MCA Final Year Project (Review I)

AI-Driven Fraud Detection: Securing Banking Transactions

Submitted to the Presidency University, Bengaluru in partial fulfillment for the award of the degree of Master of Computer Applications(MCA)

Project Number : 212

Name	Roll Number
Vijita Narayan Nayak	20232MCA0211

Under the supervision of

Mr. Sakthi S

Asst. Prof, Department of SCSE

School of Computer Science and Engineering

Content

- Problem Statement
- Literature Survey
- Tools and Technologies to be used
- Github Link
- Timeline of the Project
- References

Problem Statement

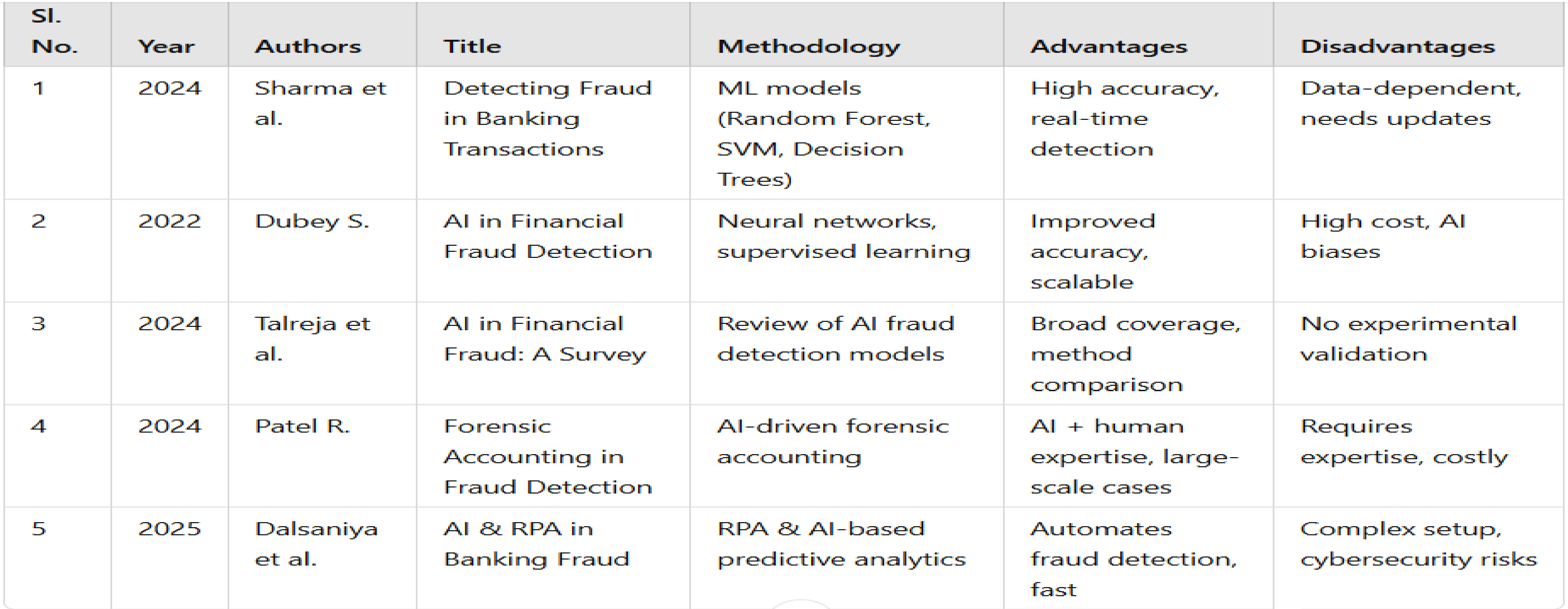
- **Rising Fraud Threat:** Financial fraud in banking leads to significant monetary losses and reduced customer trust.

- **Limitations of Current Systems:** Traditional rule-based methods fail to detect evolving fraud patterns, causing false positives and undetected fraud.

- **AI-Powered Detection** : Machine Learning models analyze transaction patterns in real time to identify anomalies and predict fraudulent activities.

- **Enhanced Security & Accuracy:** The AI-driven system improves fraud detection efficiency, minimizes false alarms, and secures banking transactions.

