Guarding transactions with AI-powered credit card fraud detection and prevention

Phase-3

Student Name: SALEEM AHMED A

Register Number: 513523104068

Institution: ANNAI MIRA COLLEGE OF ENGINEERING AND TECHNOLOGY

Department: COMPUTER SCIENCE AND ENGINEERING

Date of Submission: 06.05.2025

Github Repository Link: https://github.com/ashu19saleem/SALEEM-AHMED-A-Phase-3-DS.git

---

1.Problem Statement

With the increasing volume of online and digital transactions, credit card fraud has become a significant concern for consumers and financial institutions. Traditional fraud detection systems often struggle to identify complex, evolving fraud patterns in real time, leading to financial losses, customer dissatisfaction, and operational inefficiencies. There is a pressing need for advanced, AI-powered solutions to enhance the accuracy, speed, and adaptability of fraud detection and prevention systems, ensuring secure transactions while minimizing false positives and operational costs.

2.Abstract

As digital financial transactions continue to grow, so does the risk of credit card fraud, threatening both consumers and financial institutions. Traditional rule-based systems are no longer sufficient to combat the evolving tactics of cybercriminals. This paper explores the implementation of artificial intelligence (AI) in credit card fraud detection and prevention, highlighting how machine learning, behavioral analytics, and real-time data processing enable more accurate and efficient fraud mitigation. AI-driven systems can identify anomalous patterns, detect fraudulent behavior, and adapt to emerging threats with minimal human intervention. By leveraging vast datasets and continuously learning from new transaction trends, AI offers a dynamic, scalable, and cost-effective solution to safeguard financial transactions. The integration of AI into fraud prevention strategies marks a significant advancement in securing the integrity of digital payments in an increasingly connected world.

3.System Requirements:

Software Requirements

1. **AI/ML Frameworks & Data Tools**
   - Use Python with libraries like TensorFlow, PyTorch, Scikit-learn for model development.
   - Leverage tools like Pandas, Spark, and Kafka for data processing and real-time stream handling.
2. **Infrastructure & Integration**
   - Utilize secure, scalable backend services with REST APIs and databases (e.g., PostgreSQL, MongoDB).

- Employ monitoring, authentication, and model deployment tools (e.g., MLflow, Docker, Prometheus).

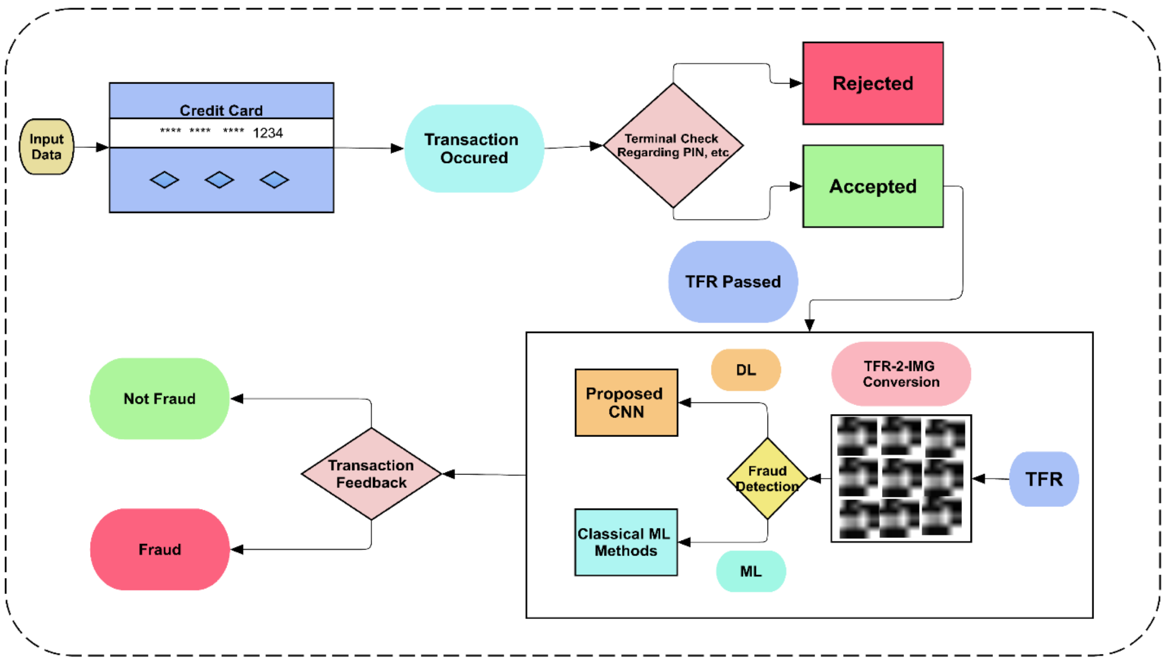
Hardware Requirements:

