HEARTDISEASE DETECTION

FUNDAMENTALS OF AI ML

[VITyarthi]

SLOT: B14+B23+D21

By

Ms. JIGYASA

Reg NO. : 25BCY10166

Submitted to

Dr.SANTOSH KUMAR SAHOO

TABLE OF CONTENTS

INTRODUCTION

Heart disease is a major global health concern and is responsible for millions of deaths every year. As technology advances, machine learning has become a very powerful means of giving assistance to early detections of such diseases.

The project is related to building a predictive model that will analyze patient clinical data and predict the probability of heart disease. Using Python, its libraries for data science, and the KNN algorithm, it is shown how machine learning can support healthcare analytics.

PROBLEM STATEMENT

Early diagnosis of heart diseases may considerably reduce mortality rates, yet traditional medical tests to detect heart conditions often take time, are costly, and inaccessible to many.

The problem addressed in this project is:

"Is it possible to create a machine learning model based on the medical attributes of patients with heart disease in assisting preliminary diagnosis?

The goal is to analyze the medical data, find out patterns, and predict whether a person has heart disease (1) or not (0).

FUNCTIONAL REQUIREMENTS

FR1 - Load Dataset

The system must load and read the heart disease dataset, heart_disease_data.csv.

FR2 - Perform Preprocessing of Data

Handle duplicates

Scale features using StandardScaler

Split Data into Training and Testing Sets

FR3 – Provide Data Visualization

Bar chart for target distribution

Correlation heatmap

FR4 – Train Machine Learning Model

Train a K-Nearest Neighbors classifier on the dataset.

FR5 – Evaluate Model

Calculate accuracy using test dataset.

FR6 – Predict Output for New Input

Allow users to input new patient values and receive a prediction of 0 or 1

NON FUNCTIONAL REQUIREMENTS

NFR1 – Performance

A model should make predictions in milliseconds.

Training should complete within seconds.

NFR2 – Usability

Easy-to-read code and output in Jupyter Notebook.

NFR3 – Reliability

Prediction should be reproducible with the same input, using random_state.

NFR4 - Maintainability Code should be modular and easy to update with new models.

NFR5 – Security Dataset handled locally, no sharing of data outside.

SYSTEM ARCHITECTURE

Then, a simple three-layer architecture is utilized:

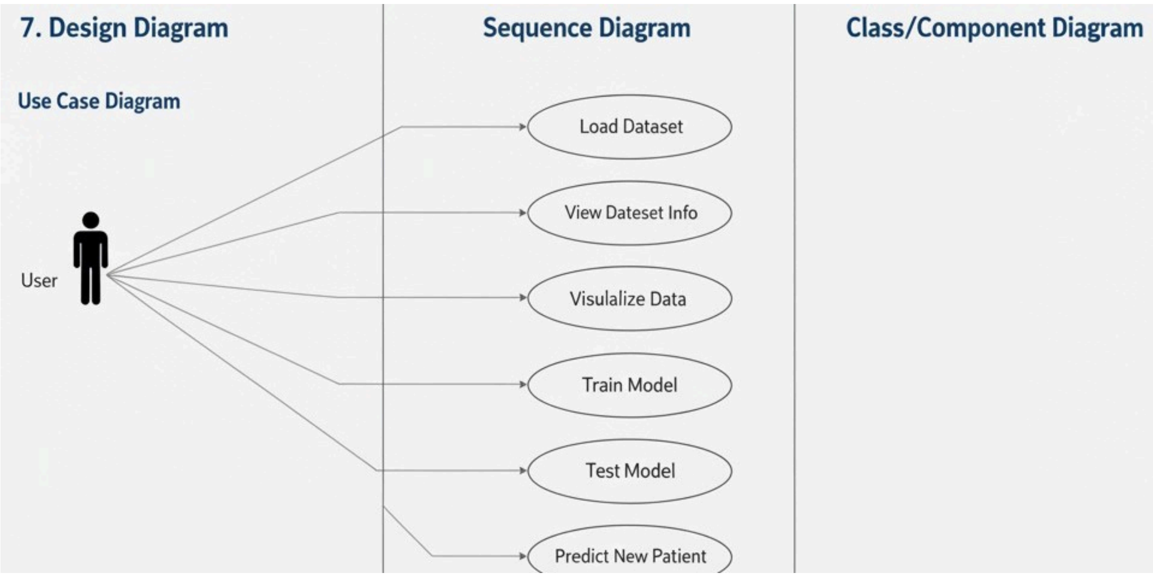
1. Data Layer
   - Dataset (heart_disease_data.csv)
   - Feature extraction

