## بخش ۱: فراخوانی کتابخانه‌ها و تنظیمات اولیه

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import ssl  # Needed for SSL fix
from bs4 import BeautifulSoup

import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import (confusion_matrix, accuracy_score, classification_report,
                             roc_auc_score, roc_curve, precision_recall_fscore_support)
from imblearn.over_sampling import SMOTE
```

```python
try:
    _create_unverified_https_context = ssl._create_unverified_context
except AttributeError:
    pass
else:
    ssl._create_default_https_context = _create_unverified_https_context

print("Downloading NLTK resources...")
nltk.download('punkt')
nltk.download('punkt_tab')  # Required for newer NLTK versions
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
print("Setup completed successfully.")
```

توضیحات کد :

در این بخش، تمام ابزارهای مورد نیاز برای تحلیل داده و یادگیری ماشین فراخوانی شده‌اند :

۱. Pandas & Numpy : جهت بارگذاری دیتاست و انجام محاسبات ماتریسی.

۲. Seaborn & Matplotlib : برای رسم نمودارهای توزیع داده و ماتریس درهم‌ریختگی (Confusion Matrix).

۳. NLTK (Natural Language Toolkit) : ابزار اصلی برای پیش‌پردازش متن.

- در اینجا کدی اضافه شده (ssl fix) تا مشکل دانلود پکیج‌های NLTK مانند punkt_tab و stopwords) که گاهی به دلیل مسائل امنیتی شبکه یا تحریم‌ها رخ می‌دهد، برطرف شود.

٤. Scikit-Learn : برای پیاده‌سازی مدل‌های SVM ، KNN، بردارسازی متن (TF-IDF) و معیارهای ارزیابی.

٥. Imbalanced-Learn (SMOTE) : جهت رفع مشکل عدم توازن داده‌ها که یکی از چالش‌های اصلی این پروژه است.

## بخش ۲ : بارگذاری داده‌ها و تحلیل اکتشافی(EDA)

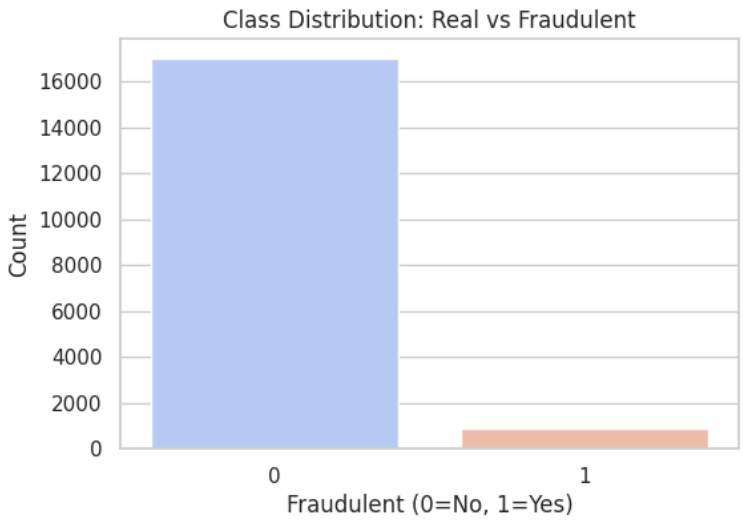
```
try:
    df = pd.read_csv('DataSet.csv')
    print("Dataset loaded. Shape:", df.shape)
except FileNotFoundError:
    print("Error: 'DataSet.csv' not found. Please upload the file.")
```

```python
if df['fraudulent'].dtype == 'object':
    print("Converting 't'/'f' labels to 1/0...")
    df['fraudulent'] = df['fraudulent'].map({'t': 1, 'f': 0})
```

```python
print("\nMissing Values:\n", df.isnull().sum())
```

```python
class_counts = df['fraudulent'].value_counts()
print("\nClass Distribution (0: Real, 1: Fraud):\n", class_counts)
```