1. **Model Training & Development Hardware**
   - High-performance servers or cloud instances with GPUs (e.g., NVIDIA A100) and 64GB+ RAM.
   - SSD storage (1TB+) for rapid access to large training datasets.
2. **Real-Time Deployment Infrastructure**
   - Efficient CPUs (quad-core+), 16–32GB RAM, and low-latency networking for real-time fraud detection.
   - Scalable cloud platforms (AWS, GCP, Azure) for load balancing and global transaction monitoring.

4.Objectives

The primary objective of this initiative is to enhance the security and integrity of digital financial transactions by leveraging artificial intelligence for real-time credit card fraud detection and prevention. It aims to develop intelligent systems capable of identifying suspicious patterns, detecting anomalies, and adapting to emerging fraud tactics with minimal human intervention. By using machine learning algorithms, behavioral analytics, and large-scale data processing, the goal is to reduce false positives, improve detection accuracy, and enable faster response to fraudulent activities. Additionally, the system seeks to minimize financial losses for both consumers and institutions, while ensuring a seamless and secure transaction experience. Ultimately, the objective is to establish a proactive, adaptive, and scalable fraud prevention framework that evolves with the digital payment landscape.

5. Flowchart of the Project Workflow



6. Dataset Description

Commonly Used Dataset:

A widely used publicly available dataset is the Kaggle Credit Card Fraud Detection dataset, which contains anonymized European credit card transactions.

Source: Kaggle - Credit Card Fraud Detection

Size: ~284,807 transactions

Fraudulent transactions: ~492 (~0.172% of total), highly imbalanced

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler,
PowerTransformer
from sklearn.metrics import classification_report, roc_auc_score,
confusion_matrix
from sklearn.model_selection import GridSearchCV,
StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from imblearn.over_sampling import ADASYN
from imblearn.under_sampling import TomekLinks
# Load Data
train_df = pd.read_csv("/kaggle/input/fraud-
detection/fraudTrain.csv")
test_df = pd.read_csv("/kaggle/input/fraud-
detection/fraudTest.csv")
train_df.head()
```

Output:

fraud

Unnamed_0	trans_date_trans_time	cc_num	category	first	last	gender	street	city	zip
0	2019-01-01 00:00:00	fraud Kilback inc	misc_net		Jennrnifer	Baker	439 S Broadway	[illegible]y	8020
1	2019-01-01 01:00:00	fraud-Kutch L 38(	grocery_pos		Stephanie	Wise	3391 Torrance	WI	5371
2	2019-01-01 02:00:00	fraud-Bogan-Mac	misc_pos		Pamela	Bond	875 High Ridge	San Diego	9211
3	2019-01-01 03:00:00	fraud-Rolfson Rolf	gas transport		Ann	Lewis	6943 Longbran	HI	96219
4	2019-01-01 04:00:00	fraud-Terry-Mohr	health_pos		Tammy	Johnson	253 Centre St	Little Rock	72205
5	2019-01-01 04:00:00	fraud-Terry-Mohr	183.94		Tammy	Johnson	253 Centre St	AR	72205

7.Data Preprocessing

Data preprocessing is a crucial step in preparing raw transactional data for machine learning. It ensures that the input data is clean, consistent, and suitable for training effective fraud detection models.

Data Cleaning:

Handle Missing Values:

Drop rows/columns with excessive missing data.

Impute missing values using mean, median, or mode (or more advanced techniques like KNN imputation).

Remove Duplicates:

Eliminate duplicate transaction records to avoid bias.

Correct Inconsistencies:

Standardize formats (e.g., timestamps, currency units).

Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler,
PowerTransformer
from sklearn.metrics import classification_report, roc_auc_score,
confusion_matrix
from sklearn.model_selection import GridSearchCV,
StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from imblearn.over_sampling import ADASYN
from imblearn.under_sampling import TomekLinks
```

```
# Load Data
train_df = pd.read_csv("/kaggle/input/fraud-detection/fraudTrain.csv")
test_df = pd.read_csv("/kaggle/input/fraud-detection/fraudTest.csv")
train_df.head()

test_df.head()
```

