

ADVANCED CREDIT CARD FRAUD DETECTION / اكتشاف الاحتيال في البطاقات الائتمانية (متقدم)

```
# =====
# ADVANCED CREDIT CARD FRAUD DETECTION / اكتشاف الاحتيال في البطاقات الائتمانية (متقدم)
# =====

# -----
# STEP 1: Import Libraries
# الخطوة 1: استيراد المكتبات
# -----

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore", category=UserWarning)
# ML Libraries
from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedKFold
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SelectKBest, mutual_info_classif, SelectFromModel
from sklearn.metrics import (classification_report, confusion_matrix,
                             roc_curve, auc, precision_recall_curve, average_precision_score,
                             f1_score, precision_score, recall_score)

# Advanced Model
import xgboost as xgb

# Visualization Settings
sns.set_style("whitegrid")
plt.rcParams['figure.figsize'] = (10, 6)

print("Step 1: Libraries Imported Successfully.")
```

Step 1: Libraries Imported Successfully.

```
# -----
# STEP 2: Load Dataset
# الخطوة 2: تحميل مجموعة البيانات
# -----

# Using TensorFlow's hosted dataset for stability
# لضمان استقرار التحميل TensorFlow استخدام رابط
# Kaggel هذه افضل مجموعة بيانات وجدتها بديلة عن تلك الموجودة في
url = "https://storage.googleapis.com/download.tensorflow.org/data/creditcard.csv"
print(f"Downloading data from {url}...")

try:
    df = pd.read_csv(url)
    print(f"Data Loaded. Shape: {df.shape}")
except Exception as e:
    print(f"Error loading data: {e}")
```

Downloading data from <https://storage.googleapis.com/download.tensorflow.org/data/creditcard.csv>...
Data Loaded. Shape: (284807, 31)

```
# -----
# STEP 3: Exploratory Data Analysis (EDA)
# الخطوة 3: تحليل البيانات الاستكشافي
# -----

# 3.1 Class Imbalance Visualization
# تصوير عدم توازن الفئات
plt.figure(figsize=(6, 4))
sns.countplot(x='Class', data=df, palette='viridis')
plt.title('Class Distribution', fontsize=14)
plt.xlabel('Class (0: Normal, 1: Fraud)')
plt.yscale('log') # Log scale helps visualize the minority class / المقياس اللوغاريتمي يوضح الفئة القليلة
plt.show()

fraud_pct = df['Class'].value_counts(normalize=True)[1] * 100
print(f"Fraud Percentage: {fraud_pct:.4f}%")

# 3.2 Time & Amount Analysis
# تحليل الوقت والمبلغ
```

```
f, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize=(12, 6))
ax1.hist(df.Time[df.Class == 1], bins=50, color='red', alpha=0.6, density=True)
ax1.set_title('Fraud Transaction Time')
ax2.hist(df.Time[df.Class == 0], bins=50, color='blue', alpha=0.3, density=True)
ax2.set_title('Normal Transaction Time')
plt.xlabel('Time (Sec)')
plt.show()
```

