```
import os
print(os.listdir())

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split

import os

print(os.getcwd())

import os

print(os.listdir())

import os

print(os.listdir('data'))

data = pd.read_csv('data/original_data/heart.csv')

data.head()

from sklearn.model_selection import train_test_split

X = data.drop('target', axis=1)
y = data['target']

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

print("تم التقسيم بنجاح!")

import os
print(os.getcwd())  # هذا سيعرض المسار الحالي الذي تعمل فيه

# تحديد المسار الصحيح للملفات
x_train_path = r'C:\Users\shab1\project_heart\data\Preprocessed_data\x_train.csv'
x_test_path = r'C:\Users\shab1\project_heart\data\Preprocessed_data\x_test.csv'
y_train_path = r'C:\Users\shab1\project_heart\data\Preprocessed_data\y_train.csv'
y_test_path = r'C:\Users\shab1\project_heart\data\Preprocessed_data\y_test.csv'

# قراءة الملفات باستخدام pandas
x_train = pd.read_csv(x_train_path)
x_test = pd.read_csv(x_test_path)
y_train = pd.read_csv(y_train_path)
y_test = pd.read_csv(y_test_path)

# طباعة عينة من البيانات للتحقق
```

```
print(x_train.head())
print(y_train.head())

# عرض أول 5 صفوف من البيانات
data.head()

# فحص وجود قيم مفقودة
data.isnull().sum()

# تقسيم البيانات إلى مدخلات (X) وهدف (y)
X = data.drop('target', axis=1)  # إزالة عمود target
y = data['target']  # عمود target هو الهدف

from sklearn.model_selection import train_test_split

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

from sklearn.preprocessing import StandardScaler

# تحجيم البيانات
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# تدريب النموذج على البيانات المحجّمة
model = LogisticRegression(max_iter=2000)
model.fit(X_train_scaled, y_train)

# التنبؤ باستخدام البيانات الاختبارية
y_pred = model.predict(X_test_scaled)

# حساب الدقة
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")

# حساب التقارير الأخرى مثل الدقة، الاسترجاع، و F1-Score
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))

import os

# تحديد المسار
folder_path = 'project_heart/data/results'

# إذا لم يكن المجلد موجودًا، نقوم بإنشائه
os.makedirs(folder_path, exist_ok=True)
```

```
# حفظ التوقعات في ملف CSV
predictions_df = pd.DataFrame(y_pred, columns=['Predictions'])
predictions_df.to_csv(f'{folder_path}/predictions_logistic_regression.csv', index=False)

import os

# تحديد المسار للمجلدات
original_data_dir = 'project_heart/data/original data'

# التأكد من وجود المجلد، وإذا لم يكن موجودًا يتم إنشاؤه
os.makedirs(original_data_dir, exist_ok=True)

# حفظ البيانات الأصلية في مجلد 'original data'
original_data_path = os.path.join(original_data_dir, 'heart.csv')
data.to_csv(original_data_path, index=False)

# تحديد المسارات للمجلدات المعالجة
preprocessed_data_dir = 'project_heart/data/preprocessed data'

# التأكد من وجود المجلد، وإذا لم يكن موجودًا يتم إنشاؤه
os.makedirs(preprocessed_data_dir, exist_ok=True)

# حفظ البيانات المُعالجة
X_train_path = os.path.join(preprocessed_data_dir, 'X.csv')
X_test_path = os.path.join(preprocessed_data_dir, 'X_test.csv')
y_train_path = os.path.join(preprocessed_data_dir, 'Y.csv')
y_test_path = os.path.join(preprocessed_data_dir, 'Y_test.csv')

# حفظ البيانات
X_train.to_csv(X_train_path, index=False)
X_test.to_csv(X_test_path, index=False)
y_train.to_csv(y_train_path, index=False)
y_test.to_csv(y_test_path, index=False)

# تحديد مسار مجلد النتائج
results_dir = 'project_heart/data/Results'

# التأكد من وجود المجلد، وإذا لم يكن موجودًا يتم إنشاؤه
os.makedirs(results_dir, exist_ok=True)

# حفظ التوقعات
predictions_path = os.path.join(results_dir, 'predictions_logistic_regression.csv')

# تحويل التوقعات إلى DataFrame ثم حفظها
predictions_df = pd.DataFrame(y_pred, columns=['Predictions'])
predictions_df.to_csv(predictions_path, index=False)
```

```
from sklearn.ensemble import RandomForestClassifier

# بناء الموديل
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

# تدريب الموديل
rf_model.fit(X_train_scaled, y_train)

# التوقعات
rf_predictions = rf_model.predict(X_test_scaled)

# حفظ التوقعات في ملف CSV داخل مجلد 'Results'
rf_predictions_df = pd.DataFrame(rf_predictions, columns=['Predictions'])
rf_predictions_df.to_csv('project_heart/data/results/predictions_RF_model.csv', index=False)

print("Random Forest model predictions saved.")