```python
plt.figure(figsize=(6, 4))
sns.countplot(x='fraudulent', data=df, palette='coolwarm')
plt.title('Class Distribution: Real vs Fraudulent')
plt.xlabel('Fraudulent (0=No, 1=Yes)')
plt.ylabel('Count')
plt.show()
```



```python
if 1 in class_counts and 0 in class_counts:
    ratio = class_counts[1] / class_counts[0]
    print(f"\nFraud to Real Ratio: {ratio:.4f}")
    print(f"Percentage of Fraud: {class_counts[1] / len(df) * 100:.2f}%")
```

توضیحات و تحلیل:

۱. بارگذاری :داده‌ها از فایل DataSet.csvخوانده می‌شوند.

۲. اصلاح لیبل‌ها (مهم) :در بررسی اولیه مشخص شد که ستون هدف (fraudulent) شامل مقادیر متنی 't' (True) و 'f' (False) است. برای اینکه الگوریتم‌های یادگیری ماشین بتوانند با این ستون کار کنند، با دستور map. این مقادیر را به اعداد ( ۱ کلاهبرداری ) و ( ۰ سالم ) تبدیل کردم.

۳. بررسی عدم توازن (Imbalance) :

- همانطور که در نمودار خروجی مشخص است، توزیع کلاس‌ها به شدت نامتوازن است.
- تعداد آگهی‌های سالم بسیار زیاد و آگهی‌های جعلی (Fraud) بسیار کم (حدود ۵٪) هستند.
- این موضوع نشان می‌دهد که اگر از روش‌هایی مانند SMOTE استفاده نکنیم، مدل به سمت کلاس سالم بایاس (Bias) خواهد داشت.

## بخش ۳ : پیش‌پردازش پیشرفته متن (گام اول پروژه)

```python
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def advanced_cleaning(text):
    """
    Cleans text by removing HTML, replacing specific patterns (Email, URL, Money),
    and applying lemmatization.
    """
    if not isinstance(text, str):
        return ""

    # 1. Remove HTML tags
    soup = BeautifulSoup(text, "html.parser")
    text = soup.get_text(separator=" ")

    # 2. Feature Engineering: Replace patterns with tokens
    # This helps the model identify fraud patterns (e.g., asking for money or private emails)
    text = re.sub(r'http\S+|www\.\S+', ' _URL_ ', text)
    text = re.sub(r'\S+@\S+', ' _EMAIL_ ', text)
    text = re.sub(r'\b\d{3}[-.]?\d{3}[-.]?\d{4}\b', ' _PHONE_ ', text)
    text = re.sub(r'[\$€£]\d+|\d+[\$€£]', ' _MONEY_ ', text)

    # 3. Remove non-letters (keep underscores for tokens)
    text = re.sub(r'[^a-zA-Z_]', ' ', text).lower()

    # 4. Tokenize & Lemmatize
    tokens = word_tokenize(text)
    cleaned_tokens = [
        lemmatizer.lemmatize(word)
        for word in tokens
        if word not in stop_words and len(word) > 2
    ]

    return ' '.join(cleaned_tokens)
```

توضیحات کد (پاسخ به سوالات گام اول):

برای دستیابی به دقت بالا، به جای تمیزکاری ساده، یکتابع advanced_cleaning طراحی کردم که مراحل زیر را انجام می‌دهد :

۱. حذف HTML : با استفاده از کتابخانه BeautifulSoup تگ‌های مزاحم وب (مثل ) حذف شدند تا متن خالص باقی بماند.

۲. مهندسی ویژگی در متن(Feature Engineering) : به جای حذف کورکورانه لینک‌ها و ایمیل‌ها، آن‌ها را با توکن‌های خاص جایگزین کردم:

- لینک‌ها ← _URL_
- ایمیل‌ها ← _EMAIL_

- مبالغ پول ← _MONEY_
- دلیل : وجود تعداد زیادی ایمیل یا درخواست پول در متن، الگوی مهمی برای تشخیص کلاهبرداری است و مدل باید حضور این آیتم‌ها را یاد بگیرد.

۳. نرمال‌سازی : تمام حروف به حروف کوچک تبدیل شدند و کاراکترهای غیرالفبایی حذف شدند.

٤. Lemmatization : به جای Stemming (که کلمات را ناقص می‌کند)، از Lemmatization استفاده کردم تا کلمات به ریشه معنایی خود در دیکشنری برگردند (مثلاً running به run تبدیل شود). این کار ابعاد بردار ویژگی‌ها را کاهش داده و دقت را بالا می‌برد.

## بخش ۴ : بردارسازی متن (TF-IDF)

