REPUBLIC OF YEMEN

SANA'A UNIVERSITY

FACULTY OF ENGINEERING

MECHATRONICS ENGINEERING DEPARTMENT

Heart Disease Prediction

(Orange & Python)

Done by:

Mohannad Abobakr Gabr Guhish (202170242)

Hadeel Adel Ali Al-Ariqi (202170321)

Group (2)

System (General)

Supervised by:

Dr. Ahmad Al-Aarashi

Eng. Yousef Al-Qiz

ABSTRACT

This project presents a comprehensive software solution for predicting the presence of heart disease in patients using machine learning techniques implemented in Python and the Orange data mining software. The dataset utilized consists of several clinical features, including age, sex, chest pain type, resting blood pressure, cholesterol levels, fasting blood sugar, resting electrocardiogram results, maximum heart rate, exercise-induced angina, ST depression, slope of the ST segment, number of major vessels, and thalassemia status.

Data preprocessing steps involved loading the dataset, examining its structure, and handling any missing values. The dataset was then split into features and target labels, followed by the application of various machine learning algorithms, including Random Forest, Logistic Regression, and Neural Networks, to classify the presence or absence of heart disease.

Performance metrics such as accuracy, confusion matrices, and classification reports were generated to evaluate the effectiveness of each model. Additionally, scaling techniques, including Standard Scaler and Min-Max Scaler, were employed to improve model accuracy. The Orange program was utilized to visualize data and model performance, enhancing interpretability and insights. The results demonstrate the capability of the implemented models to effectively predict heart disease, showcasing the potential of machine learning in medical diagnostics.

content

Introduction:

The Cardiovascular diseases continue to be a leading cause of morbidity and mortality worldwide, highlighting the urgent need for effective diagnostic tools. Early detection of heart disease can significantly improve patient outcomes, allowing for timely interventions and better management of health conditions. Traditional diagnostic methods often rely on subjective clinical assessments and invasive procedures, which can be time-consuming and costly.

In recent years, the integration of machine learning techniques into healthcare has shown great promise in enhancing diagnostic accuracy and efficiency. By analyzing large datasets, machine learning algorithms can uncover patterns and relationships that may not be evident through conventional methods. This project aims to leverage these advancements by developing a predictive model for heart disease using a dataset that includes various clinical features.

We utilized Python programming and the Orange data mining software to implement and evaluate several machine learning algorithms, including Random Forest, Logistic Regression, and Neural Networks. The project involves data preprocessing, feature selection, and model training, followed by performance evaluation using metrics such as accuracy and classification reports. Additionally, data visualization tools provided by Orange were employed to enhance the interpretability of the results.

The findings of this project not only aim to improve the diagnostic process for heart disease but also serve as a foundation for future research and development in the application of machine learning in healthcare.

Figure 1 Heart Disease Prediction

- **Introduction about our data:**
  - We took our dataset from website called Kaggle.com: Kaggle is the world's largest data science community with powerful tools and resources to help you achieve your data science goals.

Figure 2 Kaggle

  - Our features and target:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	diagnosis	age	sex	chest pain	resting blood pressure	cholesterol levels	fasting blood sugar	resting electrocardiogram	maximum heart rate	exercise induced angina	depression	slope	major vessels	thalassemia
2	1	63	1	3	145	233	1	0	150	0	2.3	0	0	1
3	1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
4	1	41	0	1	130	204	0	0	172	0	1.4	2	0	2
5	1	56	1	1	120	236	0	1	178	0	0.8	2	0	2
6	1	57	0	0	120	354	0	1	163	1	0.6	2	0	2
7	1	57	1	0	140	192	0	1	148	0	0.4	1	0	1
8	1	56	0	1	140	294	0	0	153	0	1.3	1	0	2
9	1	44	1	1	120	263	0	1	173	0	0	2	0	3
10	1	52	1	2	172	199	1	1	162	0	0.5	2	0	3

1. **In heart disease data, the variable sex typically represents the biological sex of the participants, usually coded as:**

   Male: Often represented as 1
   Female: Often represented as 0

2. **In heart disease data, if the variable cp (chest pain type) is coded from 0 to 3, it likely represents the following categories:**

   0: Typical Angina (chest pain typical of heart-related issues)
   1: Atypical Angina (chest pain that doesn't fit typical patterns but may still suggest heart problems)
   2: Non-Anginal Pain (chest pain not related to heart issues, possibly due to other conditions)
   3: Asymptomatic (no chest pain or symptoms present)

3. **Total Cholesterol: Less than 200 mg/dL is desirable; 200-239 mg/dL is borderline high; 240 mg/dL and above is high.**