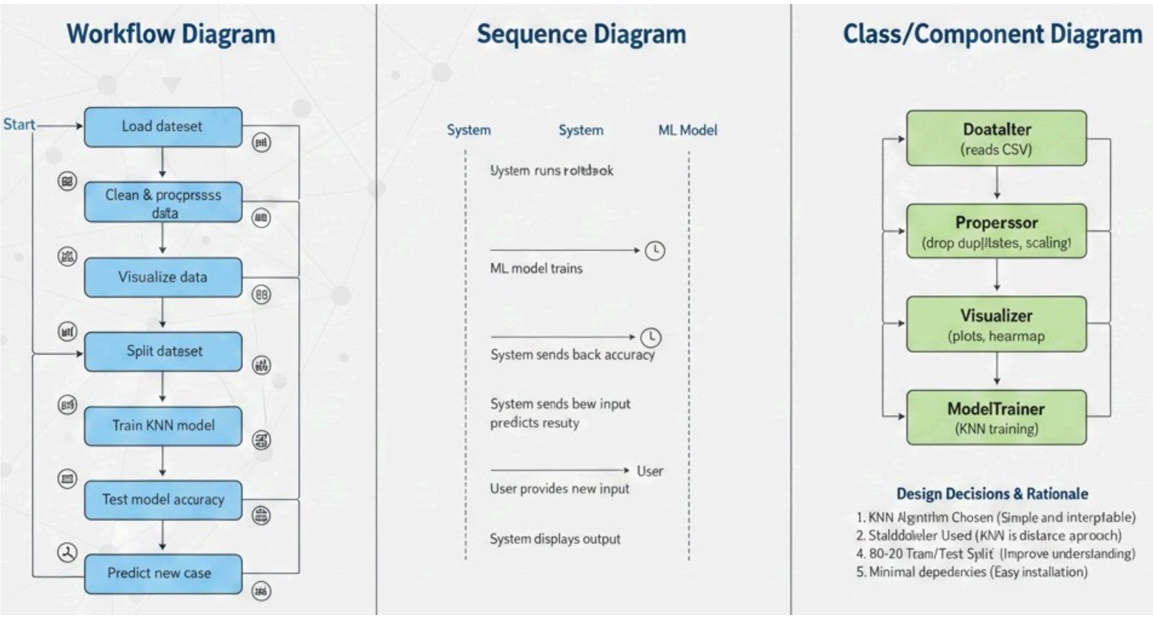
2. Processing Layer
   - Data cleaning
   - Scaling
   - KNN training and prediction

3. Output Layer
   - Visualizations
   - Accuracy score
   - Prediction result for new patient data

Design Diagrams

Here is the pictorial representation of design diagrams of various types:



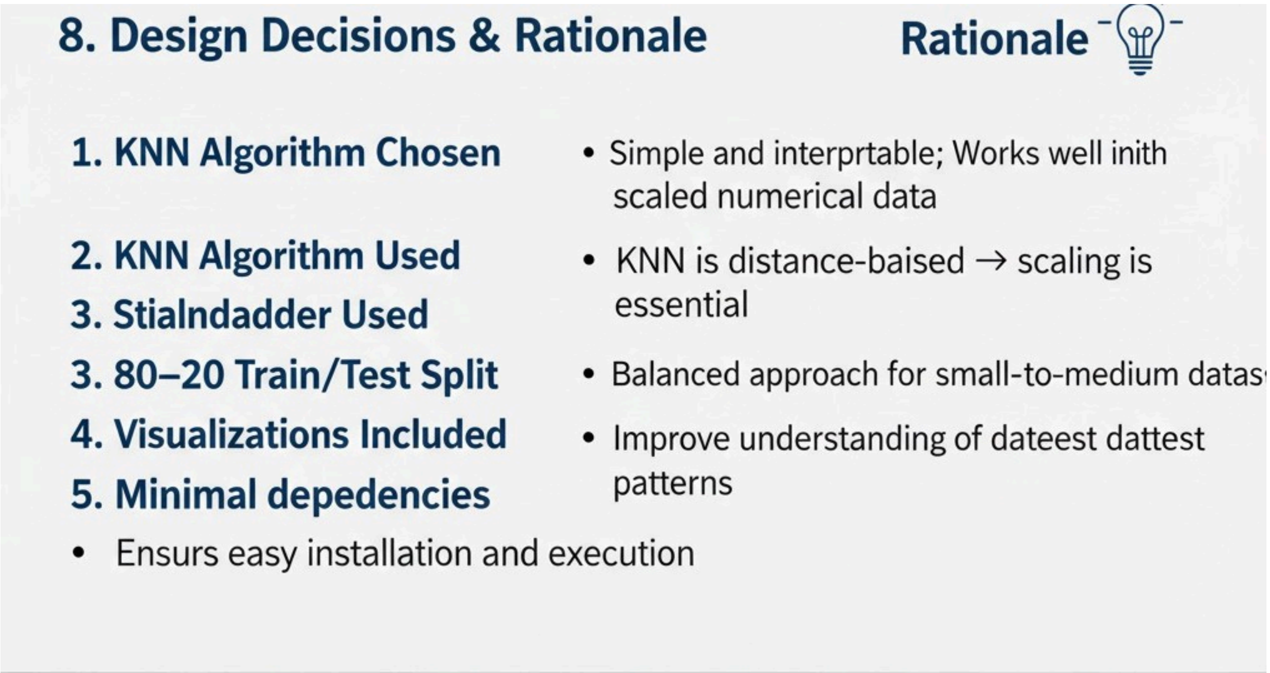


Design Decisions & Rationale

1. KNN Algntihm Chosen (Simple and interplable)
2. Stalddoleler Used (KNN is distarce aprooch)
4. 80-20 Tram/Test Split (Improve understanding)
5. Minimal depedenxies (Easy installation)

DESIGN DECISION & RATIONALE

Here is a pictorial representation of the following:



8. Design Decisions & Rationale

Rationale

1. KNN Algorithm Chosen

- Simple and interprtable; Works well inith scaled numerical data

2. KNN Algorithm Used

- KNN is distance-baised → scaling is essential

3. Stialndadder Used

3. 80–20 Train/Test Split

- Balanced approach for small-to-medium datas

4. Visualizations Included

- Improve understanding of dateest dattest patterns

5. Minimal depedencies

- Ensurs easy installation and execution

IMPLEMENTATION DETAILS

Language: Python

IDE: Jupyter Notebook

Libraries: pandas, numpy, matplotlib, seaborn, scikit-learn

Model: KNN with n_neighbors = 5

Data Handling:

- Import CSV
- Display head, info, describe

Preprocessing:

- Drop duplicates
- Scale features

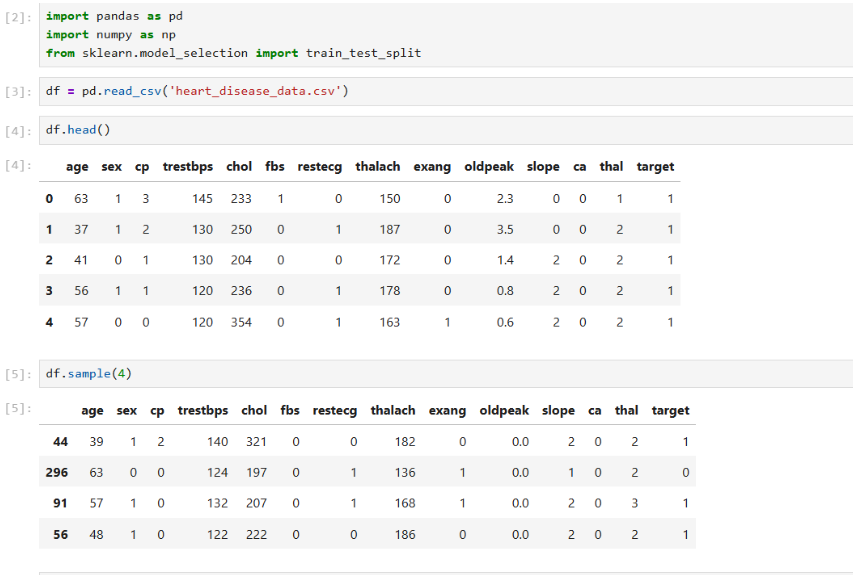
Model Training:

```
knn.fit(X_train_scaled, y_train)
```