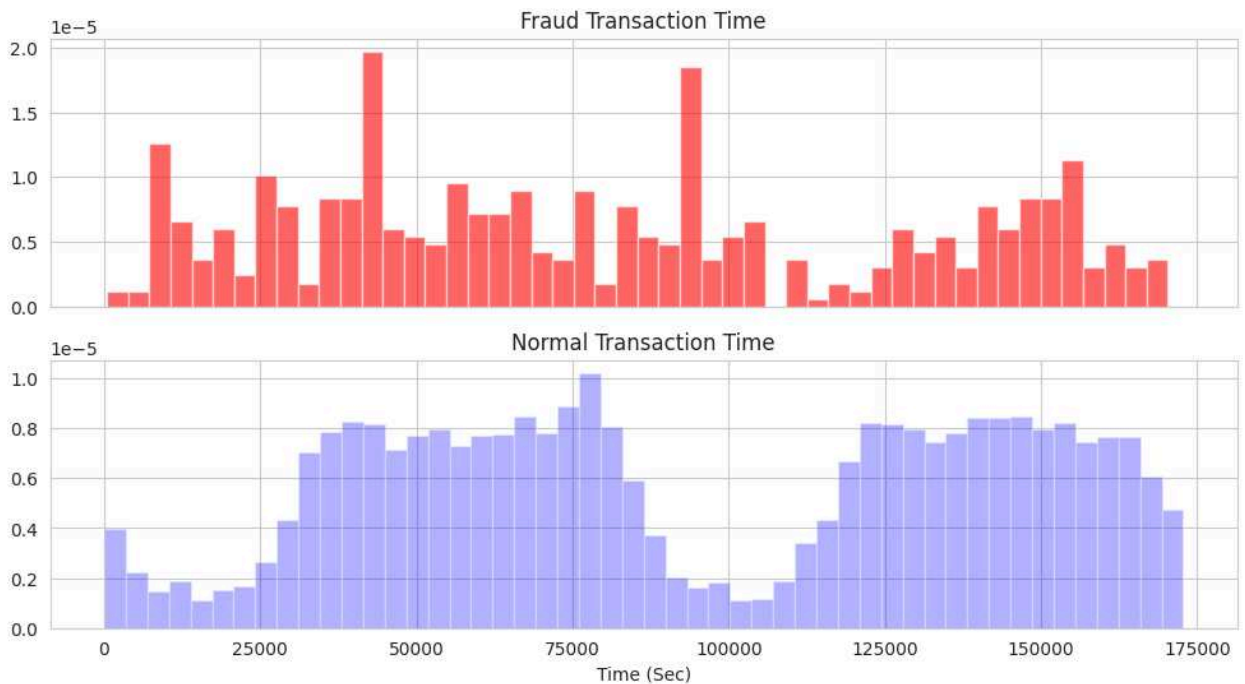
/tmp/ipython-input-4134450283.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` a

```
sns.countplot(x='Class', data=df, palette='viridis')
```



Fraud Percentage: 0.1727%



```
# -----
# STEP 4: Preprocessing (Sampling, Missing Values, Splitting)
# الخطوة 4: المعالجة الأولية (العينة، القيم المفقودة، التقسيم)
# -----

# Sampling for computational efficiency in Colab (Optional but recommended)
# نحافظ على نسبة الاحتمال باستخدام أخذ عينة لتحسين الأداء
# Original line: df_sample = df.sample(n=25000, random_state=42, stratify=df['Class'])
# Fix: Use train_test_split to perform stratified sampling as df.sample() does not support 'stratify'
_, df_sample = train_test_split(df, test_size=25000, random_state=42, stratify=df['Class'])

# Reset index for consistent behavior after sampling
df_sample = df_sample.reset_index(drop=True)

# 4.1 Handle Missing Values
```

```

# معالجة القيم المفقودة
imputer = SimpleImputer(strategy='median') # Median is better for skewed data like Amount / الوسيط أفضل للبيانات المائلة
df_imputed = pd.DataFrame(imputer.fit_transform(df_sample), columns=df_sample.columns)

# 4.2 Split Features and Target
X = df_imputed.drop('Class', axis=1)
y = df_imputed['Class']

# 4.3 Train-Test Split (Stratified)
# تقسيم البيانات مع الحفاظ على نسبة الفئات
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# 4.4 Scaling
# توحيد المقاييس
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
# Keep as DataFrame for feature names preservation
X_train_scaled_df = pd.DataFrame(X_train_scaled, columns=X.columns)
X_test_scaled_df = pd.DataFrame(X_test_scaled, columns=X.columns)

print("Step 4: Preprocessing Complete.")

```

Step 4: Preprocessing Complete.

```

# -----
# STEP 5: Helper Function for Evaluation
# الخطوة 5: دالة مساعدة للتقييم والرسم
# -----
def evaluate_model(model, X_test, y_test, model_name):
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)[:, 1]

    # Metrics
    print(f"\n--- {model_name} Evaluation ---")
    print(classification_report(y_test, y_pred))

    # 1. Confusion Matrix
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(15, 4))

    plt.subplot(1, 3, 1)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f'{model_name} Confusion Matrix')

    # 2. ROC Curve
    fpr, tpr, _ = roc_curve(y_test, y_prob)
    roc_auc_val = auc(fpr, tpr)
    plt.subplot(1, 3, 2)
    plt.plot(fpr, tpr, color='darkorange', label=f'AUC = {roc_auc_val:.2f}')
    plt.plot([0, 1], [0, 1], linestyle='--')
    plt.title(f'{model_name} ROC Curve')
    plt.legend()

    # 3. Precision-Recall Curve (Crucial for Imbalanced Data)
    precision, recall, _ = precision_recall_curve(y_test, y_prob)
    pr_auc_val = average_precision_score(y_test, y_prob)
    plt.subplot(1, 3, 3)
    plt.plot(recall, precision, color='green', label=f'PR AUC = {pr_auc_val:.2f}')
    plt.title(f'{model_name} PR Curve')
    plt.xlabel('Recall')
    plt.ylabel('Precision')
    plt.legend()

    plt.tight_layout()
    plt.show()

    return {
        'Model': model_name,
        'ROC-AUC': roc_auc_val,
        'PR-AUC': pr_auc_val,
        'F1-Score': f1_score(y_test, y_pred),
        'Recall': recall_score(y_test, y_pred)
    }
results_list = []

