

Stroke Prediction Dataset
<https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset?resource=download>
Student: Mohammed Omar Abdullah Alqathmi

```
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive
```

استيراد المكتبات

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

from sklearn.feature_selection import SelectKBest, mutual_info_classif, RFE

from sklearn.metrics import (
    accuracy_score, precision_score, recall_score,
    f1_score, confusion_matrix, classification_report
)
```

قراءة البيانات

```
df = pd.read_csv("/content/drive/MyDrive/Project 2/healthcare-dataset-stroke-data.csv")
df.head()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60162	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1

Next steps: [Generate code with df](#) [New interactive sheet](#)

وتنظيف مبدئي (Target) تحديد الهدف

```
TARGET = "stroke"

# لأنه غير مفيد id حذف عمود
if "id" in df.columns:
    df.drop(columns=["id"], inplace=True)

X = df.drop(columns=[TARGET])
y = df[TARGET].astype(int)

print(y.value_counts())

stroke
0    4861
1     249
Name: count, dtype: int64
```

تقسيم الأعمدة (رقمية / نصية)

```
categorical_cols = X.select_dtypes(include=["object"]).columns.tolist()
numerical_cols = X.select_dtypes(exclude=["object"]).columns.tolist()

print("Categorical:", categorical_cols)
print("Numerical:", numerical_cols)

Categorical: ['gender', 'ever_married', 'work_type', 'Residence_type', 'smoking_status']
Numerical: ['age', 'hypertension', 'heart_disease', 'avg_glucose_level', 'bmi']
```

معالجة القيم المفقودة + التجهيز (Preprocessing)

```
numeric_pipeline = Pipeline([
    ("imputer", SimpleImputer(strategy="median")),
    ("scaler", StandardScaler())
])

categorical_pipeline = Pipeline([
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("onehot", OneHotEncoder(handle_unknown="ignore"))
])

preprocessor = ColumnTransformer([
    ("num", numeric_pipeline, numerical_cols),
    ("cat", categorical_pipeline, categorical_cols)
])
```

Train / Test تقديم البيانات

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)
```

دالة تقييم النموذج

```
def evaluate_model(name, y_true, y_pred):
    print(f"\n==== {name} =====")
    print("Accuracy :", accuracy_score(y_true, y_pred))
    print("Precision:", precision_score(y_true, y_pred, zero_division=0))
    print("Recall   :", recall_score(y_true, y_pred, zero_division=0))
    print("F1 Score :", f1_score(y_true, y_pred, zero_division=0))
    print("\nClassification Report:\n", classification_report(y_true, y_pred, zero_division=0))

    cm = confusion_matrix(y_true, y_pred)
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
    plt.title(name)
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()
```

GridSearch (مربعات) + السمات Feature Selection

```
pipe_lr = Pipeline([
    ("prep", preprocessor),
    ("model", LogisticRegression(class_weight="balanced", max_iter=500))
])

param_lr = {
    "model__C": [0.01, 0.1, 1, 10],
    "model__solver": ["liblinear", "lbfgs"]
}

grid_lr = GridSearchCV(pipe_lr, param_lr, cv=5, scoring="f1", n_jobs=-1)
grid_lr.fit(X_train, y_train)

y_pred_lr = grid_lr.predict(X_test)
evaluate_model("Logistic Regression (Clean)", y_test, y_pred_lr)
```

```
==== Logistic Regression (Clean) ====
Accuracy : 0.7455968688845401
Precision: 0.13793183448275862
Recall   : 0.8
F1 Score : 0.23529411764785882

Classification Report:
      precision    recall  f1-score   support

0               0.99      0.74      0.85        972
1               0.14      0.88      0.24         58

accuracy          0.56
macro avg         0.77
weighted avg      0.94      0.75      0.82        1022
```

Logistic Regression (Clean)

	Predicted 0	Predicted 1	Support
Actual 0	722	250	972
Actual 1	10	40	50

Random Forest

```
pipe_rf = Pipeline([
    ("prep", preprocessor),
    ("model", RandomForestClassifier(class_weight="balanced", random_state=42))
])

