Fraud Transaction Detection Project Report

1. **Introduction**

This project aims to develop a system that can identify fraudulent transactions based on a simulated dataset. Fraud detection is crucial for financial institutions to minimize losses and protect customers.

**Objectives:**

1. Detect fraudulent transactions using specified patterns.
2. Build a machine learning model to classify transactions as fraudulent or legitimate.
3. Evaluate model performance using metrics such as accuracy, precision, recall, and F1-score.

**Dataset Description:** The dataset is a simulated series of transactions, including identifiers for each transaction, customer, and terminal, transaction amounts, and fraud labels. The dataset consists of multiple *“.pkl”* files.

1. **Data Loading and Preprocessing**

We combined multiple *“.pkl”* files from the dataset directory. After loading, we converted date columns to DateTime format and explored the dataset structure. The following code was used to load the data:

*def load\_data\_from\_pkl(folder\_path):*

*all\_data = []*

*for file in glob.glob(os.path.join(folder\_path, "\*.pkl")):*

*df = pd.read\_pickle(file)*

*all\_data.append(df)*

*return pd.concat(all\_data, ignore\_index=True)*

A screen shot of a black screen

Description automatically generated

1. **Explanatory Data Analysis**

EDA was performed to understand the dataset’s structure and identify patterns in transaction amounts and fraud labels.

* 1. **Transaction Amount Distribution**

The distribution of transaction amounts helps identify outliers and patterns in spending behavior. The histogram below shows a heavy tail, with some transactions having very high amounts, potentially indicating fraud.

*plt.figure(figsize=(10, 5))*

*sns.histplot(data['TX\_AMOUNT'], bins=50, kde=True)*

*plt.title("Distribution of Transaction Amounts")*

*plt.xlabel("Transaction Amount")*

*plt.ylabel("Frequency")*

*plt.show()*

A diagram of a distribution of transaction amount

Description automatically generated

* 1. **Fraudulent vs Legitimate Transactions**

A count plot was generated to show the proportion of fraudulent vs. legitimate transactions, giving insight into the data's imbalance.

*plt.figure(figsize=(6, 4))*

*sns.countplot(x='TX\_FRAUD', data=data)*

*plt.title("Fraudulent vs Legitimate Transactions")*

*plt.xlabel("Fraud Status (0 = Legitimate, 1 = Fraud)")*

*plt.ylabel("Count")*

*plt.show()*

A graph of a blue bar

Description automatically generated with medium confidence

1. **Feature Engineering**

To simulate real-world fraud patterns, we applied three scenarios to label transactions as fraudulent:

1. **Scenario 1:** Transactions with an amount greater than 220 are considered fraudulent.

*data['SCENARIO\_1\_FRAUD'] = data['TX\_AMOUNT'] > 220*

1. **Scenario 2:** Randomly selected terminals were marked as fraud-prone for a 28-day period.

*def mark\_scenario\_2\_fraud(data, num\_terminals=2, period=28):*

*unique\_dates = data['TX\_DATETIME'].dt.date.unique()*

*fraud\_period = datetime.timedelta(days=period)*

*for day in unique\_dates:*

*selected\_terminals = np.random.choice(data['TERMINAL\_ID'].unique(), num\_terminals, replace=False)*

*day\_start = pd.to\_datetime(day)*

*day\_end = day\_start + fraud\_period*

*data.loc[(data['TERMINAL\_ID'].isin(selected\_terminals)) &*

*(data['TX\_DATETIME'].between(day\_start, day\_end)), 'SCENARIO\_2\_FRAUD'] = 1*

*data['SCENARIO\_2\_FRAUD'] = data['SCENARIO\_2\_FRAUD'].fillna(0)*

*mark\_scenario\_2\_fraud(data)*

1. **Scenario 3:** Randomly selected customers had 1/3 of their transactions marked as fraud by multiplying amounts by 5 over a 14-day period.

*def mark\_scenario\_3\_fraud(data, num\_customers=3, fraction=0.33, period=14):*

*unique\_dates = data['TX\_DATETIME'].dt.date.unique()*

*fraud\_period = datetime.timedelta(days=period)*

*for day in unique\_dates:*

*selected\_customers = np.random.choice(data['CUSTOMER\_ID'].unique(), num\_customers, replace=False)*

*day\_start = pd.to\_datetime(day)*

*day\_end = day\_start + fraud\_period*

*mask = (data['CUSTOMER\_ID'].isin(selected\_customers)) & (data['TX\_DATETIME'].between(day\_start, day\_end))*

*fraud\_indices = data[mask].sample(frac=fraction).index*

*data.loc[fraud\_indices, 'TX\_AMOUNT'] \*= 5 # Increase amount*

*data.loc[fraud\_indices, 'SCENARIO\_3\_FRAUD'] = 1*

*data['SCENARIO\_3\_FRAUD'] = data['SCENARIO\_3\_FRAUD'].fillna(0)*

*mark\_scenario\_3\_fraud(data)*

1. **Data Splitting and Scaling**

The dataset was split into training and test sets, and transaction amounts were scaled to normalize the feature.

*scaler = StandardScaler()*

*X.loc[:, 'TX\_AMOUNT'] = scaler.fit\_transform(X[['TX\_AMOUNT']])*

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)*

1. **Model Training**

We trained a RandomForestClassifier on the training data. Random Forests are well-suited for classification tasks with high dimensionality and offer interpretability.

*model = RandomForestClassifier(n\_estimators=100, random\_state=42)*

*model.fit(X\_train, y\_train)*

1. **Model Evaluation**

The model was evaluated on the test data using accuracy, precision, recall, and F1-score. We also visualized the confusion matrix to understand model performance.

*y\_pred = model.predict(X\_test)*

*conf\_matrix = confusion\_matrix(y\_test, y\_pred)*

*sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)*

*plt.xlabel("Predicted Label")*

*plt.ylabel("True Label")*

*plt.title("Confusion Matrix")*

*plt.show()*

**Results:**

* **Accuracy:** 0.9934612453847718
* **Precision:** 0.99
* **Recall:** 0.99
* **F1-score:** 0.99

A screenshot of a computer

Description automatically generated

A blue square with white text

Description automatically generated

1. **Discussion**

The model performed well in detecting fraudulent transactions. However, since the data is simulated with specific patterns, performance in a real-world scenario would require additional tuning and feature engineering to capture more nuanced fraud patterns. Imbalanced data could be further addressed with resampling techniques if needed.

1. **Conclusion**

This project demonstrated the process of building a fraud detection system using machine learning. Through feature engineering, visualization, and model evaluation, we achieved a reliable classifier for detecting simulated fraud patterns.