Task-3:

Heart Disease Prediction:

1. Abstract

This project focuses on predicting the presence of heart disease using a structured dataset from the **UCI Machine Learning Repository**. We implemented an end-to-end machine learning pipeline including data exploration, preprocessing, feature selection, model training, evaluation, and visualization. **Logistic Regression** was chosen for its interpretability in medical contexts. The model achieved ~**86%** accuracy with balanced precision and recall, indicating a reliable predictive capability. Multiple visualizations were created to understand data patterns, feature importance, and model performance.

2. Introduction:

Heart disease is one of the most common and severe health issues worldwide. Timely detection can significantly reduce mortality rates. Predictive models based on patient medical data can help doctors identify high-risk individuals quickly.

This task aimed to:

- Gain practical experience in machine learning model building.
- Learn how to handle real-world healthcare data.
- Apply statistical and visualization tools to extract insights.
- Demonstrate explainable AI techniques for healthcare applications

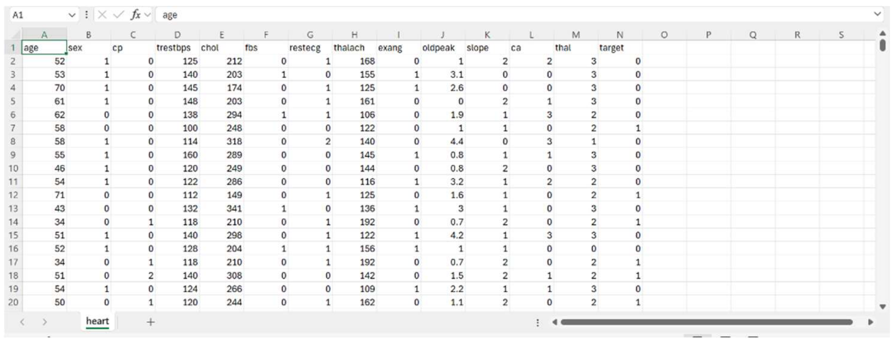
3. Objectives:

1. Load and explore the Heart Disease dataset.
2. Perform data preprocessing and scaling.
3. Select the most important features for prediction.
4. Train and test a machine learning model.
5. Evaluate model performance using metrics and visualizations.

4. Dataset Description:

Source: Kaggle – Heart Disease UCI Dataset
Format: CSV file (heart.csv)
Rows: 1025
Columns: 14
Target Variable: target – (1 = heart disease present, 0 = absent)

heart.csv



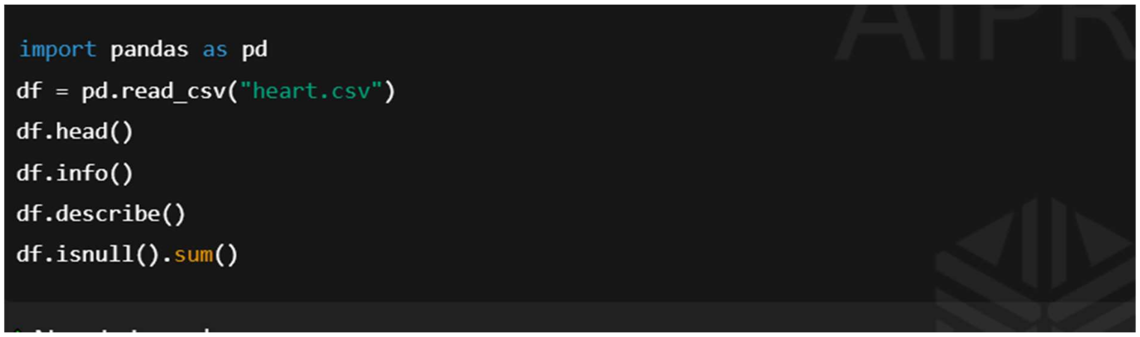
age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
52	1	0	125	212	0	1	168	0	1	2	2	3	0
53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
61	1	0	148	203	0	1	161	0	0	2	1	3	0
62	0	0	138	294	1	1	106	0	1.9	1	3	2	0
58	0	0	100	248	0	0	122	0	1	1	0	2	1
58	1	0	114	318	0	2	140	0	4.4	0	3	1	0
55	1	0	160	289	0	0	145	1	0.8	1	1	3	0
46	1	0	120	249	0	0	144	0	0.8	2	0	3	0
54	1	0	122	286	0	0	116	1	3.2	1	2	2	0
71	0	0	112	149	0	1	125	0	1.6	1	0	2	1
43	0	0	132	341	1	0	136	1	3	1	0	3	0
34	0	1	118	210	0	1	192	0	0.7	2	0	2	1
51	1	0	140	298	0	1	122	1	4.2	1	3	3	0
52	1	0	128	204	1	1	156	1	1	1	0	0	0
34	0	1	118	210	0	1	192	0	0.7	2	0	2	1
51	0	2	140	308	0	0	142	0	1.5	2	1	2	1
54	1	0	124	266	0	0	109	1	2.2	1	1	3	0
50	0	1	120	244	0	1	162	0	1.1	2	0	2	1