Output:

train_df test_df

	cc.num	merch	category	amt	first	last	street	city	zip	state	lat	long	citypoi	trans_num	unix_time	merch_lat	is_fraud
)	2.7040:	fraud- Kirlin Inc	misc_net	4.97	Jennil	Banks	4706 Poe G	4706 Poe Glade	CA `	92555	4.947	3267	172175	f6b62ee7ch	1528376000	33,8121	0
)	3.0956‹	fraud Niugn	gotor_pos	107.23	Steph	Funoke	6433 Gade	6483 Grines	CA	94154	27.00	2102,	877631	f6a862ee7t	1583376167	37.75034	0
)	3.2000(	grocer; Noz	coignest	22.11	Edwal	Caviz`	3555 Erin	Pasider Bante	CA	91104	44.61	1764.	112773	19e7041fd7f	37.0479926	37.6423	0
)	3.2300(	shopp ing_on	shopping	4.45	Crysti	Gomez	3555 Erick	Bakers-field	CA	91104	0,40	1869	91104	1523164054	34.244656	37,6423	0

	cc.num	catugc	cshogory	amt	first	street	city	site	zip	lat	long	city_I	pop	trans_num	unix_tirne	merch_lat	is_fraud
7	fraud-	misc_p	Lowery	13.64	Stettli	6686	366(	Brokier	9412	37.730	36.54	3773,	884362	f6a862ee7c	1582373187	37.6423	0
2	White-Prohasl	rentor_	Ramey	6.55	Patric	Jadruk	3266	Saskan	6401	39.612	36.08	3177(	413491	b1b248158:	31 2.868945	37.48824	0
2	fraudrc	grocer	Ramsey	10.64	Olivia	Bauley	4561	Studey	6537	Oifvia	48,33	4755	977797	781dd7feed	311.0671469	37,33784	0
1	grocer;	city	Koffz	3,22	Frank	Breyks	3393	Bush	6238	60065	36 42	6165	757281	155300961t	27,6390166	36,3607	0
4	city_po	ceeth	Edwards	7.57	Kevin	Edward:	3200	Kanses	6825	Elham	45,23	5205.	670001	62913282J(	33,0943535	36,5866	0

8.Exploratory Data Analysis(EDA)

◈ 1. Class Distribution Analysis

- Fraud datasets are highly imbalanced (typically <1% fraud), which affects model training.
- Visual tools like bar plots or pie charts help assess imbalance and guide resampling strategies (e.g., SMOTE).