   Range of Values:
   Normal resting blood pressure is usually around 120/80 mmHg.
   Values can vary widely, with higher values indicating potential hypertension:
   Normal: Less than 120/80 mmHg
   Elevated: 120-129 systolic and less than 80 diastolic
   Hypertension Stage 1: 130-139 systolic or 80-89 diastolic
   Hypertension Stage 2: 140 or higher systolic or 90 or higher diastolic

4. **In heart disease data, if the variable fbs (fasting blood sugar) is coded as 0 and 1, it typically represents:**

   0: Fasting blood sugar less than 120 mg/dL (normal)
   1: Fasting blood sugar greater than or equal to 120 mg/dL (high)

5. **In heart disease data, the variable restecg typically represents resting electrocardiogram (ECG) results. This variable indicates the findings from an ECG performed while the patient is at rest and is usually categorized into different types. The common categories are:**

   0: Normal ECG (no significant abnormalities)
   1: Having ST-T wave abnormality (possibly indicating ischemia)
   2: Showing left ventricular hypertrophy (enlargement of the heart's left ventricle)

6. **In heart disease data, the variable exang represents exercise induced angina. It is typically coded as a binary variable:**

   0: No exercise induced angina (the patient does not experience chest pain during exercise)
   1: Exercise induced angina (the patient experiences chest pain during exercise)

7. **In heart disease data, the variable slope typically represents the slope of the ST segment during exercise testing, measured during a stress test. It is usually categorized into three types:**

   0: Upsloping - The ST segment slopes upward, which can be considered a normal response during exercise.
   1: Flat - The ST segment remains flat, which may indicate a more concerning response.
   2: Downsloping - The ST segment slopes downward, often associated with myocardial ischemia and a higher risk of heart disease.

8. **If the variable ca (number of major vessels) in your heart disease data is coded from 0 to 4, it might represent:**

   0: No major vessels affected
   1: One major vessel affected
   2: Two major vessels affected
   3: Three major vessels affected
   4: Four major vessels affected (this could indicate severe coronary artery disease or a specific classification in your dataset)

9. **If the variable thal in your heart disease data is coded from 0 to 3, it typically represents the following categories:**

   0: Normal (no thalassemia or related issues)
   1: Fixed defect (indicating a region of the heart that does not receive adequate blood flow at rest or during stress)
   2: Reversible defect (indicating a region that is not receiving adequate blood flow during stress but is normal at rest)
   3: Thalassemia or another specific abnormality related to blood flow or heart function

Objectives:

1. To collect and preprocess a dataset containing clinical features related to heart disease.
2. To perform exploratory data analysis to identify patterns and correlations within the dataset.
3. To implement and train various machine learning algorithms, including Random Forest and Logistic Regression, for predicting heart disease as a classification problem.
4. To evaluate model performance using metrics such as accuracy, precision, and confusion matrices.
5. To utilize Orange for visualizing data distributions and model results to enhance interpretability.
6. To discuss the potential of machine learning in improving heart disease diagnosis and management.

Methodology:

- **Software:**

  a) Python:

Figure 3 Python

Python is a widely used general-purpose, high-level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code. Python is a programming language that lets you work quickly and integrate systems more efficiently. There are two major Python versions: **Python 2 and Python 3**. Both are quite different.

b) Orange3:

Figure 4 Orange3

Orange3 is an open-source software package designed for data visualization, machine learning, data mining, and data analysis. It features a visual programming front-end, allowing users to create workflows by linking various components called widgets. These widgets can perform tasks such as data visualization, subset selection, preprocessing, and predictive modeling.
Orange3 is written in Python and uses common scientific computing libraries like numpy, scipy, and scikit-learn. It supports multiple operating systems, including macOS, Windows, and Linux[1]. Users can interactively explore data visualizations and feed selected subsets into other widgets for further analysis.

Equipments:

- **Orange3:**

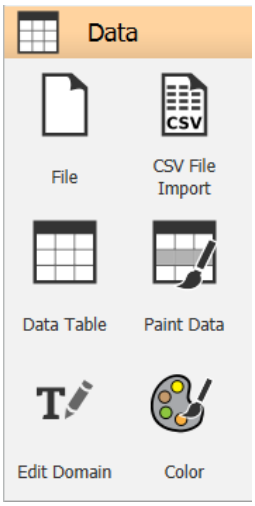


Figure 5 Data Equipments

Figure 6 Transform Equipments

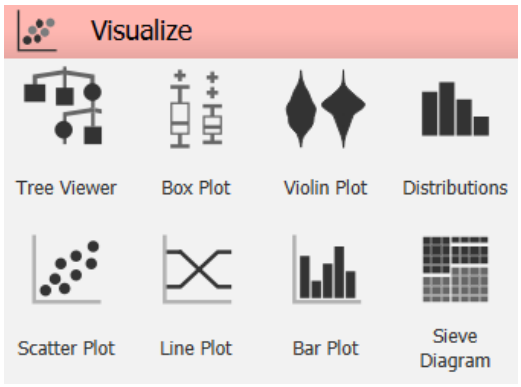


Figure 7 Visualize Equipments

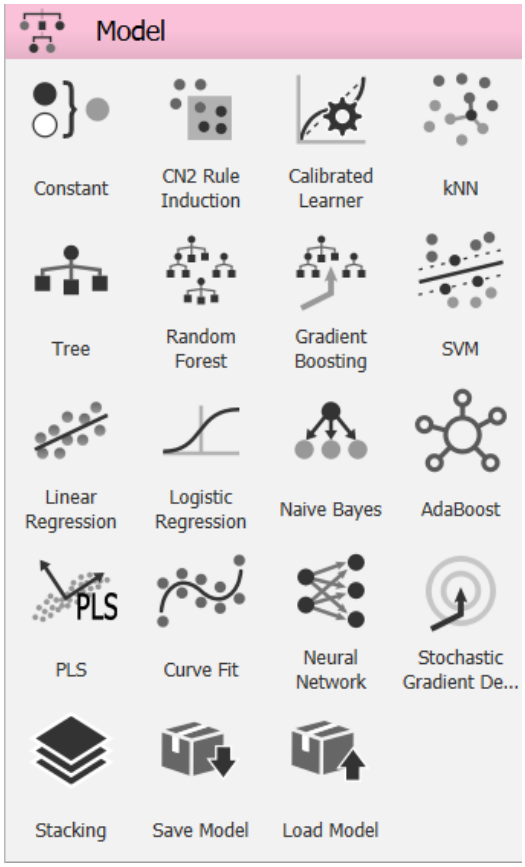


Figure 8 Model Equipments

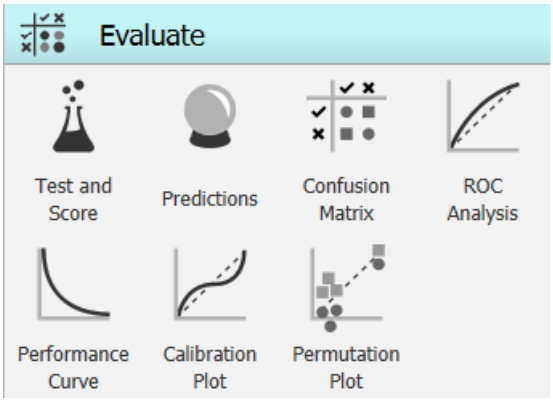


Figure 9 Evaluate Equipments

Procedures:

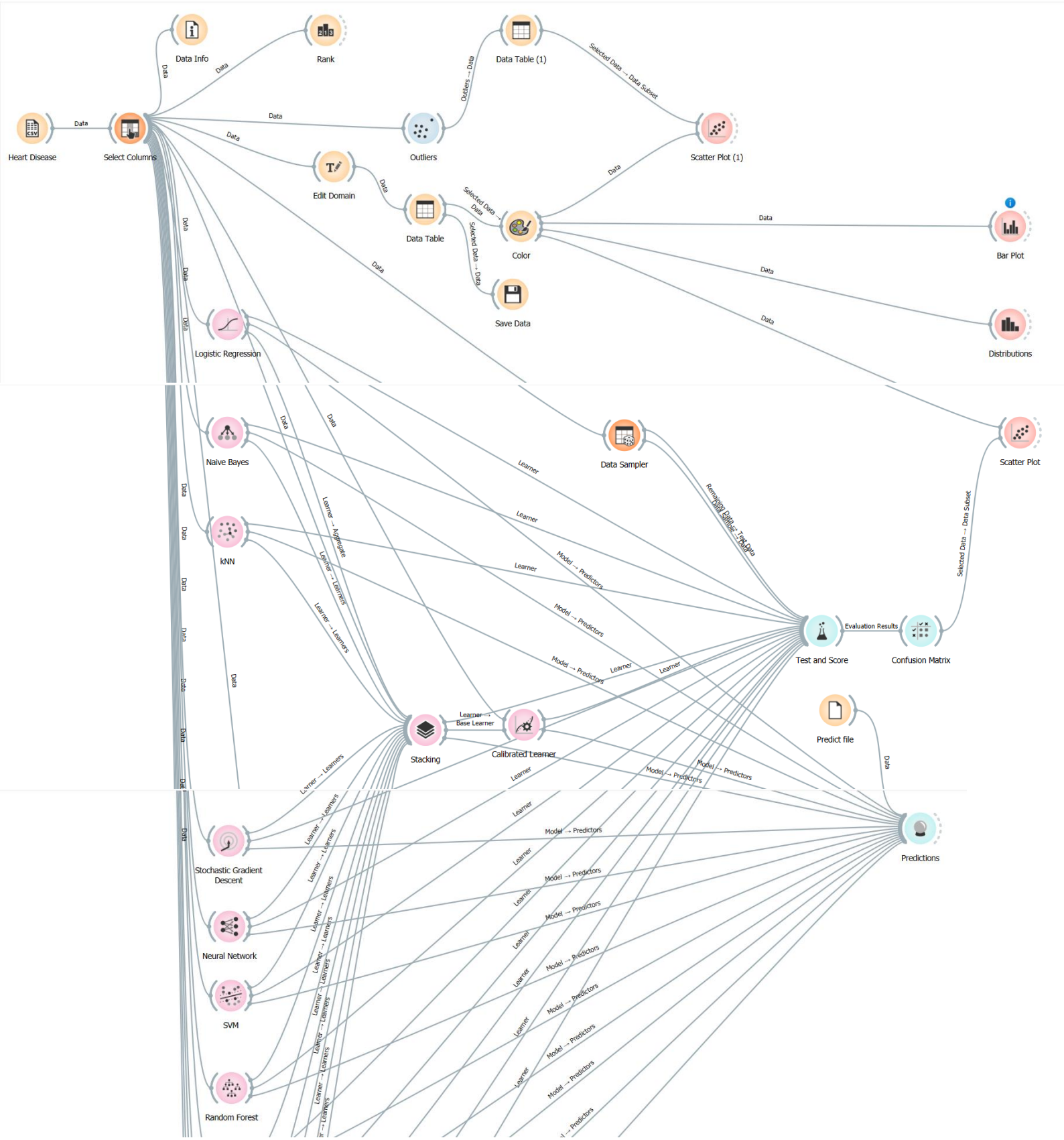
- **Python:**

a. We import the dataset and display the first and last 20 rows to understand its structure.
b. We determine the number of rows and columns in the dataset and analyze its information to check for null values.
c. We split the dataset into features and target labels, focusing on clinical attributes such as age, sex, and blood pressure.
d. We prepare the target variable, which is the diagnosis of heart disease.
e. We utilize seaborn to generate plots illustrating the distribution of diagnoses and relationships between features.
f. We create a heatmap to show correlations among the variables for better feature selection.
g. We apply machine learning algorithms, starting with the Random Forest classifier.
h. We train the model and evaluate its performance using accuracy scores and confusion matrices.
i. We implement different scaling techniques, including Standard Scaler and Min-Max Scaler, to enhance model accuracy.
j. We utilize Logistic Regression and Neural Networks to improve prediction capabilities.
k. We ensure robust evaluation through classification reports and confusion matrices.
l. We allow for input data to be provided for real-time predictions, determining the likelihood of heart disease based on clinical metrics.

- **Orange3:**
  a. We connected CSV File Import to Select Columns.
  b. We connected Select Columns to Data Info, Rank, Outliers, Edit Domin, Data Sampler, Logistic Regression, Naive Bayes, kNN, Stacking, Calibrated Learner, Stochastic Gradient Descent, AdaBoost, Random Forest, Constant, CN2 Rule Induction, Tree.
  c. We connected Outliers to Data Table (1).
  d. We connected Data Table (1) to Scatter Plot (1).
  e. We connected Edit Domain to Data Table.
  f. We connected Data Table to Color and Save Data.
  g. We connected Color to Scatter Plot (1), Bar Plot, Distributions, Scatter Plot.
  h. We connected Data Sampler to Test and Score.
  i. We connected Logistic Regression, Naive Bayes, kNN, Stacking, Calibrated Learner, Stochastic Gradient Descent, AdaBoost, Random Forest, Constant, CN2 Rule Induction, Tree to Test and Score.
  j. We connected Logistic Regression, Naive Bayes, kNN, Stacking, Calibrated Learner, Stochastic Gradient Descent, AdaBoost, Random Forest, Constant, CN2 Rule Induction, Tree to Predictions.
  k. We connected Logistic Regression, Naive Bayes, kNN, Stochastic Gradient Descent, AdaBoost, Random Forest, Constant, CN2 Rule Induction, Tree to Stacking.
  l. We connected Stacking to Calibrated Learner.
  m. We connected CN2 Rule Induction to CN2 Rule Viewer.
  n. We connected Tree to Tree Viewer.
  o. We connected Test and Score to Confusion Matrix.
  p. We connected Confusion Matrix to Scatter Plot.
  q. We connected File to Predictions.

