```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from matplotlib import pyplot as plt

data_true =
pd.read_csv("/content/drive/MyDrive/Heart_Disease_Prediction.csv")
df = data_true

x = df[['BP','Cholesterol']]
y = df['Heart Disease']

k = 3
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(x,y)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
# New data point to predict on
new_data = np.array([[322,361]])
prediction = knn.predict(new_data)

# Check the prediction and print the result
if prediction[0] == 'Presence':  # Access the prediction string from
the array
    print("absence")
else:
    print("presence")
```

```
absence
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439:
UserWarning: X does not have valid feature names, but
KNeighborsClassifier was fitted with feature names
  warnings.warn(
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

data = {
    'bp': [130, 120, 140, 110, 150, 125, 135, 115, 145, 105],
    'cholesterol': [200, 180, 220, 160, 240, 210, 230, 170, 250, 150],
    'heartdisease': [1, 0, 1, 0, 1, 0, 1, 0, 1, 0]
}

df = pd.DataFrame(data)
```

```
df.to_csv('/content/drive/MyDrive/Heart_Disease_Prediction.csv',
index=False)

df =
pd.read_csv('/content/drive/MyDrive/Heart_Disease_Prediction.csv')

X = df[['bp', 'cholesterol']]
y = df['heartdisease']

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

# Initialize and train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression()
```

```
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

```
Mean Squared Error: 0.061753902662993326
R-squared: 0.7529843893480267
```

```
# Predict the likelihood of heart disease for a new individual
new_individual = pd.DataFrame({'bp': [128], 'cholesterol': [190]})
predicted_heartdisease = model.predict(new_individual)


print("Predicted likelihood of heart disease for the new individual:",
predicted_heartdisease[0])
```

```
Predicted likelihood of heart disease for the new individual:
0.6363636363636358
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df=pd.read_csv("/content/drive/MyDrive/Heart_Disease_Prediction.csv")

df.columns
```

```
Index(['bp', 'cholesterol', 'heartdisease'], dtype='object')
```

```
df.head(5)
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 10,\n \"fields\": [\n{\n   \"column\": \"bp\",\n   \"properties\": {\n\"dtype\": \"number\",\n   \"std\": 15,\n   \"min\": 105,\n\"max\": 150,\n   \"num_unique_values\": 10,\n\"samples\": [\n   145,\n   120,\n   125\n],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n}\n   },\n   {\n   \"column\": \"cholesterol\",\n\"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 34,\n   \"min\": 150,\n   \"max\": 250,\n\"num_unique_values\": 10,\n   \"samples\": [\n   250,\n180,\n   210\n   ],\n   \"semantic_type\": \"\",\n\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"heartdisease\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n\"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   0,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n\"column\": \"target\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n\"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   1,\n   0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"df"}

```
a=[[130,120,0,]]
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   bp            10 non-null     int64
 1   cholesterol   10 non-null     int64
 2   heartdisease  10 non-null     int64
 3   target        10 non-null     int64
dtypes: int64(4)
memory usage: 448.0 bytes
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.linear_model import LogisticRegression
```

```
x=df.drop("target",axis=1)
```

```
xtrain.shape
```

```
(8, 3)
```

```
xtest.shape
```

```
(2, 3)
```

```
LOR=LogisticRegression()
```

```
LOR.fit(xtrain,ytrain)
```

```
LogisticRegression()
```

```
pred=LOR.predict(xtest)
```

```
from sklearn.metrics import accuracy_score
```

```
ac=accuracy_score(ytest,pred)
```

```
ac*100
```

```
50.0
```

```
a=np.array([130,120,0,]).reshape(1,-1)
```

```
ans=LOR.predict(a)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439:
UserWarning: X does not have valid feature names, but
LogisticRegression was fitted with feature names
  warnings.warn(
```

```
if(int(ans)==1):
    print("person has heart disease")
else:
    print("person does not have heart disease")
```

```
person has heart disease
```

```
<ipython-input-68-fb7ff118f1f8>:1: DeprecationWarning: Conversion of
an array with ndim > 0 to a scalar is deprecated, and will error in
future. Ensure you extract a single element from your array before
performing this operation. (Deprecated NumPy 1.25.)
  if(int(ans)==1):
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
import matplotlib.pyplot as plt
```

```
df =
pd.read_csv('/content/drive/MyDrive/Heart_Disease_Prediction.csv')
```

```
print(df.head())
```

```
    bp  cholesterol  heartdisease
0  130          200             1
1  120          180             0
2  140          220             1
3  110          160             0
4  150          240             1
```

```
X = df.drop(columns=['heartdisease'])  # Drop the target column
y = df['heartdisease']

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

dt_classifier = DecisionTreeClassifier(random_state=42)

dt_classifier.fit(X_train, y_train)
```

```
DecisionTreeClassifier(random_state=42)
```

```
# Predict on the test set
y_pred = dt_classifier.predict(X_test)

# Evaluate performance
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
```

```
Accuracy: 1.00
```

```
# Display classification report
print('Classification Report:')
print(classification_report(y_test, y_pred))
```

