

```

# (Social Media Trend) مشروع: تحليل بيانات انتشار المحتوى الرقمي
# اخترت هذا الموضوع لاهتمامي بتحليل الإعلام الرقمي وفهم الفيروسية
# (Chi2 + RFE) واختيار الميزات، Grid Search، يتضمن: تنظيف البيانات

# استيراد جميع المكتبات
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report
from sklearn.feature_selection import SelectKBest, chi2, RFE

from google.colab import files

# رفع الملف وعرض أسماء الأعمدة
print("📁 من جهازك (CSV) الرجاء اختيار ملف البيانات")
uploaded = files.upload()
file_name = list(uploaded.keys())[0]

# قراءة الملف
df = pd.read_csv(file_name)

print("\n✅ تم تحميل الملف بنجاح")
print("\n📄 أسماء جميع الأعمدة في ملفك:")
print(df.columns.tolist())

# تحديد العمود المستهدف
target_column = None
for col in df.columns:
    if 'viral' in col.lower():
        target_column = col
        break

if target_column is None:
    print("\n❌ في بياناتك 'viral' خطأ: لم أجد عموداً يحوي كلمة")
    print("\n.رجاء انظر إلى قائمة الأعمدة أعلاه، وابحث عن العمود الذي يعني (فيروسى / انتشار)")
    print("\n.في الكود واعد التشغيل 'target_column' ثم غير قيمة")
    # نوقف التنفيذ هنا عشان ما يكمل غلط
    raise ValueError("المستهدف على العمود المستهدف")

print(f"\n🎯 الهدف: '{target_column}' : العمود المستهدف (الهدف) هو")
print("\n(توزيع الفئات (0 = غير فيروسى، 1 = فيروسى))")
print(df[target_column].value_counts())

# الخطوة 2: معالجة القيم المفقودة
print("\n" + "="*60)
print("\nالخطوة 2: معالجة القيم المفقودة")
print("="*60)

print("\n(عدد القيم المفقودة قبل المعالجة (لكل عمود))")
print(df.isnull().sum())

# تعويض الأعمدة الرقمية بالمتوسط
for col in df.select_dtypes(include=['float64', 'int64']).columns:
    if df[col].isnull().sum() > 0:
        df[col].fillna(df[col].mean(), inplace=True)
        print(f"✅ تم تعويض الفراغات في العمود الرقمي {col}")

# تعويض الأعمدة الفئوية بالمعنى
for col in df.select_dtypes(include=['object']).columns:
    if df[col].isnull().sum() > 0:
        df[col].fillna(df[col].mode()[0], inplace=True)
        print(f"✅ تم تعويض الفراغات في العمود الفئوى {col}")

# حذف أي صفوف متبقية بها فراغات
df.dropna(inplace=True)
print(f"\n🧹 إجمالي القيم المفقودة بعد التنظيف: {df.isnull().sum().sum()}")