Literature Review

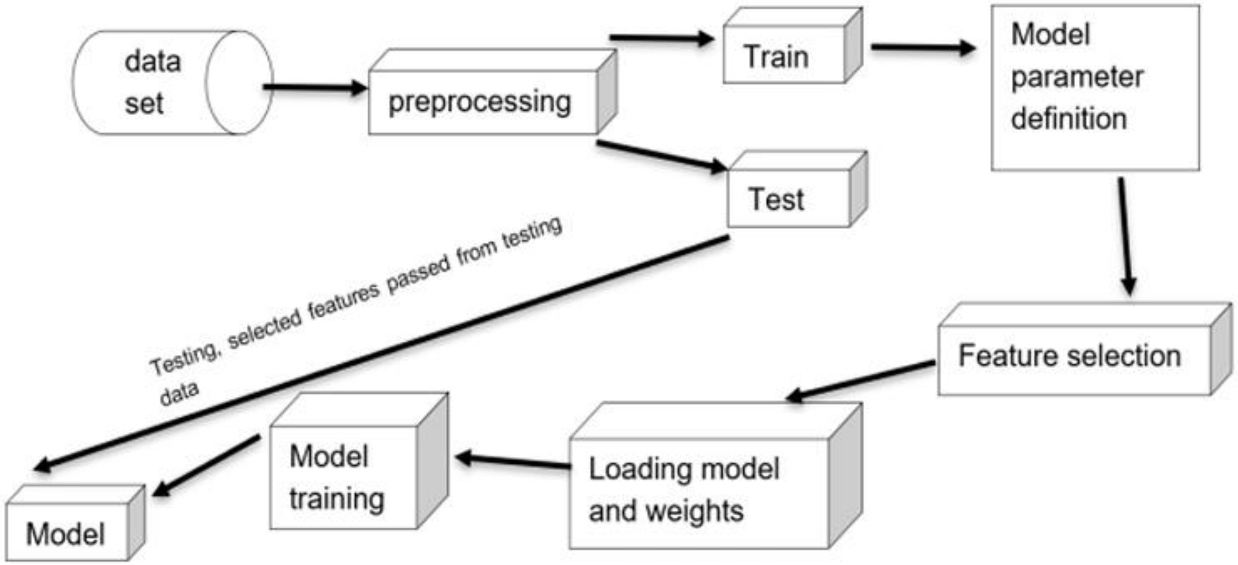


Sl. No.	Year	Authors	Title	Methodology	Advantages	Disadvantages
1	2024	Sharma et al.	Detecting Fraud in Banking Transactions	ML models (Random Forest, SVM, Decision Trees)	High accuracy, real-time detection	Data-dependent, needs updates
2	2022	Dubey S.	AI in Financial Fraud Detection	Neural networks, supervised learning	Improved accuracy, scalable	High cost, AI biases
3	2024	Talreja et al.	AI in Financial Fraud: A Survey	Review of AI fraud detection models	Broad coverage, method comparison	No experimental validation
4	2024	Patel R.	Forensic Accounting in Fraud Detection	AI-driven forensic accounting	AI + human expertise, large-scale cases	Requires expertise, costly
5	2025	Dalsaniya et al.	AI & RPA in Banking Fraud	RPA & AI-based predictive analytics	Automates fraud detection, fast	Complex setup, cybersecurity risks

Module Design

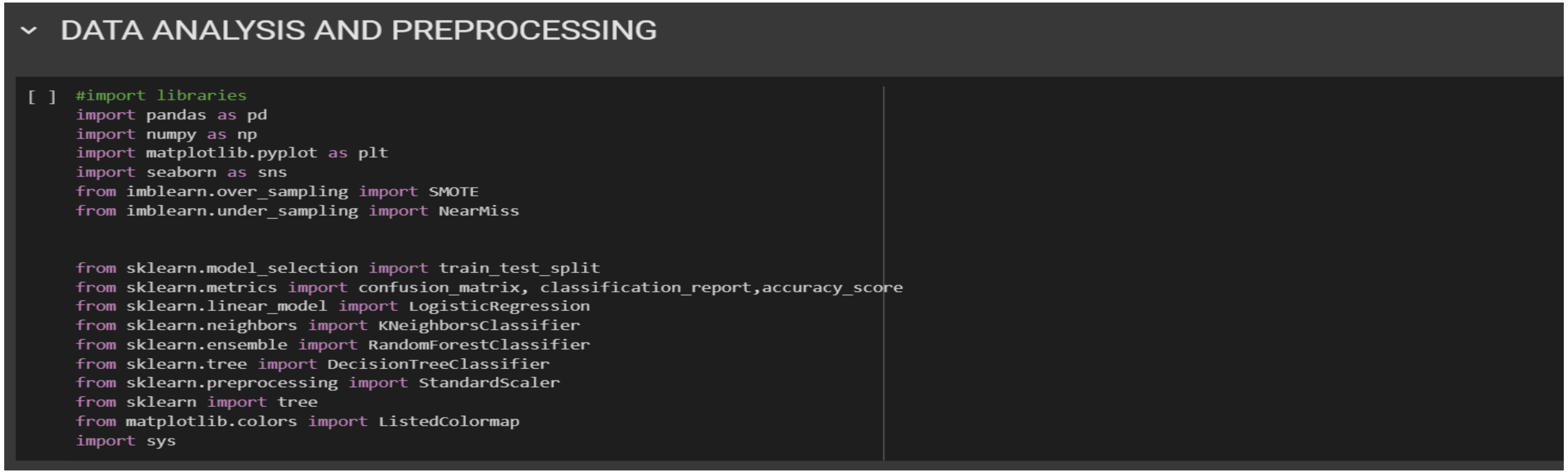
Brief overview of the project’s modular architecture

The fraud detection system is designed using a modular architecture to ensure **scalability, efficiency, and flexibility**. Each module is responsible for a specific function, working together to detect fraudulent transactions in real time.


data set
preprocessing
Train
Model parameter definition
Test
Testing, selected features passed from testing data
Feature selection
Loading model and weights
Model training
Model

Modular Breakdown

Module 1: Data Analysis and Data Preprocessing



DATA ANALYSIS AND PREPROCESSING

```
[ ]  #import libraries
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     from imblearn.over_sampling import SMOTE
     from imblearn.under_sampling import NearMiss


     from sklearn.model_selection import train_test_split
     from sklearn.metrics import confusion_matrix, classification_report,accuracy_score
     from sklearn.linear_model import LogisticRegression
     from sklearn.neighbors import KNeighborsClassifier
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.preprocessing import StandardScaler
     from sklearn import tree
     from matplotlib.colors import ListedColormap
     import sys
```

Importance:

- Improves model performance by reducing noise in data.