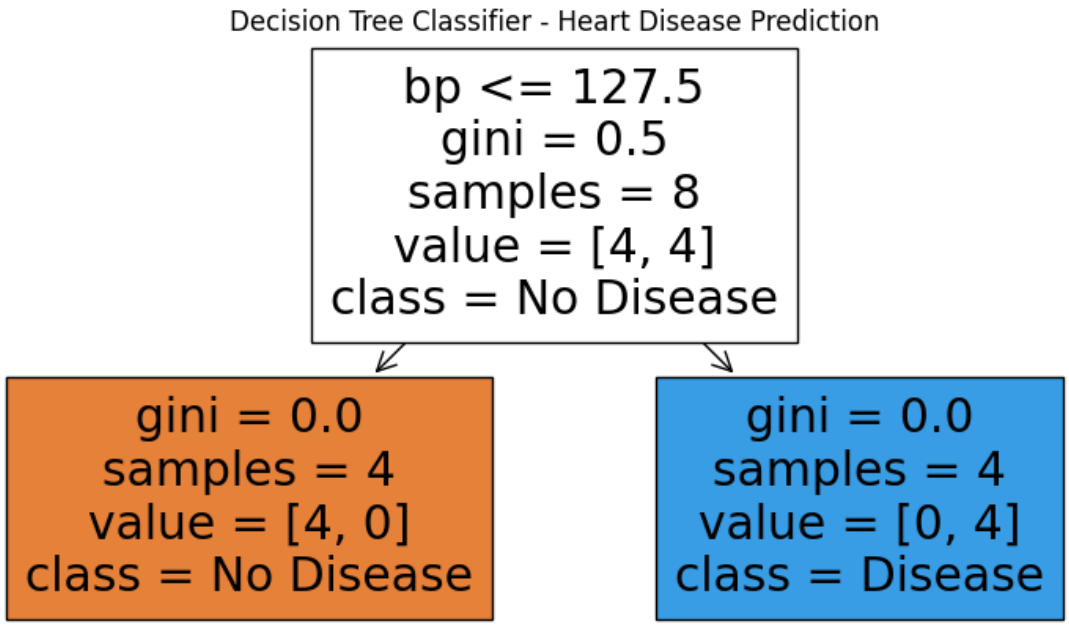
```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         1

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2
```

```
print('Confusion Matrix:')
print(confusion_matrix(y_test, y_pred))
```

```
Confusion Matrix:
[[1 0]
 [0 1]]
```

```
# Visualize the decision tree
plt.figure(figsize=(10, 5))
```

```
plot_tree(dt_classifier, feature_names=X.columns, class_names=['No
Disease', 'Disease'], filled=True)
plt.title("Decision Tree Classifier - Heart Disease Prediction")
plt.show()
```




```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
import matplotlib.pyplot as plt

# Load heart disease dataset (example dataset)
df =
pd.read_csv('/content/drive/MyDrive/Heart_Disease_Prediction.csv')

print(df.head())
```

```
    bp  cholesterol  heartdisease
0  130          200             1
1  120          180             0
2  140          220             1
3  110          160             0
4  150          240             1
```

```
X = df[['bp','cholesterol']]
y = df['heartdisease']

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

