomplishments

3. Autoencoders Implementation



Model Architecture

- **Input Layer**: Handles input_dim features per transaction.
- **Encoder**: Compresses input via three layers (128, 64, 32 neurons) with ReLU activation.
- **Decoder**: Reconstructs input through two layers (64, 128 neurons) with ReLU, and an output layer matching input dimensions with sigmoid activation.

Functionality:

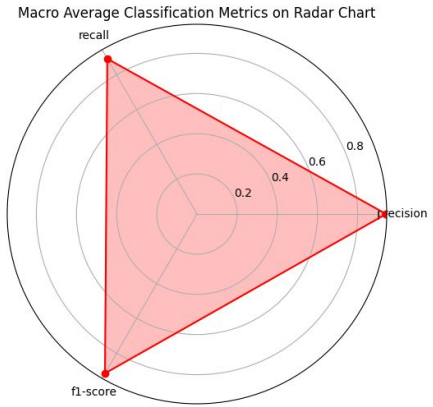
- **Loss Function**: Utilizes Mean Squared Error (MSE) to measure reconstruction loss.
- **Optimizer**: Employs Adam for efficient optimization during training.

Performance Metrics with Autoencoders

```
[[56857     7]
 [   15    83]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.92      0.85      0.88        98

    accuracy                           1.00     56962
   macro avg       0.96      0.92      0.94     56962
weighted avg       1.00      1.00      1.00     56962
```



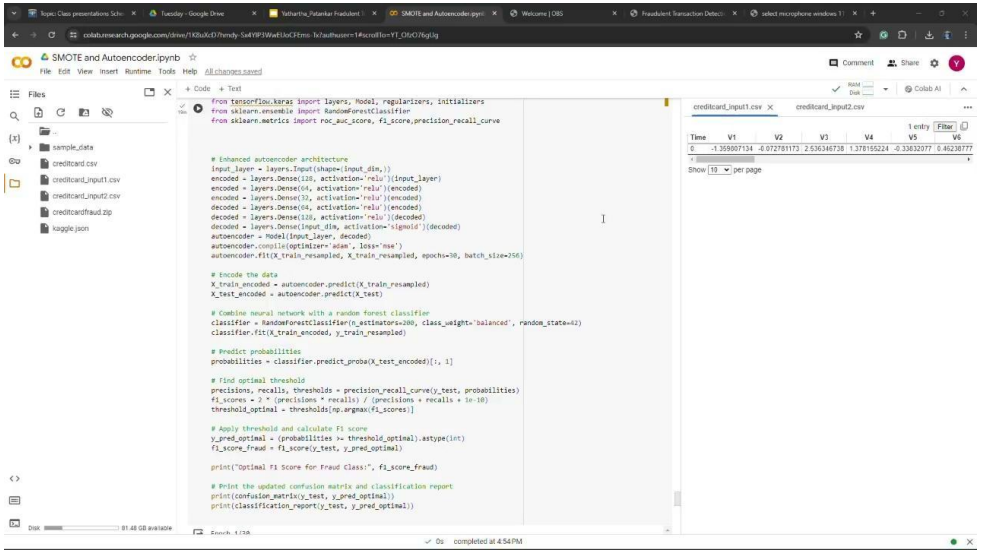
Future Scope

- Autoencoders have demonstrated significant success in achieving robust and balanced F1 scores. The implementation of autoencoders in conjunction with a Gradient Boosting Classifier has further improved the recall score to 87%. There is potential for further enhancements through meticulous hyperparameter tuning. This exploration could yield even more optimized performance metrics.

Google Colaboratory Link:

1. SMOTE : CreditCard-SMOTE.ipynb
2. Autoencoder : SMOTE and Autoencoder.ipynb
3. GitHub : Link

Brief Demo


SMOTE and Autoencoder.ipynb
Files
sample_data
creditcard.csv
creditcard_input1.csv
creditcard_input2.csv
creditcardfraud.zip
kaggle.json
from tensorflow.keras import layers, Model, regularizers, initializers
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_auc_score, f1_score,precision_recall_curve
# Enhanced autoencoder architecture
input_layer = layers.Input(shape=(input_dim,))
encoded = layers.Dense(128, activation='relu')(input_layer)
encoded = layers.Dense(64, activation='relu')(encoded)
encoded = layers.Dense(32, activation='relu')(encoded)
decoded = layers.Dense(64, activation='relu')(encoded)
decoded = layers.Dense(128, activation='relu')(decoded)
decoded = layers.Dense(input_dim, activation='sigmoid')(decoded)
autoencoder = Model(input_layer, decoded)
autoencoder.compile(optimizer='adam', loss='mse')
autoencoder.fit(X_train_resampled, X_train_resampled, epochs=30, batch_size=256)
# Encode the data
X_train_encoded = autoencoder.predict(X_train_resampled)
X_test_encoded = autoencoder.predict(X_test)
# Combine neural network with a random forest classifier
classifier = RandomForestClassifier(n_estimators=200, class_weight='balanced', random_state=42)
classifier.fit(X_train_encoded, y_train_resampled)
# Predict probabilities
probabilities = classifier.predict_proba(X_test_encoded)[:, 1]
# Find optimal threshold
precisions, recalls, thresholds = precision_recall_curve(y_test, probabilities)
f1_scores = 2 * (precisions * recalls) / (precisions + recalls + 1e-10)
threshold_optimal = thresholds[np.argmax(f1_scores)]
# Apply threshold and calculate F1 score
y_pred_optimal = (probabilities >= threshold_optimal).astype(int)
f1_score_fraud = f1_score(y_test, y_pred_optimal)
print("Optimal F1 Score for Fraud Class:", f1_score_fraud)
# Print the updated confusion matrix and classification report
print(confusion_matrix(y_test, y_pred_optimal))
print(classification_report(y_test, y_pred_optimal))
creditcard_input1.csv
creditcard_input2.csv
Time V1 V2 V3 V4 V5 V6
0 -1.359807134 -0.072781173 2.536346738 1.378155224 -0.33832077 0.462387778
Show 10 per page

References

1. A novel combined approach based on deep Autoencoder and deep classifiers for credit card fraud detection - ScienceDirect
2. Fraud Detection in Mobile Payment Systems using an XGBoost-based Framework | Information Systems Frontiers

Thank you