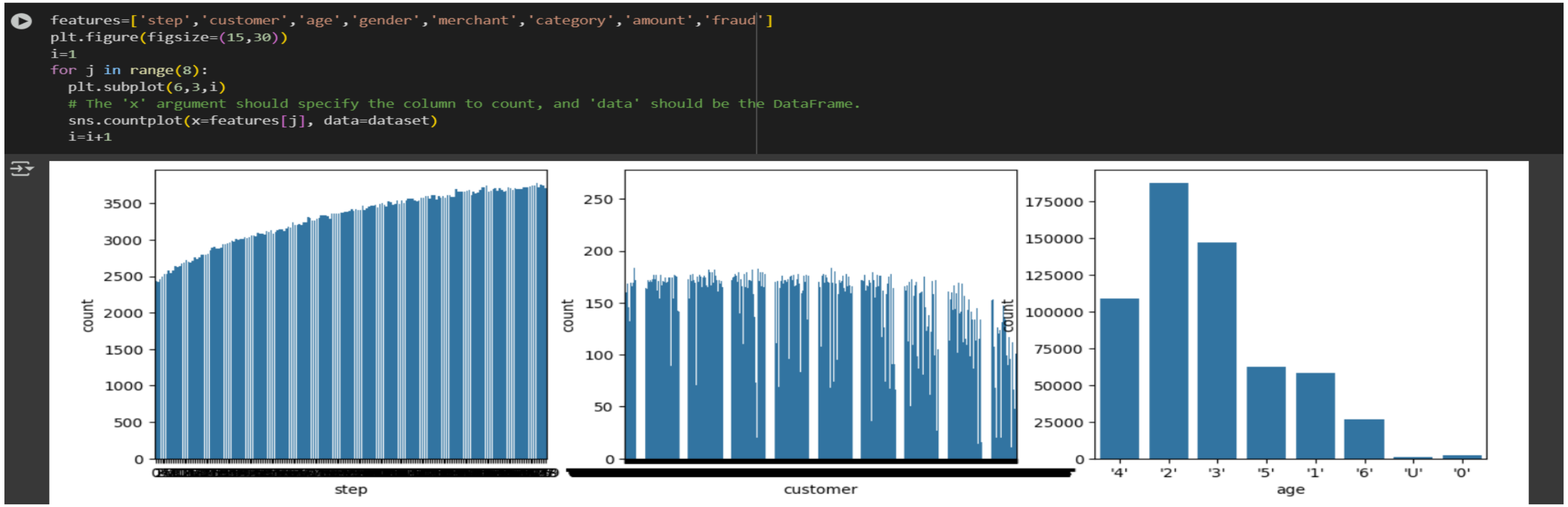
Module 2: Feature Engineering & Selection

Functionality:

- Selects the most relevant features for fraud detection models.
- Uses statistical methods and ML techniques to enhance detection.

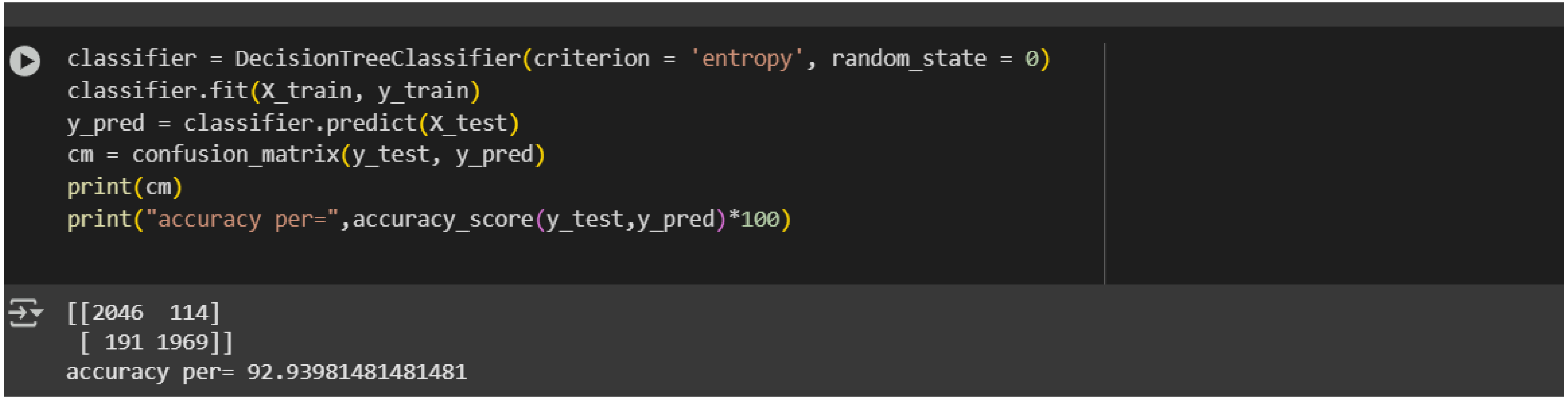
Importance:

- Improves fraud detection accuracy by refining critical features



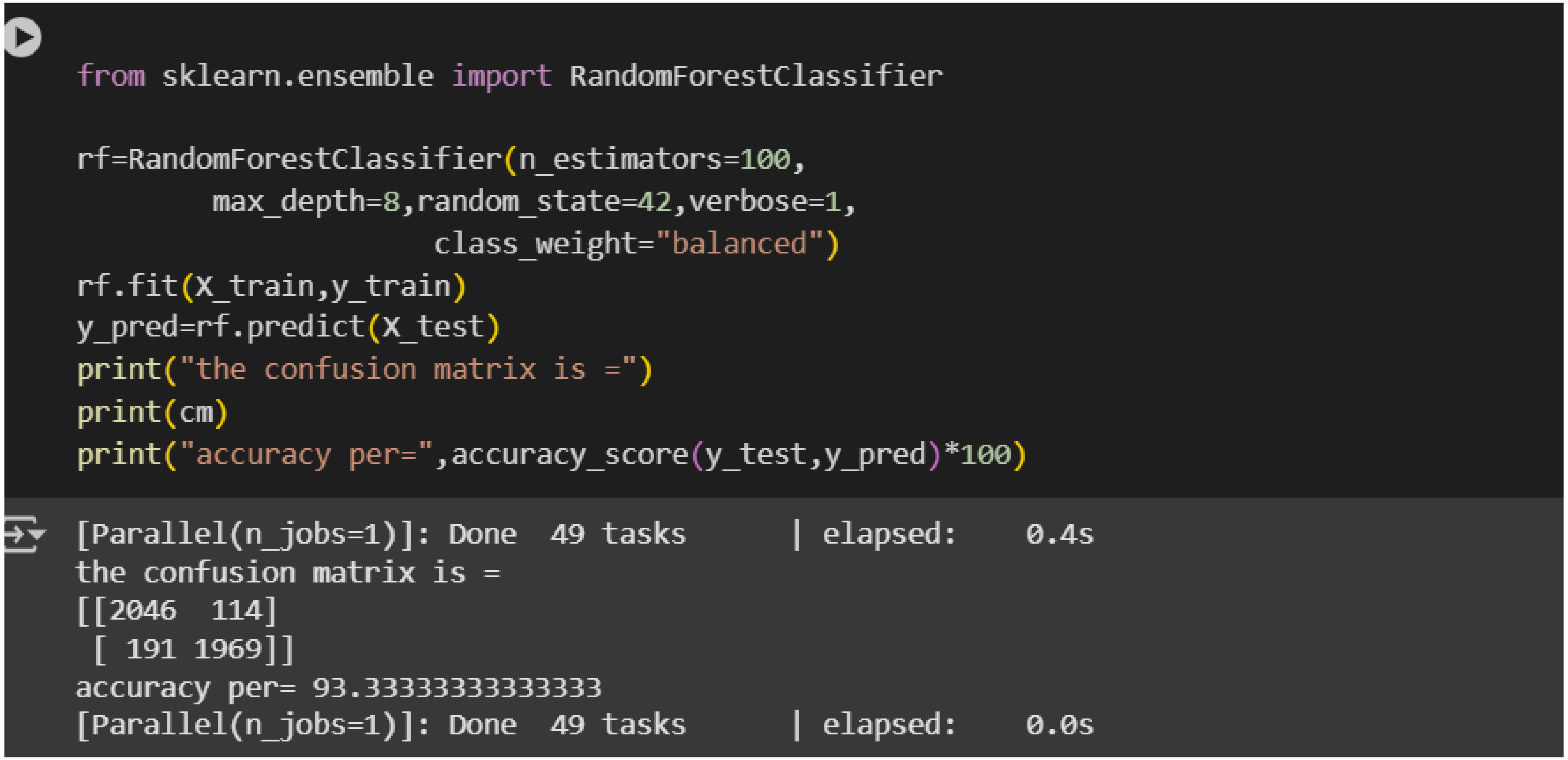
```
features=['step','customer','age','gender','merchant','category','amount','fraud']
plt.figure(figsize=(15,30))
i=1
for j in range(8):
  plt.subplot(6,3,i)
  # The 'x' argument should specify the column to count, and 'data' should be the DataFrame.
  sns.countplot(x=features[j], data=dataset)
  i=i+1
```