```
rf_classifier = RandomForestClassifier(n_estimators=100,
random_state=42)

rf_classifier.fit(X_train, y_train)
```

```
RandomForestClassifier(random_state=42)
```

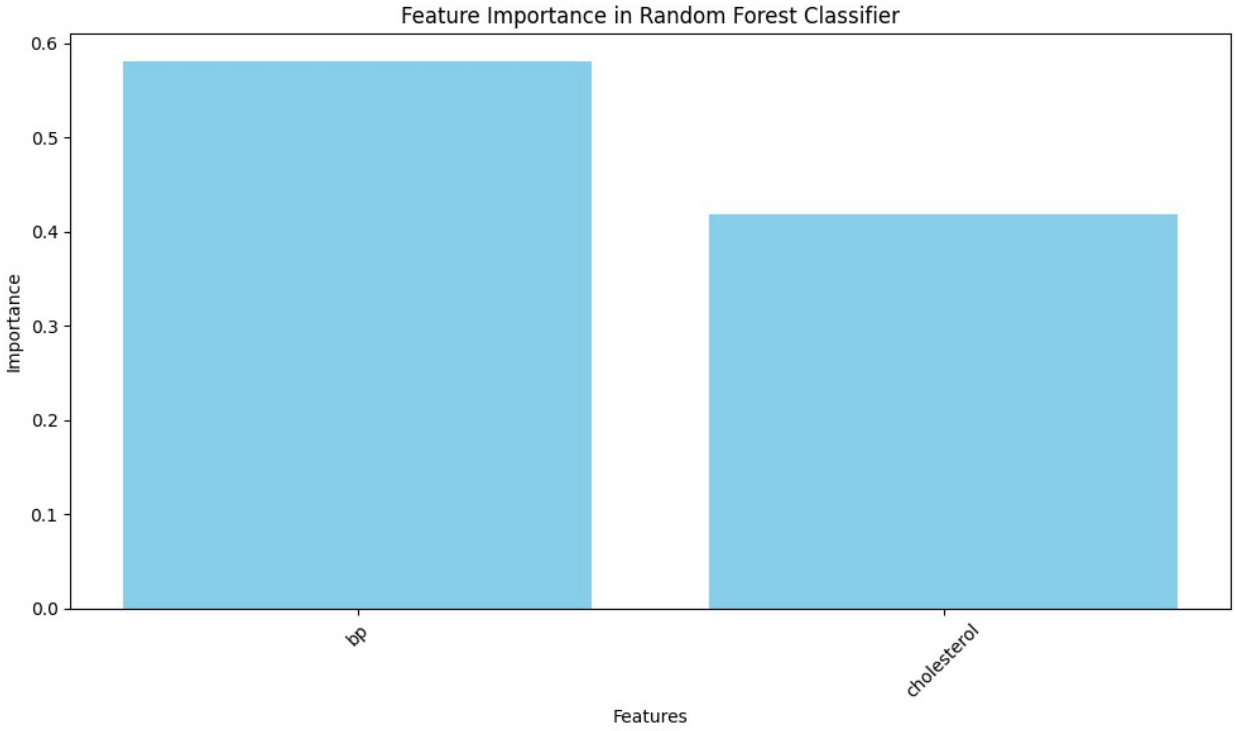
```
y_pred = rf_classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 1.0
```

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         1

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2
```

```
feature_importances = rf_classifier.feature_importances_
features = X.columns

# Visualize feature importance
plt.figure(figsize=(10, 6))
plt.bar(features, feature_importances, color='skyblue')
plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importance in Random Forest Classifier')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```


Feature Importance in Random Forest Classifier
Importance
0.0
0.1
0.2
0.3
0.4
0.5
0.6
bp
cholesterol
Features