Data Analysis Project on Heart Disease Prediction

Presented by:
Name:Pothuri Akish
Mail:akishpothuri@gmail.com

- **Table of Contents:**

- **Table Of Figures/Graphs**

●Abstract

Heart disease is one of the most critical human diseases in the world and affects human life very badly. In heart disease, the heart is unable to push the required amount of blood to other parts of the body. Accurate and on time diagnosis of heart disease is important for heart failure prevention and treatment. The diagnosis of heart disease through traditional medical history has been considered as not reliable in many aspects. To classify the healthy people and people with heart disease, noninvasive-based methods such as machine learning are reliable and efficient.

● CHAPTERS

Importing Data Visualization libraries in python

```
import sklearn
import numpy as np
import pandas as pd
import plotly as plot
import plotly.express as px
import plotly.graph_objs as go
import cufflinks as cf
import matplotlib.pyplot as plt
import seaborn as sns
import os
from sklearn.metrics import accuracy_score
import plotly.offline as pyo
from plotly.offline import init_notebook_mode,plot,iplot
```

- Before going to analysis any datasets we have to ensure that the above libraries should installed in our local machine.if it not installed.Install the libraries by using pip or conda commad(if you are using jupyter-notebook)

Example to install libraries is:
pip install pandas
The pandas library will be install.

- import pandas as pd

  The meaning of the above commad is to import the pandas library in the working file.If we want to call it any where in the working file use it by pd instead pandas.

  The pandas library is used to make dataframe and importing the dataframe from the local machine to the working file.

- import numpy as np

  numpy libraby used for numerical operations in python it consists of multidimensional array objects.

  The above command is import numpy in our working file instead of calling numpy we call it as np required place.

- import sklearn

  Scikit-learn(sklearn) is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

- from sklearn.metrics import accuracy_score

In the sklearn their is a sublibrary called metrics from this we use accuracy_acore module to check the accuracy of the predicted output of the model we trained by testing with the remaining data.

- import plotly as plot

  The plotly Python library is an interactive, open-source plotting library that supports over 40 unique chart types covering a wide range of statistical, financial, geographic, scientific, and 3-dimensional use-cases.

- import plotly.express as px
- import plotly.graph_objs as go

express,graph_objs are the sub-libraries of the plotly library.
Instead of the command plotly.express we can use px and on the same way graph_objs we can use go where ever we wana use them.

- import plotly.offline as pyo
- from plotly.offline import init_notebook_mode,plot,iplot
  Plotly allows you to generate graphs offline and save them in local machine. The plotly. offline. plot() function creates a standalone HTML that is saved locally and opened inside your web browser.

- import cufflinks as cf
  Cufflinks is another library that connects the Pandas data frame with Plotly enabling users to create visualizations directly from Pandas. The library binds the power of Plotly with the flexibility of Pandas for easy plotting.
  In the above command we are importing the cufflinks library and creating a shortcut as cf
- Import matplotlib.pyplot as plt
  Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.
  %matplotlib inline
  sets the backend of matplotlib to the 'inline' backend.With this backend, the output of plotting commands is displayed inline within frontends like the Jupyter notebook
- import seaborn as sns
  Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
- import os
  The OS module in python provides functions for interacting with the operating system

- pyo.init_notebook_mode(connected=True)
- cf.go_offline()
  Enabling the offline mode for interactive plotting locally

- heart=pd.read_csv(r'G:\datascience_internship\project\heart.csv')
- heart

	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

With pd.read_csv we read the dataset in the local machine and do operations on it.CSV represents the 'comma separated values'

- info = ["age","1: male, 0: female","chest pain type, 0: typical angina, 1: atypical angina, 2: non-anginal pain, 3: asymptomatic","resting blood pressure"," serum cholestoral in mg/dl","fasting blood sugar > 120 mg/dl","resting electrocardiographic results (values 0,1,2)"," maximum heart rate achieved","exercise induced angina","oldpeak = ST depression induced by exercise relative to rest","the slope of the peak exercise ST segment","number of major vessels (0-3) colored by flourosopy","thal: 3 = normal; 6 = fixed defect; 7 = reversable defect"]

- for i in range(len(info)):
    print(heart.columns[i]+":\t\t\t"+info[i])

```
age:			age
sex:			1: male, 0: female
cp:			chest pain type, 0: typical angina, 1: atypical angina, 2: non-anginal pain, 3: asymptomatic
trestbps:			resting blood pressure
chol:			 serum cholestoral in mg/dl
fbs:			fasting blood sugar > 120 mg/dl
restecg:			resting electrocardiographic results (values 0,1,2)
thalach:			 maximum heart rate achieved
exang:			exercise induced angina
oldpeak:			oldpeak = ST depression induced by exercise relative to rest
slope:			the slope of the peak exercise ST segment
ca:			number of major vessels (0-3) colored by flourosopy
thal:			thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
```

In the info object information of the dataset was stored as list.

- heart['target']

```
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
```

- heart.groupby('target').size()

```
target
0    138
1    165
dtype: int64
```

- heart.groupby('target').sum()

target	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	7811	114	66	18547	34650	22	62	19196	76	218.8	161	161	351
1	8662	93	227	21335	39968	23	98	26147	23	96.2	263	60	350

- heart.shape

  (303, 14)

- heart.size

  4242

- heart.describe()

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
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	303.000000
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	0.544554
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	0.498835
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	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	0.000000
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	1.000000
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	1.000000
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	1.000000

- heart.info()