Module 3: Machine Learning-Based Fraud Detection



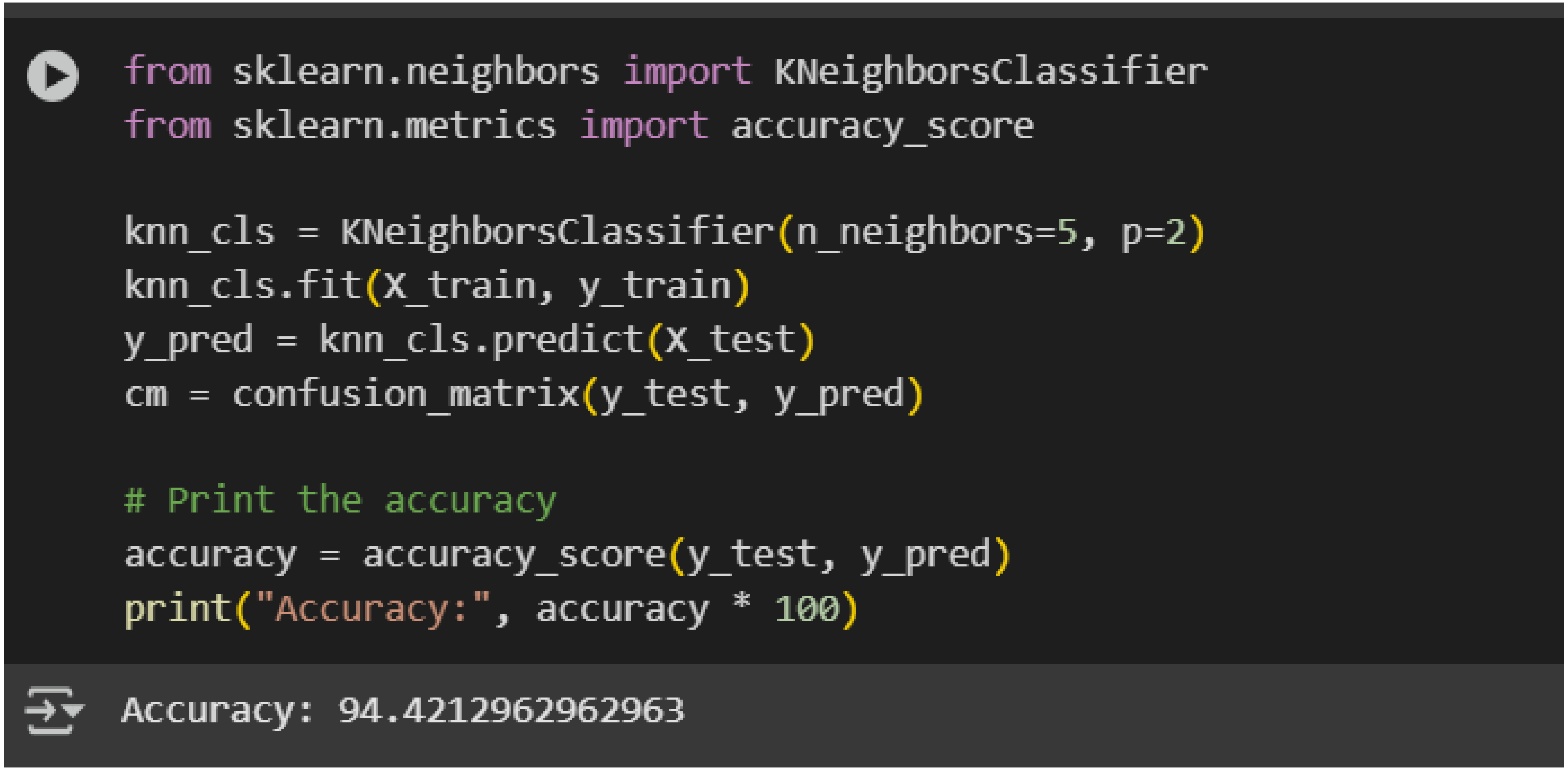
```
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print(cm)
print("accuracy per=",accuracy_score(y_test,y_pred)*100)
```

```
[[2046  114]
 [ 191 1969]]
accuracy per= 92.93981481481481
```



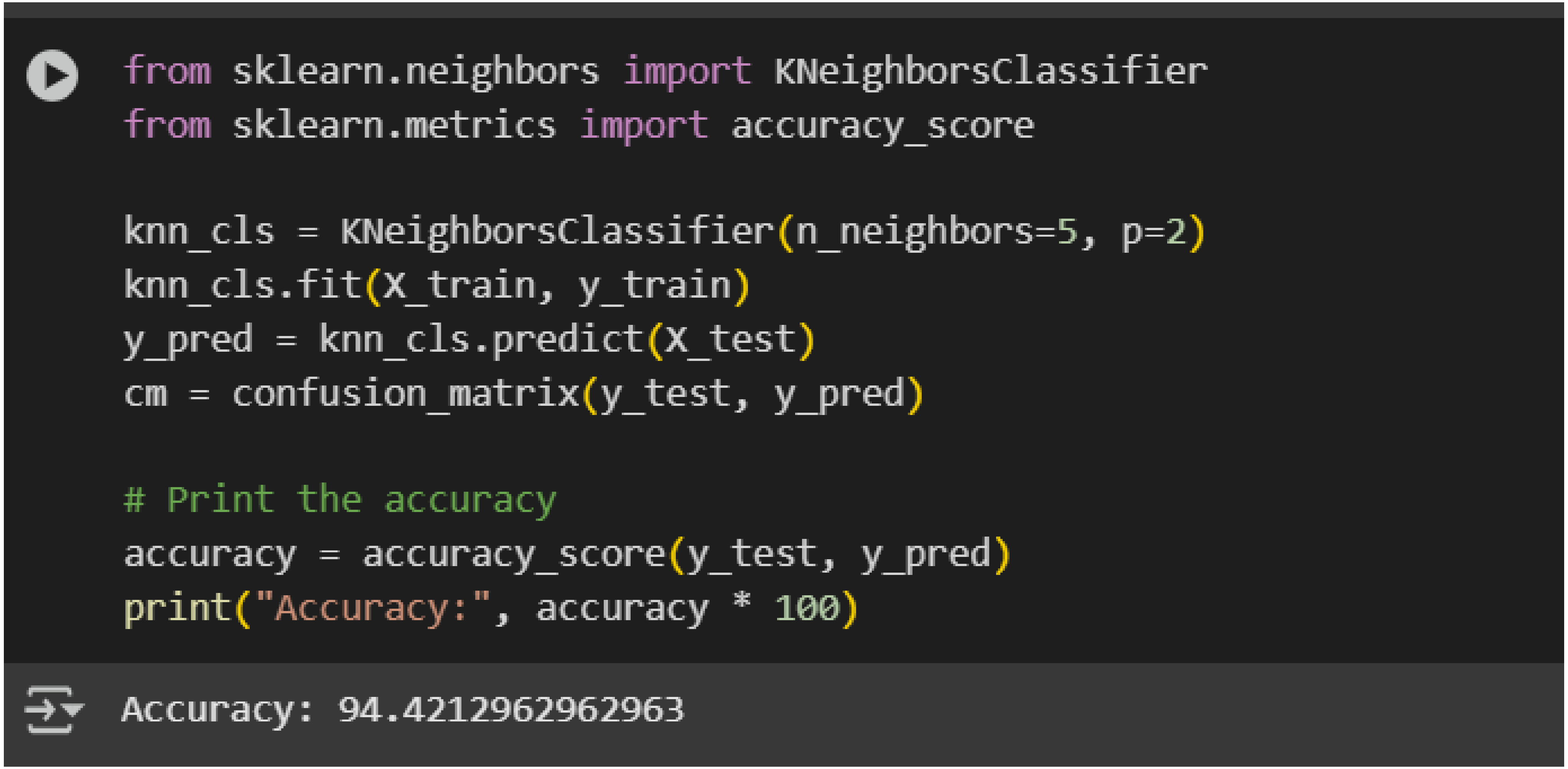
```
from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier(n_estimators=100,
        max_depth=8,random_state=42,verbose=1,
                     class_weight="balanced")
rf.fit(X_train,y_train)
y_pred=rf.predict(X_test)
print("the confusion matrix is =")
print(cm)
print("accuracy per=",accuracy_score(y_test,y_pred)*100)
```

```
[Parallel(n_jobs=1)]: Done  49 tasks      | elapsed:    0.4s
the confusion matrix is =
[[2046  114]
 [ 191 1969]]
accuracy per= 93.33333333333333
[Parallel(n_jobs=1)]: Done  49 tasks      | elapsed:    0.0s
```



```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

knn_cls = KNeighborsClassifier(n_neighbors=5, p=2)
knn_cls.fit(X_train, y_train)
y_pred = knn_cls.predict(X_test)
cm = confusion_matrix(y_test, y_pred)

# Print the accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy * 100)
```

```
Accuracy: 94.4212962962963
```