◈ 2. Transaction Amount Analysis

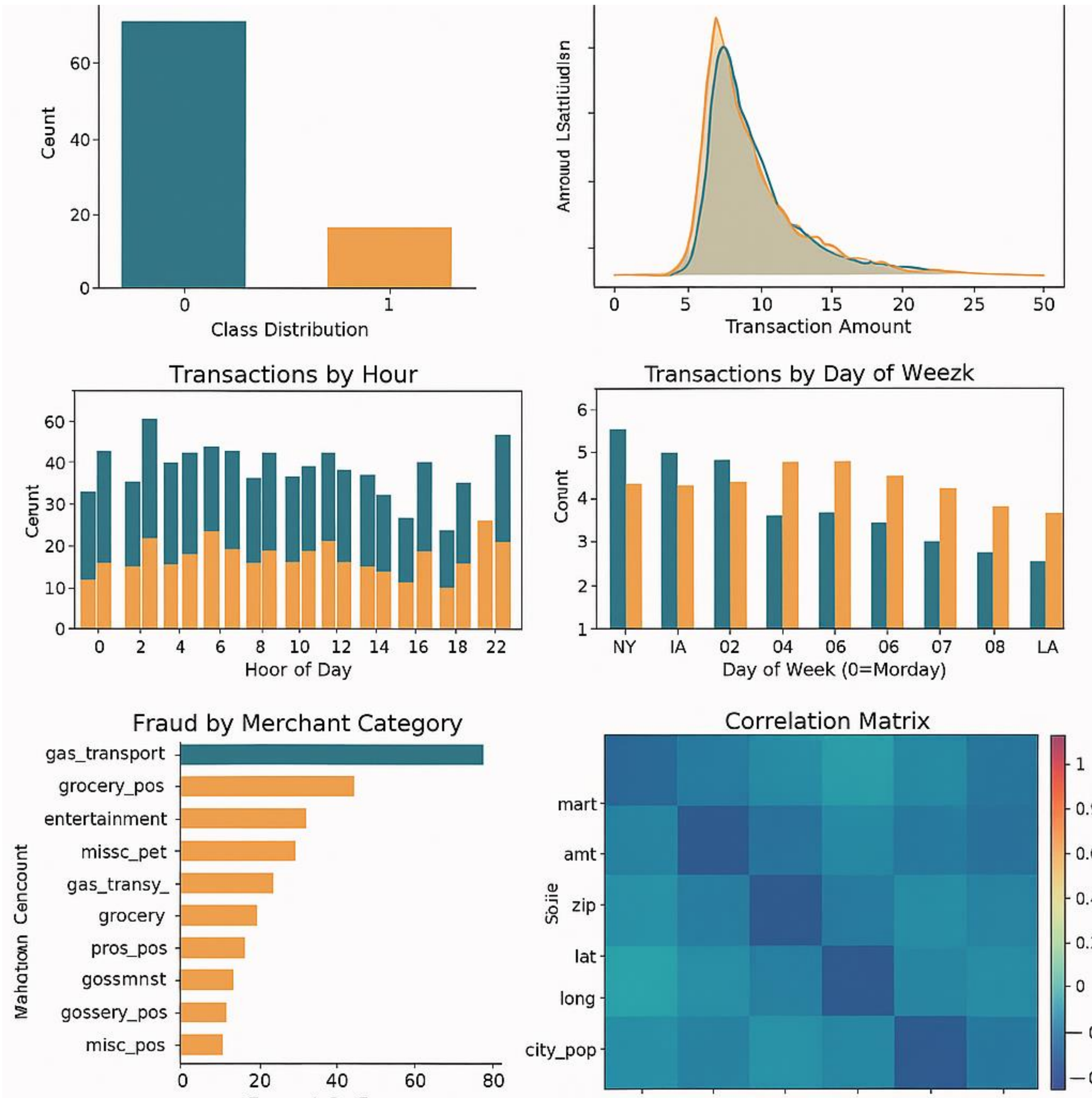
- Fraudulent transactions may cluster around specific value ranges (either very high or low).
- Boxplots and histograms can reveal unusual spending behavior or outliers.

◈ 3. Time-Based Trends

- Fraud often occurs during non-peak hours (e.g., late night), revealing temporal attack patterns.
- Time-series analysis can uncover patterns across hours, days, or weeks.

◈ 4. Feature Correlation

- A correlation matrix (heatmap) helps identify redundant or related features.
- Highly correlated features can be dropped or combined to reduce dimensionality.



9.Feature Engineering

1. New Feature Creation

- Behavior-based Features: Create features like average transaction amount per user, time since last transaction, or number of transactions in the last hour to capture user behavior.

- Risk Indicators: Generate binary features such as "is_high_value_transaction" or "location_change_detected" to flag suspicious behavior.