Results:

- **Project in Orange3:**



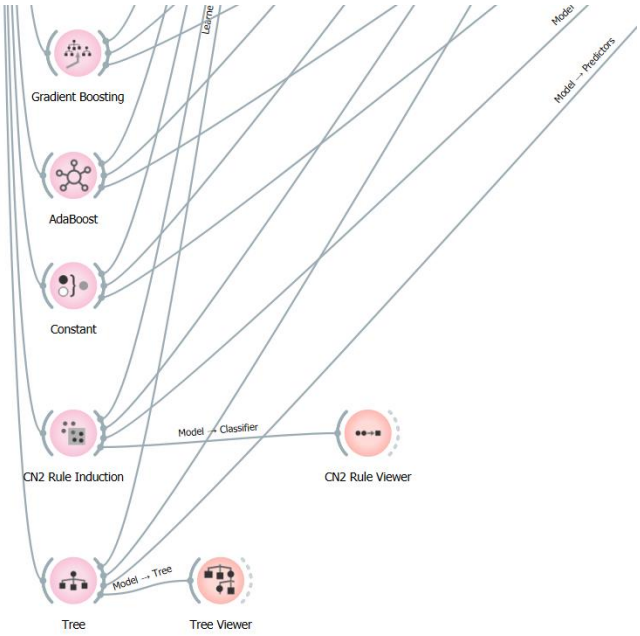


Figure 10 Project in Orange3

- **Project in Python:**

```
# Import the the data file with the first 20 rows:
import numpy as np
import pandas as pd
df = pd.read_csv('heart.csv')
df.head(20)
```

[260] Python

	age	sex	chest pain	resting blood pressure	cholesterol levels	fasting blood sugar	resting electrocardiogram	maximum heart rate	exercise induced angina	depression	slope	major vessels	thalassemia	diagnosis
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1

```
# Import the the data file with the last 20 rows:
import numpy as np
import pandas as pd
df = pd.read_csv('heart.csv')
df.tail(20)
```

[261] Python

	age	sex	chest pain	resting blood pressure	cholesterol levels	fasting blood sugar	resting electrocardiogram	maximum heart rate	exercise induced angina	depression	slope	major vessels	thalassemia	diagnosis
283	40	1	0	152	223	0	1	181	0	0.0	2	0	3	0
284	61	1	0	140	207	0	0	138	1	1.9	2	1	3	0
285	46	1	0	140	311	0	1	120	1	1.8	1	2	3	0
286	59	1	3	134	204	0	1	162	0	0.8	2	2	2	0
287	57	1	1	154	232	0	0	164	0	0.0	2	1	2	0

```python
# Number of Rows and Columns:
import numpy as np
import pandas as pd
df = pd.read_csv('heart.csv')
df.shape
```

[264] Python

```
(303, 14)
```

```python
# Data file information and nulls:
import numpy as np
import pandas as pd
df = pd.read_csv('heart.csv')

df.info()

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

[263] Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   age                       303 non-null    int64
 1   sex                       303 non-null    int64
 2   chest pain                303 non-null    int64
 3   resting blood pressure    303 non-null    int64
 4   cholesterol levels        303 non-null    int64
 5   fasting blood sugar       303 non-null    int64
 6   resting electrocardiogram 303 non-null    int64
 7   maximum heart rate        303 non-null    int64
 8   exercise induced angina   303 non-null    int64
 9   depression                303 non-null    float64
 10  slope                     303 non-null    int64
 11  major vessels             303 non-null    int64
 12  thalassemia               303 non-null    int64
 13  diagnosis                 303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

```
age                       0
sex                       0
chest pain                0
resting blood pressure    0
cholesterol levels        0
fasting blood sugar       0
resting electrocardiogram 0
maximum heart rate        0
exercise induced angina   0
depression                0
slope                     0
major vessels             0
thalassemia               0
diagnosis                 0
dtype: int64
```

```python
#Splitting the Features and Target
import numpy as np
import pandas as pd
df = pd.read_csv('heart.csv')
columns = ['age', 'sex', 'chest pain', 'resting blood pressure', 'cholesterol levels', 'fasting blood sugar', 'resting electrocardiogram', 'maximum
labels = df['diagnosis'].values
features = df[list(columns)].values
labels
```

[364] Python

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

```python
#Splitting the Features and Target
import numpy as np
import pandas as pd
df = pd.read_csv('heart.csv')
columns = ['age', 'sex', 'chest pain', 'resting blood pressure', 'cholesterol levels', 'fasting blood sugar', 'resting electrocardiogram', 'maximum
labels = df['diagnosis'].values
features = df[list(columns)].values
features
```

[365] Python