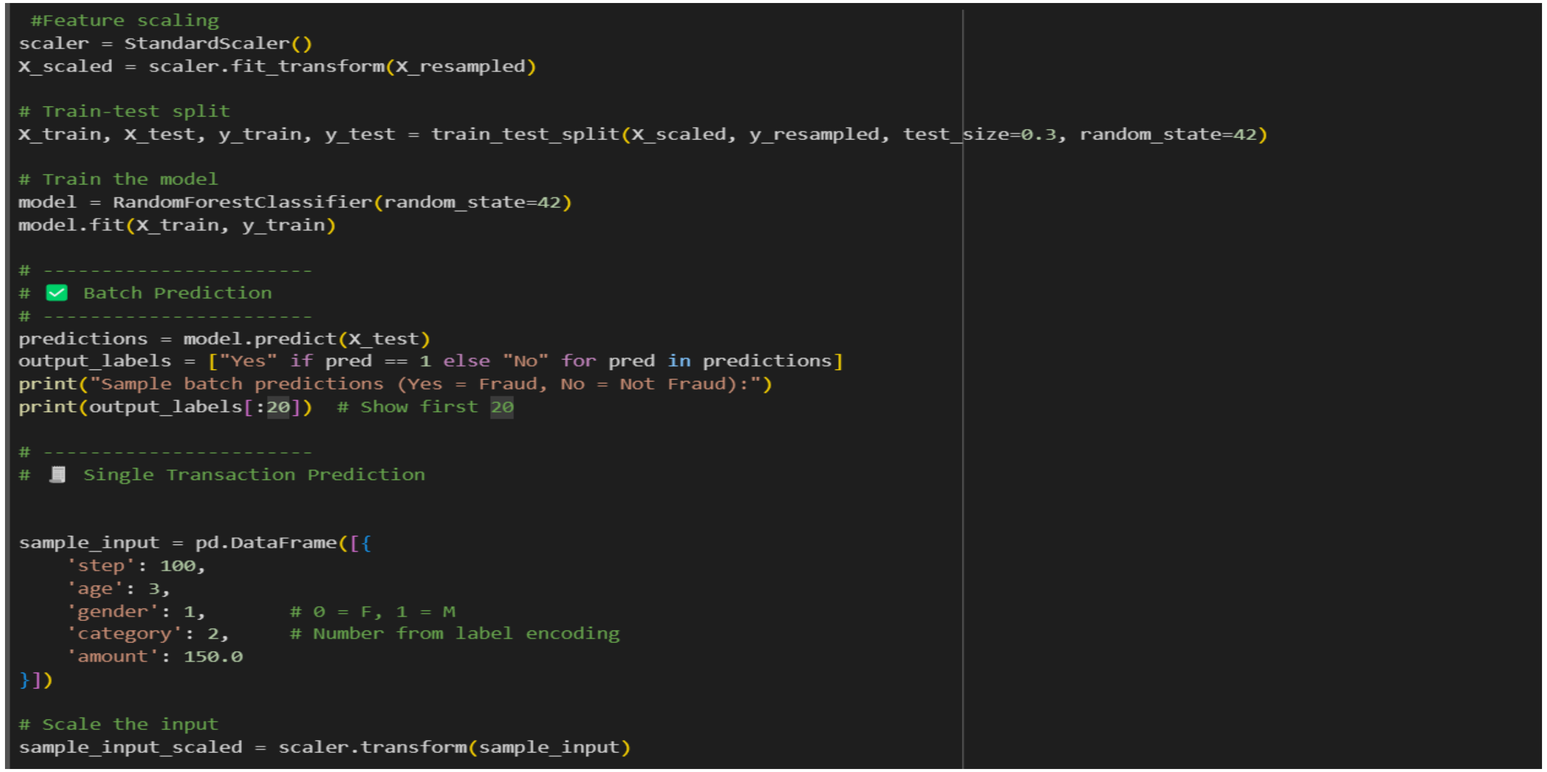
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

knn_cls = KNeighborsClassifier(n_neighbors=5, p=2)
knn_cls.fit(X_train, y_train)
y_pred = knn_cls.predict(X_test)
cm = confusion_matrix(y_test, y_pred)