Prediction:

```
knn.predict(new.reshape(1,-1))
```

SCREENSHOTS



[2]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
```

[3]:
```
df = pd.read_csv('heart_disease_data.csv')
```

[4]:
```
df.head()
```

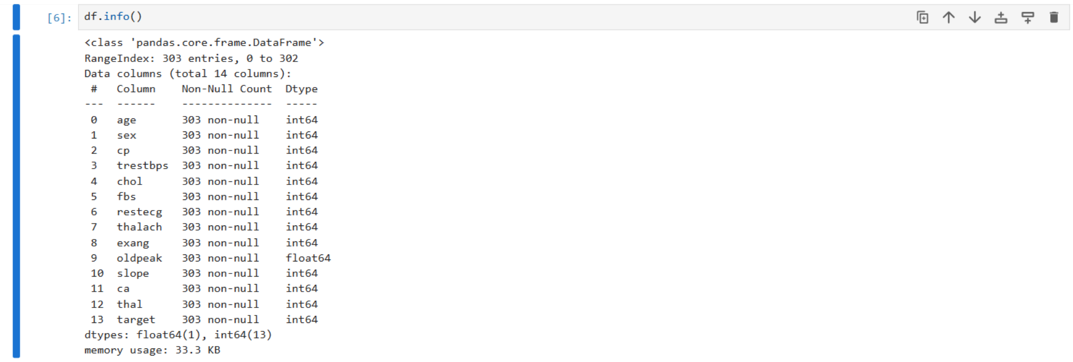
[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
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]:
```
df.sample(4)
```

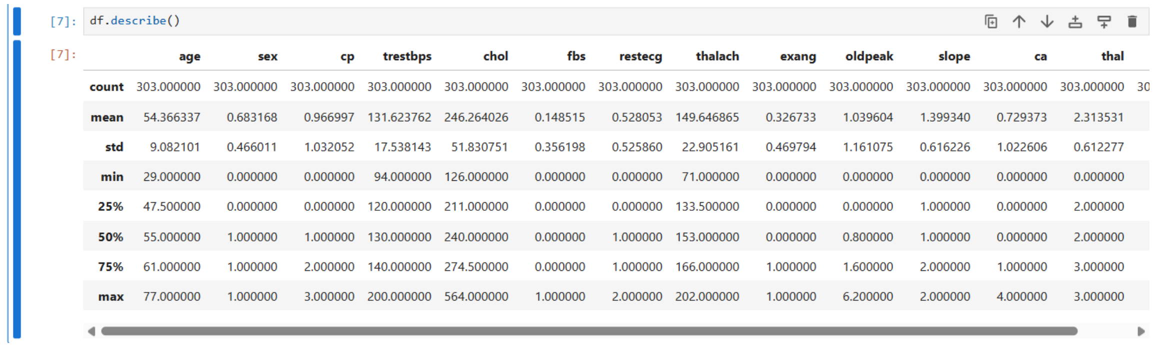
[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
44	39	1	2	140	321	0	0	182	0	0.0	2	0	2	1
296	63	0	0	124	197	0	1	136	1	0.0	1	0	2	0
91	57	1	0	132	207	0	1	168	1	0.0	2	0	3	1
56	48	1	0	122	222	0	0	186	0	0.0	2	0	2	1



[6]:
```
df.info()
```

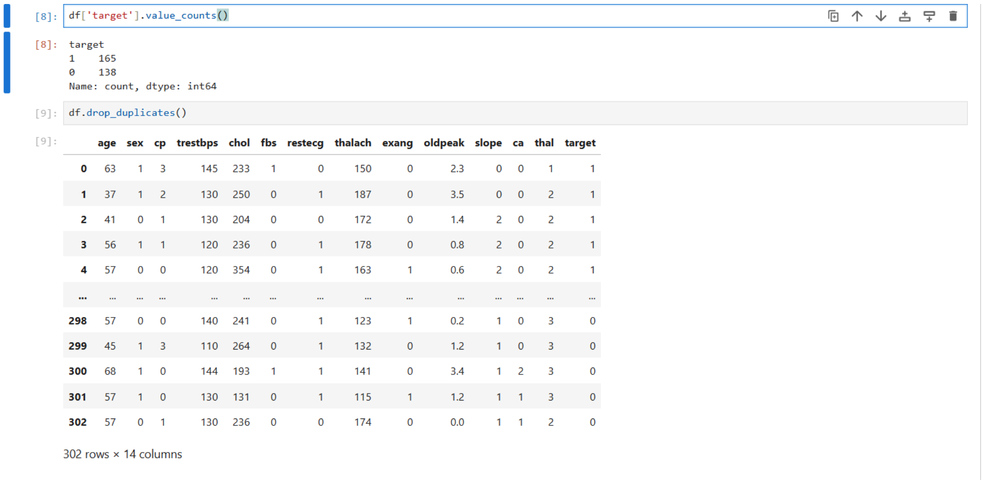
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```



[7]:
```
df.describe()
```

[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	30
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	