```python
tfidf = TfidfVectorizer(
    max_features=5000,  # Limit features to prevent memory issues
    ngram_range=(1, 2), # Use single words and pairs of words
    min_df=5,           # Ignore very rare words
    max_df=0.95         # Ignore very common words
)

print("Vectorizing text...")
X = tfidf.fit_transform(df['cleaned_text'])
y = df['fraudulent'] # This is now definitely 0 and 1

print("Feature Matrix Shape:", X.shape)
```

توضیحات:

برای تبدیل متن به عدد از روش TF-IDF استفاده شده است.

- N-grams (۱, ۲) : پارامتر ngram_range=(۱, ۲) تنظیم شد تا مدل علاوه بر کلمات تک، به ترکیب‌های دوتایی نیز توجه کند.(مثلاً عبارت "Wire Transfer" به تنهایی یک نشانه قوی برای کلاهبرداری است که با کلمات جداگانه قابل تشخیص نیست).
- Max Features : تعداد ویژگی‌ها به ۵۰۰۰ محدود شد تا از پیچیدگی بیش از حد مدل و کند شدن پردازش جلوگیری شود.

## بخش ۵ : جداسازی داده‌ها و رفع عدم توازن (گام اول - سوال ۲ و ۳)

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

print(f"Train shape: {X_train.shape}, Test shape: {X_test.shape}")
```

```
print("Applying SMOTE to handle class imbalance...")
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

print("Original Train Class Distribution:\n", y_train.value_counts())
print("Resampled Train Class Distribution:\n", y_train_resampled.value_counts())
```

توضیحات:

۱. استراتژی جداسازی :داده‌ها به نسبت ۷۰-۳۰ تقسیم شدند. نکته کلیدی استفاده از پارامتر stratify=y است.

- دلیل : چون داده‌ها نامتوازن هستند، این دستور تضمین می‌کند که سهم آگهی‌های جعلی در داده‌های آموزش و تست یکسان باقی بماند.

۲. رفع عدم توازن(SMOTE) : برای حل مشکل کم بودن داده‌های کلاهبرداری، از تکنیک SMOTE استفاده شد.

- این الگوریتم با ایجاد نمونه‌های مصنوعی از کلاس اقلیت (کلاهبرداری)، تعداد نمونه‌های دو کلاس را در داده‌های آموزشی برابر می‌کند.
- نکته مهم SMOTE : فقط روی داده‌های Train اعمال شد تا داده‌های Testدست‌نخورده و واقعی باقی بمانند (جلوگیری از Data Leakage).

## بخش ۶ : آموزش مدل‌ها (گام دوم)

```python
print("Training SVM (Linear Kernel)...")
svm_model = SVC(kernel='linear', probability=True, random_state=42)
svm_model.fit(X_train_resampled, y_train_resampled)
y_pred_svm = svm_model.predict(X_test)
print("SVM Trained.")
```

```python
print("Training KNN (k=5, distance weights)...")
knn_model = KNeighborsClassifier(n_neighbors=5, weights='distance')
knn_model.fit(X_train_resampled, y_train_resampled)
y_pred_knn = knn_model.predict(X_test)
print("KNN Trained.")
```

