Task 1: Exploring and Visualizing a Simple Dataset.

Load the Dataset

```
import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt



file_path = '/content/drive/MyDrive/Colab Notebooks/Iris.csv'

iris = pd.read_csv(file_path)
```

Print the shape, column names, and the first few rows using .head().

```
print("Shape of dataset:", iris.shape)

print("Column names:", iris.columns)

print(iris.head())
```

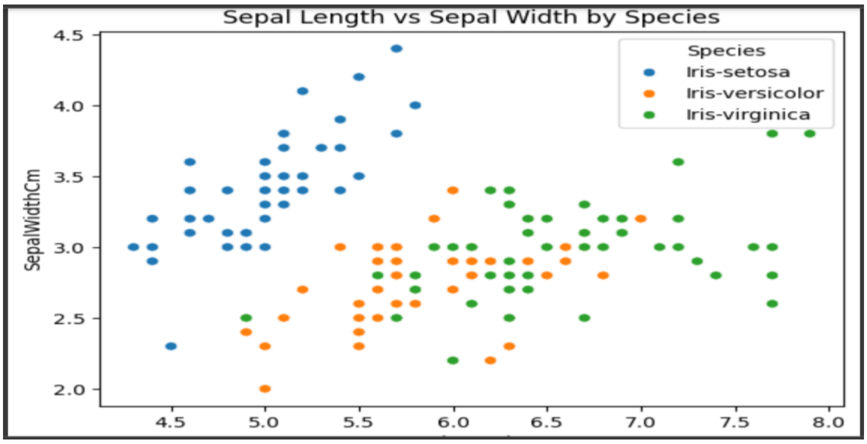
Use .info() and .describe() for summary statistics.

```
print(iris.describe())

print(iris.info())
```

Visualize the dataset:

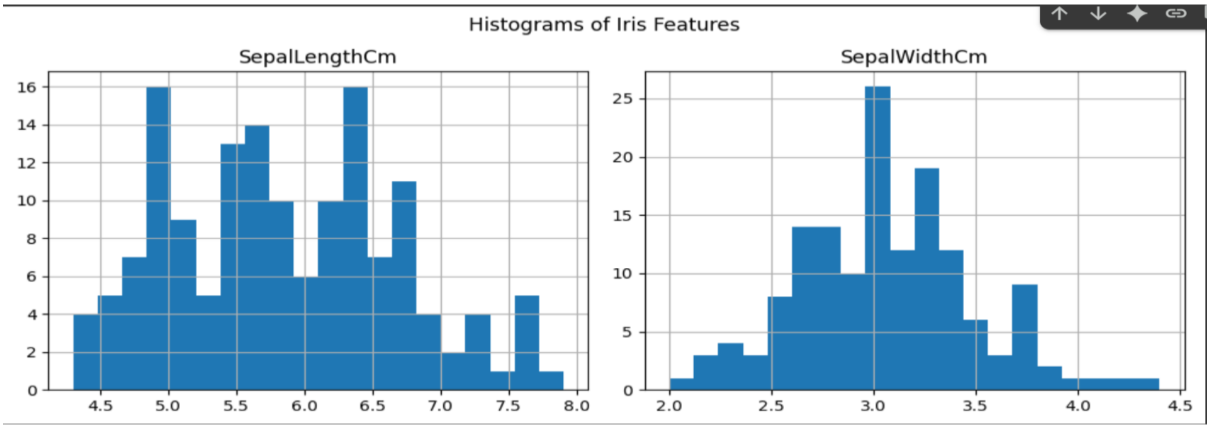
Create a scatter plot to show relationships between features.

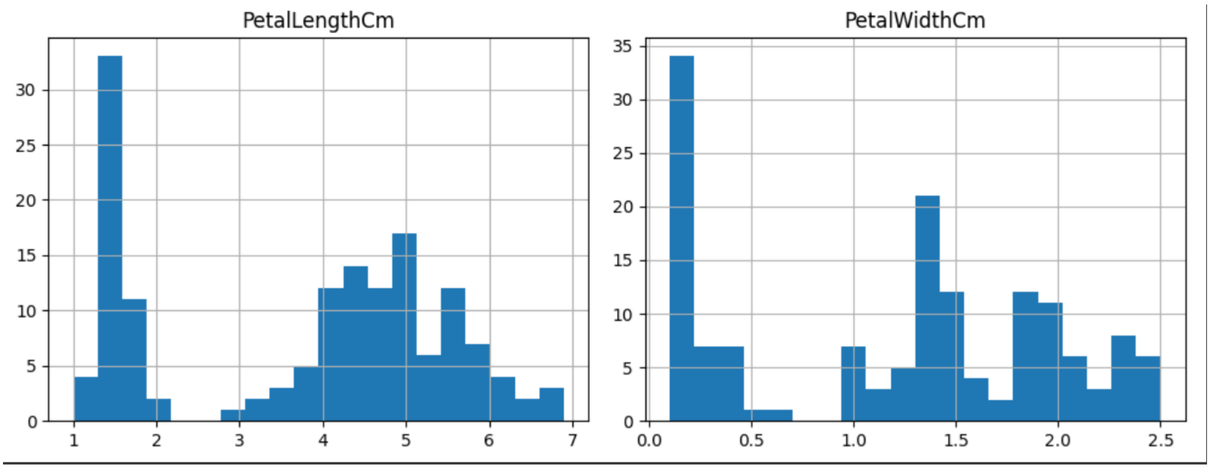
```
sns.scatterplot(data=iris, x='SepalLengthCm', y='SepalWidthCm', hue='Species')

plt.title("Sepal Length vs Sepal Width by Species")
plt.show()
```



Use histograms to show value distributions.