```
[8]: df['target'].value_counts()
```

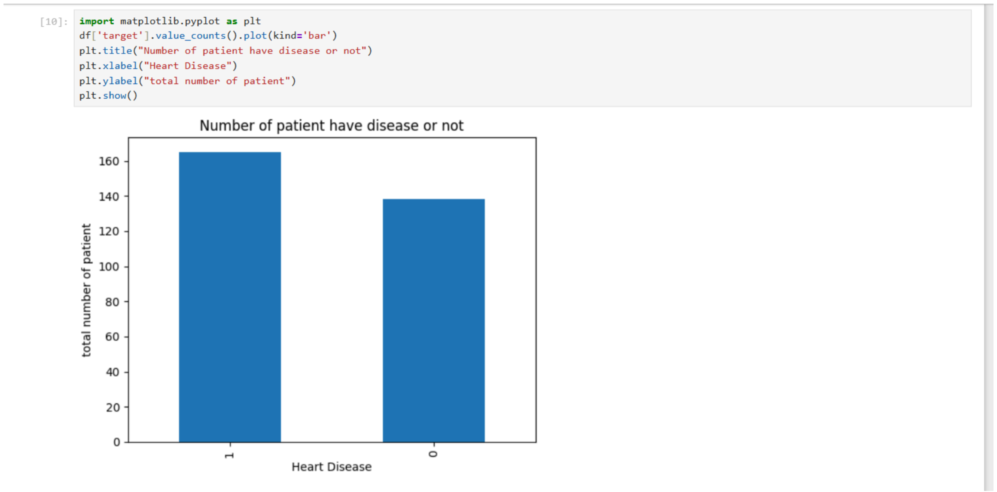
```
[8]: target
     1    165
     0    138
     Name: count, dtype: int64
```