from sklearn.svm import SVC

# بناء الموديل
svm_model = SVC(random_state=42)

# تدريب الموديل
svm_model.fit(X_train_scaled, y_train)

# التوقعات
svm_predictions = svm_model.predict(X_test_scaled)

# حفظ التوقعات في ملف CSV داخل مجلد 'Results'
svm_predictions_df = pd.DataFrame(svm_predictions, columns=['Predictions'])
svm_predictions_df.to_csv('project_heart/data/results/predictions_SVM_model.csv',
index=False)

print("SVM model predictions saved.")


# Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Load Original Data
original_data = pd.read_csv('data/original_data/heart.csv')

# Check for missing values
print(original_data.isnull().sum())

# Preprocessing
X = original_data.drop('target', axis=1)
y = original_data['target']

# Normalize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split Data
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3,
random_state=42)

# Save Preprocessed Data
pd.DataFrame(X_train).to_csv('data/Preprocessed_data/X_train.csv', index=False)
pd.DataFrame(X_test).to_csv('data/Preprocessed_data/X_test.csv', index=False)
pd.DataFrame(y_train).to_csv('data/Preprocessed_data/y_train.csv', index=False)
pd.DataFrame(y_test).to_csv('data/Preprocessed_data/y_test.csv', index=False)

# Create results directory if it doesn't exist
os.makedirs('data/results', exist_ok=True)

# Models
models = {
    "Logistic Regression": LogisticRegression(),
    "Support Vector Machine": SVC(),
    "Random Forest": RandomForestClassifier()
}

# Train, Predict and Save Results
results = {}
for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    results[model_name] = accuracy

    # Save Predictions
    pred_df = pd.DataFrame(y_pred, columns=["Prediction"])
    pred_df.to_csv(f'data/results/predictions_{model_name.replace(" ", "_").lower()}.csv',
index=False)
```

```
# Display Results
for model_name, accuracy in results.items():
    print(f"{model_name}: {accuracy:.4f}")

# Visualizations
# Correlation Heatmap
plt.figure(figsize=(10,8))
sns.heatmap(original_data.corr(), annot=True, cmap='coolwarm')
plt.title('Feature Correlation Heatmap')
plt.show()

# Distribution of Target
sns.countplot(x='target', data=original_data)
plt.title('Distribution of Target Variable')
plt.show()

# Accuracy Table
accuracy_df = pd.DataFrame(list(results.items()), columns=['Model', 'Accuracy'])
print(accuracy_df)

# Bar plot of model accuracy
sns.barplot(x='Model', y='Accuracy', data=accuracy_df)
plt.title('Model Accuracy Comparison')
plt.ylim(0,1)
plt.show()


data = pd.read_csv('data/original_data/heart.csv')


import os
print(os.listdir('data'))

data = pd.read_csv('data/original_data/heart.csv')


import pandas as pd
import matplotlib.pyplot as plt
import os

# إنشاء مجلد لحفظ الرسومات إذا لم يكن موجوداً
os.makedirs('data/result/figures', exist_ok=True)

# تحميل البيانات من المسار الصحيح
data = pd.read_csv('data/original_data/heart.csv')

# مثال 1: رسم توزيع الأعمار
```

```
plt.figure(figsize=(8,6))
plt.hist(data['age'], bins=20, color='skyblue', edgecolor='black')
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Count')
plt.grid(True)
plt.tight_layout()
plt.savefig('data/result/figures/age_distribution.png')
plt.close()

# مثال 2: رسم علاقة بين العمر والضغط الدموي
plt.figure(figsize=(8,6))
plt.scatter(data['age'], data['trestbps'], color='red', alpha=0.6)
plt.title('Age vs Resting Blood Pressure')
plt.xlabel('Age')
plt.ylabel('Resting Blood Pressure')
plt.grid(True)
plt.tight_layout()
plt.savefig('data/result/figures/age_vs_blood_pressure.png')
plt.close()

# مثال 3: رسم أعمدة للعدد حسب الجنس
plt.figure(figsize=(6,5))
data['sex'].value_counts().plot(kind='bar', color=['blue', 'pink'])
plt.title('Gender Count')
plt.xlabel('Sex (0 = Female, 1 = Male)')
plt.ylabel('Count')
plt.tight_layout()
plt.savefig('data/result/figures/gender_count.png')
plt.close()
```