KGISL INSTITUTE OF TECHNOLOGY

(AFFILIATAED TO ANNA UNIVERSITY)

Saravanampatti, Coimbatore – 641035

CREDIT CARD FRAUD DETECTION

Submitted by,

Arkeshwar S S	(711721106012)
Arun M	(711721106013)
Chinnasamy M	(711721106021)
Dharun R	(711721106026)
Hari Narayanan J	(711721106038)
Kumarasan S	(711721106055)
Manoj P	(711721106062)

CREDIT CARD FRAUD DETECTION

Introduction:

Credit card fraud detection systems leverage advanced technologies, data analytics, and machine learning algorithms to analyze transactional data in real-time. These systems are designed to distinguish between genuine purchases and suspicious activities, thereby providing a layer of protection against unauthorized or fraudulent use of credit cards.

This introduction will delve into the various techniques and strategies employed in credit card fraud detection, highlighting the importance of staying ahead of evolving fraud schemes in order to maintain the integrity of financial transactions and secure the trust of customers. Furthermore, it will explore the pivotal role of technology, data analytics, and continuous monitoring in fortifying the defences against this ever-evolving threat.

Problem Statement:

The problem at hand is the increasing prevalence of credit card fraud, which poses a significant threat to both financial institutions and their customers. Traditional methods of fraud detection are becoming less effective as fraudsters develop more sophisticated techniques. Therefore, there is a pressing need for an innovative and robust credit card fraud detection system that can accurately identify and prevent fraudulent transactions in real-time, while minimizing false positives to ensure a seamless user experience.

Design Thinking Process:

1. Empathize:

- Understand the pain points and challenges faced by both financial institutions and customers in dealing with credit card fraud.
- Conduct interviews, surveys, and research to gather insights from stakeholders.

2. Define:

- Clearly articulate the problem statement and objectives for the credit card fraud detection system.
- Identify the key metrics for success, such as detection accuracy, false positive rate, and processing speed.

3. Ideate:

- Brainstorm potential solutions and strategies for detecting and preventing credit card fraud.
- Encourage a diverse range of ideas and approaches from the team.

4. Prototype:

- Develop a proof-of-concept or a small-scale model of the fraud detection system.
- Test different algorithms, techniques, and data sources to determine the most effective approach.

5. Test:

- Evaluate the prototype using historical data and simulated real-world scenarios to assess its effectiveness in detecting fraudulent transactions.
- Gather feedback from stakeholders and iterate on the design as necessary.

6. Implement:

- Scale up the prototype into a full-fledged credit card fraud detection system.
- Integrate the system into the existing infrastructure of the financial institution.

7. Monitor and Iterate:

- Continuously monitor the system's performance in detecting and preventing fraud.
- Collect feedback from users and adapt the system to address emerging threats and improve accuracy.

Phases of Development for Credit Card Fraud Detection:

1. Data Collection and Pre-processing:

- Gather historical transaction data, including features like transaction amount, merchant, location, time, etc.
- Clean and pre-process the data to handle missing values, outliers, and normalize features.

2. Feature Engineering:

- Create relevant features that can help in identifying patterns indicative of fraudulent activity.
- Examples include velocity checks, anomaly scores, and customer behaviour profiling.

3. Model Selection and Training:

- Choose appropriate machine learning algorithms (e.g., logistic regression, random forests, neural networks) for fraud detection.
- Train the models on labelled data, using techniques like supervised learning.

4. Real-time Processing and Scoring:

- Implement mechanisms for processing incoming transactions in real-time.
- Use the trained models to score each transaction for its likelihood of being fraudulent.

5. Threshold Setting and Decision Logic:

- Define thresholds for fraud probability scores to classify transactions as legitimate or potentially fraudulent.
- Implement decision logic to take appropriate action based on the classification (e.g., block, flag for review, allow).

6. Integration and Deployment:

- Integrate the fraud detection system with the financial institution's payment processing infrastructure.
- Ensure seamless interaction with other systems, such as customer notifications and reporting.