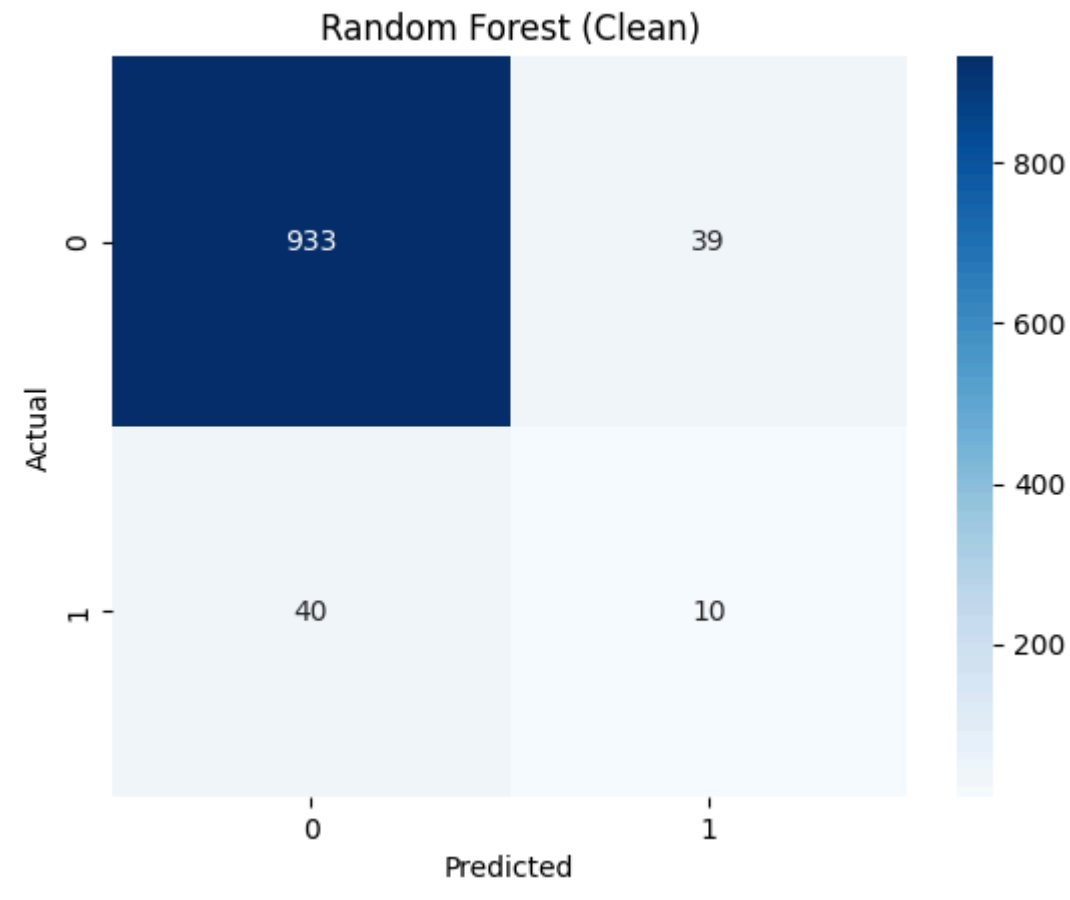
param_rf = {
    "model__n_estimators": [100, 200],
    "model__max_depth": [None, 10, 20]
}

grid_rf = GridSearchCV(pipe_rf, param_rf, cv=5, scoring="f1", n_jobs=-1)
grid_rf.fit(X_train, y_train)

y_pred_rf = grid_rf.predict(X_test)
evaluate_model("Random Forest (Clean)", y_test, y_pred_rf)
```

==== Random Forest (Clean) =====
Accuracy : 0.9227085870841487
Precision: 0.20408163265306123
Recall : 0.2
F1 Score : 0.20202020202020202

Classification Report:				
	precision	recall	f1-score	support
0	0.96	0.96	0.96	972
1	0.20	0.20	0.20	50
accuracy				0.92 1022
macro avg	0.58	0.58	0.58	1022
weighted avg	0.92	0.92	0.92	1022



SVM

```
pipe_svm = Pipeline([
    ("prep", preprocessor),
    ("model", SVC(class_weight="balanced"))
])