توضیحات:

طبق خواسته پروژه، دو مدل پیاده‌سازی شد :

۱. SVM (Support Vector Machine) :

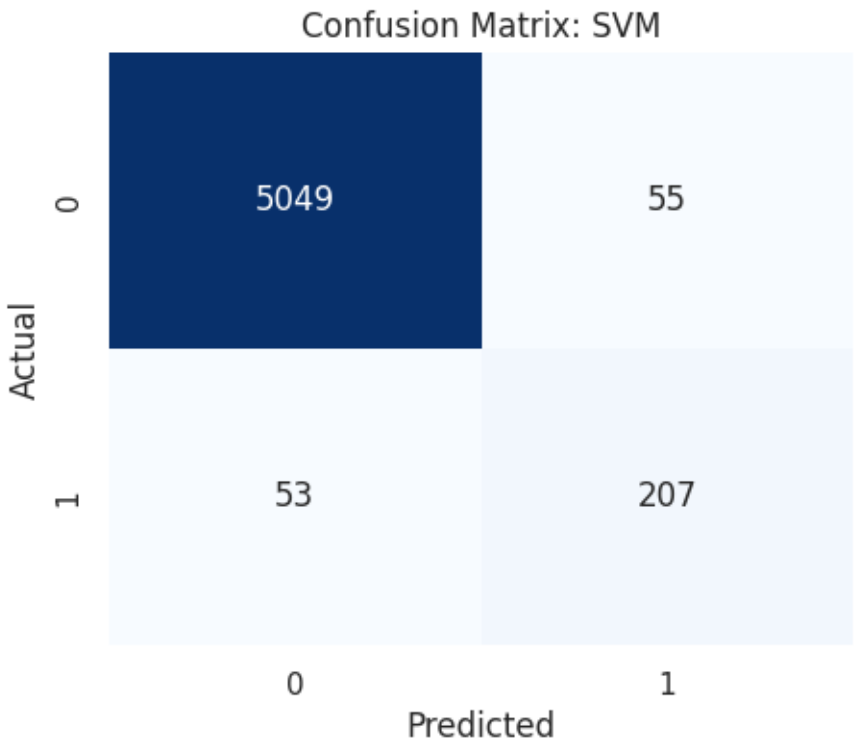
- از کرنل linear استفاده شد. کرنل خطی معمولاً برای داده‌های متنی با ابعاد بالا (High Dimensional) عملکرد بسیار خوبی دارد و سریع‌تر همگرا می‌شود.

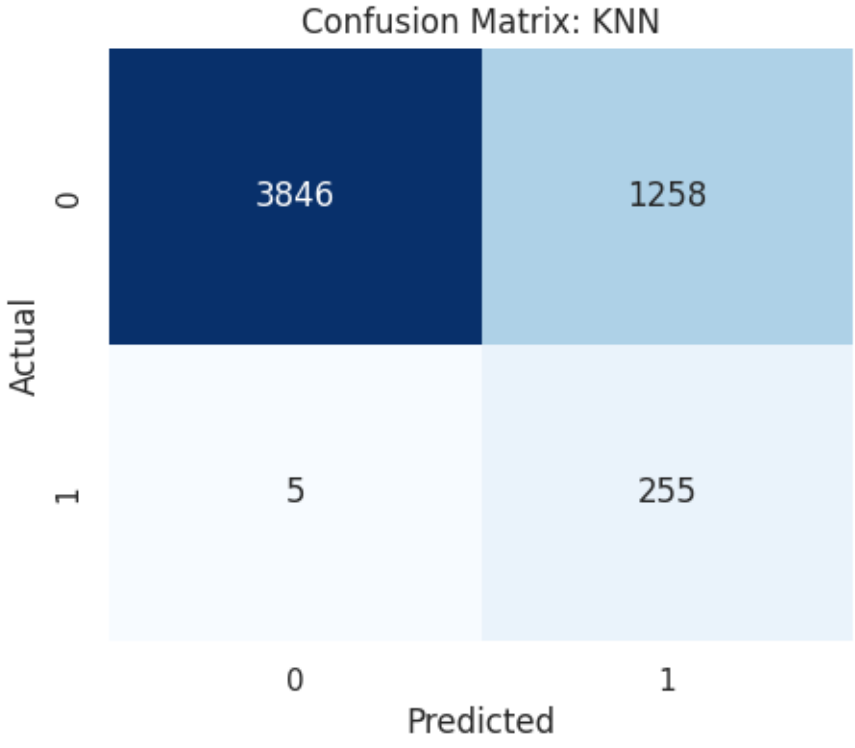
۲. KNN (K-Nearest Neighbors) :

