heart

December 1, 2023

Importing the Dependencies

[ ]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Data Collection and Processing

[ ]:
```
# loading the csv data to a Pandas DataFrame
heart_data = pd.read_csv('/content/heart.csv')
```

[ ]:
```
# print first 5 rows of the dataset
heart_data.head()
```

[ ]:

	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

[ ]:
```
# print last 5 rows of the dataset
heart_data.tail()
```

[ ]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak
298	57	0	0	140	241	0	1	123	1	0.2
299	45	1	3	110	264	0	1	132	0	1.2
300	68	1	0	144	193	1	1	141	0	3.4
301	57	1	0	130	131	0	1	115	1	1.2

```
302   57    0   1       130   236    0        0       174       0       0.0

     slope  ca  thal  target
298      1   0     3       0
299      1   0     3       0
300      1   2     3       0
301      1   1     3       0
302      1   1     2       0
```

[ ]:
```
# number of rows and columns in the dataset
heart_data.shape
```

[ ]:
```
(303, 14)
```

[ ]:
```
# getting some info about the data
heart_data.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
```

[ ]:
```
# checking for missing values
heart_data.isnull().sum()
```

[ ]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
```

```
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

[ ]:
```
# statistical measures about the data
heart_data.describe()
```

[ ]:
```
              age         sex          cp    trestbps        chol         fbs  \
count  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000
mean    54.366337    0.683168    0.966997  131.623762  246.264026    0.148515
std      9.082101    0.466011    1.032052   17.538143   51.830751    0.356198
min     29.000000    0.000000    0.000000   94.000000  126.000000    0.000000
25%     47.500000    0.000000    0.000000  120.000000  211.000000    0.000000
50%     55.000000    1.000000    1.000000  130.000000  240.000000    0.000000
75%     61.000000    1.000000    2.000000  140.000000  274.500000    0.000000
max     77.000000    1.000000    3.000000  200.000000  564.000000    1.000000

          restecg     thalach       exang     oldpeak       slope          ca  \
count  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000
mean     0.528053  149.646865    0.326733    1.039604    1.399340    0.729373
std      0.525860   22.905161    0.469794    1.161075    0.616226    1.022606
min      0.000000   71.000000    0.000000    0.000000    0.000000    0.000000
25%      0.000000  133.500000    0.000000    0.000000    1.000000    0.000000
50%      1.000000  153.000000    0.000000    0.800000    1.000000    0.000000
75%      1.000000  166.000000    1.000000    1.600000    2.000000    1.000000
max      2.000000  202.000000    1.000000    6.200000    2.000000    4.000000

             thal      target
count  303.000000  303.000000
mean     2.313531    0.544554
std      0.612277    0.498835
min      0.000000    0.000000
25%      2.000000    0.000000
50%      2.000000    1.000000
75%      3.000000    1.000000
max      3.000000    1.000000
```

[ ]:
```
# checking the distribution of Target Variable
heart_data['target'].value_counts()
```