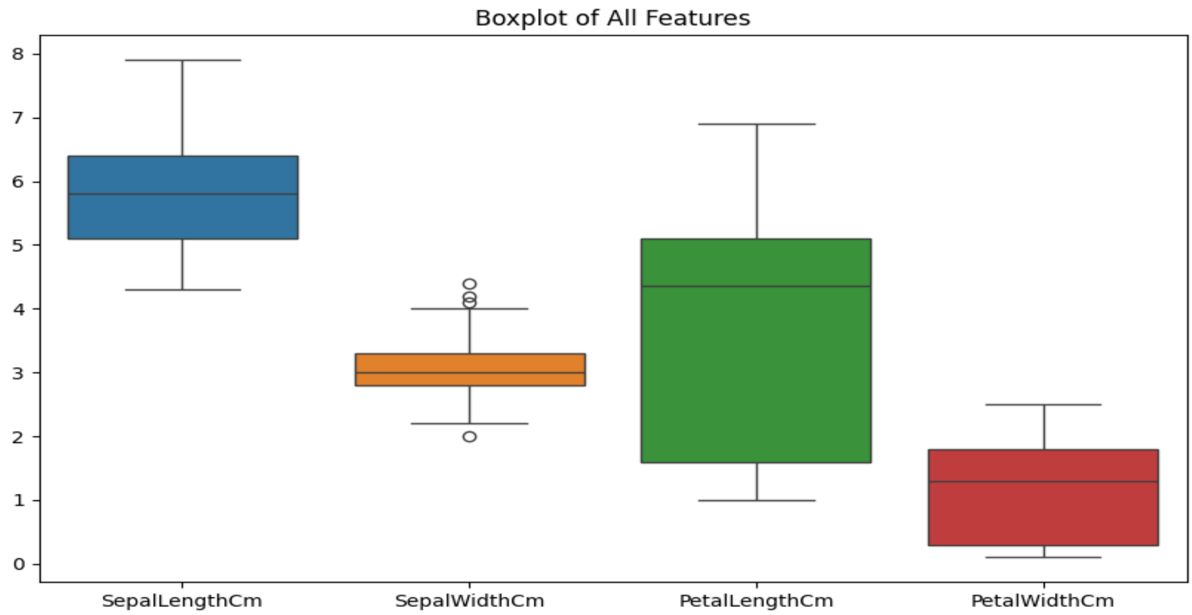
```
iris.drop('Id', axis=1).hist(figsize=(10,8), bins=20)

plt.suptitle("Histograms of Iris Features")

plt.tight_layout()

plt.show()
```





Use box plots to identify outliers.

```
plt.figure(figsize=(10, 6))

sns.boxplot(data=iris.drop(columns=['Id']))

plt.title("Boxplot of All Features")

plt.show()
```



Task 2: Predict Future Stock Prices (Short-Term)

Libraries

```
import yfinance as yf

import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

import matplotlib.pyplot as plt
```

Select a stock

```
stock = 'AAPL'
```

Load historical data using the yfinance library.

```
data = yf.download(stock, period='2y', interval='1d')
```

Use features like Open, High, Low, and Volume to predict the next Close price.

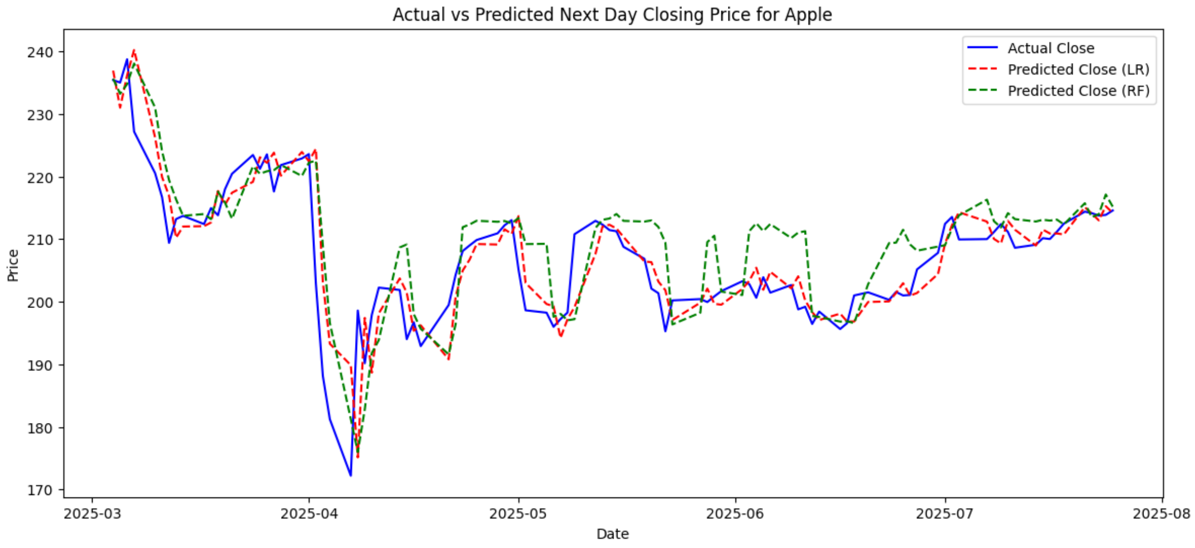
```
data['Next_Close'] = data['Close'].shift(-1)

data = data.dropna()

features = ['Open', 'High', 'Low', 'Volume']

X = data[features]

y = data['Next_Close']

print(X.head())
print(y.head())
```