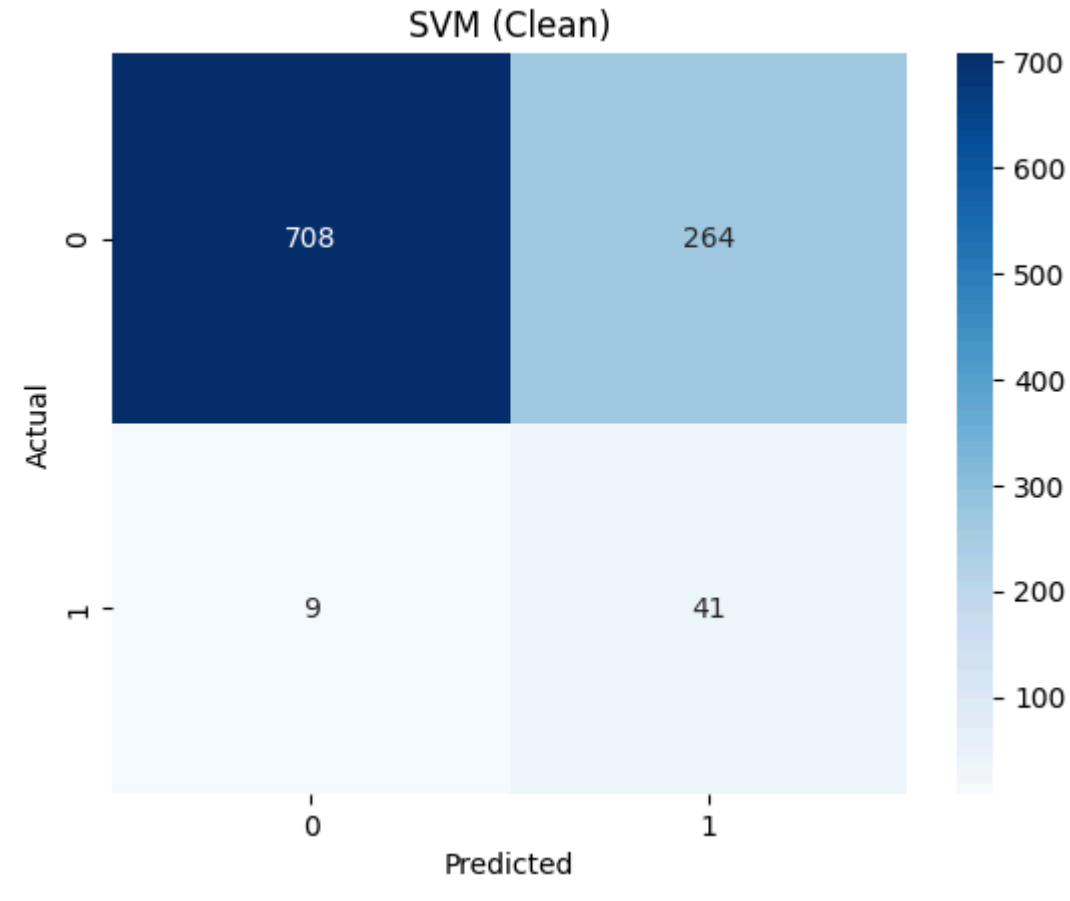
param_svm = {
    "model__C": [0.1, 1, 10],
    "model__kernel": ["linear", "rbf"]
}

grid_svm = GridSearchCV(pipe_svm, param_svm, cv=5, scoring="f1", n_jobs=-1)
grid_svm.fit(X_train, y_train)

y_pred_svm = grid_svm.predict(X_test)
evaluate_model("SVM (Clean)", y_test, y_pred_svm)
```

==== SVM (Clean) =====
Accuracy : 0.7328767123287672
Precision: 0.13442622950819672
Recall : 0.82
F1 Score : 0.230908591549295774

Classification Report:				
	precision	recall	f1-score	support
0	0.99	0.73	0.84	972
1	0.13	0.82	0.23	50
accuracy				0.73 1022
macro avg	0.56	0.77	0.53	1022
weighted avg	0.95	0.73	0.81	1022



9 Feature Selection – الطريقة الألي – (SelectKBest)

```
kbest = SelectKBest(score_func=mutual_info_classif, k=15)