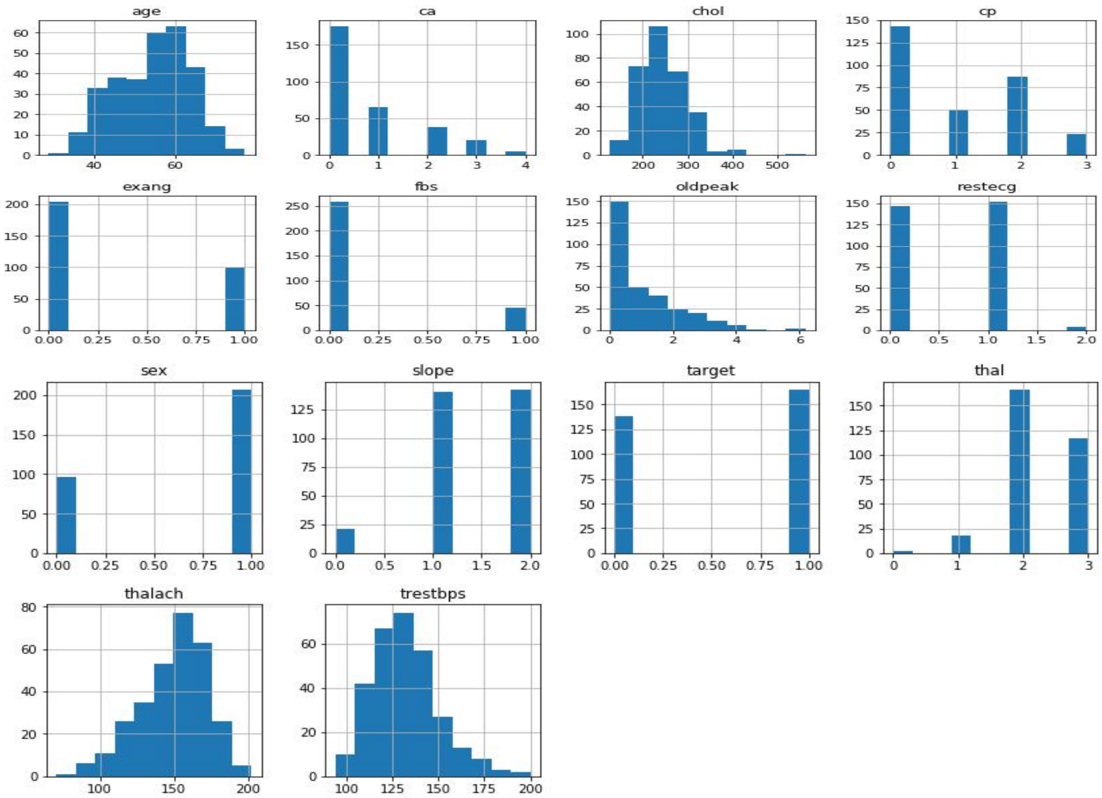
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  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
```

- heart['target'].unique()

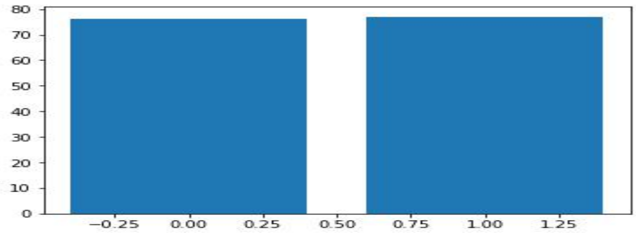
```
array([1, 0], dtype=int64)
```

Visualization

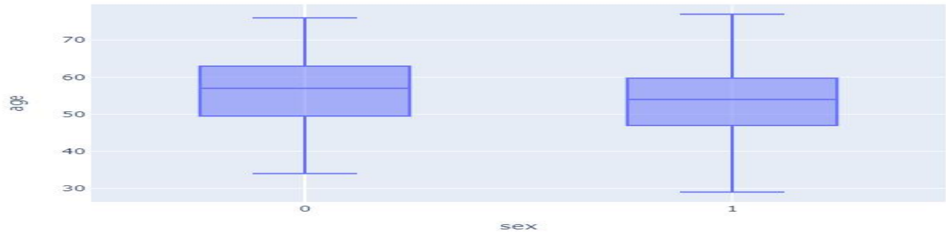
- heart.hist(figsize=(14,14))
- plt.show()



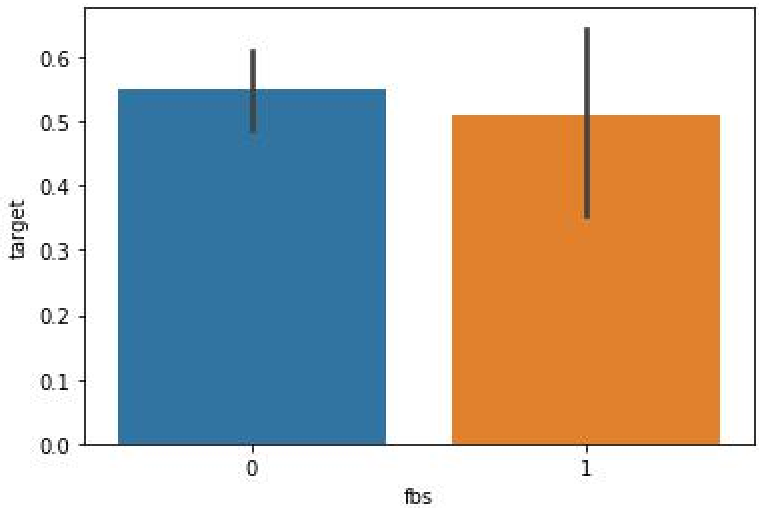
- plt.bar(x=heart['sex'],height=heart['age'])
- plt.show()



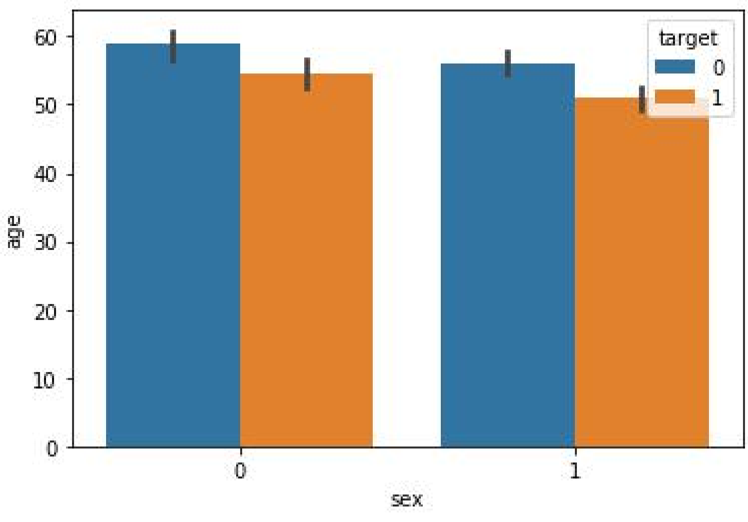
- px.box(heart,heart['sex'],heart['age'])



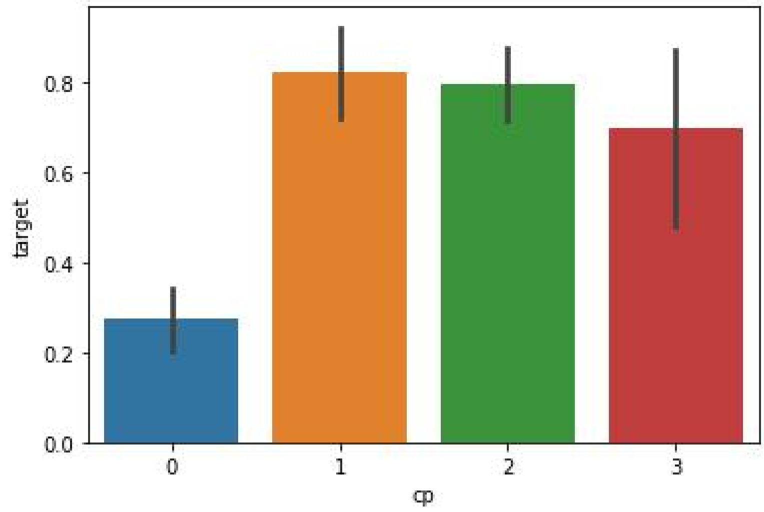
- sns.barplot(x="fbs", y="target", data=heart)
- plt.show()



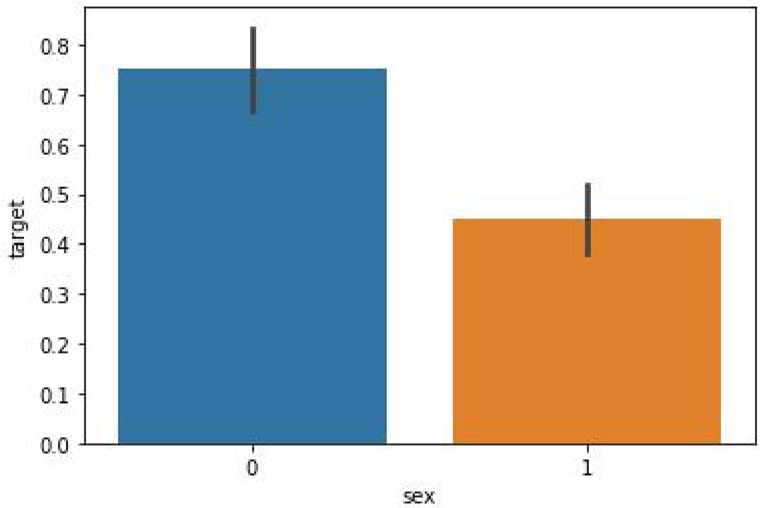