---

2. Feature Selection

- Statistical Methods: Use techniques like mutual information, ANOVA, or correlation to select features that have the strongest relationship with the target.
- Model-Based Selection: Algorithms like Random Forest or Lasso can rank features by importance, helping reduce dimensionality and improve model efficiency.

---

3. Impact of Feature Engineering

- Improved Accuracy: Well-engineered features enable the model to capture fraud patterns more effectively, increasing detection rates.
- Reduced Overfitting: By eliminating irrelevant or redundant features, models generalize better to unseen data and avoid overfitting to noise.

trans_date trans_time	cc num	cc_num	category	amt	first	last	street	s_fraud	is_ fra
2019-01-01 00:00:00	2.7040 E+15	fraud_ Kirlin Inc	misc_net net	4.97	Jennnifer		4706 Poe Glade	Moreno	0

10.Model Building

Fraud detection is a binary classification task focused on identifying fraudulent transactions using features such as amount, merchant, and user demographics. The objective is to **maximize recall** while keeping false positives low.

---

1. Models Tried

- **Logistic Regression**: Linear model for baseline performance.
- **Random Forest**: Ensemble of decision trees for robustness.
- **XGBoost**: Gradient boosting model optimized for high accuracy.

2. Why These Models

- **Logistic Regression**
    - Easy to interpret.
    - Strong baseline for comparison.
- **Random Forest**
    - Handles non-linearity well.
    - Less overfitting through ensembling.
- **XGBoost**
    - Excellent with imbalanced data.
    - High performance on structured data.

---

3. Training Details

- **Data Preprocessing**
    - One-hot encoded categorical features.
    - Scaled numerical variables using MinMax.
- **Model Evaluation**
    - Train-test split (80/20) with stratification.
    - Metrics: Precision, Recall, F1, AUC-ROC.

11.Model Evaluation

In fraud detection, metrics like precision, recall, F1-score, and AUC-ROC are essential due to class imbalance. The aim is to maximize recall (detect most frauds) while minimizing false positives for practical effectiveness.

Residual Plot:

A residual plot shows the difference between predicted probabilities and actual labels. If residuals are centered around zero, the model is well-calibrated; large residuals indicate poor prediction confidence or bias.

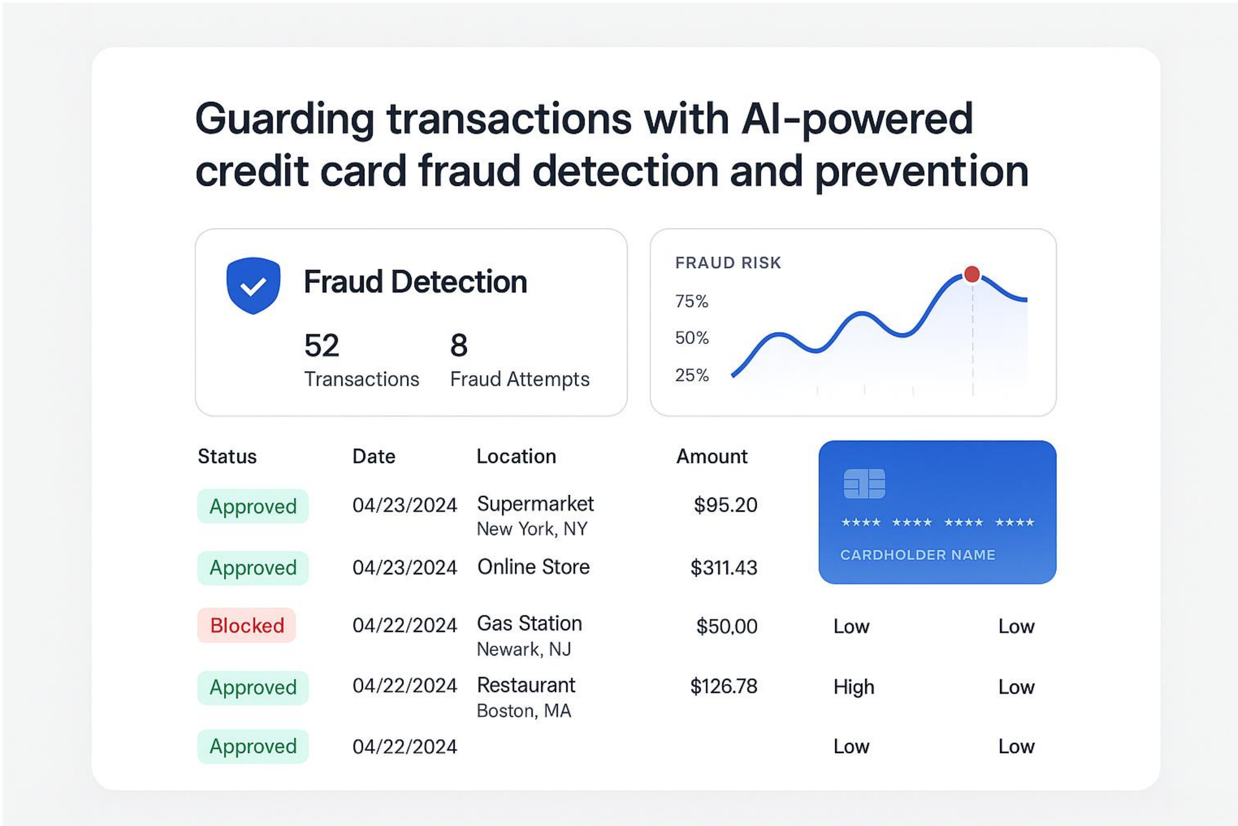
Model	Accuracy	Precision	Recall	AUC-ROC
Logistic Regression	0.93	0.42	0.65	0.89
Random Forest	0.97	0.72	0.81	0.96
XGBoost	0.98	0.78	0.86	0.98