```

```

# تحويل الأعمدة النصية إلى أرقام

print("\n🔄 تحويل الأعمدة النصية إلى أرقام")
for col in df.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col].astype(str))
print("✅ تم التحويل بنجاح")

# فصل الميزات (X) عن الهدف (y)
X = df.drop(target_column, axis=1)
y = df[target_column]

print(f"\n📊 (الأعمدة المؤثرة) : {X.shape[1]}")
print(f"\n📊 (الصفوف) : {X.shape[0]}")

# تقسيم البيانات: 80% تدريب، 20% اختبار
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

print(f"\n📊 حجم بيانات التدريب : {X_train.shape[0]}")
print(f"\n📊 حجم بيانات الاختبار : {X_test.shape[0]}")

# دالة التقييم (عشان تطلع لنا الأرقام والجداول)
def evaluate_model(model_name, y_true, y_pred):
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred, average='weighted', zero_division=0)
    recall = recall_score(y_true, y_pred, average='weighted', zero_division=0)
    f1 = f1_score(y_true, y_pred, average='weighted', zero_division=0)

    print(f"\n--- {model_name} ---")
    print(f"✅ الدقة (Accuracy): {accuracy:.4f}")
    print(f"✅ الدقة الإيجابية (Precision): {precision:.4f}")
    print(f"✅ الاسترجاع (Recall): {recall:.4f}")
    print(f"✅ درجة F1: {f1:.4f}")

    # مصفوفة الارتباك (Confusion Matrix)
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(4, 3))
    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()

    return {'accuracy': accuracy, 'precision': precision, 'recall': recall, 'f1': f1}

# البحث عن أفضل الإعدادات Grid Search (الخطوة 3: تطبيق النماذج مع)

print("\n" + "="*60)
print("Grid Search (الخطوة 3: تطبيق النماذج مع)")
print("="*60)

models = {
    'Random Forest': {
        'model': RandomForestClassifier(random_state=42),
        'params': {'n_estimators': [50, 100], 'max_depth': [None, 10]}
    },
    'Decision Tree': {
        'model': DecisionTreeClassifier(random_state=42),
        'params': {'max_depth': [None, 10], 'min_samples_split': [2, 5]}
    },
    'SVM': {
        'model': SVC(random_state=42),
        'params': {'C': [0.1, 1], 'kernel': ['rbf', 'linear']}
    },
    'Logistic Regression': {
        'model': LogisticRegression(random_state=42, max_iter=1000),
        'params': {'C': [0.1, 1]}
    },
    'KNN': {
        'model': KNeighborsClassifier(),
        'params': {'n_neighbors': [3, 5, 7]}
    }
}

results_all = []
for name, config in models.items():
    print(f"\n--- 🟡 جاري تدريب نموذج {name} ---")
    gs = GridSearchCV(config['model'], config['params'], cv=3, scoring='accuracy', n_jobs=-1)

```

```

gs.fit(X_train, y_train)
print(f"أفضل إعدادات تم العثور عليها: {gs.best_params_}")
y_pred = gs.predict(X_test)
res = evaluate_model(name, y_test, y_pred)
res['model'] = name
results_all.append(res)

# اختيار الميزات بطريقتين
print("\n" + "="*60)
print("الخطوة 4: اختيار الميزات - الطريقتين المطلوبتين")
print("="*60)

# --- الطريقة الأولى: Chi-Square (Filter Method) ---
print("\n--- الطريقة 1: Chi-Square Test ---")
# Chi-Square يشترط أن تكون كل القيم موجبة، لذلك نزيحها
X_train_pos = X_train - X_train.min() + 1e-6
selector_chi2 = SelectKBest(score_func=chi2, k=8) # اختر أفضل 8 ميزات
X_train_chi2 = selector_chi2.fit_transform(X_train_pos, y_train)
X_test_chi2 = selector_chi2.transform(X_test - X_train.min() + 1e-6)
features_chi2 = X.columns[selector_chi2.get_support()].tolist()
print(f"✅ Chi-Square: الميزات المختارة بواسطة {features_chi2}")

# --- الطريقة الثانية: RFE (Wrapper Method) ---
print("\n--- الطريقة 2: Recursive Feature Elimination (RFE) ---")
estimator = RandomForestClassifier(n_estimators=50, random_state=42)
rfe = RFE(estimator=estimator, n_features_to_select=8)
rfe.fit(X_train, y_train)
features_rfe = X.columns[rfe.support_].tolist()
print(f"✅ RFE: الميزات المختارة بواسطة {features_rfe}")

# إعادة تطبيق النماذج على الميزات المختارة
print("\n" + "="*60)
print("الخطوة 5: إعادة تطبيق النماذج على الميزات المختارة")
print("="*60)

# لكننا نختار الأولى، أو ChiSquare نختار مجموعة الميزات من
X_train_sel = X_train[features_chi2]
X_test_sel = X_test[features_chi2]

results_sel = []
for name, config in models.items():
    print(f"\n--- (على الميزات المختارة) {name} :إعادة تدريب ---")
    gs = GridSearchCV(config['model'], config['params'], cv=3, scoring='accuracy', n_jobs=-1)
    gs.fit(X_train_sel, y_train)
    y_pred = gs.predict(X_test_sel)
    res = evaluate_model(name + " (مختار)", y_test, y_pred)
    res['model'] = name
    results_sel.append(res)

# المقارنة النهائية (جدول + رسم بياني)
print("\n" + "="*60)
print("(الميزات المختارة vs جميع الميزات) المقارنة النهائية")
print("="*60)

comparison = pd.DataFrame({
    'Model': [r['model'] for r in results_all],
    'الدقة (جميع الميزات)': [r['accuracy'] for r in results_all],
    'الدقة (الميزات المختارة)': [r['accuracy'] for r in results_sel]
})
print(comparison)

# رسم بياني للمقارنة
plt.figure(figsize=(10,6))
x = np.arange(len(comparison))
width = 0.35
plt.bar(x - width/2, comparison['الدقة (جميع الميزات)', width, label='جميع الميزات', color='skyblue')
plt.bar(x + width/2, comparison['الدقة (الميزات المختارة)', width, label='الميزات المختارة', color='lightcoral')
plt.xlabel('النموذج')
plt.ylabel('الدقة')
plt.title('مقارنة أداء النماذج: جميع الميزات مقابل الميزات المختارة')
plt.xticks(x, comparison['Model'], rotation=45)
plt.legend()
plt.tight_layout()
plt.show()

print("\n🎉! تم إنجاز المشروع بنجاح")
print("🌟. الآن: تأكد من ظهور الجداول والأرقام والرسوم أعلاه")
print("📄 (سمّ الملف باسمك الكامل) File → Print → Save as PDF: ثم اذهب إلى 🌟")