```

```
# -----
# STEP 6: Logistic Regression (Baseline)
# الخطوة 6: الانحدار اللوجستي (النموذج الأساسي)
# -----
print("\nTraining Baseline: Logistic Regression...")

# Parameters for GridSearch
# class_weight='balanced' is key for imbalanced datasets
lr_params = {'C': [0.1, 1, 10], 'penalty': ['l2']}
lr_model = LogisticRegression(solver='lbfgs', max_iter=1000, class_weight='balanced')

lr_grid = GridSearchCV(lr_model, lr_params, cv=3, scoring='f1')
lr_grid.fit(X_train_scaled, y_train)

best_lr = lr_grid.best_estimator_
res_lr = evaluate_model(best_lr, X_test_scaled, y_test, "Logistic Regression (All Features)")
results_list.append(res_lr)
```

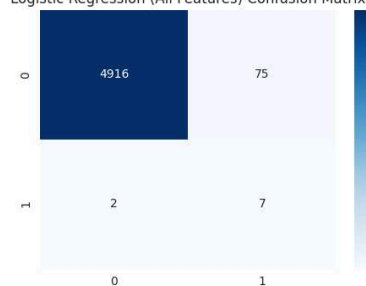
Training Baseline: Logistic Regression...

--- Logistic Regression (All Features) Evaluation ---

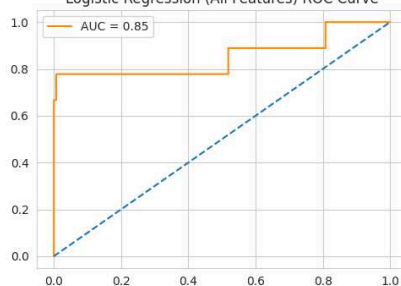
	precision	recall	f1-score	support
0.0	1.00	0.98	0.99	4991
1.0	0.09	0.78	0.15	9

accuracy			0.98	5000
macro avg	0.54	0.88	0.57	5000
weighted avg	1.00	0.98	0.99	5000

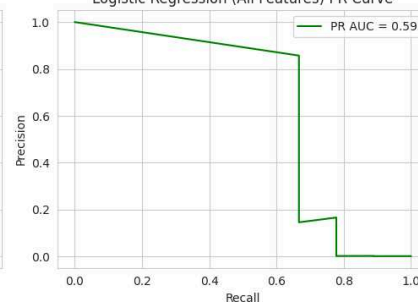
Logistic Regression (All Features) Confusion Matrix



Logistic Regression (All Features) ROC Curve



Logistic Regression (All Features) PR Curve



```
# -----
# STEP 7: XGBoost (Advanced Model)
# الخطوة 7: XGBoost (النموذج المتقدم)
# -----
print("\nTraining Advanced: XGBoost...")

# Calculate scale_pos_weight for imbalance
# حساب وزن الفئة الموجبة للتعامل مع عدم التوازن
ratio = float(np.sum(y_train == 0)) / np.sum(y_train == 1)

xgb_model = xgb.XGBClassifier(eval_metric='logloss', use_label_encoder=False)

# Params: scale_pos_weight helps with imbalance
xgb_params = {
    'n_estimators': [50, 100],
    'max_depth': [3, 5],
    'learning_rate': [0.1],
    'scale_pos_weight': [ratio]
}

xgb_grid = GridSearchCV(xgb_model, xgb_params, cv=3, scoring='f1')
xgb_grid.fit(X_train_scaled, y_train)

best_xgb = xgb_grid.best_estimator_
res_xgb = evaluate_model(best_xgb, X_test_scaled, y_test, "XGBoost (All Features)")
results_list.append(res_xgb)
```