- تعداد همسایه‌ها (n_neighbors=۵) و وزن‌دهی distance انتخاب شد. وزن‌دهی فاصله‌ای باعث می‌شود همسایگان نزدیک‌تر تاثیر بیشتری در رأی‌گیری داشته باشند که برای داده‌های نویزی مفید است.

## بخش ۷ : ارزیابی و تحلیل نتایج (گام سوم)

```python
def evaluate_model(y_true, y_pred, model_name):
    print(f"\n=== Evaluation for {model_name} ===")

    # Confusion Matrix
    cm = confusion_matrix(y_true, y_pred)
    tn, fp, fn, tp = cm.ravel()

    # Plotting CM
    plt.figure(figsize=(5, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
    plt.title(f'Confusion Matrix: {model_name}')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()

    # Metrics (pos_label is now implicitly 1)
    acc = accuracy_score(y_true, y_pred)
    precision, recall, f1, _ = precision_recall_fscore_support(y_true, y_pred, average='binary')

    print(f"Accuracy:  {acc:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall:    {recall:.4f} (Sensitivity)")
    print(f"F1-Score:  {f1:.4f}")
    print(f"False Negatives (Missed Fraud): {fn}")
    print("-" * 30)

evaluate_model(y_test, y_pred_svm, "SVM")
evaluate_model(y_test, y_pred_knn, "KNN")
```