```
array([[63.,  1.,  3., ...,  0.,  0.,  1.],
       [37.,  1.,  2., ...,  0.,  0.,  2.],
       [41.,  0.,  1., ...,  2.,  0.,  2.],
       ...,
       [68.,  1.,  0., ...,  1.,  2.,  3.],
       [57.,  1.,  0., ...,  1.,  1.,  3.],
       [57.,  0.,  1., ...,  1.,  1.,  2.]])
```

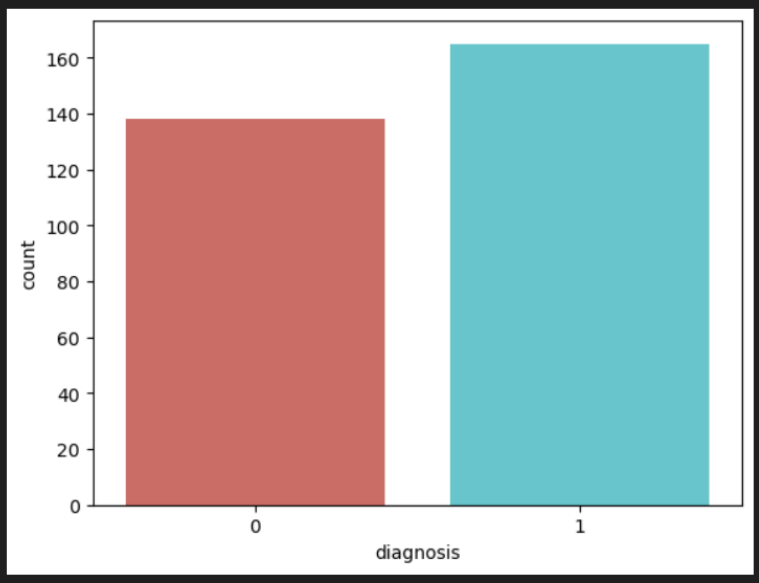
```python
# Ploting
import numpy as np
import pandas as pd
import seaborn as sb

df = pd.read_csv('heart.csv')
columns = ['age', 'sex', 'chest pain', 'resting blood pressure', 'cholesterol levels', 'fasting blood sugar', 'resting electrocardiogram', 'maximum
labels = df['diagnosis'].values
features = df[list(columns)].values

sb.countplot(x='diagnosis',data=df, palette='hls')
#sb.countplot(x='age',data=df, palette='hls')
#sb.heatmap(df.corr())
#sb.pairplot(df, hue="diagnosis")
```

[1] ✓ 4.3s Python

```
<Axes: xlabel='diagnosis', ylabel='count'>
```




```python
# Random forest algorithm:
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

df = pd.read_csv('heart.csv')
columns = ['age', 'sex', 'chest pain', 'resting blood pressure', 'cholesterol levels', 'fasting blood sugar', 'resting electrocardiogram', 'maximum
labels = df['diagnosis'].values
features = df[list(columns)].values

X_train, X_test, y_train, y_test = train_test_split(features, labels, train_size=0.80)

clf = RandomForestClassifier(n_estimators=1)
clf = clf.fit(X_train,y_train)

accurecy_train = clf.score(X_train, y_train)
print (accurecy_train*100)

accurecy_test = clf.score(X_test, y_test)
print (accurecy_test*100)

ypredict = clf.predict(X_train)
print ('\nTraining classification report\n', classification_report(y_train, ypredict))
print ("\n Confusion matrix of training \n", confusion_matrix(y_train, ypredict))

ypredict = clf.predict(X_test)
print ('\nTest classification report\n', classification_report(y_test, ypredict))
print ("\n Confusion matrix of training \n", confusion_matrix(y_test, ypredict))
```

```python
input_data = (23,0,3,110,180,0,0,100,0,1.2,0,0,0)

input_data_as_numpy_array= np.asarray(input_data)

input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = clf.predict(input_data_reshaped)
print(prediction)

if (prediction[0]== 0):
  print('\n\nThe Person does not have a Heart Disease')
else:
  print('\n\nThe Person has Heart Disease')
```

```
90.49586776859503
72.1311475409836

Training classification report
              precision    recall  f1-score   support

           0       0.92      0.88      0.89       112
           1       0.90      0.93      0.91       130

    accuracy                           0.90       242
   macro avg       0.91      0.90      0.90       242
weighted avg       0.91      0.90      0.90       242


 Confusion matrix of training
 [[ 98  14]
 [  9 121]]

Test classification report
              precision    recall  f1-score   support

           0       0.68      0.65      0.67        26
           1       0.75      0.77      0.76        35

    accuracy                           0.72        61
...
[1]


The Person has Heart Disease
```

```python
# Improving Random forest algorithm using standard scaler:
import numpy as np
import pandas as pd
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

df = pd.read_csv('heart.csv')
columns = ['age', 'sex', 'chest pain', 'resting blood pressure', 'cholesterol levels', 'fasting blood sugar', 'resting electrocardiogram', 'maximum
labels = df['diagnosis'].values
features = df[list(columns)].values