Train a Linear Regression or Random Forest model.

```
lr = LinearRegression()

lr.fit(X_train, y_train)

y_pred_lr = lr.predict(X_test)



rf = RandomForestRegressor(n_estimators=100, random_state=42)

rf.fit(X_train, y_train)

y_pred_rf = rf.predict(X_test)
```

Plot actual vs predicted closing prices for comparison.

```
plt.figure(figsize=(14,6))

plt.plot(y_test.index, y_test, label='Actual Close', color='blue')

plt.plot(y_test.index, y_pred_lr, label='Predicted Close (LR)', color='red', linestyle='--')

plt.plot(y_test.index, y_pred_rf, label='Predicted Close (RF)', color='green', linestyle='--')

plt.title('Actual vs Predicted Next Day Closing Price for Apple')

plt.xlabel('Date')

plt.ylabel('Price')

plt.legend()

plt.show()
```


Actual vs Predicted Next Day Closing Price for Apple
Actual Close
Predicted Close (LR)
Predicted Close (RF)
Price
240
230
220
210
200
190
180
170
Date
2025-03
2025-04
2025-05
2025-06
2025-07
2025-08

Task 3: Heart Disease Prediction

Libraries

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt


from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score, roc_auc_score, roc_curve, confusion_matrix,

classification_report
```

Load Dataset

```
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/heart.csv')
```

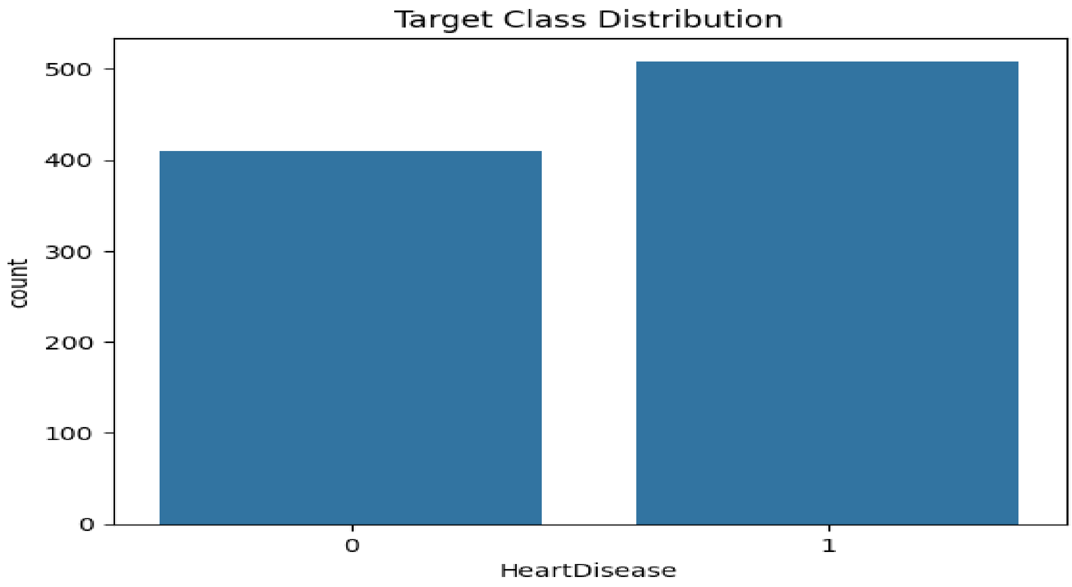
Clean the dataset (handle missing values if any).