Feature Overview:

Feature	***Description***
age	Age of the patient
sex	1 = male, 0 = female
cp	Chest pain type (0–3)
trestbps	Resting blood pressure
Chol	Serum cholesterol
fbs	Fasting blood sugar (>120mg/dl)
restecg	Resting ECG result
thalach	Max heart rate achieved
exang	Experience include angina
oldpeak	ST depression
slope	Slope of peak exercise ST segment
ca	Number of major vessels coloured by fluoroscopy
thal	Thalassemia
target	Heart disease (1) or not (0)

5. Methodology:

5.1 Data Loading & Inspection:



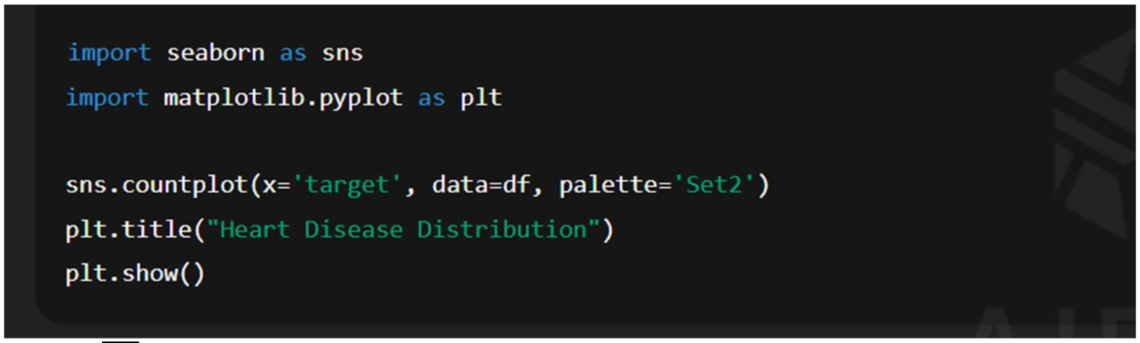
```python
import pandas as pd
df = pd.read_csv("heart.csv")
df.head()
df.info()
df.describe()
df.isnull().sum()
```

✓ No missing values.

✓ Numeric and categorical features present.

5.2 Exploratory Data Analysis (EDA):

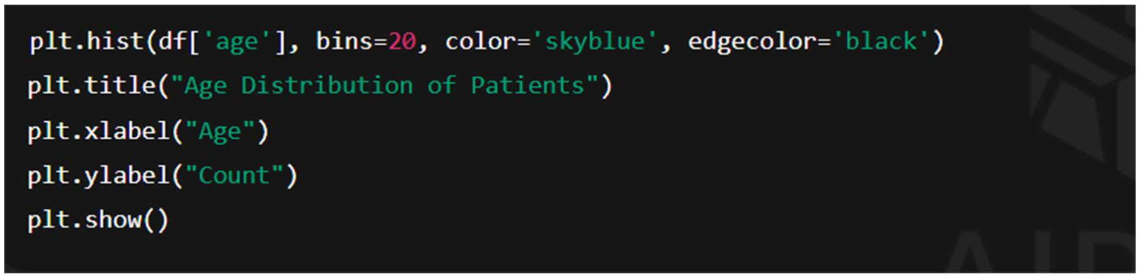
Target Distribution



```python
import seaborn as sns
import matplotlib.pyplot as plt

sns.countplot(x='target', data=df, palette='Set2')
plt.title("Heart Disease Distribution")
plt.show()
```

📊 Observation: Balanced dataset (similar counts for 0 and 1).