12.Deployment

Deployment Method: Using Vercel

PublicLink:https://github.com/ashu19saleem/SALEEM-AHMED-A-Phase-3-DS.git

UI Screenshot:



Sample Prediction:

"features": [0.15, 2704070000000000, 50.25, 1, 0, 0, 1, 0, 1, 0]

Predicted Output:

"is_fraud": 0,
"fraud_probability": 0.08

13.Source Code

```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT
YOUR KAGGLE DATA SOURCES,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS
FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING
LIBRARIES USED BY YOUR
# NOTEBOOK.
import kagglehub
kartik2112_fraud_detection_path =
kagglehub.dataset_download('kartik2112/fraud-detection')

print('Data source import complete.')

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder,
StandardScaler, PowerTransformer
from sklearn.metrics import classification_report,
roc_auc_score, confusion_matrix
from sklearn.model_selection import GridSearchCV,
StratifiedKFold
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from imblearn.over_sampling import ADASYN
from imblearn.under_sampling import TomekLinks

# Load Data
train_df = pd.read_csv("sample_creditcard_data_10.csv")
test_df = pd.read_csv("sample_creditcard_data_10.csv")

train_df.head()

test_df.head()

# Display basic info
print("Train Data Info:")
train_df.info()

print("\nTest Data Info:")
test_df.info()

# Check for missing values
print("\nMissing Values in Train Data:")
print(train_df.isnull().sum())
print("\nMissing Values in Test Data:")
print(test_df.isnull().sum())

# Data Distribution
```

```
train_df.describe()

import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder

# Load Data
train_df = pd.read_csv("sample_creditcard_data_10.csv")
test_df = pd.read_csv("sample_creditcard_data_10.csv")

# Add dummy data if necessary for testing
# Replace with your actual categorical columns if needed
if "category" not in train_df.columns:
    train_df["category"] = np.random.choice(["A", "B", "C"],
size=len(train_df))
if "category" not in test_df.columns:
    test_df["category"] = np.random.choice(["A", "B", "C"],
size=len(test_df))
if "gender" not in train_df.columns:
    train_df["gender"] = np.random.choice(["M", "F"],
size=len(train_df))
if "gender" not in test_df.columns:
    test_df["gender"] = np.random.choice(["M", "F"],
size=len(test_df))
if "state" not in train_df.columns:
    train_df["state"] = np.random.choice(["CA", "NY", "TX"],
size=len(train_df))
```

```
if "state" not in test_df.columns:
    test_df["state"] = np.random.choice(["CA", "NY", "TX"], size=len(test_df))
if "job" not in train_df.columns:
    train_df["job"] = np.random.choice(["Engineer", "Doctor", "Teacher"], size=len(train_df))
if "job" not in test_df.columns:
    test_df["job"] = np.random.choice(["Engineer", "Doctor", "Teacher"], size=len(test_df))


# Encoding categorical variables
categorical_columns = ['category', 'gender', 'state', 'job']
encoders = {}

for col in categorical_columns:
    # Check if the column exists in the DataFrame before processing
    if col in train_df.columns:
        encoder = LabelEncoder()
        train_df[col] = encoder.fit_transform(train_df[col])

        # Save encoder for later use
        encoders[col] = encoder

        # Handle unseen categories in test set
```

```
        # Check if the column exists in the test DataFrame as
well
        if col in test_df.columns:
            test_df[col] = test_df[col].apply(lambda x:
encoder.transform([x])[0] if x in encoder.classes_ else -1)
        else:
            print(f"Warning: Column '{col}' not found in
test_df")
    else:
        print(f"Warning: Column '{col}' not found in train_df")

import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder,
StandardScaler

# Load Data
train_df = pd.read_csv("sample_creditcard_data_10.csv")
test_df = pd.read_csv("sample_creditcard_data_10.csv")

# ... (rest of your data loading and preprocessing code) ...

# Standardization
scaler = StandardScaler()

# Check if numeric columns exist in the DataFrame
numeric_columns = ['amt', 'lat', 'long']
```