- sns.barplot(x=heart['sex'],y=heart['age'],hue=heart['target'])



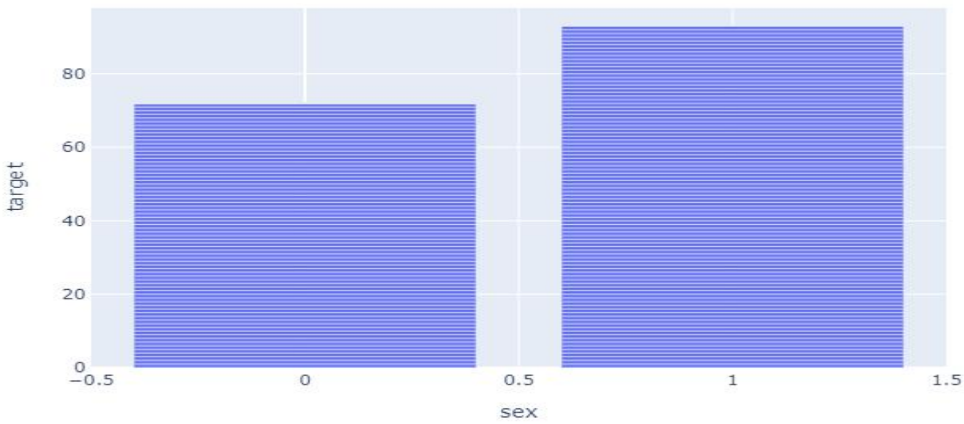
● sns.barplot(heart["cp"],heart['target'])



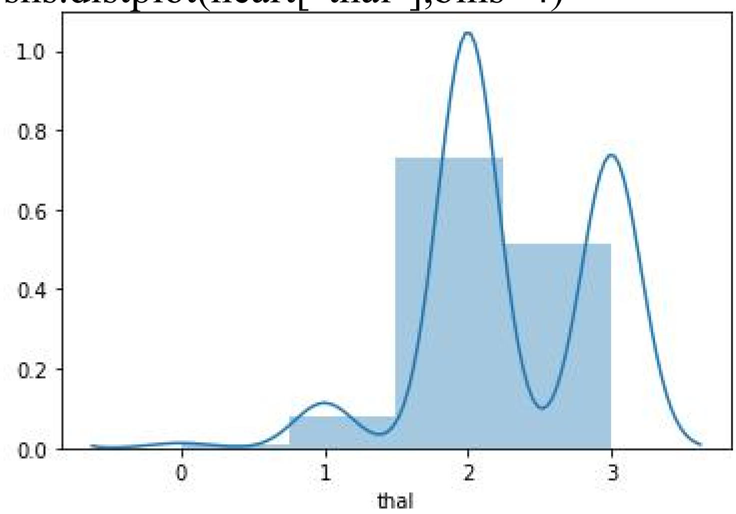
● sns.barplot(heart["sex"],heart['target'])



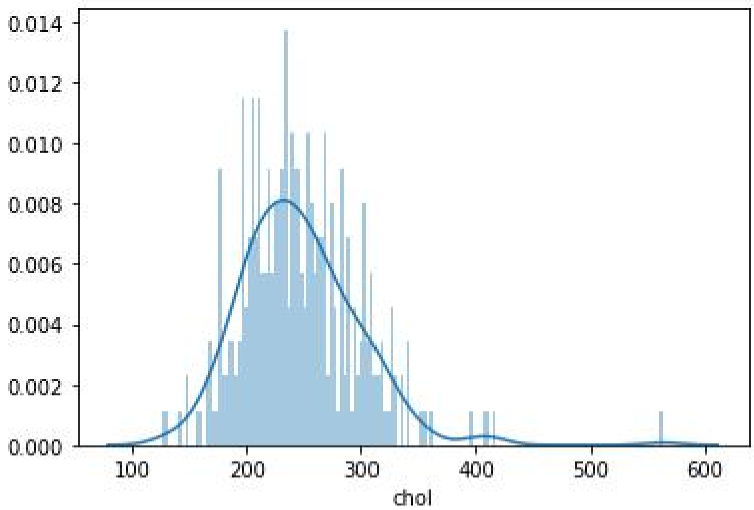
● px.bar(heart,heart['sex'],heart['target'])



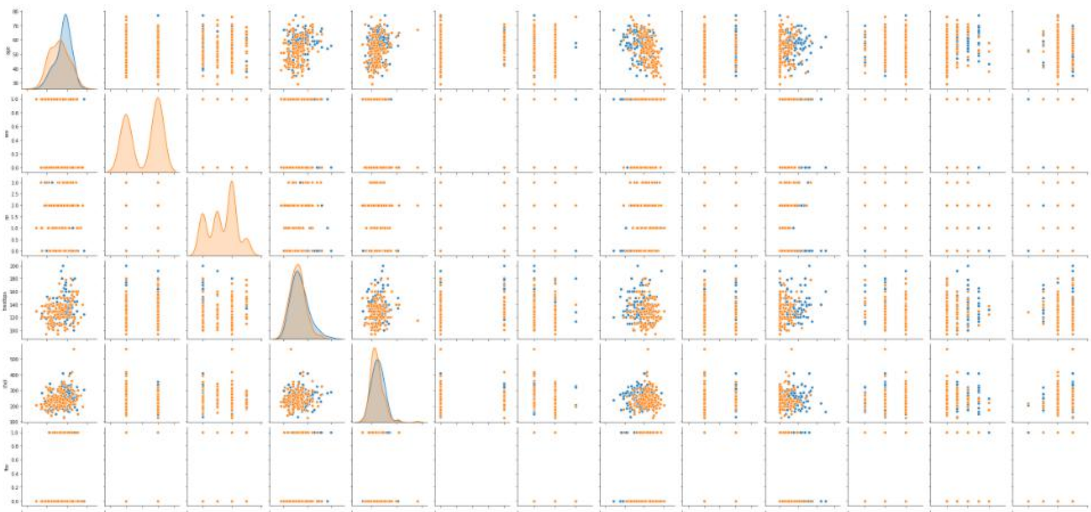
- sns.distplot(heart["thal"],bins=4)

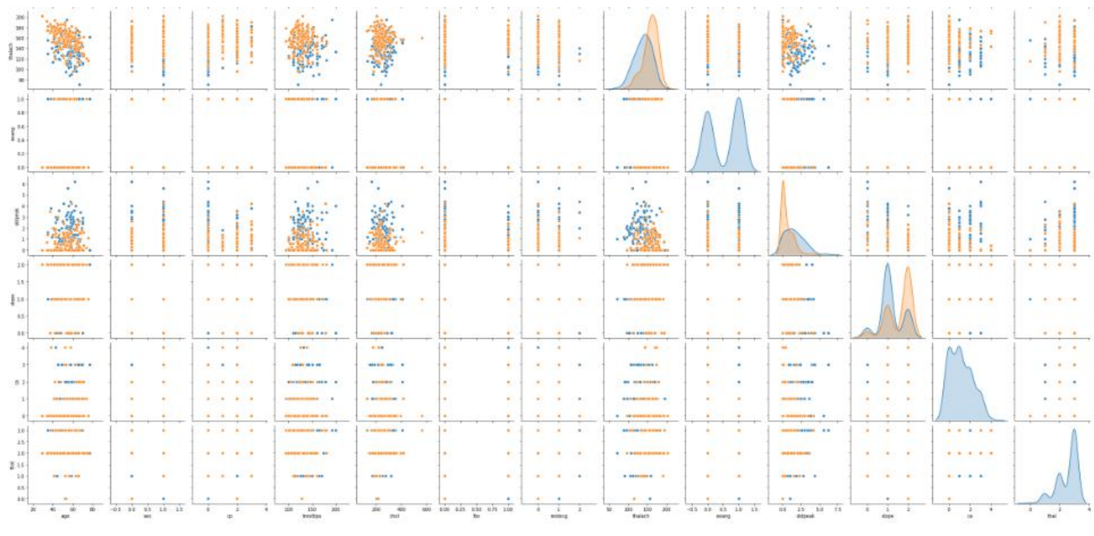


- sns.distplot(heart["chol"],bins=152)

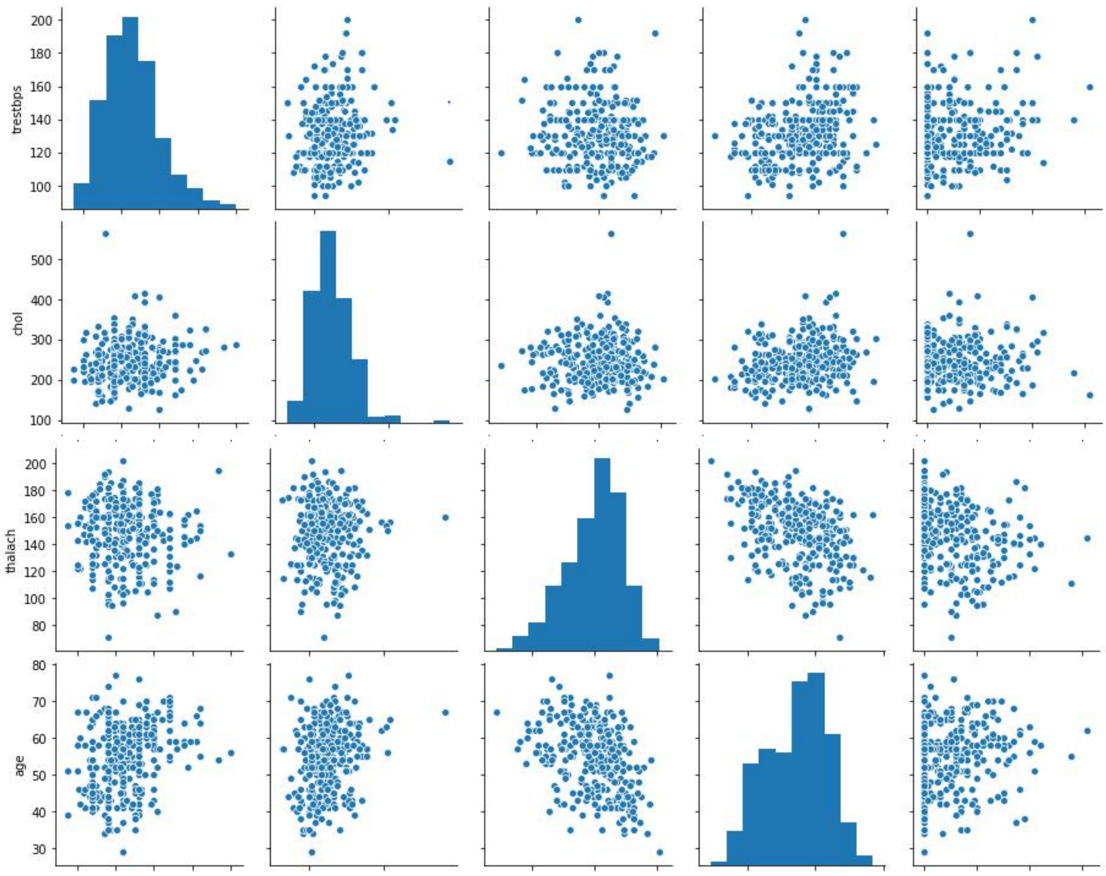


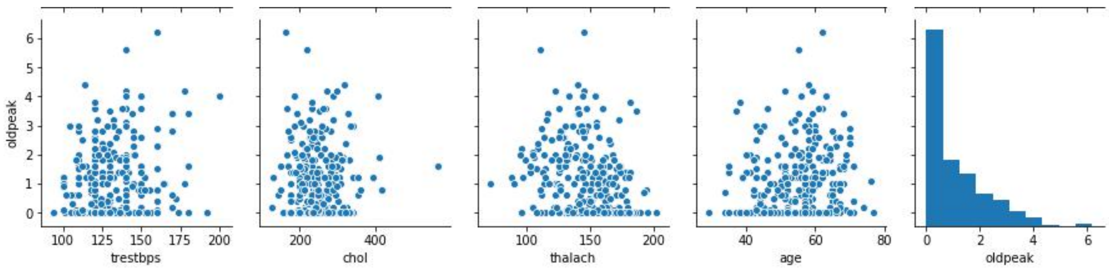
- sns.pairplot(heart,hue='target')





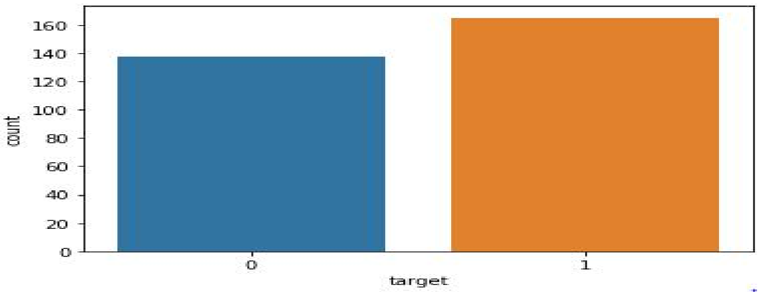
- numeric_columns=['trestbps','chol','thalach','age','oldpeak']