```
[9]: df.drop_duplicates()
```

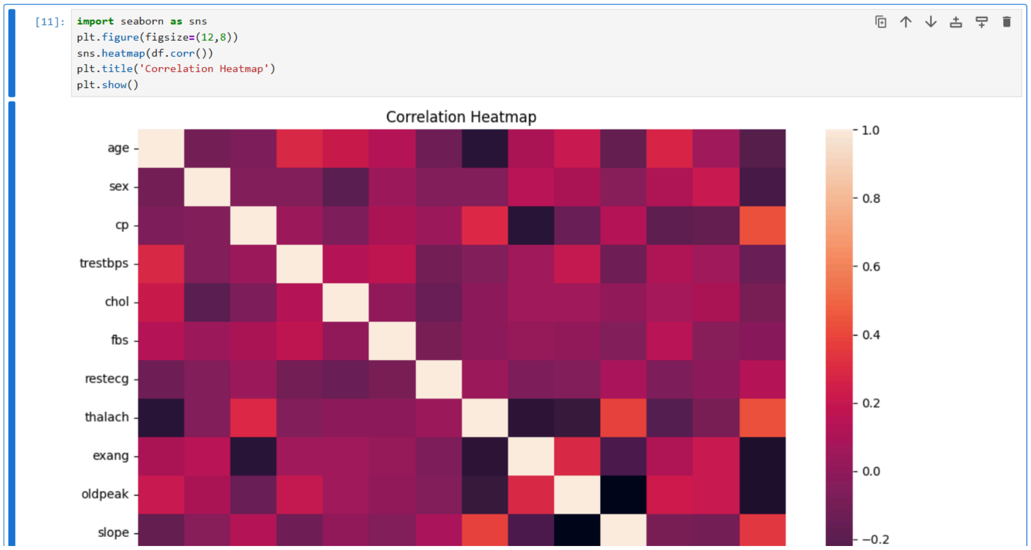
[9]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

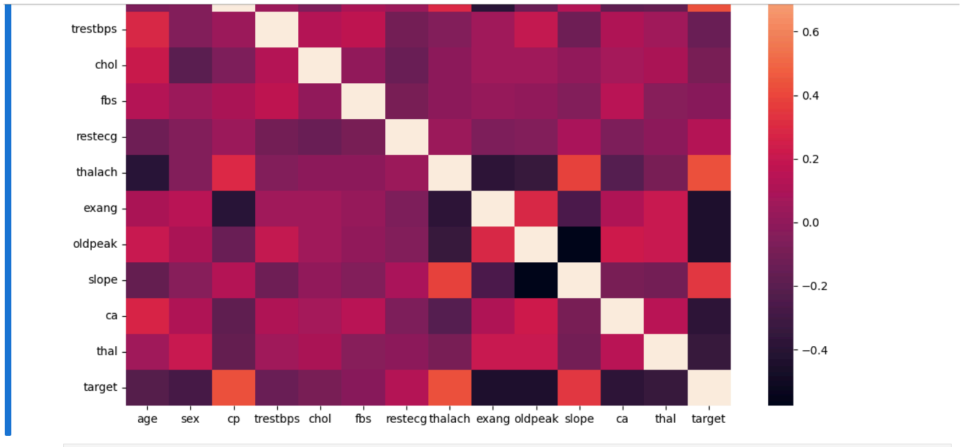
302 rows × 14 columns

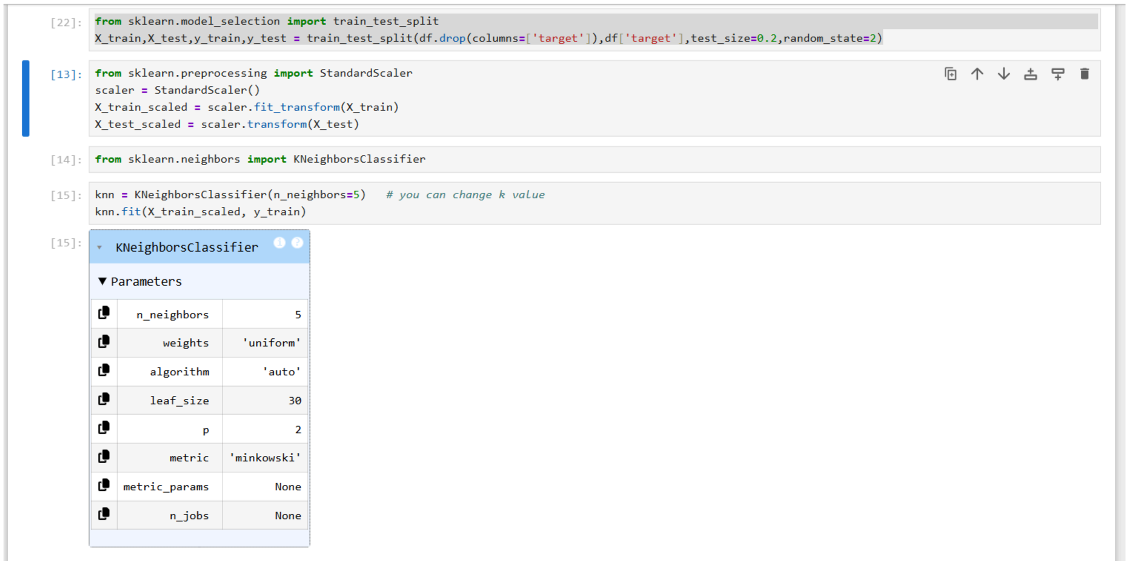