# Print the accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy * 100)
```

```
Accuracy: 94.4212962962963
```

Module 4: Behavioral Analysis and Prediction Results

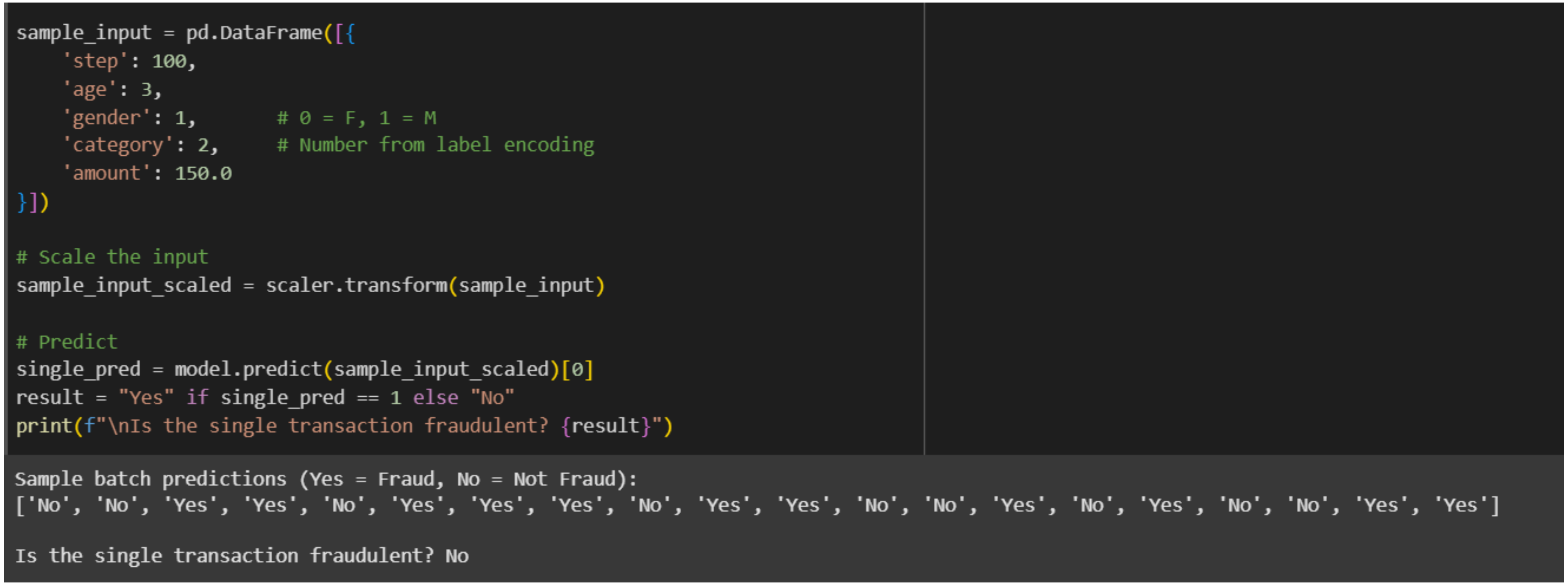
Functionality:

- Implements predefined fraud rules .



```
 #Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_resampled)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_resampled, test_size=0.3, random_state=42)

# Train the model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# -----------------------
# ✅ Batch Prediction
# -----------------------
predictions = model.predict(X_test)
output_labels = ["Yes" if pred == 1 else "No" for pred in predictions]
print("Sample batch predictions (Yes = Fraud, No = Not Fraud):")
print(output_labels[:20])  # Show first 20

# -----------------------
# 🧾 Single Transaction Prediction


sample_input = pd.DataFrame([{
    'step': 100,
    'age': 3,
    'gender': 1,       # 0 = F, 1 = M
    'category': 2,     # Number from label encoding
    'amount': 150.0
}])

# Scale the input
sample_input_scaled = scaler.transform(sample_input)
```

RESULT :



```
sample_input = pd.DataFrame([{
    'step': 100,
    'age': 3,
    'gender': 1,        # 0 = F, 1 = M
    'category': 2,      # Number from label encoding
    'amount': 150.0
}])

# Scale the input
sample_input_scaled = scaler.transform(sample_input)

# Predict
single_pred = model.predict(sample_input_scaled)[0]
result = "Yes" if single_pred == 1 else "No"
print(f"\nIs the single transaction fraudulent? {result}")
```

```
Sample batch predictions (Yes = Fraud, No = Not Fraud):
['No', 'No', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No', 'No', 'Yes', 'No', 'Yes', 'No', 'No', 'Yes', 'Yes']