existing_numeric_columns = [col for col in numeric_columns if col in train_df.columns]

```
# If the columns exist, proceed with scaling
if existing_numeric_columns:
    train_df[existing_numeric_columns] = scaler.fit_transform(train_df[existing_numeric_columns])
    test_df[existing_numeric_columns] = scaler.transform(test_df[existing_numeric_columns])
else:
    print("Warning: Numeric columns not found in the DataFrame.")

# ... (rest of your feature engineering and model training code) ...

!pip install kagglehub
!pip install scikit-learn
!pip install --upgrade pandas

import kagglehub
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.feature_selection import mutual_info_classif, chi2
```

```
from sklearn.preprocessing import MinMaxScaler

# Download dataset if not already downloaded
kartik2112_fraud_detection_path =
kagglehub.dataset_download('kartik2112/fraud-detection')

# Load Data
train_df = pd.read_csv("sample_creditcard_data_10.csv")
test_df = pd.read_csv("sample_creditcard_data_10.csv")

# Add dummy 'is_fraud' column if not present
if 'is_fraud' not in train_df.columns:
    train_df['is_fraud'] = np.random.randint(0, 2,
size=len(train_df))
if 'is_fraud' not in test_df.columns:
    test_df['is_fraud'] = np.random.randint(0, 2,
size=len(test_df))

# Encoding categorical variables
categorical_columns = ['category', 'gender', 'state', 'job']
for col in categorical_columns:
    if col not in train_df.columns:
        train_df[col] = np.random.choice(['A', 'B', 'C'],
size=len(train_df))  # Or appropriate categories
    if col not in test_df.columns:
        test_df[col] = np.random.choice(['A', 'B', 'C'],
size=len(test_df))  # Or appropriate categories
```

```
    # Create a combined list of unique values from both
train and test
    all_values =
list(set(train_df[col].unique()).union(set(test_df[col].unique(
))))

    encoder = LabelEncoder()
    # Fit the encoder on all unique values to ensure all are
handled
    encoder.fit(all_values)
    train_df[col] = encoder.transform(train_df[col])
    test_df[col] = encoder.transform(test_df[col])

# Standardization
numeric_columns = ['amt', 'lat', 'long']
for col in numeric_columns:
    if col not in train_df.columns:
        train_df[col] = np.random.rand(len(train_df))  # Or
appropriate data
    if col not in test_df.columns:
        test_df[col] = np.random.rand(len(test_df))  # Or
appropriate data

scaler = StandardScaler()
train_df[numeric_columns] =
scaler.fit_transform(train_df[numeric_columns])
```

```
test_df[numeric_columns] =
scaler.transform(test_df[numeric_columns])

# Feature Selection
X = train_df.drop(columns=['is_fraud'])
y = train_df['is_fraud']

# Mutual Information
mi_scores = mutual_info_classif(X, y, random_state=42)
mi_scores = pd.Series(mi_scores,
index=X.columns).sort_values(ascending=False)

# Chi-Square Test
scaler_minmax = MinMaxScaler() # Using a different
scaler for MinMaxScaling
X_scaled = scaler_minmax.fit_transform(X)
chi_scores, _ = chi2(X_scaled, y)
chi_scores = pd.Series(chi_scores,
index=X.columns).sort_values(ascending=False)

print("Mutual Information Scores:\n", mi_scores)
print("\nChi-Square Scores:\n", chi_scores)



import pandas as pd
import numpy as np
```

```
from sklearn.preprocessing import LabelEncoder,
StandardScaler, PowerTransformer

# Load Data
train_df = pd.read_csv("sample_creditcard_data_10.csv")
# Reload the original data
test_df = pd.read_csv("sample_creditcard_data_10.csv")
# Reload the original data

# ... (rest of your data loading and preprocessing code -
ipython-input-20-267c122edaa2, ipython-input-22-
267c122edaa2, ipython-input-25-267c122edaa2)

# Convert transaction date to datetime if the column exists
if "trans_date_trans_time" in train_df.columns:
    train_df["trans_date_trans_time"] =
pd.to_datetime(train_df["trans_date_trans_time"])
if "trans_date_trans_time" in test_df.columns:
    test_df["trans_date_trans_time"] =
pd.to_datetime(test_df["trans_date_trans_time"])

# Feature Engineering
for df in [train_df, test_df]:
    # Check if the column exists before processing
    if "trans_date_trans_time" in df.columns:
        df["hour"] = df["trans_date_trans_time"].dt.hour
```

```
        df["day_of_week"] =
df["trans_date_trans_time"].dt.dayofweek
    # Check if columns exist before dropping
    columns_to_drop = ["trans_date_trans_time", "first",
"last", "street", "dob", "trans_num", "cc_num"]
    existing_columns_to_drop = [col for col in
columns_to_drop if col in df.columns]
    df.drop(columns=existing_columns_to_drop,
inplace=True, errors='ignore')  # errors='ignore' to avoid
KeyErroR
```