- sns.pairplot(heart[numeric_columns])



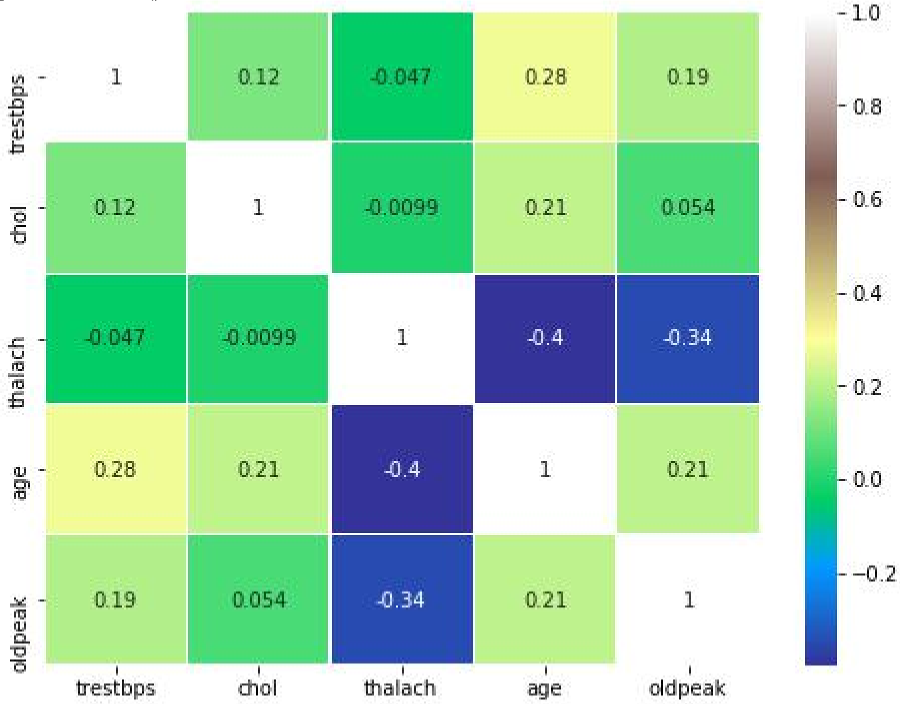


- y = heart["target"]
- sns.countplot(y)
- target_temp = heart.target.value_counts()
- print(target_temp)

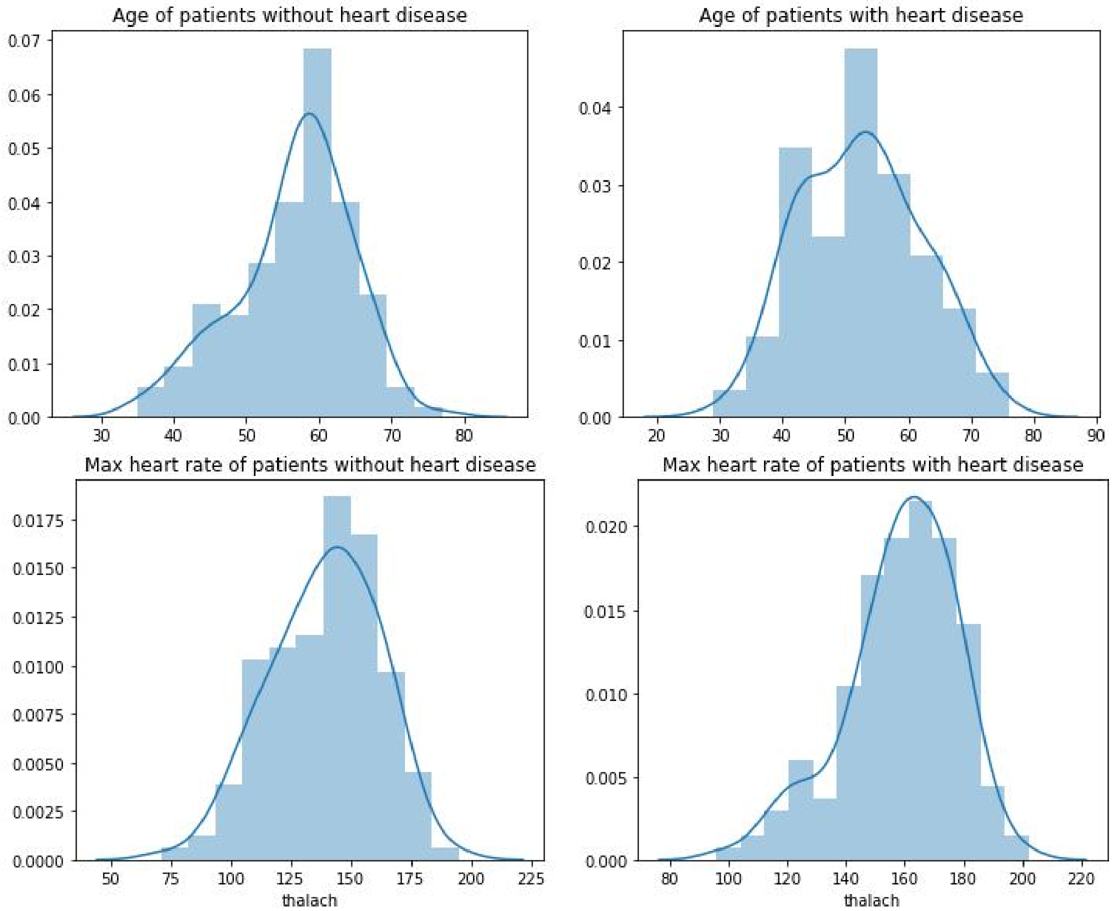
```
1    165
0    138
Name: target, dtype: int64
```



- sns.heatmap(heart[numeric_columns].corr(),annot=True, cmap='terrain', linewidths=0.1)
- fig=plt.gcf()
- fig.set_size_inches(8,6)
- plt.show()

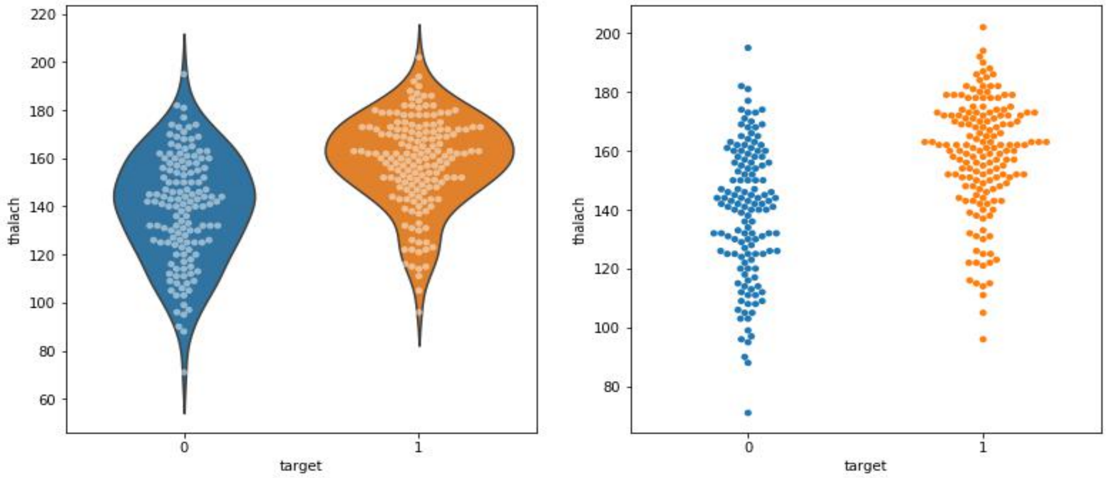


- # create four distplots
- plt.figure(figsize=(12,10))
- plt.subplot(221)
- sns.distplot(heart[heart['target']==0].age)
- plt.title('Age of patients without heart disease')
- plt.subplot(222)
- sns.distplot(heart[heart['target']==1].age)