```


...من جهازك (CSV) الرجاء اختيار ملف البيانات

اختيار الملفات realistic_soci...a_dataset.csv

realistic_social_media_dataset.csv(text/csv) - 200661 bytes, last modified: 15100% - 2026/8/ done

Saving realistic_social_media_dataset.csv to realistic_social_media_dataset (5).csv

✓ تم تحميل الملف بنجاح

أسماء جميع الأعمدة في ملفك:

['post_id', 'platform', 'content_type', 'category', 'creator_type', 'follower_count', 'account_age_days', 'hook_strength', '

🎯 هو 'viral': العمود المستهدف (الهدف)

توزيع الفئات (0 = غير فيروسي، 1 = فيروسي)

viral

1 755

0 745

Name: count, dtype: int64

الخطوة 2: معالجة القيم المفقودة

عدد القيم المفقودة قبل المعالجة (لكل عمود):

```
post_id      0
platform     0
content_type  0
category     0
creator_type  0
follower_count  0
account_age_days  0
hook_strength  0
relatability_score  0
thumbnail_quality  0
caption_length  0
hashtag_count  0
trending_audio  0
call_to_action  0
video_duration  0
post_time     0
day_of_week   0
emotion_trigger  0
views         0
likes         0
comments      0
shares        0
saves         0
impressions   0
reach         0
watch_time_avg  0
completion_rate  0
engagement_rate  0
first_3sec_retention  0
viral         0
dtype: int64
```