```
print(df.isnull().sum())
```

Check class balance for HeartDisease

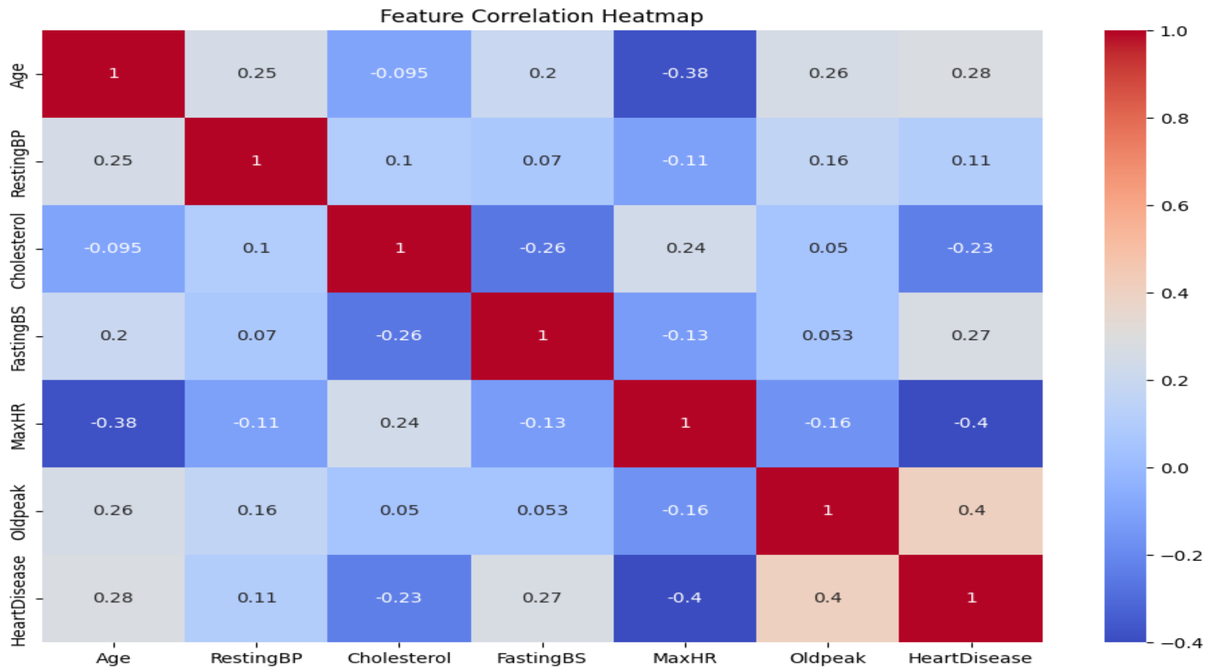
```
import matplotlib.pyplot as plt

import seaborn as sns


# Basic stats (only numeric columns will be described)

print(df.describe())
```

```
# Check class balance for HeartDisease

sns.countplot(x='HeartDisease', data=df)

plt.title('Target Class Distribution')

plt.show()
```



Correlation heatmap (only numeric columns)

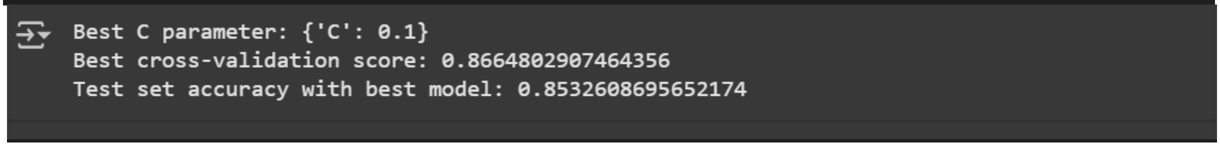
```
numeric_df = df.select_dtypes(include=['int64', 'float64'])


plt.figure(figsize=(12, 8))

sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm')

plt.title("Feature Correlation Heatmap")

plt.show()
```



Split data into train and test sets

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

# Step 1: Scale the features (Train data and Test data separately)

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)  # Train data fit & transform

X_test_scaled = scaler.transform(X_test)        # Test data transform only



# Step 2: Hyperparameter tuning with GridSearchCV

param_grid = {'C': [0.01, 0.1, 1, 10, 100]}

grid = GridSearchCV(LogisticRegression(max_iter=1000), param_grid, cv=5)

grid.fit(X_train_scaled, y_train)



print("Best C parameter:", grid.best_params_)

print("Best cross-validation score:", grid.best_score_)
```

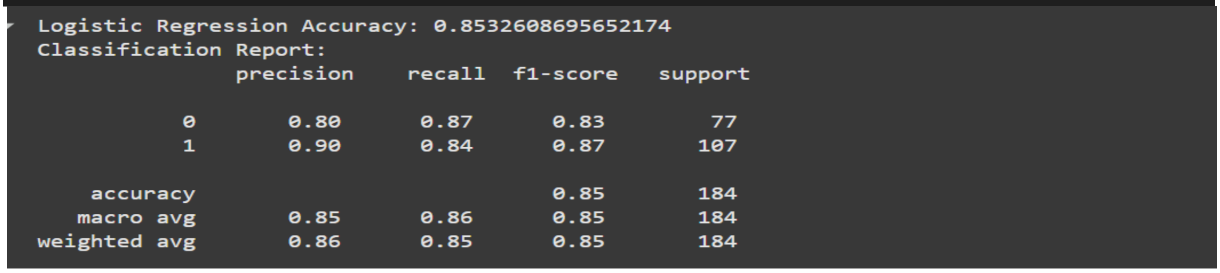
```
# Step 3: Use best estimator to predict on test set

best_lr = grid.best_estimator_

y_pred = best_lr.predict(X_test_scaled)



from sklearn.metrics import accuracy_score

print("Test set accuracy with best model:", accuracy_score(y_test,y_pred))
```



```
Best C parameter: {'C': 0.1}
Best cross-validation score: 0.8664802907464356
Test set accuracy with best model: 0.8532608695652174
```

