```
# Imports
%pip install matplotlib seaborn --quiet
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, confusion_matrix, classification_report

from sklearn.ensemble import RandomForestClassifier,
ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier

import warnings
warnings.filterwarnings('ignore')
```

```
Note: you may need to restart the kernel to use updated packages.
```

Load and preprocess data

```
# Load dataset
df = pd.read_csv(r"C:\Users\hp\Desktop\Fraud_Payment_Detection2\
Fraud_Detection_Dataset\PS_20174392719_1491204439457_log.csv")

# Encode categorical variables
le = LabelEncoder()
df['type'] = le.fit_transform(df['type'])

# Drop non-numeric, irrelevant, or user ID columns
df.drop(['nameOrig', 'nameDest'], axis=1, inplace=True)

# Features and target
X = df.drop('isFraud', axis=1)
y = df['isFraud']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42, stratify=y)
```

Define and Train Models

```
models = {
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(n_estimators=100),
    "Extra Trees": ExtraTreesClassifier(n_estimators=100),
```

```
    "Support Vector Machine": SVC(kernel='linear', probability=True),
    "XGBoost": XGBClassifier(use_label_encoder=False,
eval_metric='logloss')
}

results = []
```

Evaluate the Models

```
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    acc = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred)
    rec = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)

    results.append({
        'Model': name,
        'Accuracy': round(acc, 4),
        'Precision': round(prec, 4),
        'Recall': round(rec, 4),
        'F1 Score': round(f1, 4)
    })
```

Creating comparision table

```
results_df = pd.DataFrame(results)
results_df = results_df.sort_values(by='F1 Score', ascending=False)
results_df.reset_index(drop=True, inplace=True)
results_df
```

Visualizing results

```
plt.figure(figsize=(10,6))
sns.barplot(x='F1 Score', y='Model', data=results_df,
palette='viridis')
plt.title('Model Comparison (F1 Score)')
plt.xlabel('F1 Score')
plt.ylabel('Model')
plt.tight_layout()
plt.show()
```