```
[10]: import matplotlib.pyplot as plt
      df['target'].value_counts().plot(kind='bar')
      plt.title("Number of patient have disease or not")
      plt.xlabel("Heart Disease")
      plt.ylabel("total number of patient")
      plt.show()
```



```
[11]: import seaborn as sns
      plt.figure(figsize=(12,8))
      sns.heatmap(df.corr())
      plt.title('Correlation Heatmap')
      plt.show()
```





[22]:
```python
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(df.drop(columns=['target']),df['target'],test_size=0.2,random_state=2)
```

[13]:
```python
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[14]:
```python
from sklearn.neighbors import KNeighborsClassifier
```

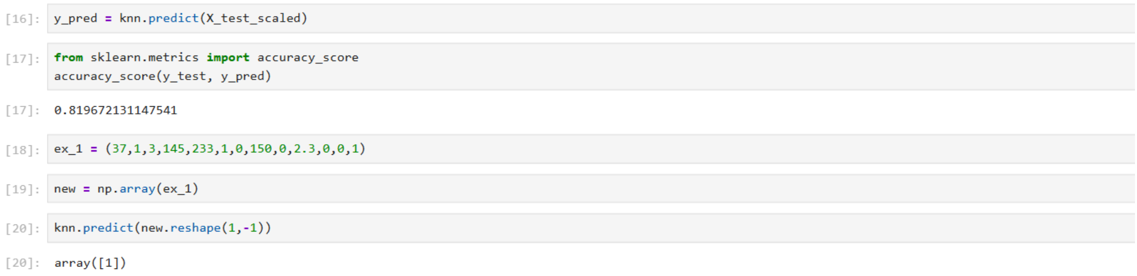
[15]:
```python
knn = KNeighborsClassifier(n_neighbors=5)   # you can change k value
knn.fit(X_train_scaled, y_train)
```

[15]:

KNeighborsClassifier

▼ Parameters

	n_neighbors	5
	weights	'uniform'
	algorithm	'auto'
	leaf_size	30
	p	2
	metric	'minkowski'
	metric_params	None
	n_jobs	None



[16]:
```python
y_pred = knn.predict(X_test_scaled)
```

[17]:
```python
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

[17]:
```
0.819672131147541
```

[18]:
```python
ex_1 = (37,1,3,145,233,1,0,150,0,2.3,0,0,1)
```

[19]:
```python
new = np.array(ex_1)
```

[20]:
```python
knn.predict(new.reshape(1,-1))
```

[20]:
```
array([1])
```

TESTING APPROACH

Testing Types Used:

1. **Unit Testing**
   Verified model predictions using sample inputs

2. **Functional Testing**
   Ensured each code section executed correctly

3. **Data Testing**
   Checks for missing values, duplicates

4. **Performance Test Model**
   Training speed Prediction response Accuracy Testing then uses accuracy_score(y_test, y_pred) to evaluate the model.

CHALLENGES FACED

- Ensuring proper scaling prior to feeding data into KNN.
- Understanding correlations in dataset
- Avoiding overfitting with small data Choosing the correct value of k

KEY LEARNINGS AND TAKEAWAYS

- Importance of data preprocessing
- Visualization helps in clear understanding of dataset
- KNN works well on scaled numerical data.
- Evaluation of machine learning models is essential.
- Workflow of ML projects: data → preprocessing → modeling → testing → prediction

FUTURE ENHANCEMENTS

- Add more ML algorithms: Logistic Regression, Random Forest, SVM
- Build a GUI using Tkinter or Streamlit
- Deploy model as a web app
- Improve accuracy using hyperparameter tuning
- Add real-time data import

REFERENCES

In order to work on this project websites are referred by me during the various

phases of development of the project.

1) www.youtube.com

2) www.python.com

OTHER THAN THE MENTIONED THING I HAVE ALSO SEEKED HELP AND INFORMATION FROM MY TEACHERS WHO MADE ME UNDERSTAND EACH AND EVERY DETAIL OF THE PROJECT