Accuracy and Classification report (text)

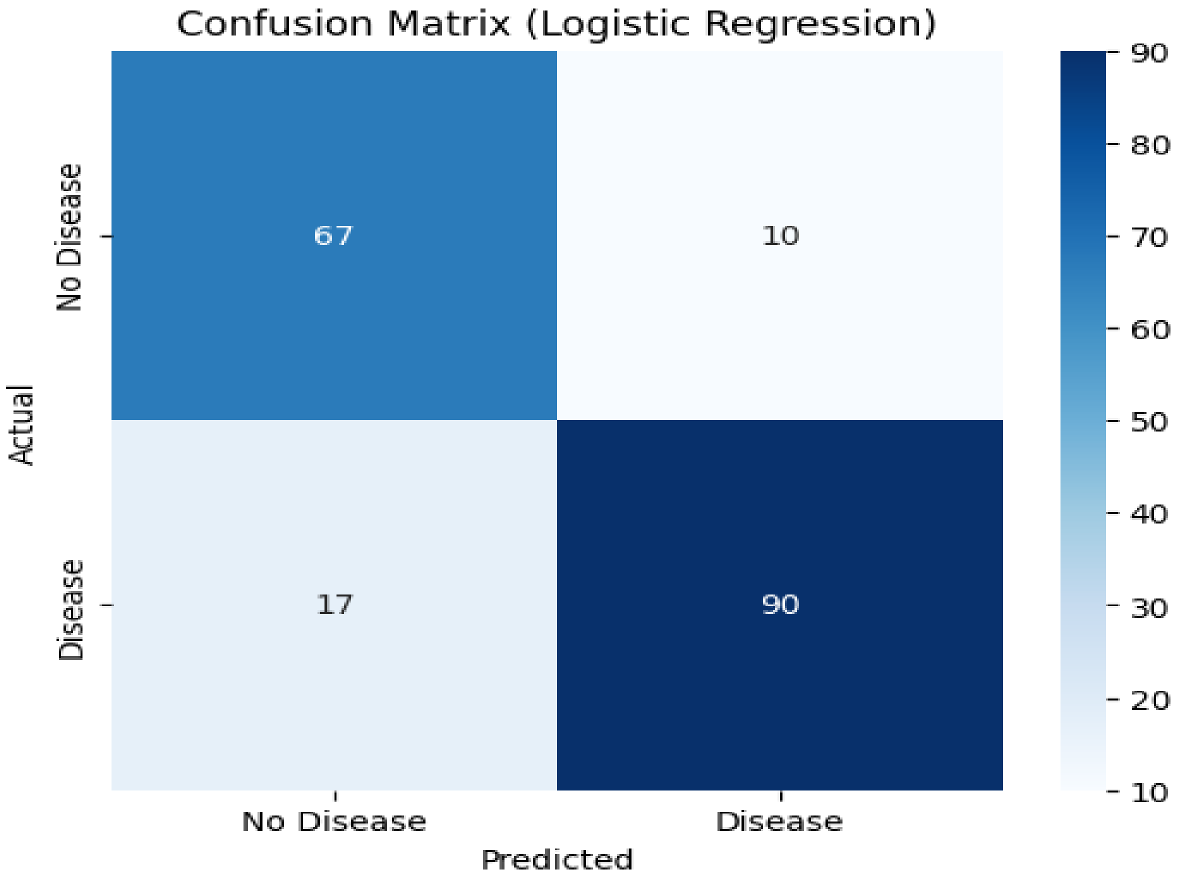
```
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_lr))

print("Classification Report:\n", classification_report(y_test, y_pred_lr))
```



```
Logistic Regression Accuracy: 0.8532608695652174
Classification Report:
               precision    recall  f1-score   support

           0       0.80      0.87      0.83        77
           1       0.90      0.84      0.87       107

    accuracy                           0.85       184
   macro avg       0.85      0.86      0.85       184
weighted avg       0.86      0.85      0.85       184
```