Training Advanced: XGBoost...

```

--- XGBoost (All Features) Evaluation ---
      precision    recall  f1-score   support

      0.0         1.00      1.00      1.00      4991
      1.0         0.67      0.89      0.76         9

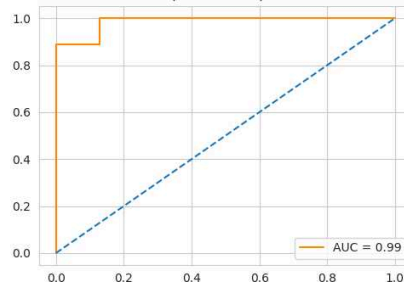
 accuracy         1.00      5000
 macro avg        0.83      0.94      0.88      5000
 weighted avg     1.00      1.00      1.00      5000

```

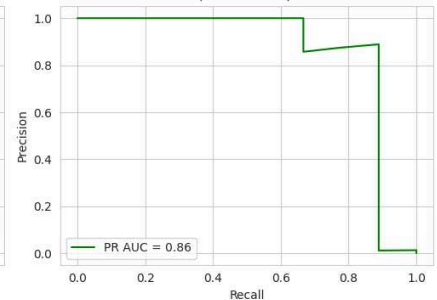
XGBoost (All Features) Confusion Matrix



XGBoost (All Features) ROC Curve



XGBoost (All Features) PR Curve



```

# -----
# STEP 8: Feature Selection 1: Filter-Based (Mutual Information)
# الخطوة 8: اختيار الميزات 1: طريقة التصفية (المعلومات المتبادلة)
# -----
print("\n--- Feature Selection: Filter-Based (Mutual Info) ---")

# Using Mutual Information because it captures non-linear relationships
# لأنها تلتقط العلاقات غير الخطية Mutual Info استخدام
selector_filter = SelectKBest(score_func=mutual_info_classif, k=10)
selector_filter.fit(X_train_scaled, y_train)

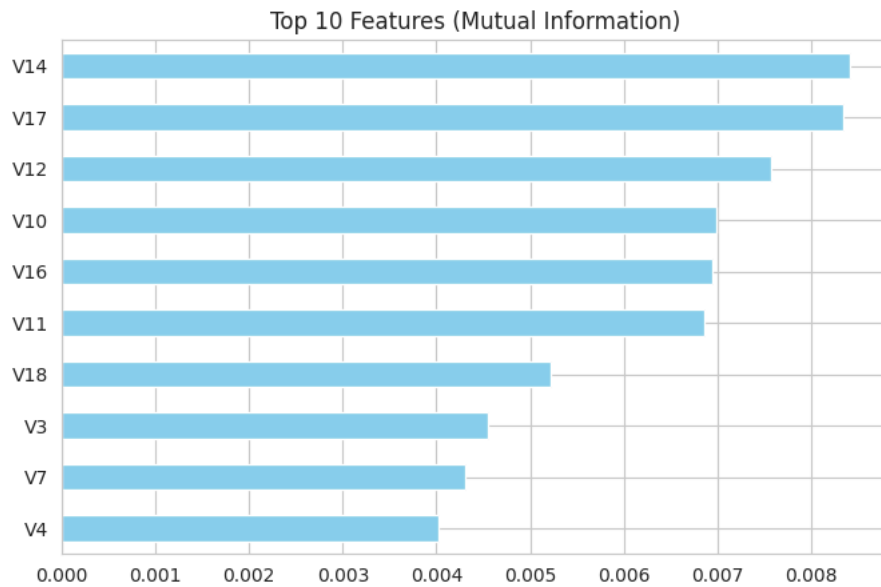
# Visualize Scores
# تصوير أهمية الميزات
filter_scores = pd.Series(selector_filter.scores_, index=X.columns)
top_filter_features = filter_scores.nlargest(10)

plt.figure(figsize=(8, 5))
top_filter_features.sort_values().plot(kind='barh', color='skyblue')
plt.title('Top 10 Features (Mutual Information)')
plt.show()

# Transform Data
X_train_filter = selector_filter.transform(X_train_scaled)
X_test_filter = selector_filter.transform(X_test_scaled)