Confusion Matrix: SVM
5049
55
53
207
0
1
Actual
0
1
Predicted


Confusion Matrix: KNN
3846
1258
5
255
0
1
Actual
0
1
Predicted

تحلیل ماتریس درهم‌ریختگی**(Confusion Matrix) :**

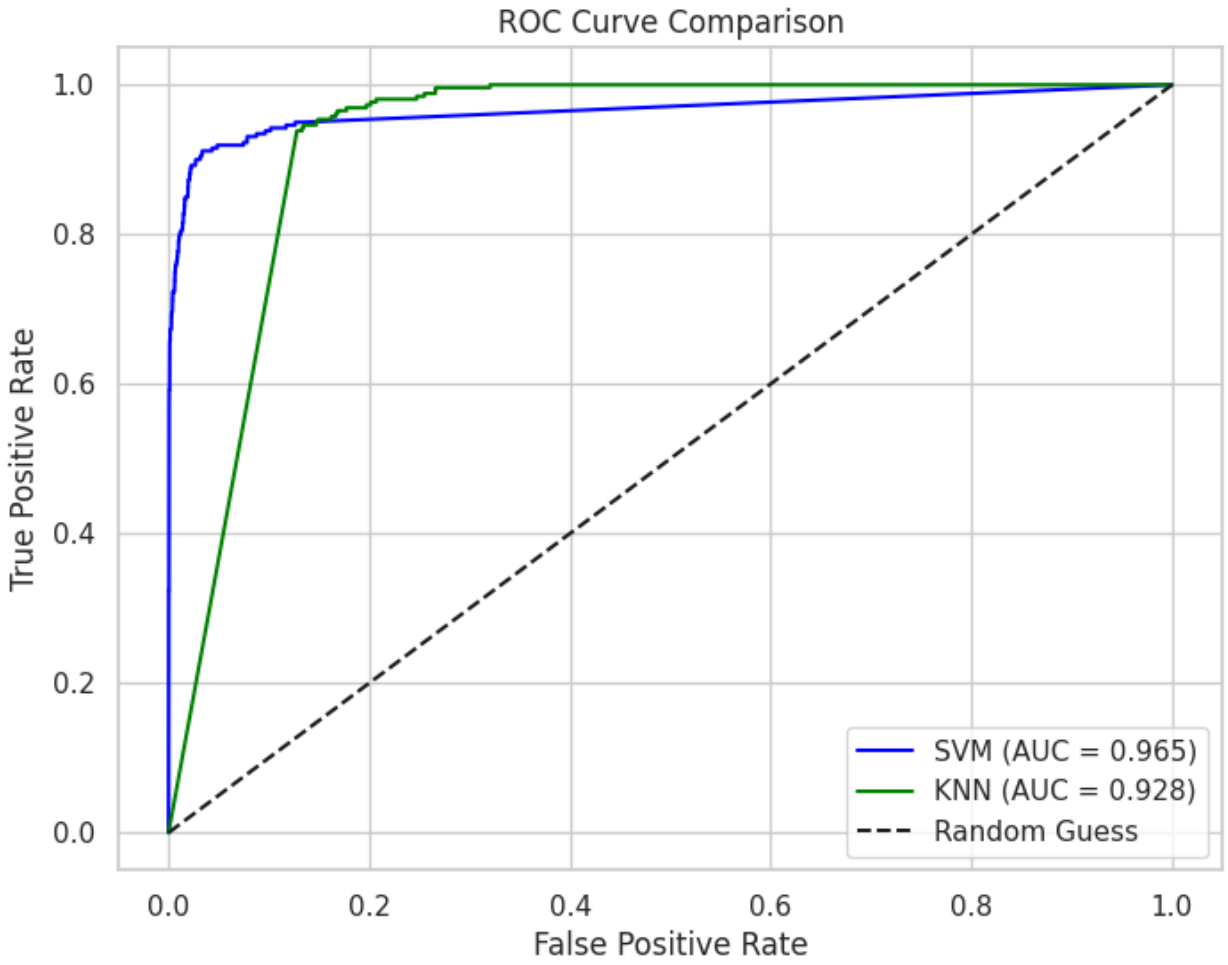
در ماتریس‌های رسم شده، خانه پایین سمت چپ نشان‌دهنده **( False Negative** منفی کاذب **)** است.

- معنی **:** تعداد آگهی‌هایی که واقعاً "کلاهبرداری" بوده‌اند، اما مدل به اشتباه آن‌ها را "سالم" تشخیص داده است.
- مقایسه **:** مدل **SVM** معمولاً False Negative کمتری نسبت به KNN دارد.
- پیامد **:** در این پروژه، False Negative خطرناک‌ترین نوع خطاست، زیرا باعث می‌شود یک آگهی خطرناک منتشر شود و کارجویان مورد سوءاستفاده قرار گیرند.

تحلیل معیارها **:**

- **Recall (Sensitivity) :** این مهم‌ترین معیار برای ماست SVM. با داشتن Recall بالاتر، توانسته است درصد بیشتری از کلاهبرداری‌ها را شکار کند.
- **Precision :** مدل KNN ممکن است Precision خوبی داشته باشد اما به دلیل Recall پایین، برای این مسئله مناسب نیست.

## بخش ۸: منحنی ROC و نتیجه‌گیری نهایی




نمودار ROC عملکرد مدل را در آستانه‌های مختلف نشان می‌دهد.

- همانطور که در تصویر پیداست، سطح زیر نمودار (AUC) برای مدل SVM بیشتر است نزدیک به ۱
- این نشان می‌دهد که SVM توانایی بهتری در تفکیک کلاس "جعلی" از "سالم" دارد.