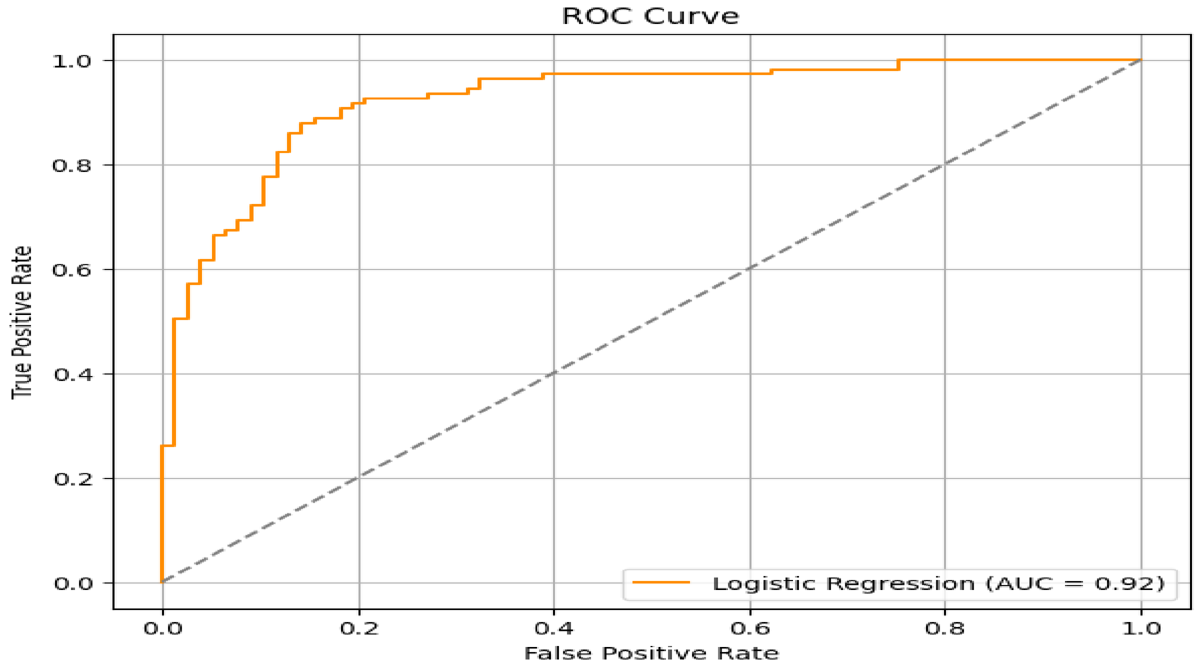
Confusion Matrix heatmap

```
plt.figure(figsize=(6,5))

sns.heatmap(confusion_matrix(y_test, y_pred_lr), annot=True, fmt='d', cmap='Blues',

            xticklabels=['No Disease', 'Disease'], yticklabels=['No Disease', 'Disease'])

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix (Logistic Regression)')
plt.show()
```



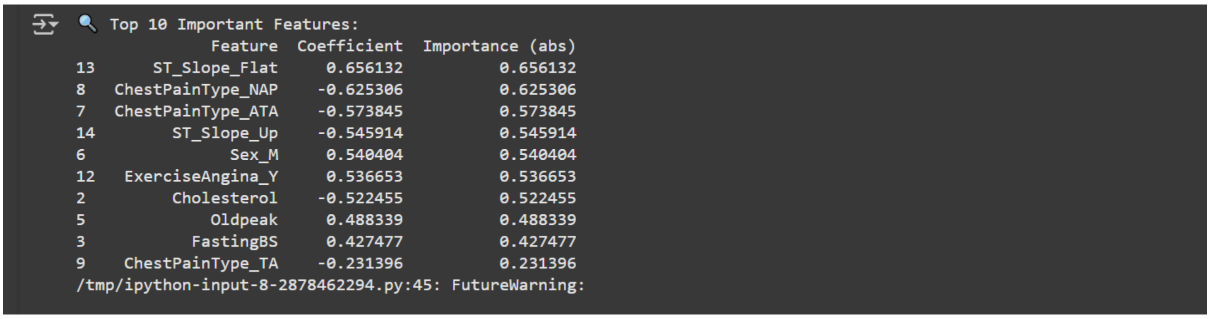
ROC Curve plot

```
y_proba_lr = lr_model.predict_proba(X_test)[:, 1]

fpr, tpr, _ = roc_curve(y_test, y_proba_lr)

auc_score = roc_auc_score(y_test, y_proba_lr)



plt.figure(figsize=(7,6))

plt.plot(fpr, tpr, label=f'Logistic Regression (AUC = {auc_score:.2f})', color='darkorange')

plt.plot([0, 1], [0, 1], linestyle='--', color='gray')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve')
plt.legend(loc='lower right')

plt.grid(True)

plt.show()
```