```

--- Feature Selection: Filter-Based (Mutual Info) ---



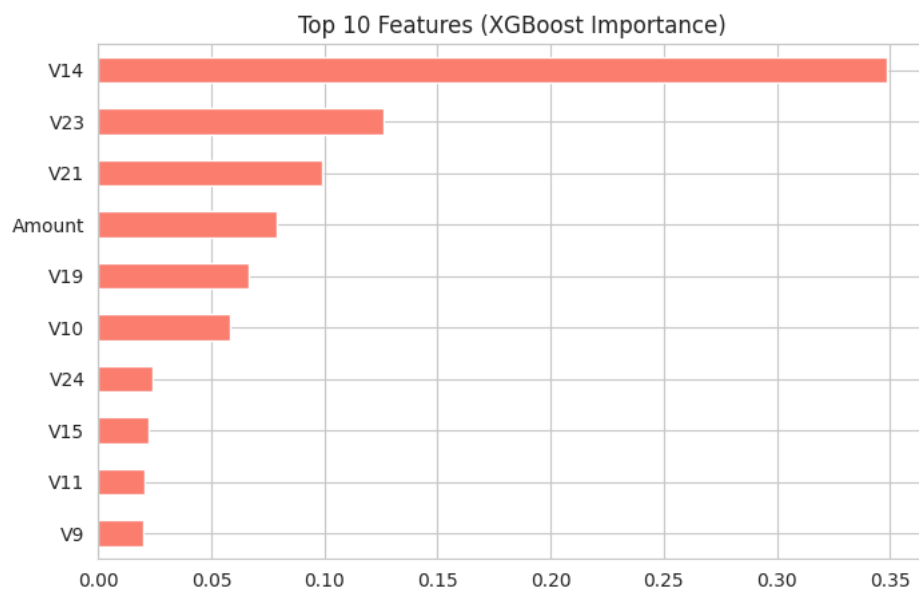
```
# -----
# STEP 9: Feature Selection 2: Model-Based (XGBoost Importance)
# الخطوة 9: اختيار الميزات 2: طريقة تعتمد على النموذج
# -----
print("\n--- Feature Selection: Model-Based (XGBoost) ---")

# Extract feature importance from the trained XGBoost model
# المدرب سابقاً XGBoost استخراج أهمية الميزات من نموذج
importances = best_xgb.feature_importances_
feature_imp_df = pd.Series(importances, index=X.columns).nlargest(10)

plt.figure(figsize=(8, 5))
feature_imp_df.sort_values().plot(kind='barh', color='salmon')
plt.title('Top 10 Features (XGBoost Importance)')
plt.show()

# Transform Data using SelectFromModel
selection_model = SelectFromModel(best_xgb, prefit=True, max_features=10)
X_train_model = selection_model.transform(X_train_scaled)
X_test_model = selection_model.transform(X_test_scaled)
```

--- Feature Selection: Model-Based (XGBoost) ---



```
# -----
# STEP 10: Retrain & Compare on Selected Features
# الخطوة 10: إعادة التدريب والمقارنة على الميزات المختارة
# -----
```

```

print("\nRetraining Models on Selected Features...")

# 1. Logistic Regression with Filter-Based Features
lr_filter = LogisticRegression(solver='lbfgs', max_iter=1000, class_weight='balanced', C=lr_grid.best_params_['C'])
lr_filter.fit(X_train_filter, y_train)
res_lr_filt = evaluate_model(lr_filter, X_test_filter, y_test, "LogReg (Filter Features)")
results_list.append(res_lr_filt)

# 2. XGBoost with Model-Based Features
xgb_model_sel = xgb.XGBClassifier(eval_metric='logloss', use_label_encoder=False,
                                  **xgb_grid.best_params_)
xgb_model_sel.fit(X_train_model, y_train)
res_xgb_sel = evaluate_model(xgb_model_sel, X_test_model, y_test, "XGBoost (Model Features)")
results_list.append(res_xgb_sel)

```

```

# -----
# STEP 11: Final Comparison Summary
# الخطوة 11: ملخص المقارنة النهائي
# -----
results_df = pd.DataFrame(results_list)
print("\n=== Final Performance Comparison / مقارنة الأداء النهائية ===")
print(results_df)

# Plot Comparison
# رسم مقارنة النتائج
results_melted = results_df.melt(id_vars="Model", var_name="Metric", value_name="Score")
plt.figure(figsize=(10, 6))
sns.barplot(data=results_melted, x='Model', y='Score', hue='Metric', palette='muted')
plt.title("Model Performance Comparison ")
plt.xticks(rotation=15)
plt.ylim(0.7, 1.0)
plt.legend(loc='lower right')
plt.show()

```

```

=== Final Performance Comparison / مقارنة الأداء النهائية ===

```

	Model	ROC-AUC	PR-AUC	F1-Score	Recall
0	Logistic Regression (All Features)	0.851755	0.590537	0.153846	0.777778
1	XGBoost (All Features)	0.985752	0.864200	0.761905	0.888889
2	XGBoost (All Features)	0.985752	0.864200	0.761905	0.888889