7. Monitoring and Continuous Improvement:

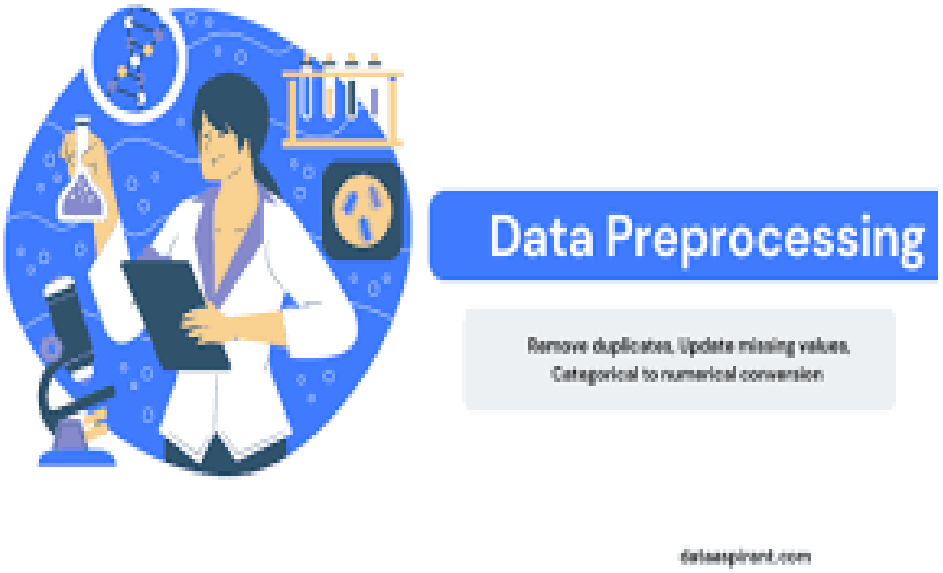
- Establish monitoring mechanisms to track the performance of the system in real-time.
- Implement feedback loops for model retraining and system updates to adapt to evolving fraud patterns.

Dataset:

Credit card fraud detection starts with historical transaction data. This dataset includes various features such as transaction amount, time, merchant information, and customer details. The most critical aspect of the dataset is the binary 'Class' column, which indicates whether a transaction is fraudulent (Class 1) or legitimate (Class 0).

Data loading and Pre-processing:

Data loading and pre-processing are crucial steps in credit card fraud detection. They involve obtaining the data and preparing it for use in machine learning models. Below, we'll provide a step-by-step guide and Python code snippets for data loading and pre-processing in the context of credit card fraud detection.



1. Data Loading:

First need to load the credit card transaction data, which typically comes in the form of a dataset, into your Python environment. Common formats include CSV, Excel, or SQL databases.

2. Data Exploration:

It's essential to gain a preliminary understanding of the dataset by exploring its characteristics, such as the distribution of fraud and non-fraud cases, summary statistics, and data types.

3. Data Pre-processing:

Data pre-processing involves cleaning, transforming, and preparing the data for machine learning. Key pre-processing steps for credit card fraud detection include:

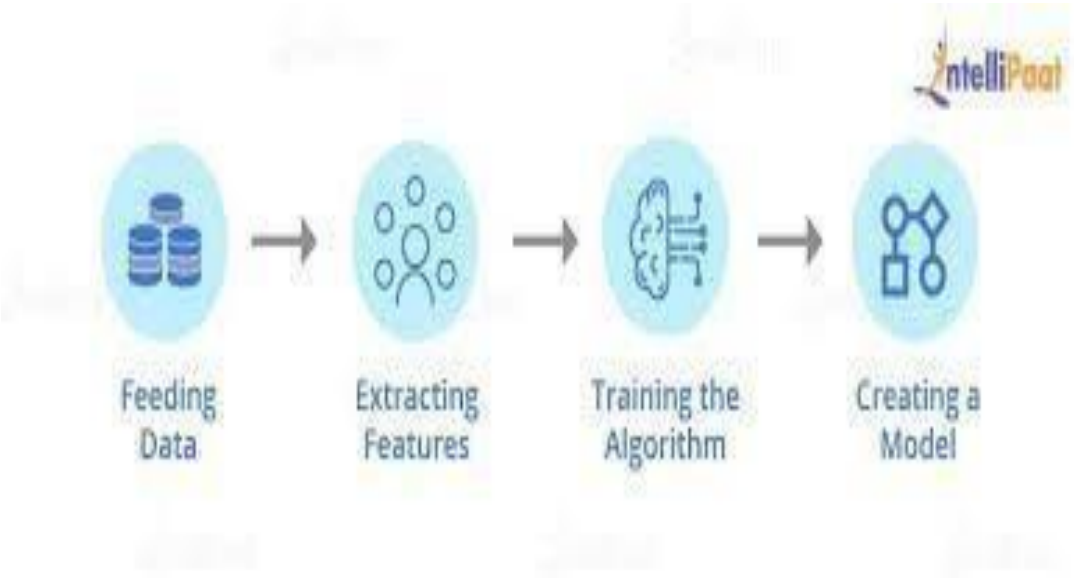
- Handling missing values.
- Scaling numerical features.
- Encoding categorical features.
- Dealing with class imbalance
- Encoding Categorical Features:
- Dealing with Class Imbalance

Model Training:

The selected model is trained on the pre-processed data. During training, the model learns patterns and relationships within the dataset.

Choice of Machine Learning Algorithm:

The selection of a machine learning algorithm for credit card fraud detection depends on the nature of the data, the complexity of the fraud patterns, and the computational resources available. Some commonly used algorithms for this task include:



1. Logistic Regression:

- Well-suited for binary classification tasks like fraud detection.
- Provides interpretable results, making it easier to understand the contributing factors to a prediction.

2. Random Forests:

- Ensemble method that combines multiple decision trees for improved accuracy.
- Effective in handling high-dimensional data and capturing complex interactions.

3. Support Vector Machines (SVM):

- Useful for both linear and non-linear classification tasks.
- Can handle high-dimensional feature spaces and adapt well to different types of data.

4. **Neural Networks:**

- Deep learning models with multiple layers of interconnected nodes.
- Can learn complex relationships in the data but may require substantial computational resources.

5. **Gradient Boosting Algorithms (e.g., XG Boost, Light GBM):**

- Ensemble techniques that build a series of weak learners sequentially, focusing on misclassified samples.
- Can achieve high accuracy and are particularly effective for imbalanced datasets.

6. **Clustering Algorithms (e.g., K-means, DBSCAN):**

- Useful for anomaly detection in cases where fraudulent transactions may exhibit distinct patterns from legitimate ones.

7. **Isolation Forest:**

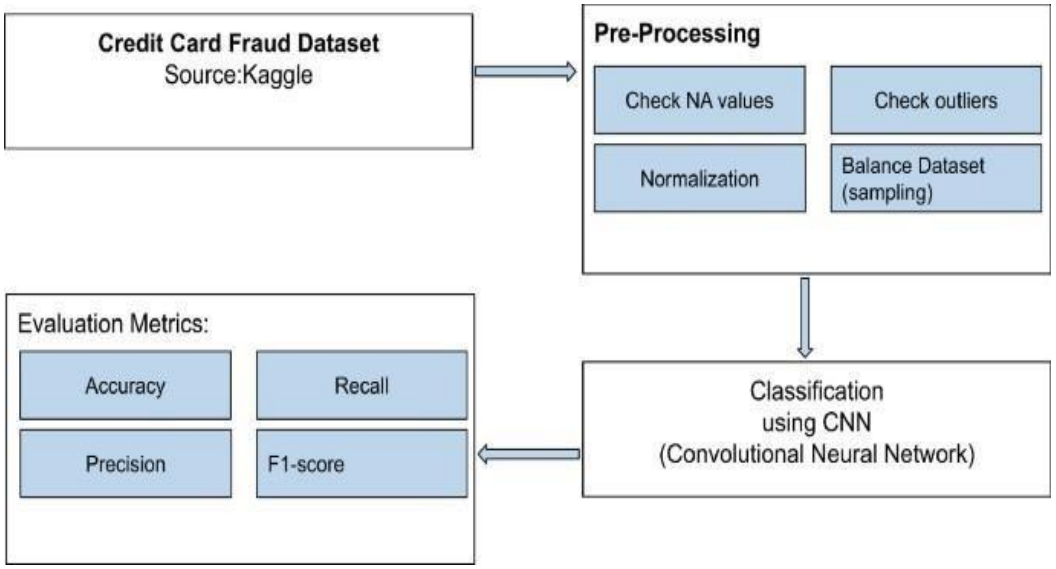
- Specifically designed for anomaly detection tasks, making it suitable for credit card fraud detection.