- plt.title('Age of patients with heart disease')
- plt.subplot(223)
- sns.distplot(heart[heart['target']==0].thalach)
- plt.title('Max heart rate of patients without heart disease')
- plt.subplot(224)
- sns.distplot(heart[heart['target']==1].thalach)
- plt.title('Max heart rate of patients with heart disease')
- plt.show()

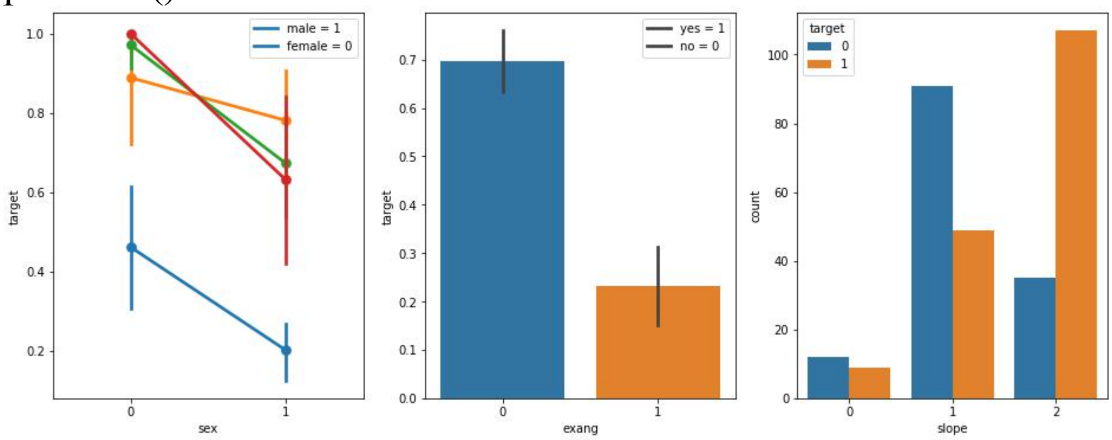


- plt.figure(figsize=(13,6))
- plt.subplot(121)

- sns.violinplot(x="target", y="thalach", data=heart, inner=None)
- sns.swarmplot(x="target", y="thalach", data=heart, color='w', alpha=0.5)
- plt.subplot(122)
- sns.swarmplot(x="target", y="thalach", data=heart)
- plt.show()



- # create pairplot and two barplots
- plt.figure(figsize=(16,6))
- plt.subplot(131)
- sns.pointplot(x="sex", y="target", hue='cp', data=heart)
- plt.legend(['male = 1', 'female = 0'])
- plt.subplot(132)
- sns.barplot(x="exang", y="target", data=heart)
- plt.legend(['yes = 1', 'no = 0'])
- plt.subplot(133)
- sns.countplot(x="slope", hue='target', data=heart)
- plt.show()



DATA Preprocessing

- heart['target'].value_counts()

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

- heart['target'].isnull()

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

- heart['target'].sum()

165

- heart['target'].unique()

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

- heart.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
```