Is the single transaction fraudulent? No
```

Importance:

- Strengthens fraud detection by combining AI with traditional banking security measures.
- Reduces false positives by considering user habits.

Tools And Technologies To Be Used

1. Development Tools:

- **Google Colab / Jupyter Notebook** – used for coding, testing, and visualizing machine learning models.

2. Programming Language:

- **Python** – The predominant language for implementing machine learning models and data analysis in fraud detection systems..

3. Frameworks & Libraries:

- **Pandas & Numpy** – For data manipulation analysis and numerical computations.
- **Scikit-Learn --** Offers simple and efficient tools for predictive data analysis.
- **TensorFlow/Keras --** Used for building and training deep learning models.

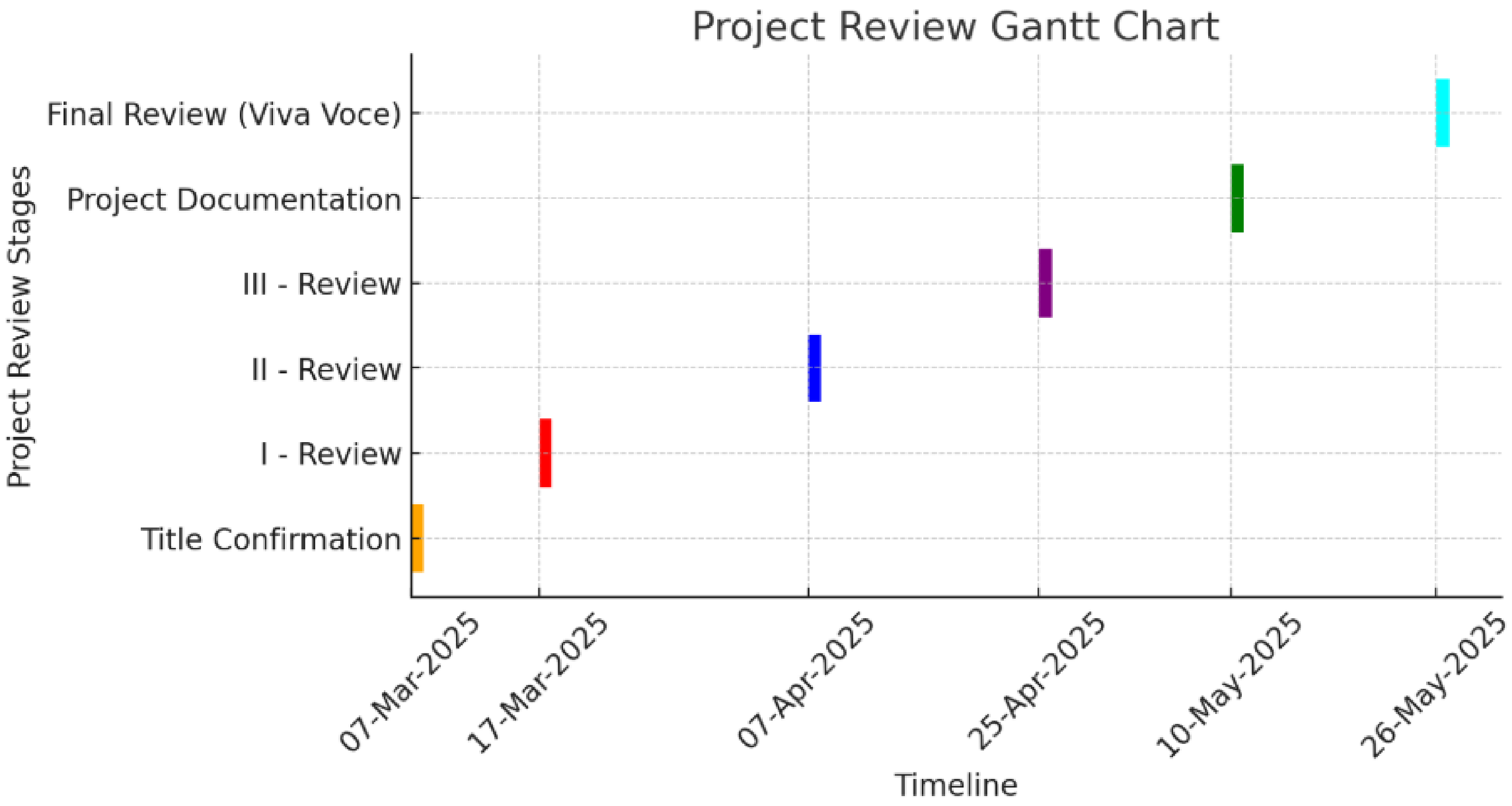
- **4. Additional Tools:**

- **Matplotlib/Seaborn:** For creating static, animated, and interactive visualizations.

GITHUB LINK

- https://github.com/VijitaNayak/AI-Driven-Fraud-Detection-Banking-Transaction

Timeline of the Project



References

1. M. S. Islam and N. Rahman, "AI-Driven Fraud Detections in Financial Institutions: A Comprehensive Study," *Journal of Computer Science and Technology Studies*,vol. 7, no.   1, pp. 100–112, 2025.
2. S. Dubey, "Artificial Intelligence in Financial Fraud Detection " *Innovative Research Thoughts*, vol. 8, no. 4, 2022.
3. P. Kamuangu, "A Review on Financial Fraud Detection using AI and Machine Learning," *Journal of Economics, Finance and Accounting Studies*, vol. 6, no. 1, pp. 67–77, Feb. 2024.
4. A. Dalsaniya, K. Patel, and P. R. Swaminarayan, "Challenges and Opportunities: Implementing RPA and AI in Fraud Detection in the Banking Sector," *World Journal of Advanced Research and Reviews*, vol. 25, no. 01, pp. 296–308, 2025.
5. S. Kalisetty, C. Pandugula, L. R. K. Sondinti, G. Mallesham, and P. R. S. Rani, "AI-Driven Fraud Detection Systems: Enhancing Security in Card-Based Transactions Using Real-Time Analytics," *Journal of Electrical Systems*, vol. 20, no. 11s, 2024.

Thank
You!