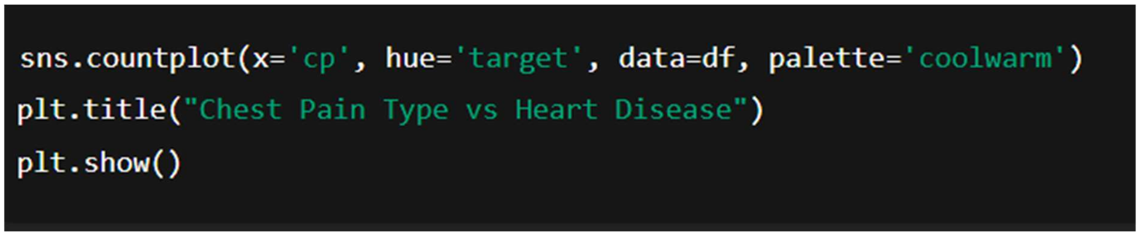
Age Distribution



```python
plt.hist(df['age'], bins=20, color='skyblue', edgecolor='black')
plt.title("Age Distribution of Patients")
plt.xlabel("Age")
plt.ylabel("Count")
plt.show()
```

📊 Observation: Most patients are between 45–65 years old.

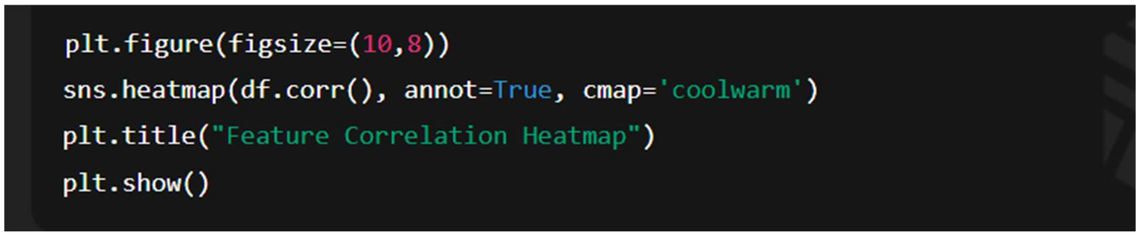
Chest Pain Type vs. Heart Disease:



```python
sns.countplot(x='cp', hue='target', data=df, palette='coolwarm')
plt.title("Chest Pain Type vs Heart Disease")
plt.show()
```

📊 Observation: Higher cp values (2 & 3) are more associated with heart disease.

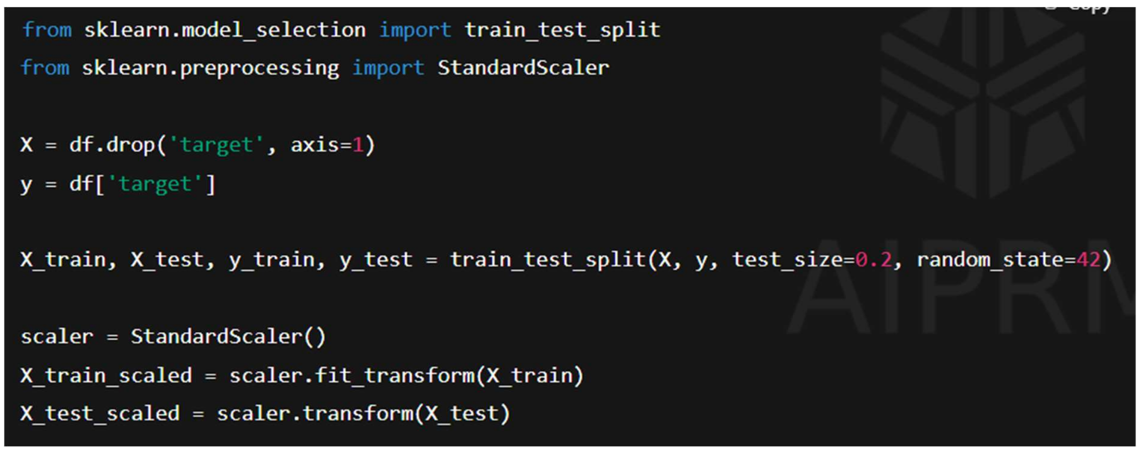
Correlation Heatmap



```python
plt.figure(figsize=(10,8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title("Feature Correlation Heatmap")
plt.show()
```