X_train, X_test, y_train, y_test = train_test_split(features, labels, train_size=0.80)

scaler = StandardScaler()

scaler.fit(X_train)
X_train = scaler.transform(X_train)

X_test = scaler.transform(X_test)

clf = RandomForestClassifier(n_estimators=1)
clf = clf.fit(X_train,y_train)

accurecy_train = clf.score(X_train, y_train)
print (accurecy_train*100)

accurecy_test = clf.score(X_test, y_test)
print (accurecy_test*100)
```

```
ypredict = clf.predict(X_train)
print ('\nTraining classification report\n', classification_report(y_train, ypredict))
print ("\n Confusion matrix of training \n", confusion_matrix(y_train, ypredict))

ypredict = clf.predict(X_test)
print ('\nTraining classification report\n', classification_report(y_test, ypredict))
print ("\n Confusion matrix of training \n", confusion_matrix(y_test, ypredict))

input_data = (23,1,3,110,180,0,0,100,0,1.2,0,0,0)

input_data_as_numpy_array= np.asarray(input_data)

input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = clf.predict(input_data_reshaped)
print(prediction)

if (prediction[0]== 0):
  print('\n\nThe Person does not have a Heart Disease')
else:
  print('\n\nThe Person has Heart Disease')
```

```
93.80165289256198
80.32786885245902

Training classification report
              precision    recall  f1-score   support

           0       0.95      0.92      0.93       113
           1       0.93      0.95      0.94       129

    accuracy                           0.94       242
   macro avg       0.94      0.94      0.94       242
weighted avg       0.94      0.94      0.94       242


 Confusion matrix of training
 [[104   9]
 [  6 123]]

Training classification report
              precision    recall  f1-score   support

           0       0.81      0.68      0.74        25
           1       0.80      0.89      0.84        36

    accuracy                           0.80        61
...
[0]


The Person does not have a Heart Disease
```

```
# Improving Random forest algorithm using min max scaler:
import numpy as np
import pandas as pd
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn import preprocessing

df = pd.read_csv('heart.csv')
columns = ['age', 'sex', 'chest pain', 'resting blood pressure', 'cholesterol levels', 'fasting blood sugar', 'resting electrocardiogram', 'maximum
labels = df['diagnosis'].values
features = df[list(columns)].values

X_train, X_test, y_train, y_test = train_test_split(features, labels, train_size=0.80)

scaler = preprocessing.MinMaxScaler()
```

```
scaler.fit(X_train)
X_train = scaler.transform(X_train)

X_test = scaler.transform(X_test)

clf = RandomForestClassifier(n_estimators=1)
clf = clf.fit(X_train,y_train)

accurecy_train = clf.score(X_train, y_train)
print (accurecy_train*100)

accurecy_test = clf.score(X_test, y_test)
print (accurecy_test*100)

ypredict = clf.predict(X_train)
print ('\nTraining classification report\n', classification_report(y_train, ypredict))
print ("\n Confusion matrix of training \n", confusion_matrix(y_train, ypredict))

ypredict = clf.predict(X_test)
print ('\nTraining classification report\n', classification_report(y_test, ypredict))
print ("\n Confusion matrix of training \n", confusion_matrix(y_test, ypredict))

input_data = (23,1,3,110,180,0,0,100,0,1.2,0,0,0)

input_data_as_numpy_array= np.asarray(input_data)

input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = clf.predict(input_data_reshaped)
print(prediction)

if (prediction[0]== 0):
  print('\n\nThe Person does not have a Heart Disease')
else:
  print('\n\nThe Person has Heart Disease')
```

```
91.32231404958677
77.04918032786885

Training classification report
              precision    recall  f1-score   support

           0       0.87      0.95      0.91       109
           1       0.96      0.88      0.92       133

    accuracy                           0.91       242
   macro avg       0.91      0.92      0.91       242
weighted avg       0.92      0.91      0.91       242


 Confusion matrix of training
 [[104   5]
 [ 16 117]]

Training classification report
              precision    recall  f1-score   support

           0       0.74      0.79      0.77        29
           1       0.80      0.75      0.77        32

    accuracy                           0.77        61
...
[0]


The Person does not have a Heart Disease
```

```
# Logistic Regression algorithm :
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

df = pd.read_csv('heart.csv')
columns = ['age', 'sex', 'chest pain', 'resting blood pressure', 'cholesterol levels', 'fasting blood sugar', 'resting electrocardiogram', 'maximum
labels = df['diagnosis'].values
features = df[list(columns)].values

