


```

}

results = []

for name, model in models.items():
    model.fit(X_train_test, y_train_test)
    y_pred = model.predict(X_test)

    print(f'({name}) ({name}) ({name})')
    print(classification_report(y_test, y_pred, target_names=labels['Category'].classes))

    results.append({
        'Model': name,
        'Accuracy': accuracy_score(y_test, y_pred)
    })

# Model Accuracy
results_df = pd.DataFrame(results)
print(results_df)
print(results_df)

===== Random Forest =====
precision    recall    f1-score   support

 Fraud      0.35    0.27    0.30    251
 Marketing  0.12    0.12    0.12    172
 Rent       0.11    0.11    0.11    203
 Salary     0.00    0.00    0.00    161
 Supplies   0.24    0.22    0.23    104
 Travel     0.00    0.00    0.00    117
 Utilities  0.20    0.18    0.19    207

 accuracy   0.18    1000
 macro avg  0.15    0.15    0.15    1000
 weighted avg 0.18    0.18    0.18    1000

===== Decision Tree =====
precision    recall    f1-score   support

 Fraud      0.26    0.20    0.23    251
 Marketing  0.14    0.12    0.12    172
 Rent       0.13    0.13    0.13    203
 Salary     0.00    0.00    0.00    161
 Supplies   0.24    0.19    0.21    104
 Travel     0.07    0.10    0.08    117
 Utilities  0.15    0.15    0.15    207

 accuracy   0.16    1000
 macro avg  0.15    0.15    0.15    1000
 weighted avg 0.16    0.16    0.17    1000

===== SVM =====
precision    recall    f1-score   support

 Fraud      0.35    0.05    0.06    251
 Marketing  0.12    0.10    0.11    172
 Rent       0.12    0.15    0.13    203
 Salary     0.00    0.00    0.00    161
 Supplies   0.00    0.00    0.00    104
 Travel     0.00    0.00    0.00    117
 Utilities  0.10    0.00    0.00    207

 accuracy   0.18    1000
 macro avg  0.14    0.11    0.10    1000
 weighted avg 0.10    0.10    0.08    1000

```

```

~ Model Accuracy
0 Random Forest 0.18000
1 Decision Tree 0.16000
2 SVM 0.16000

```

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, roc_auc_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
warnings.filterwarnings("ignore")

# Read data
df = pd.read_csv('content/sample_data/Marketing.csv')

print(df.info())
print(df.describe())
print(df['Category'].value_counts())

# Feature Engineering
df['Merchant'] = df['Merchant'].fillna('Unknown')
df['Description'] = df['Description'].fillna('Unknown')

# Feature Engineering
le = LabelEncoder()
df['Merchant'] = le.fit_transform(df['Merchant'])
df['Description'] = le.fit_transform(df['Description'])
df['Account_Type'] = le.fit_transform(df['Account_Type'])

# Feature Engineering
features = ['Merchant', 'Processing_Time_min', 'Risk_Score',
            'Days_Since_Last', 'Is_Returning', 'Previous_Amount',
            'Merchant_Acc', 'Description_Acc', 'Account_Type_Acc']

X = df[features]

# Feature Scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Model Training
logit = LogisticRegression()
logit.fit(X_scaled, df['Category'])

# Model Evaluation
y_pred = logit.predict(X_scaled)

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)

# Accuracy Score
accuracy = accuracy_score(y_test, y_pred)

# ROC Curve
roc_auc = roc_auc_score(y_test, y_pred)

# Plotting
plt.figure(figsize=(10, 10))

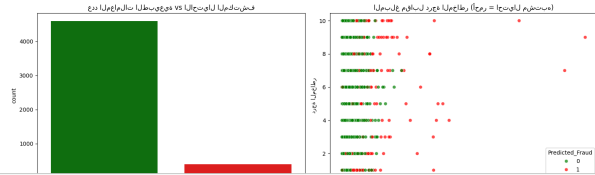
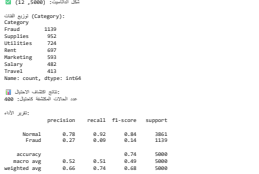
# 1. Confusion Matrix
plt.subplot(2, 2, 1)
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')