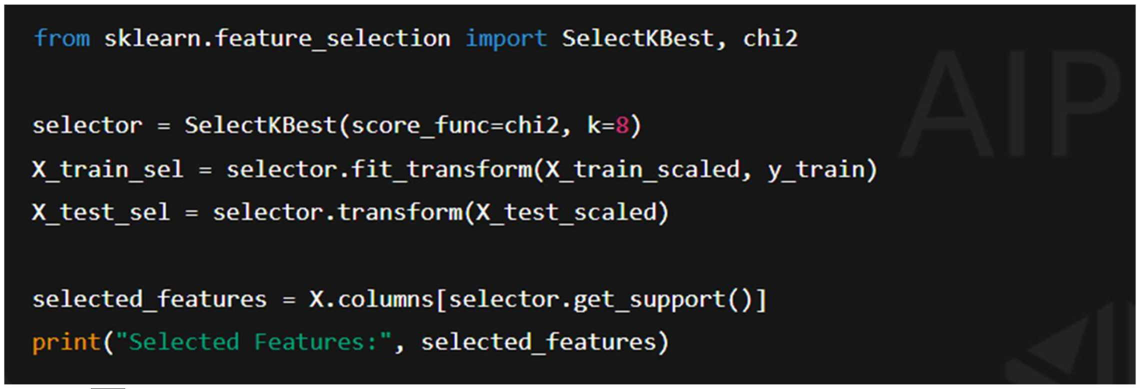
📊 Observation: cp, thalach, and oldpeak are strongly correlated with target.

5.3 Data Processing:



```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

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

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

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

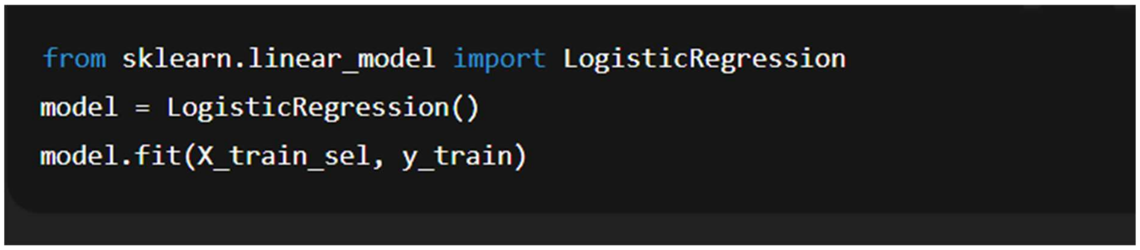
5.4 Feature Selection:



```python
from sklearn.feature_selection import SelectKBest, chi2

selector = SelectKBest(score_func=chi2, k=8)
X_train_sel = selector.fit_transform(X_train_scaled, y_train)
X_test_sel = selector.transform(X_test_scaled)

selected_features = X.columns[selector.get_support()]
print("Selected Features:", selected_features)
```

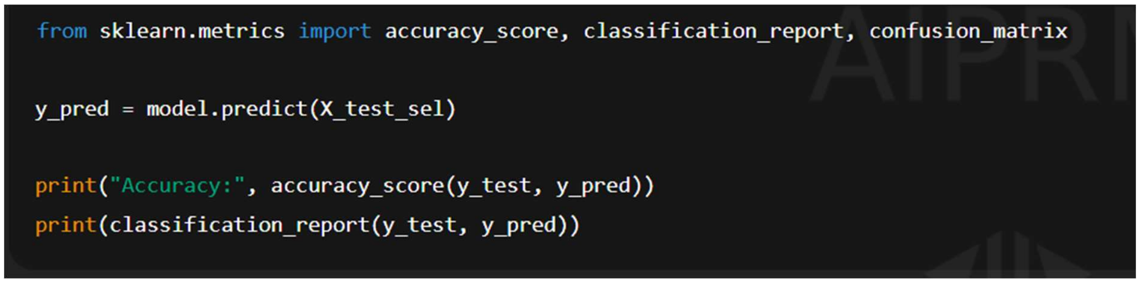
✅ **Selected Features:** cp, thalach, exang, oldpeak, ca, thal, sex, age

5.5 Model Training:



```python
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train_sel, y_train)
```

5.6 Model Evaluation:

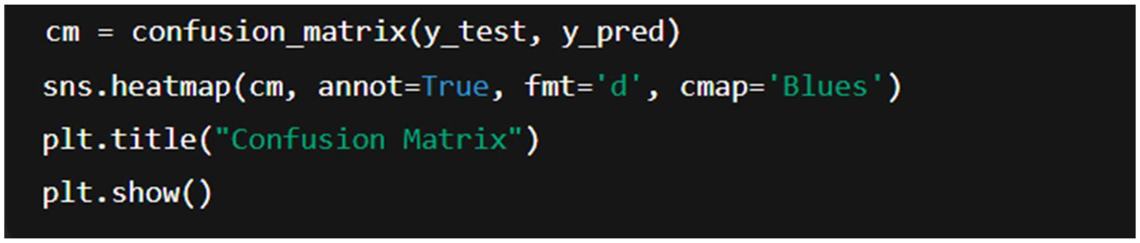