- X,y=heart.loc[:,:'thal'],heart.loc[:,'target']
- X

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2

303 rows × 13 columns

- 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
```

- X.shape
  (303, 13)

- y.shape
  (303,)

- from sklearn.model_selection import train_test_split
- from sklearn.preprocessing import StandardScaler

- X=heart.drop(['target'],axis=1)

- X

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2

303 rows × 13 columns

- X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=10,test_size=0.3,shuffle=True)
- X_test

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
246	56	0	0	134	409	0	0	150	1	1.9	1	2	3
183	58	1	2	112	230	0	0	165	0	2.5	1	1	3
229	64	1	2	125	309	0	1	131	1	1.8	1	0	3
126	47	1	0	112	204	0	1	143	0	0.1	2	0	2
184	50	1	0	150	243	0	0	128	0	2.6	1	0	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...
69	62	0	0	124	209	0	1	163	0	0.0	2	0	2
21	44	1	2	130	233	0	1	179	1	0.4	2	0	2
210	57	1	2	128	229	0	0	150	0	0.4	1	1	3
78	52	1	1	128	205	1	1	184	0	0.0	2	0	2
174	60	1	0	130	206	0	0	132	1	2.4	1	2	3

91 rows × 13 columns

- y_test

```
246    0
183    0
229    0
126    1
184    0
      ..
69     1
21     1
210    0
78     1
174    0
Name: target, Length: 91, dtype: int64
```

- print ("train_set_x shape: " + str(X_train.shape))
- print ("train_set_y shape: " + str(y_train.shape))
- print ("test_set_x shape: " + str(X_test.shape))
- print ("test_set_y shape: " + str(y_test.shape))

  train_set_x shape: (212, 13)
  train_set_y shape: (212,)
  test_set_x shape: (91, 13)
  test_set_y shape: (91,)

Model

Decision Tree Classifier

- Catagory=['No....but i pray you dont get Heart Disease or at leaset Corona Virus Soon...','Yes you have Heart Disease....RIP in Advance']
- from sklearn.tree import DecisionTreeClassifier
- dt=DecisionTreeClassifier()
- dt.fit(X_train,y_train)

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes
                       =None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

- prediction=dt.predict(X_test)
- accuracy_dt=accuracy_score(y_test,prediction)*100
- accuracy_dt

  73.62637362637363

- print("Accuracy on training set: {:.3f}".format(dt.score(X_ train, y_train)))
- print("Accuracy on test set: {:.3f}".format(dt.score(X_test, y_test)))

Accuracy on training set: 1.000

Accuracy on test set: 0.736

- y_test

```
  246    0
183    0
229    0
126    1
184    0
      ..
69     1
21     1
210    0
78     1
174    0
Name: target, Length: 91, dtype: int64
```

- Prediction

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

- X_DT=np.array([[63 ,1, 3,145,233,1,0,150,0,2.3,0,0,1]])
- X_DT_prediction=dt.predict(X_DT)
- X_DT_prediction[0]

  1
- print(Catagory[int(X_DT_prediction[0])])

  Yes you have Heart Disease....RIP in Advance

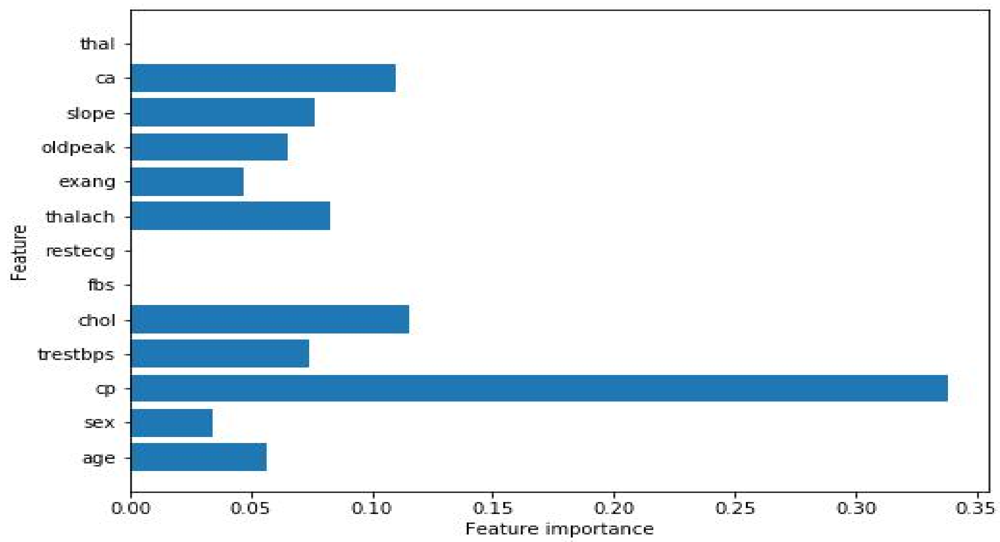
●Feature Importance in Decision Trees

- print("Feature importances:\n{}".format(dt.feature_importances_))