✓ إجمالي القيم المفقودة بعد التنظيف: 0

🔄 تحويل الأعمدة النصية إلى أرقام

✓ تم التحويل بنجاح

📊 عدد الميزات (الأعمدة المؤثرة): 29

📊 عدد العينات (الصفوف): 1500

📊 حجم بيانات التدريب: 1200

📊 حجم بيانات الاختبار: 300

الخطوة 3: تطبيق النموذج مع Grid Search

--- 🌲 Random Forest ---

🔍 أفضل إعدادات تم العثور عليها: {'max_depth': None, 'n_estimators': 100}

--- Random Forest ---

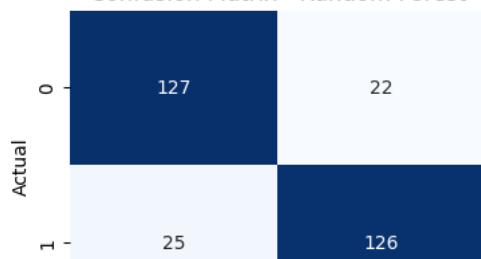
✓ الدقة (Accuracy): 0.8433

✓ الدقة الإيجابية (Precision): 0.8435

✓ الاسترجاع (Recall): 0.8433

✓ درجة F1: 0.8433

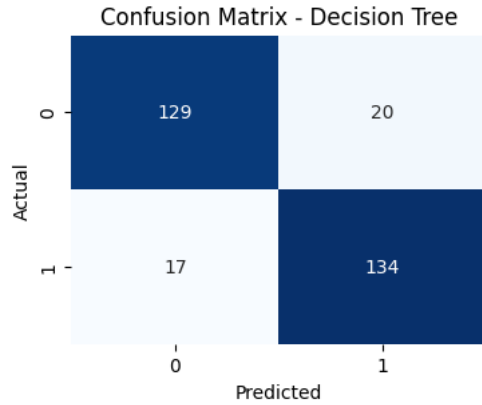
Confusion Matrix - Random Forest





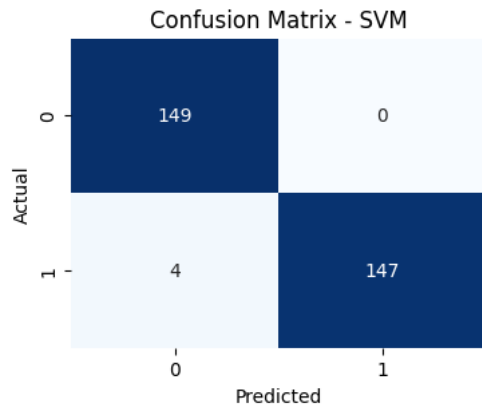
---  جاري تدريب نموذج Decision Tree ---
أفضل إعدادات تم العثور عليها: {'max_depth': None, 'min_samples_split': 5}

--- Decision Tree ---
✓ الدقة (Accuracy): 0.8767
✓ الدقة الإيجابية (Precision): 0.8768
✓ الاسترجاع (Recall): 0.8767
✓ درجة F1: 0.8766



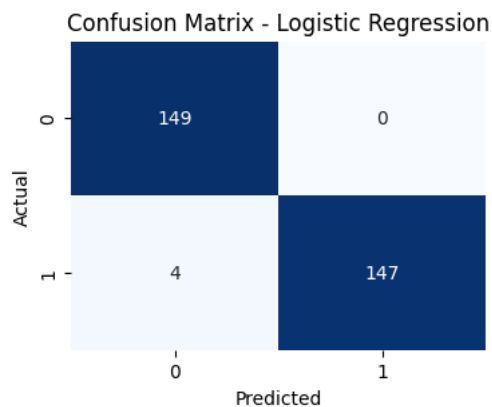
---  جاري تدريب نموذج SVM ---
أفضل إعدادات تم العثور عليها: {'C': 0.1, 'kernel': 'linear'}

--- SVM ---
✓ الدقة (Accuracy): 0.9867
✓ الدقة الإيجابية (Precision): 0.9870
✓ الاسترجاع (Recall): 0.9867
✓ درجة F1: 0.9867



---  جاري تدريب نموذج Logistic Regression ---
أفضل إعدادات تم العثور عليها: {'C': 0.1}

--- Logistic Regression ---
✓ الدقة (Accuracy): 0.9867
✓ الدقة الإيجابية (Precision): 0.9870
✓ الاسترجاع (Recall): 0.9867
✓ درجة F1: 0.9867

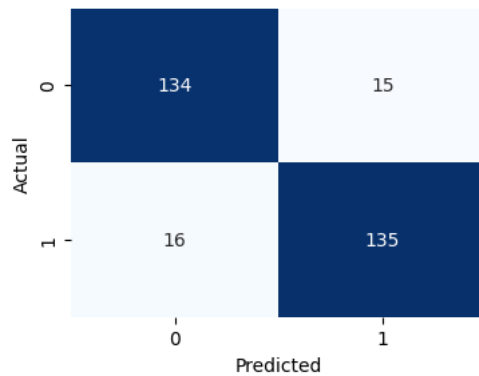


---  جاري تدريب نموذج KNN ---
أفضل إعدادات تم العثور عليها: {'n_neighbors': 7}

--- KNN ---

✓ الدقة (Accuracy): 0.8967
✓ الدقة الإيجابية (Precision): 0.8967
✓ الاسترجاع (Recall): 0.8967
✓ درجة F1: 0.8967

Confusion Matrix - KNN



=====

الخطوة 4: اختيار الميزات - الطريقتين المطلوبتين

=====

--- الطريقة 1: Chi-Square Test ---

✓ Chi-Square: ['follower_count', 'views', 'likes', 'comments', 'shares', 'saves', 'impressions', 'reach']

--- الطريقة 2: Recursive Feature Elimination (RFE) ---

✓ RFE: ['follower_count', 'views', 'likes', 'comments', 'shares', 'saves', 'impressions', 'reach']