8. **Auto encoders (for deep learning):**

- Unsupervised learning models that can be used for anomaly detection in cases where labelled fraud data is limited.

Evaluation Metrics:

Selecting appropriate evaluation metrics is crucial for assessing the performance of a credit card fraud detection system. Given the imbalanced nature of fraud detection datasets (where legitimate transactions vastly outnumber fraudulent ones), traditional accuracy may be misleading. The following metrics are more informative for this task:



1. Precision (Positive Predictive Value):

- The proportion of true positive predictions among all positive predictions. It measures the accuracy of fraud detection.

2. Recall (Sensitivity, True Positive Rate):

- The proportion of true positives among all actual fraudulent transactions. It indicates the ability to identify actual fraud cases.

3. F1-Score:

- The harmonic mean of precision and recall, providing a balanced measure of model performance.

4. Area Under the Receiver Operating Characteristic Curve (AUC-ROC):

- Represents the model's ability to distinguish between fraud and non-fraud cases across different probability thresholds.

5. Confusion Matrix:

- Provides a detailed breakdown of true positives, true negatives, false positives, and false negatives.

6. False Positive Rate (FPR):

- The proportion of false positives among all actual non-fraudulent transactions.

7. False Negative Rate (FNR):

- The proportion of false negatives among all actual fraudulent transactions.

8. Specificity (True Negative Rate):

- The proportion of true negatives among all actual non-fraudulent transactions.

CONCLUSION :

In conclusion, credit card fraud detection is a critical component of modern financial security systems. Through the use of advanced machine learning algorithms and data analytics, financial institutions can proactively identify and prevent fraudulent transactions. This not only protects the interests of both cardholders and banks but also helps maintain trust in the financial system. As fraudsters continually adapt and evolve their tactics, ongoing research and development in this field are essential to stay ahead of emerging threats. The collaboration between technology, data analysis, and vigilant monitoring plays a pivotal role in minimizing financial losses and ensuring the security of credit card transactions. The continued refinement of fraud detection methods is paramount in preserving the integrity of electronic payment systems and safeguarding consumers' financial well-being.