```
Feature importances:
[0.06177978 0.03461456 0.33832546 0.0909254  0.08967398 0.
 0.         0.07455742 0.04724994 0.05595161 0.0792335  0.11946876
 0.0082196 ]
```

- def plot_feature_importances_diabetes(model):

```
def plot_feature_importances_diabetes(model):
    plt.figure(figsize=(8,6))
    n_features = 13
    plt.barh(range(n_features), model.feature_importances_,
 align='center')
    plt.yticks(np.arange(n_features), X)
    plt.xlabel("Feature importance")
    plt.ylabel("Feature")
    plt.ylim(-1, n_features)
plot_feature_importances_diabetes(dt)
plt.savefig('feature_importance')
```



KNN

- sc=StandardScaler().fit(X_train)
- X_train_std=sc.transform(X_train)
- X_test_std=sc.transform(X_test)
- X_test_std

```
array([[ 0.18111199, -1.35154233, -0.97043553, ..., -0.6067969 ,
         1.33369489,  1.22676132],
       [ 0.39865161,  0.73989544,  0.97963397, ..., -0.6067969 ,
         0.33105902,  1.22676132],
       [ 1.05127045,  0.73989544,  0.97963397, ..., -0.6067969 ,
        -0.67157686,  1.22676132],
       ...,
       [ 0.2898818 ,  0.73989544,  0.97963397, ..., -0.6067969 ,
         0.33105902,  1.22676132],
       [-0.25396724,  0.73989544,  0.00459922, ...,  0.98136289,
        -0.67157686, -0.41927286],
       [ 0.61619122,  0.73989544, -0.97043553, ..., -0.6067969 ,
         1.33369489,  1.22676132]])
```

- from sklearn.neighbors import KNeighborsClassifier
- knn=KNeighborsClassifier(n_neighbors=4)
- knn.fit(X_train_std,y_train)

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=4, p=2,
                     weights='uniform')
```

- prediction_knn=knn.predict(X_test_std)
- accuracy_knn=accuracy_score(y_test,prediction_knn)*100
- accuracy_knn

  84.61538461538461

- print("Accuracy on training set: {:.3f}".format(knn.score(X_train, y_train)))
- print("Accuracy on test set: {:.3f}".format(knn.score(X_test, y_test)))

```
Accuracy on training set: 0.373
Accuracy on test set: 0.516
```

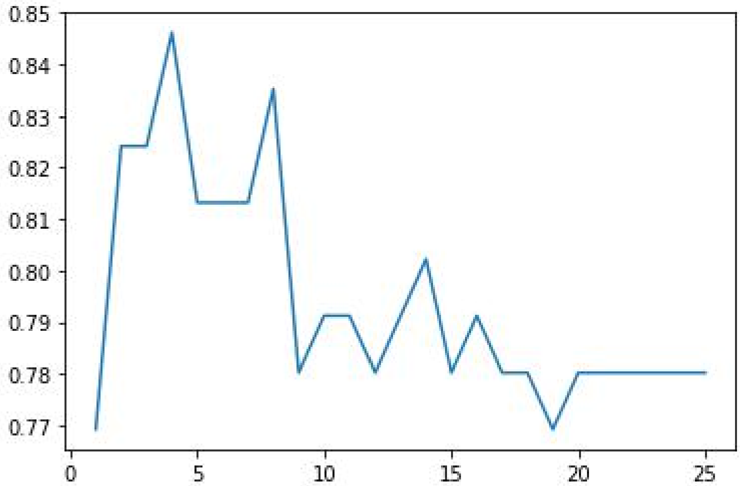
- k_range=range(1,26)

- scores={}
- scores_list=[]
-
- for k in k_range:
        knn=KNeighborsClassifier(n_neighbors=k)
        knn.fit(X_train_std,y_train)
        prediction_knn=knn.predict(X_test_std)
        scores[k]=accuracy_score(y_test,prediction_knn)
        scores_list.append(accuracy_score(y_test,prediction_kn
n))

- scores

```
{1: 0.7692307692307693,
 2: 0.8241758241758241,
 3: 0.8241758241758241,
 4: 0.8461538461538461,
 5: 0.8131868131868132,
 6: 0.8131868131868132,
 7: 0.8131868131868132,
 8: 0.8351648351648352,
 9: 0.7802197802197802,
 10: 0.7912087912087912,
 11: 0.7912087912087912,
 12: 0.7802197802197802,
 13: 0.7912087912087912,
 14: 0.8021978021978022,
 15: 0.7802197802197802,
 16: 0.7912087912087912,
 17: 0.7802197802197802,
 18: 0.7802197802197802,
 19: 0.7692307692307693,
 20: 0.7802197802197802,
 21: 0.7802197802197802,
 22: 0.7802197802197802,
 23: 0.7802197802197802,
 24: 0.7802197802197802,
 25: 0.7802197802197802}
```

- plt.plot(k_range,scores_list)



- px.line(x=k_range,y=scores_list)

- X_knn=np.array([[63 ,1, 3,145,233,1,0,150,0,2.3,0,0,1]])
- X_knn_std=sc.transform(X_knn)
- X_knn_prediction=dt.predict(X_knn)
- X_knn_std

```
array([[ 0.94250064,  0.73989544,  1.95466871,  0.75961822, -0.30064937,
         2.37170825, -0.9841849 ,  0.01848325, -0.6723502 ,  1.10653103,
        -2.1949567 , -0.67157686, -2.06530703]])
```

- (X_knn_prediction[0])

  1
- print(Catagory[int(X_knn_prediction[0])])

```
Yes you have Heart Disease....RIP in Advance
```

- algorithms=['Decision Tree','KNN']
- scores=[accuracy_dt,accuracy_knn]

- sns.set(rc={'figure.figsize':(15,7)})
- plt.xlabel("Algorithms")
- plt.ylabel("Accuracy score")
- sns.barplot(algorithms,scores)