Get Feature Importants

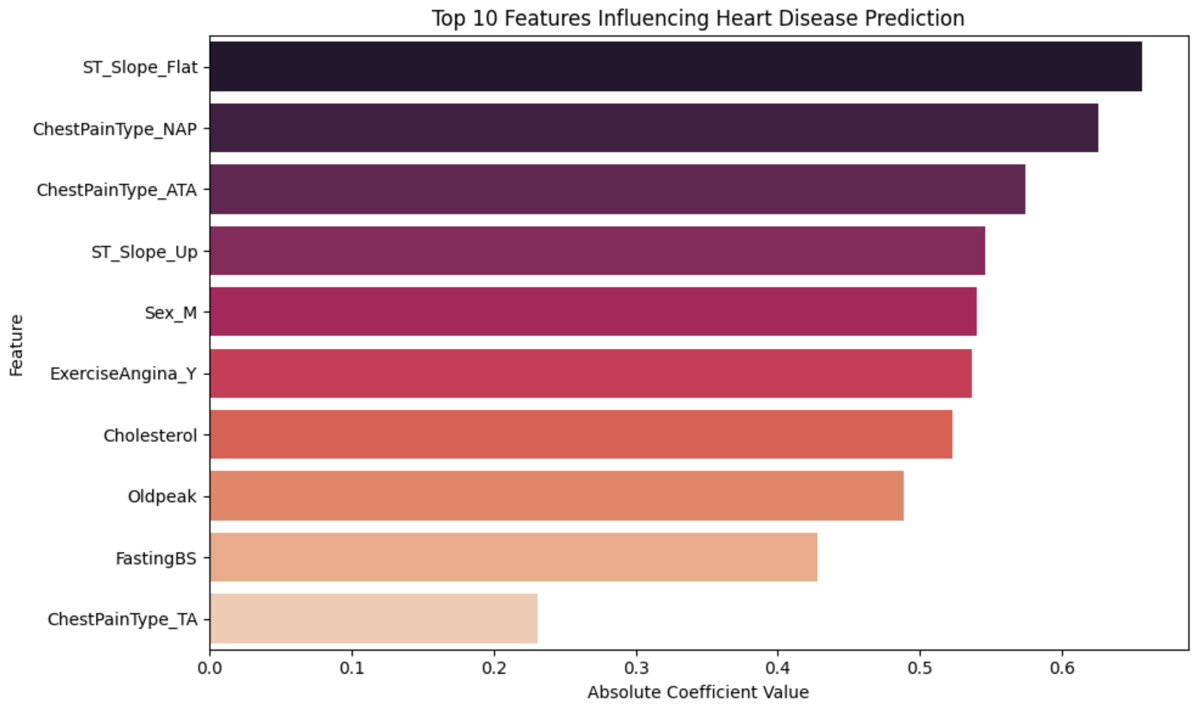
```
coefficients = model.coef_[0]



feature_importance = pd.DataFrame({

    'Feature': feature_names,

    'Coefficient': coefficients,

    'Importance (abs)': np.abs(coefficients)

}).sort_values(by='Importance (abs)', ascending=False)
```



```
🔍 Top 10 Important Features:
              Feature  Coefficient  Importance (abs)
13      ST_Slope_Flat     0.656132          0.656132
8   ChestPainType_NAP    -0.625306          0.625306
7   ChestPainType_ATA    -0.573845          0.573845
14        ST_Slope_Up    -0.545914          0.545914
6               Sex_M     0.540404          0.540404
12   ExerciseAngina_Y     0.536653          0.536653
2         Cholesterol    -0.522455          0.522455
5             Oldpeak     0.488339          0.488339
3           FastingBS     0.427477          0.427477
9    ChestPainType_TA    -0.231396          0.231396
/tmp/ipython-input-8-2878462294.py:45: FutureWarning:
```

Visualize top features

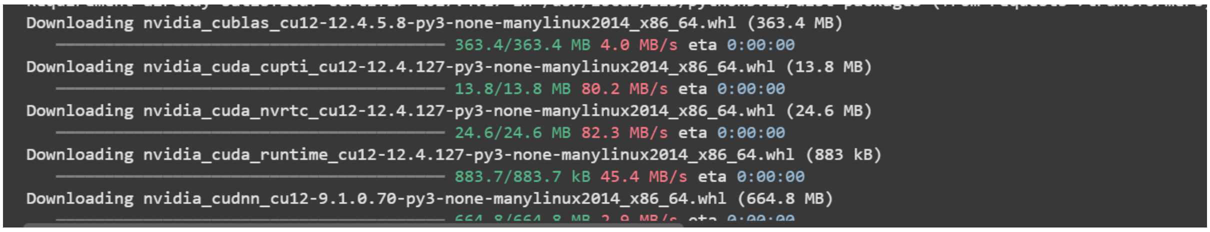
```
plt.figure(figsize=(10, 6))

sns.barplot(data=feature_importance.head(10), x='Importance (abs)', y='Feature', palette='rocket')

plt.title('Top 10 Features Influencing Heart Disease Prediction')

plt.xlabel('Absolute Coefficient Value')

plt.tight_layout()

plt.show()
```


Top 10 Features Influencing Heart Disease Prediction
Feature
ST_Slope_Flat
ChestPainType_NAP
ChestPainType_ATA
ST_Slope_Up
Sex_M
ExerciseAngina_Y
Cholesterol
Oldpeak
FastingBS
ChestPainType_TA
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Absolute Coefficient Value

TASK 5:General Health Query Chatbot (Prompt Engineering Based)