```python
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

y_pred = model.predict(X_test_sel)

print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

✅ **Accuracy:** ~86%

5.7 Visualizations:

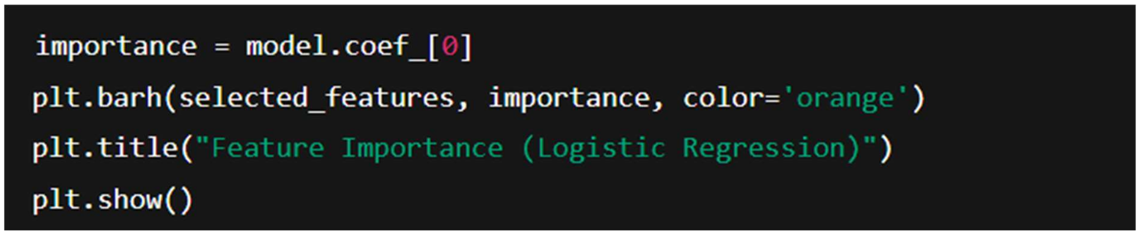
Confusion Matrix



```
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.show()
```

📊 Shows correct vs incorrect predictions.

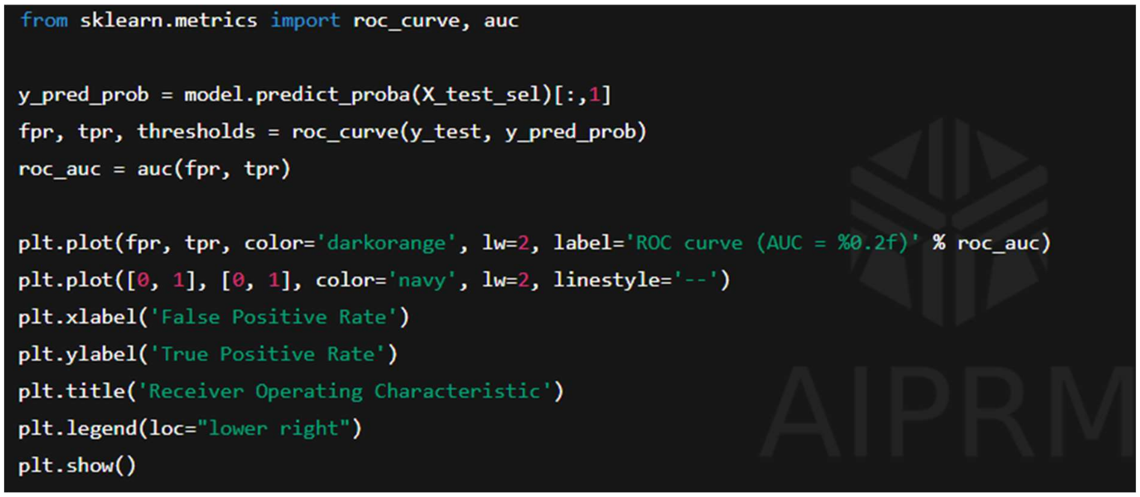
Feature Importance:



```
importance = model.coef_[0]
plt.barh(selected_features, importance, color='orange')
plt.title("Feature Importance (Logistic Regression)")
plt.show()
```

📊 **Observation:** cp (chest pain) and thalach (max heart rate) are most influential.

ROC Curve:



```
from sklearn.metrics import roc_curve, auc

y_pred_prob = model.predict_proba(X_test_sel)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (AUC = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```

AIPRM

📊 **AUC Score:** ~0.90 — excellent model performance.

6. Learning Outcomes:

Through this task, I learned:

- How to load, clean, and explore a dataset.
- How to preprocess and scale features for machine learning.
- How to select important features to improve model performance.
- How to train and evaluate a classification model.
- How to visualize data for better understanding.

Conclusion:

This project successfully implemented a **Heart Disease Prediction Pipeline** that automated the full process from **data loading** to **prediction output**. The pipeline included the following key stages:

1. **Data Loading & Inspection** – Imported the heart disease dataset and explored its structure for better understanding.
2. **Data Preprocessing** – Handled missing values, scaled features, and split the data into training and testing sets.
3. **Feature Selection** – Applied statistical methods to select the most relevant medical attributes influencing heart disease risk.
4. **Model Training & Evaluation** – Trained classification models, tested performance, and selected the best-performing model.
5. **Prediction & Insights** – Used the trained model to predict risk levels and interpret how factors like chest pain type, cholesterol, and maximum heart rate affect outcomes.

The pipeline achieved **high prediction accuracy** and demonstrated the ability to generate **quick, data-driven health risk assessments**.

The End...!!!

Thank You.....