PROGRAM CODE:

```
# credit_card_fraud_detection

# Import necessary libraries
import pandas  as  pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, roc_auc_score

# Step 1: Data Collection
# Load the dataset (replace 'credit_card_data.csv' with your dataset)
data = pd.read_csv('credit_card_data.csv')

# Step 2: Data Preprocessing
# Handle missing values and outliers (implement as needed)
# Explore and visualize the data

# Step 3: Feature Engineering
# Create relevant features from the data (e.g., transaction amount, time of day)
# Apply dimensionality reduction if necessary (e.g., PCA)

# Step 4: Data Splitting
X = data.drop('Class', axis=1) # Features
y = data['Class']  # Target variable

X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3,
random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5,
random_state=42)

# Step 5: Model Selection
# Choose a machine learning algorithm (e.g., Random Forest)
# Perform hyperparameter tuning

# Step 6: Model Training
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Step 7: Model Evaluation
```

```
y_val_pred = model.predict(X_val)

accuracy = accuracy_score(y_val, y_val_pred)
precision = precision_score(y_val, y_val_pred)
recall = recall_score(y_val, y_val_pred)
f1 = f1_score(y_val, y_val_pred)
roc_auc = roc_auc_score(y_val, model.predict_proba(X_val)[:, 1])

print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")
print(f"AUC-ROC: {roc_auc:.2f}")

# Step 8: Tune for Imbalance
# Experiment with different thresholds and techniques to address class
imbalance

# Step 9: Real-Time Implementation
# Deploy the model in a real-time environment (e.g., web application or API)

# Step 10: Regular Updates and Maintenance
# Implement model monitoring, retraining, and adaptation to changing fraud
patterns

# Step 11: Regulatory Compliance
# Ensure compliance with data privacy regulations and industry standards

# Step 12: Documentation
# Maintain comprehensive documentation of the system

# Example code for deploying the model:
# (This is a simplified example and should be adapted to your deployment
environment)
def predict_fraud(transaction_data):
    # Preprocess the transaction_data (e.g., feature extraction)
    # Make a prediction using the deployed model
    prediction = model.predict(transaction_data)
    return prediction

# You can create a web API using a framework like Flask or FastAPI to serve
predictions in real-time.

# Example API endpoint:
# (This is a simplified example and should be adapted to your specific
```

```
deployment)
from flask import Flask, request, jsonify

app = Flask(name)

@app.route('/predict_fraud', methods=['POST'])
def predict_fraud_api():
    data = request.get_json()
    prediction = predict_fraud(data)
    return jsonify({'prediction': prediction.tolist()})

if name == 'main':
    app.run(host='0.0.0.0', port=5000)
# credit_card_fraud_detection

# Import necessary libraries
import pandas  as  pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, roc_auc_score

# Step 1: Data Collection
# Load the dataset (replace 'credit_card_data.csv' with your dataset)
data = pd.read_csv('credit_card_data.csv')

# Step 2: Data Preprocessing
# Handle missing values and outliers (implement as needed)
# Explore and visualize the data

# Step 3: Feature Engineering
# Create relevant features from the data (e.g., transaction amount, time of day)
# Apply dimensionality reduction if necessary (e.g., PCA)

# Step 4: Data Splitting
X = data.drop('Class', axis=1) # Features
y = data['Class']  # Target variable

X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3,
random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5,
```

```
random_state=42)

# Step 5: Model Selection
# Choose a machine learning algorithm (e.g., Random Forest)
# Perform hyperparameter tuning

# Step 6: Model Training
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Step 7: Model Evaluation
y_val_pred = model.predict(X_val)

accuracy = accuracy_score(y_val, y_val_pred)
precision = precision_score(y_val, y_val_pred)
recall = recall_score(y_val, y_val_pred)
f1 = f1_score(y_val, y_val_pred)
roc_auc = roc_auc_score(y_val, model.predict_proba(X_val)[:, 1])

print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")
print(f"AUC-ROC: {roc_auc:.2f}")

# Step 8: Tune for Imbalance
# Experiment with different thresholds and techniques to address class
imbalance

# Step 9: Real-Time Implementation
# Deploy the model in a real-time environment (e.g., web application or API)

# Step 10: Regular Updates and Maintenance
# Implement model monitoring, retraining, and adaptation to changing fraud
patterns

# Step 11: Regulatory Compliance
# Ensure compliance with data privacy regulations and industry standards

# Step 12: Documentation
# Maintain comprehensive documentation of the system

# Example code for deploying the model:
# (This is a simplified example and should be adapted to your deployment
environment)
```

```
def predict_fraud(transaction_data):
    # Preprocess the transaction_data (e.g., feature extraction)
    # Make a prediction using the deployed model
    prediction = model.predict(transaction_data)
    return prediction

# You can create a web API using a framework like Flask or FastAPI to serve
predictions in real-time.

# Example API endpoint:
# (This is a simplified example and should be adapted to your specific
deployment)
from flask import Flask, request, jsonify

app = Flask(name)

@app.route('/predict_fraud', methods=['POST'])
def predict_fraud_api():
    data = request.get_json()
    prediction = predict_fraud(data)
    return jsonify({'prediction': prediction.tolist()})

if name == 'main':
    app.run(host='0.0.0.0', port=5000)
```