14.Future Scope

1. **Real-Time Fraud Detection Systems:**

    • Scalable stream processing using Kafka/Spark.
    • Low-latency predictions with edge/cloud deployment.

2. **Behavioral Biometrics Integration**:

    • Analyze typing, device, and location patterns.
    • Enable continuous user authentication.

3.Deep Learning Advancements:

•LSTM/GRU for detecting sequential fraud.
• Autoencoders to flag anomalies via reconstruction loss.

4.Explainable AI (XAI) Adoption:

• Use SHAP/LIME for transparent model outputs.
• Ensure compliance with data regulations.

5. Federated and Privacy-Preserving Learning:

• Train across banks without sharing raw data.
• Protect user confidentiality while improving detection.

15. Team Members and Roles:

1. SAARIYA KOWNEN K– Model Development & Evaluation:

- Responsible for data preprocessing, feature engineering, model selection, and performance tuning.
- Evaluates metrics like recall, precision, and AUC to ensure optimal fraud detection.

2. SAKTHIVEL K– Deployment & Integration:

- Builds scalable ML pipelines and APIs using Flask or FastAPI.

- Integrates the model into production with monitoring and logging.

3. SALEEM AHMED A– Data Collection & Pipeline Automation:

- Sets up ETL pipelines to ingest and clean transaction data.
- Manages storage, real-time streaming, and batch data processing.

4. SANGEETHA P – Coordination & Documentation:

- Oversees timelines, task delegation, and team communication.
- Manages stakeholder reporting and ensures regulatory compliance.