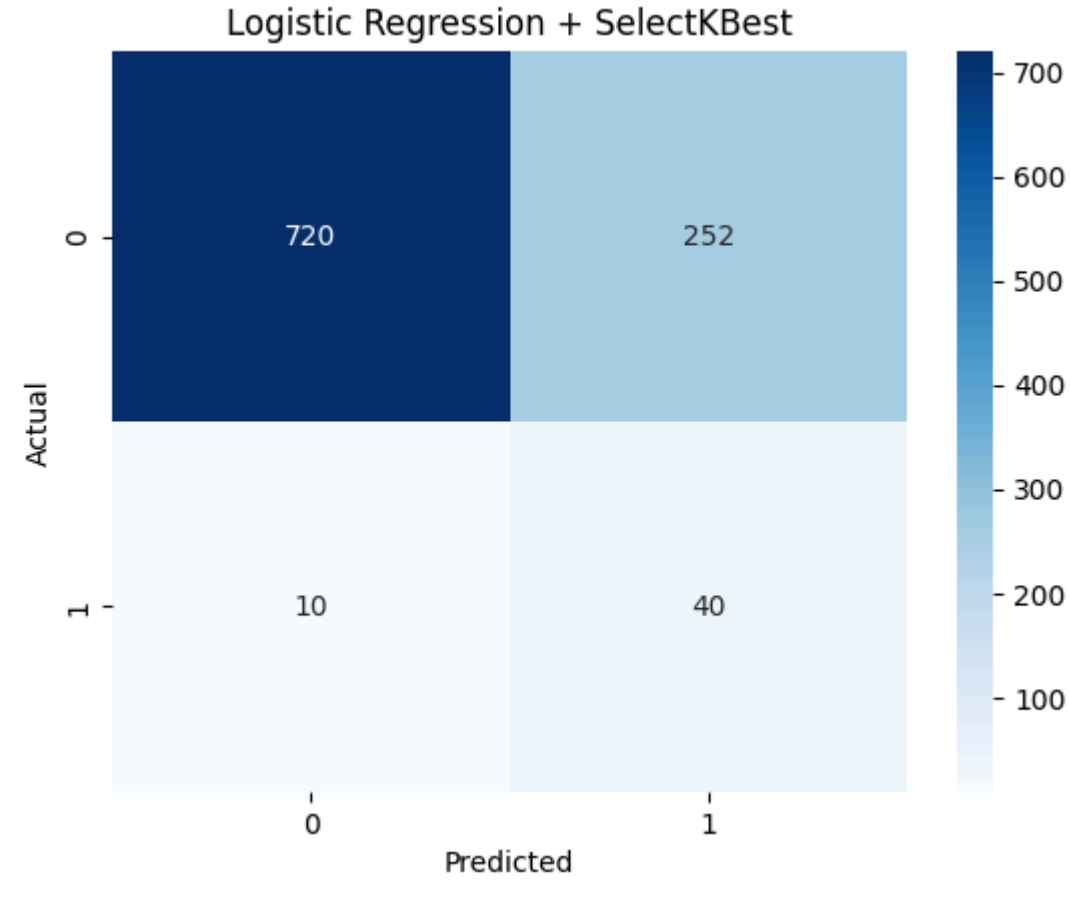
pipe_kbest = Pipeline([
    ("prep", preprocessor),
    ("select", kbest),
    ("model", LogisticRegression(class_weight="balanced", max_iter=500))
])

pipe_kbest.fit(X_train, y_train)
y_pred_kbest = pipe_kbest.predict(X_test)

evaluate_model("Logistic Regression + SelectKBest", y_test, y_pred_kbest)
```

==== Logistic Regression + SelectKBest =====
Accuracy : 0.7436399217221135
Precision: 0.136086301360863
Recall : 0.8
F1 Score : 0.23391812865497075

Classification Report:				
	precision	recall	f1-score	support
0	0.99	0.74	0.85	972
1	0.14	0.80	0.23	50
accuracy				0.74 1022
macro avg	0.56	0.77	0.54	1022
weighted avg	0.94	0.74	0.82	1022



10 Feature Selection – الطريقة التكرارية – (RFE)

```
rfe = RFE(
    estimator=LogisticRegression(class_weight="balanced", max_iter=500),
    n_features_to_select=15
)

pipe_rfe = Pipeline([
    ("prep", preprocessor),
    ("select", rfe),
    ("model", LogisticRegression(class_weight="balanced", max_iter=500))
])

pipe_rfe.fit(X_train, y_train)
y_pred_rfe = pipe_rfe.predict(X_test)

evaluate_model("Logistic Regression + RFE", y_test, y_pred_rfe)
```

==== Logistic Regression + RFE =====
Accuracy : 0.744618395303269
Precision: 0.13745704467353953
Recall : 0.8
F1 Score : 0.23460410557184752

Classification Report:				
	precision	recall	f1-score	support
0	0.99	0.74	0.85	972
1	0.14	0.80	0.23	50
accuracy				0.74 1022
macro avg	0.56	0.77	0.54	1022
weighted avg	0.94	0.74	0.82	1022