Install transformers torch

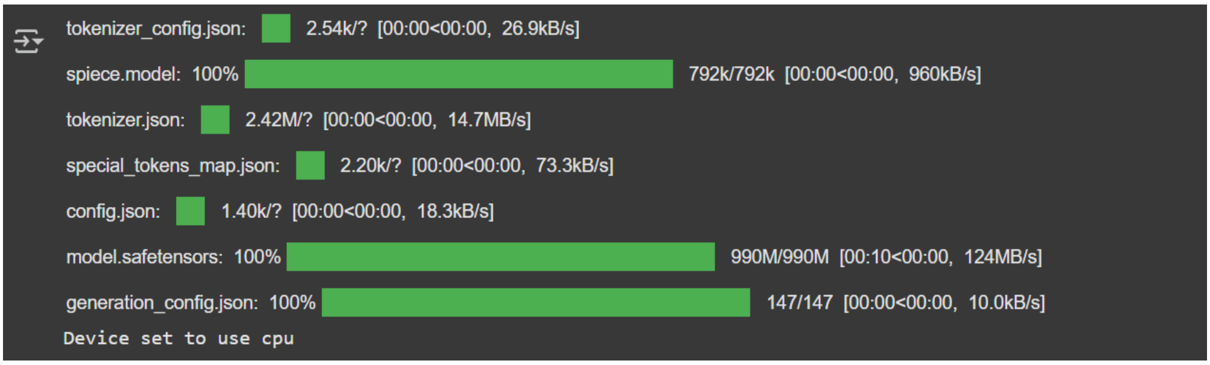
```
pip install transformers torch
```



```
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl (363.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 4.0 MB/s eta 0:00:00
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (13.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 80.2 MB/s eta 0:00:00
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (24.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 82.3 MB/s eta 0:00:00
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (883 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 45.4 MB/s eta 0:00:00
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl (664.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 2.9 MB/s eta 0:00:00
```

Load model and tokenizer

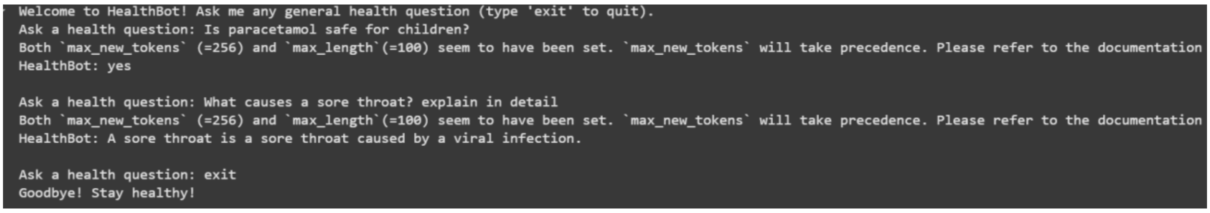
```
model_name = "google/flan-t5-base"

tokenizer = AutoTokenizer.from_pretrained(model_name)

model = AutoModelForSeq2SeqLM.from_pretrained(model_name)



pipe = pipeline("text2text-generation", model=model, tokenizer=tokenizer)



# Safety filter to block unsafe queries

def is_safe_query(query):

    unsafe_keywords = ["suicide", "overdose", "kill", "emergency", "poison", "self-harm"]

    return not any(word in query.lower() for word in unsafe_keywords)
```



```
tokenizer_config.json: 🟩 2.54k/? [00:00<00:00, 26.9kB/s]
spiece.model: 100% 🟩 792k/792k [00:00<00:00, 960kB/s]
tokenizer.json: 🟩 2.42M/? [00:00<00:00, 14.7MB/s]
special_tokens_map.json: 🟩 2.20k/? [00:00<00:00, 73.3kB/s]
config.json: 🟩 1.40k/? [00:00<00:00, 18.3kB/s]
model.safetensors: 100% 🟩 990M/990M [00:10<00:00, 124MB/s]
generation_config.json: 100% 🟩 147/147 [00:00<00:00, 10.0kB/s]
Device set to use cpu
```

Function to query the model

```
def ask_health_bot(query):
      prompt = f"You are a helpful medical assistant. Answer clearly and safely.\nQuestion: {query}"
    result = pipe(prompt, max_length=100)[0]['generated_text']
    return result

print("Welcome to HealthBot! Ask me any general health question (type 'exit' to quit).")

while True:
    user_input = input("Ask a health question: ")
    if user_input.lower() == "exit":
        print("Goodbye! Stay healthy!")
        break
    if not is_safe_query(user_input):
      print("HealthBot: Sorry, I can't assist with that. Please contact a healthcare professional.\n")
        continue
    reply = ask_health_bot(user_input)
    print(f"HealthBot: {reply}\n")
```



```
Welcome to HealthBot! Ask me any general health question (type 'exit' to quit).
Ask a health question: Is paracetamol safe for children?
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been set. `max_new_tokens` will take precedence. Please refer to the documentation
HealthBot: yes

Ask a health question: What causes a sore throat? explain in detail
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been set. `max_new_tokens` will take precedence. Please refer to the documentation
HealthBot: A sore throat is a sore throat caused by a viral infection.

Ask a health question: exit
Goodbye! Stay healthy!
```