=====

الخطوة 5: إعادة تطبيق النماذج على الميزات المختارة

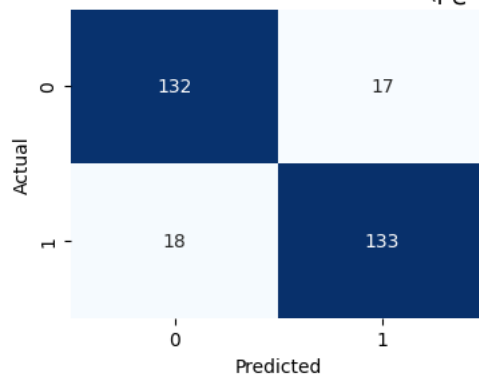
=====

--- (على الميزات المختارة) Random Forest: إعادة تدريب ---

--- Random Forest (مختار) ---

✓ الدقة (Accuracy): 0.8833
✓ الدقة الإيجابية (Precision): 0.8834
✓ الاسترجاع (Recall): 0.8833
✓ درجة F1: 0.8833

Confusion Matrix - Random Forest (راتخم)

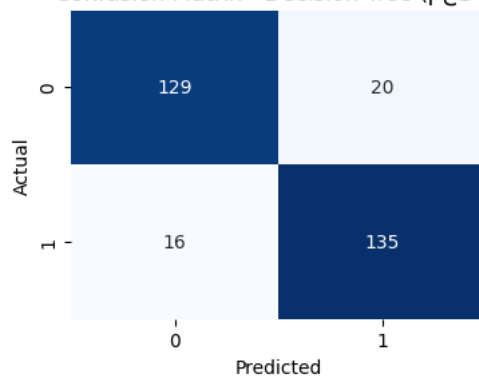


--- (على الميزات المختارة) Decision Tree: إعادة تدريب ---

--- Decision Tree (مختار) ---

✓ الدقة (Accuracy): 0.8800
✓ الدقة الإيجابية (Precision): 0.8802
✓ الاسترجاع (Recall): 0.8800
✓ درجة F1: 0.8800

Confusion Matrix - Decision Tree (راتخم)

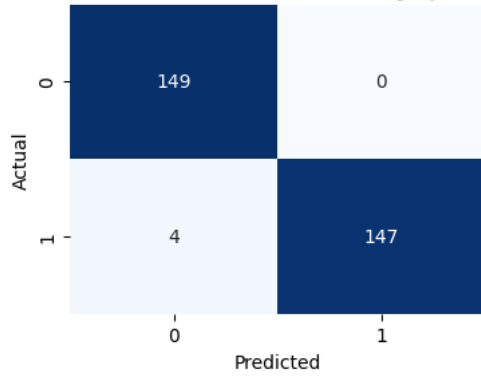


--- (على الميزات المختارة) SVM: إعادة تدريب ---

--- (مختار) SVM ---

✓ الدقة (Accuracy): 0.9867
✓ الدقة الإيجابية (Precision): 0.9870
✓ الاسترجاع (Recall): 0.9867
✓ F1: 0.9867

Confusion Matrix - SVM (راتخم)

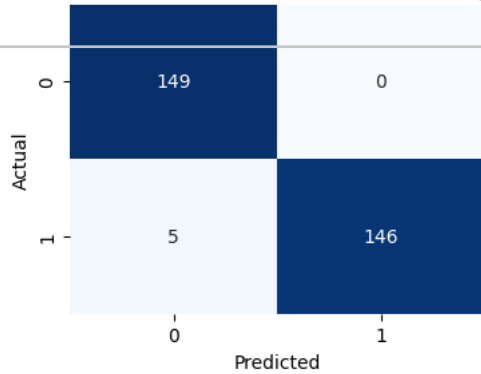


--- (على الميزات المختارة) Logistic Regression: إعادة تدريب ---

--- (مختار) Logistic Regression ---

✓ الدقة (Accuracy): 0.9833
✓ الدقة الإيجابية (Precision): 0.9839
✓ الاسترجاع (Recall): 0.9833
✓ F1: 0.9833

Confusion Matrix - Logistic Regression (راتخم)



--- (على الميزات المختارة) KNN: إعادة تدريب ---

--- (مختار) KNN ---

✓ الدقة (Accuracy): 0.9167