X_train, X_test, y_train, y_test = train_test_split(features, labels, train_size=0.80)

clf = LogisticRegression()
clf = clf.fit(X_train,y_train)

accurecy_train = clf.score(X_train, y_train)
print (accurecy_train*100)

accurecy_test = clf.score(X_test, y_test)
print (accurecy_test*100)

ypredict = clf.predict(X_train)
print ('\nTraining classification report\n', classification_report(y_train, ypredict))
print ("\n Confusion matrix of training \n", confusion_matrix(y_train, ypredict))

ypredict = clf.predict(X_test)
print ('\nTraining classification report\n', classification_report(y_test, ypredict))
print ("\n Confusion matrix of training \n", confusion_matrix(y_test, ypredict))

input_data = (23,1,3,110,180,0,0,100,0,1.2,0,0,0)
```

```
input_data_as_numpy_array= np.asarray(input_data)

input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = clf.predict(input_data_reshaped)
print(prediction)

if (prediction[0]== 0):
  print('\n\nThe Person does not have a Heart Disease')
else:
  print('\n\nThe Person has Heart Disease')
```

```
85.53719008264463
85.24590163934425

Training classification report
              precision    recall  f1-score   support

           0       0.90      0.76      0.82       107
           1       0.83      0.93      0.88       135

    accuracy                           0.86       242
   macro avg       0.86      0.85      0.85       242
weighted avg       0.86      0.86      0.85       242


 Confusion matrix of training
 [[ 81  26]
 [  9 126]]

Training classification report
              precision    recall  f1-score   support

           0       0.92      0.77      0.84        31
           1       0.80      0.93      0.86        30

    accuracy                           0.85        61
...
[1]


The Person has Heart Disease
```

```
# Neural Network algorithm
import numpy as np
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix



df = pd.read_csv('heart.csv')
columns = ['age', 'sex', 'chest pain', 'resting blood pressure', 'cholesterol levels', 'fasting blood sugar', 'resting electrocardiogram', 'maximum heart rate'
labels = df['diagnosis'].values
features = df[list(columns)].values

X_train, X_test, y_train, y_test = train_test_split(features, labels, train_size=0.80)

clf = MLPClassifier(hidden_layer_sizes=(100, 50), max_iter=500, random_state=42)
clf = clf.fit(X_train,y_train)

accurecy_train = clf.score(X_train, y_train)
print (accurecy_train*100)

accurecy_test = clf.score(X_test, y_test)
print (accurecy_test*100)

ypredict = clf.predict(X_train)
print ('\nTraining classification report\n', classification_report(y_train, ypredict))
print ("\n Confusion matrix of training \n", confusion_matrix(y_train, ypredict))

ypredict = clf.predict(X_test)
print ('\nTraining classification report\n', classification_report(y_test, ypredict))
print ("\n Confusion matrix of training \n", confusion_matrix(y_test, ypredict))
```

```
input_data = (23,1,3,110,180,0,0,100,0,1.2,0,0,0)

input_data_as_numpy_array= np.asarray(input_data)

input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = clf.predict(input_data_reshaped)
print(prediction)

if (prediction[0]== 0):
  print('\n\nThe Person does not have a Heart Disease')
else:
  print('\n\nThe Person has Heart Disease')
```

```
86.36363636363636
78.68852459016394

Training classification report
              precision    recall  f1-score   support

           0       0.90      0.79      0.84       110
           1       0.84      0.92      0.88       132

    accuracy                           0.86       242
   macro avg       0.87      0.86      0.86       242
weighted avg       0.87      0.86      0.86       242


 Confusion matrix of training
 [[ 87  23]
 [ 10 122]]

Training classification report
              precision    recall  f1-score   support

           0       0.86      0.64      0.73        28
           1       0.75      0.91      0.82        33

    accuracy                           0.79        61
...
[1]


The Person has Heart Disease
```

Figure 11 Project in Python

Discussion & conclusion:

This project successfully demonstrates the application of machine learning techniques to predict heart disease using clinical data. Through a systematic approach, we imported and explored a comprehensive dataset, ensuring its integrity and understanding its structure. By performing exploratory data analysis, we identified key relationships among features, which informed our model development.

We implemented several machine learning algorithms, including Random Forest, Logistic Regression, and Neural Networks, evaluating their performance through accuracy scores, confusion matrices, and classification reports. The use of scaling techniques, such as Standard Scaler and Min-Max Scaler, proved beneficial in enhancing model accuracy.

The results indicate that machine learning can significantly improve diagnostic capabilities in the healthcare domain, providing a valuable tool for early detection of heart disease. By allowing for real-time predictions based on patient data, our model offers practical implications for clinical decision-making, potentially leading to better patient outcomes.

This project highlights the promising role of machine learning in medical diagnostics and opens avenues for further research, including the integration of additional datasets and the exploration of more advanced algorithms. Future work could also focus on addressing ethical considerations and ensuring the responsible application of these technologies in healthcare settings.

References:

Kaggle: Your Home for Data Science
Welcome to Python.org
https://orangedatamining.com/