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

1 –> Defective Heart

0 –> Healthy Heart

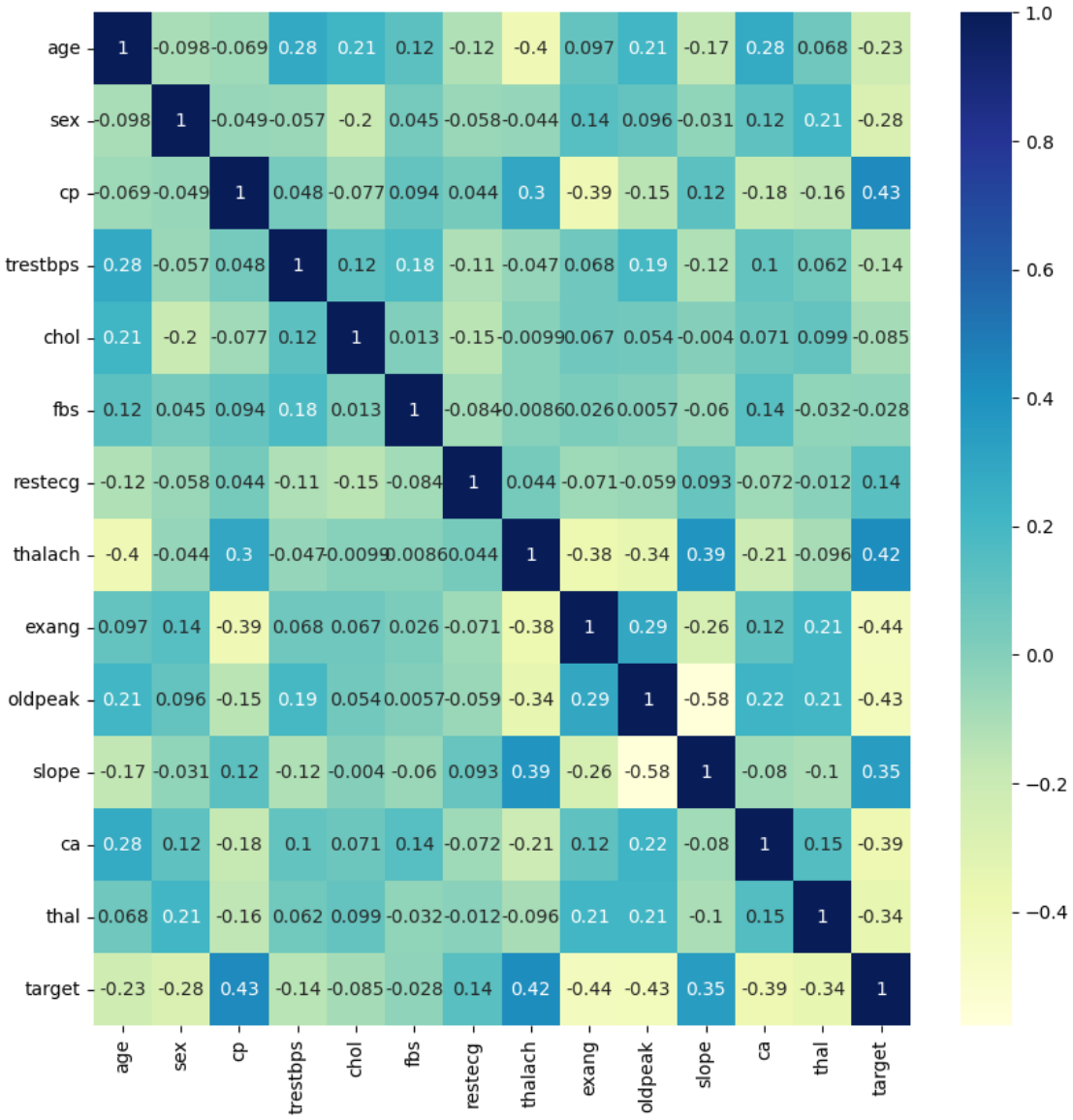
[ ]:
```
# To find the corelation in data set we use corr()
import matplotlib.pyplot as plt
import seaborn as sns
corrMatrix=heart_data.corr()
corrMatrix
```

[ ]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.398522	0.096801	0.210013	-0.168814	0.276326	0.068001	-0.225439
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.044020	0.141664	0.096093	-0.030711	0.118261	0.210041	-0.280937
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.295762	-0.394280	-0.149230	0.119717	-0.181053	-0.161736	0.433798
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.046698	0.067616	0.193216	-0.121475	0.101389	0.062210	-0.144931
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.009940	0.067023	0.053952	-0.004038	0.070511	0.098803	-0.085239
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.008567	0.025665	0.005747	-0.059894	0.137979	-0.032019	-0.028046
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.044123	-0.070733	-0.058770	0.093045	-0.072042	-0.011981	0.137230
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.000000	-0.378812	-0.344187	0.386784	-0.213177	-0.096439	0.421741
exang	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.378812	1.000000	0.288223	-0.257748	0.115739	0.206754	-0.436757
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.344187	0.288223	1.000000	-0.577537	0.222682	0.210244	-0.430696
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.386784	-0.257748	-0.577537	1.000000	-0.080155	-0.104764	0.345877
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.213177	0.115739	0.222682	-0.080155	1.000000	0.151832	-0.391724
thal	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.096439	0.206754	0.210244	-0.104764	0.151832	1.000000	-0.344029
target	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.421741	-0.436757	-0.430696	0.345877	-0.391724	-0.344029	1.000000

```
sex       0.210041 -0.280937
cp       -0.161736  0.433798
trestbps  0.062210 -0.144931
chol      0.098803 -0.085239
fbs      -0.032019 -0.028046
restecg  -0.011981  0.137230
thalach  -0.096439  0.421741
exang     0.206754 -0.436757
oldpeak   0.210244 -0.430696
slope    -0.104764  0.345877
ca        0.151832 -0.391724
thal      1.000000 -0.344029
target   -0.344029  1.000000
```