# 2. Accuracy Score
plt.subplot(2, 2, 2)
accuracy = accuracy_score(y_test, y_pred)
plt.title('Accuracy Score')

# 3. ROC Curve
plt.subplot(2, 2, 3)
roc_auc = roc_auc_score(y_test, y_pred)
plt.title('ROC Curve')

# 4. Feature Importance
plt.subplot(2, 2, 4)
feature_importance = logit.coef_[0]
plt.title('Feature Importance')

```



```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, roc_auc_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, roc_auc_score
warnings.filterwarnings("ignore")

# Read data
df = pd.read_csv('content/sample_data/Marketing.csv')

print(df.info())
print(df.describe())
print(df['Category'].value_counts())

# Feature Engineering
df['Merchant'] = df['Merchant'].fillna('Unknown')
df['Description'] = df['Description'].fillna('Unknown')

# Feature Engineering
le = LabelEncoder()
df['Merchant'] = le.fit_transform(df['Merchant'])
df['Description'] = le.fit_transform(df['Description'])
df['Account_Type'] = le.fit_transform(df['Account_Type'])

# Feature Engineering
features = ['Merchant', 'Processing_Time_min', 'Risk_Score', 'Days_Since_Last',
            'Is_Returning', 'Previous_Amount', 'Merchant_Acc',
            'Description_Acc', 'Account_Type_Acc']

X = df[features]

# Feature Scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Model Training
logit = LogisticRegression()
logit.fit(X_scaled, df['Category'])

# Model Evaluation
y_pred = logit.predict(X_scaled)

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)

# Accuracy Score
accuracy = accuracy_score(y_test, y_pred)

# ROC Curve
roc_auc = roc_auc_score(y_test, y_pred)

# Plotting
plt.figure(figsize=(10, 10))

# 1. Confusion Matrix
plt.subplot(2, 2, 1)
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')

# 2. Accuracy Score
plt.subplot(2, 2, 2)
accuracy = accuracy_score(y_test, y_pred)
plt.title('Accuracy Score')

# 3. ROC Curve
plt.subplot(2, 2, 3)
roc_auc = roc_auc_score(y_test, y_pred)
plt.title('ROC Curve')

# 4. Feature Importance
plt.subplot(2, 2, 4)
feature_importance = logit.coef_[0]
plt.title('Feature Importance')

```

```

plt.title("الخرات حر فر")
plt.tight_layout()
plt.show()

# ===== 5. مخطط شجرة سدا =====
# مخطط شجرة سدا (Fract)
fract_indices = df[df["Category"] == "Fraud"].index[1:]

print("مخطط شجرة سدا")
for i, row in enumerate(fract_indices):
    print("Row %d: %s" % (i+1, Transaction_ID, df.loc[row, "Transaction_ID"] ...))
    print("Risk Score: %f" % Risk_Score, df.loc[row, "Risk_Score"])

# Force Plot
shop_force_plot(explanator.expected_value,
                shop_value[shop],
                x.index[shop],
                map_predict=True,
                show=False)

plt.title("مخطط شجرة سدا")
plt.show()

```

```

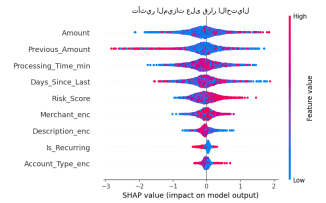
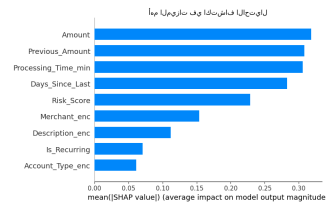
# مخطط شجرة سدا (Fract)
fract_indices = df[df["Category"] == "Fraud"].index[1:]

print("مخطط شجرة سدا")
for i, row in enumerate(fract_indices):
    print("Row %d: %s" % (i+1, Transaction_ID, df.loc[row, "Transaction_ID"] ...))
    print("Risk Score: %f" % Risk_Score, df.loc[row, "Risk_Score"])

# Force Plot
shop_force_plot(explanator.expected_value,
                shop_value[shop],
                x.index[shop],
                map_predict=True,
                show=False)

plt.title("مخطط شجرة سدا")
plt.show()

```



مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

مخطط شجرة سدا (Fract)