[ ]:
```
import seaborn as sns
sns.heatmap(corrMatrix,cmap="YlGnBu",annot=True)
plt.gcf().set_size_inches(10, 10)
```



Splitting the Features and Target

[ ]:
```
X = heart_data.drop(columns='target', axis=1)
Y = heart_data['target']
```

[ ]:
```
print(X)
```

```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0     63    1   3       145   233    1        0      150      0      2.3
1     37    1   2       130   250    0        1      187      0      3.5
2     41    0   1       130   204    0        0      172      0      1.4
3     56    1   1       120   236    0        1      178      0      0.8
```

```
4     57    0    0       120    354    0       1       163      1       0.6
..    ...   ...  ..      ...    ...    ...     ...     ...      ...
298   57    0    0       140    241    0       1       123      1       0.2
299   45    1    3       110    264    0       1       132      0       1.2
300   68    1    0       144    193    1       1       141      0       3.4
301   57    1    0       130    131    0       1       115      1       1.2
302   57    0    1       130    236    0       0       174      0       0.0

     slope  ca  thal
0        0   0     1
1        0   0     2
2        2   0     2
3        2   0     2
4        2   0     2
..     ...  ..   ...
298      1   0     3
299      1   0     3
300      1   2     3
301      1   1     3
302      1   1     2

[303 rows x 13 columns]
```

[ ]:
```
print(Y)
```

```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

Splitting the Data into Training data & Test Data

[ ]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
↪stratify=Y, random_state=2)
```

[ ]:
```
print(X.shape, X_train.shape, X_test.shape)
```

```
(303, 13) (242, 13) (61, 13)
```

Model Training

Random Forest

```
[ ]: from sklearn.ensemble import RandomForestClassifier
     model=RandomForestClassifier(n_estimators=10)
```

```
[ ]: # training the LogisticRegression model with Training data
     model.fit(X_train, Y_train)
```

```
[ ]: RandomForestClassifier(n_estimators=10)
```

Model Evaluation

Accuracy Score

```
[ ]: # accuracy on training data
     X_train_prediction = model.predict(X_train)
     training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

```
[ ]: print('Accuracy on Training data : ', training_data_accuracy)
```

```
Accuracy on Training data :  0.987603305785124
```

```
[ ]: # accuracy on test data
     X_test_prediction = model.predict(X_test)
     test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

```
[ ]: print('Accuracy on Test data : ', test_data_accuracy)
```

```
Accuracy on Test data :  0.7540983606557377
```

Building a Predictive System

```
[ ]: input_data = (62,0,0,140,268,0,0,160,0,3.6,0,2,2)

     # change the input data to a numpy array
     input_data_as_numpy_array= np.asarray(input_data)

     # reshape the numpy array as we are predicting for only on instance
     input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

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

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

```
[0]
The Person does not have a Heart Disease
```

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

Saving the trained model

[ ]:
```
import pickle
```

[ ]:
```
filename = 'heart_disease_model.sav'
pickle.dump(model, open(filename, 'wb'))
```

[ ]:
```
# loading the saved model
loaded_model = pickle.load(open('heart_disease_model.sav', 'rb'))
```

[ ]:
```
for column in X.columns:
  print(column)
```

```
age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
```

[ ]:
```

```