**46**



**Data Analysis Project on Heart Disease Prediction**

Presented by:

Name:Pothuri Akish

Mail:akishpothuri@gmail.com

* **Table of Contents:**

|  |  |
| --- | --- |
| **1** | TABLE OF FIGURES |
| **2** | ABSTRACT |
| **3** | CHAPTERS |
| **4** | COMPLETE CODE  (attached ipynb file) |
| **5** | CONCLUSION |

* **Table Of Figures/Graphs**

1. *Heart Disease data*

*Total data of heart disease*

*Columns details in the data*

*Target of the project*

*Description of the data*

*Information of the data*

1. *Visualizing the data*

*Hist Graph of the target*

*Bar Graph of the data*

*Box Plot of the data*

*Distplot of the data*

*Pairplot of the data*

*Swarmplot of the data*

*Heatmap of the data*

1. *Preprocessing of the data*

*Preparing train and testdata*

*Training data*

*Testing data*

1. *Models*
2. *Decision Tree Classifier*

*Feature importance*

1. *K-Nearest Neighbour*

*Accuracy graph*

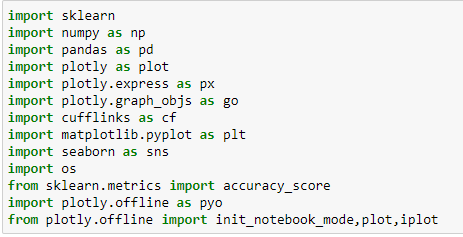
*Barplot plot*

* **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**



* 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](https://matplotlib.org/). It provides a high-level interface for drawing attractive and informative statistical graphics.

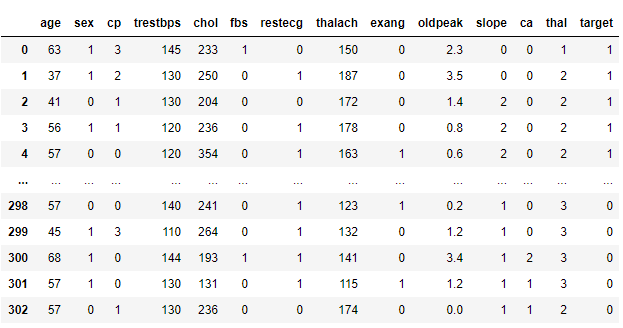
* import os

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

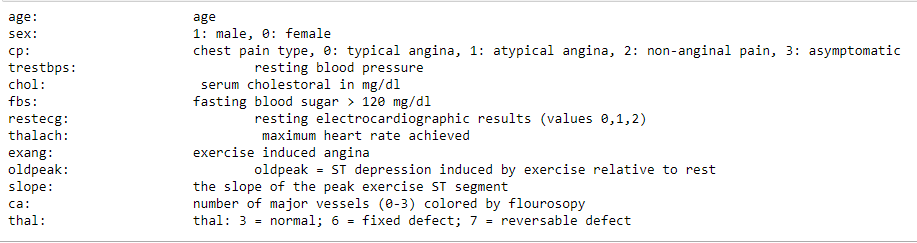
* heart=pd.read\_csv(r'G:\datascience\_internship\project\heart.csv')
* heart



With pd.read\_csv we read the dataset in the local machine and do operations on it.CSV represents the ‘comma separated values’

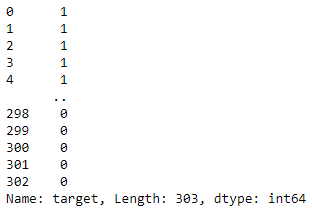
* info = ["age","1: male, 0: female","chest pain type, 0: typical angina, 1: atypical angina, 2: non-anginal pain, 3: asymptomatic","resting blood pressure"," serum cholestoral in mg/dl","fasting blood sugar > 120 mg/dl","resting electrocardiographic results (values 0,1,2)"," maximum heart rate achieved","exercise 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])

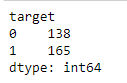


In the info object information of the dataset was stored as list.

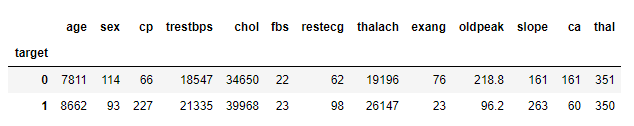
* heart['target']



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



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



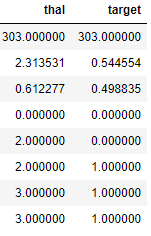
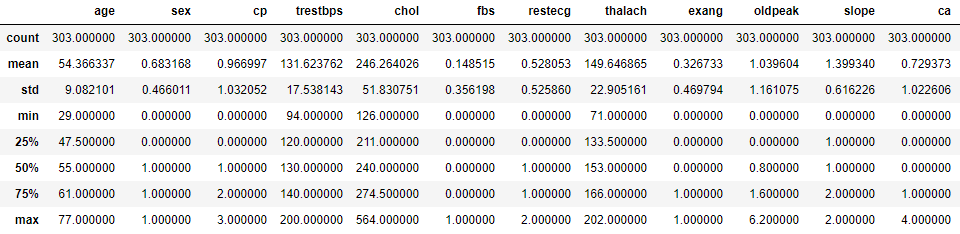
* heart.shape

(303, 14)

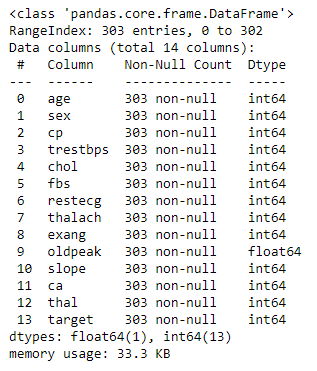
* heart.size

4242

* heart.describe()



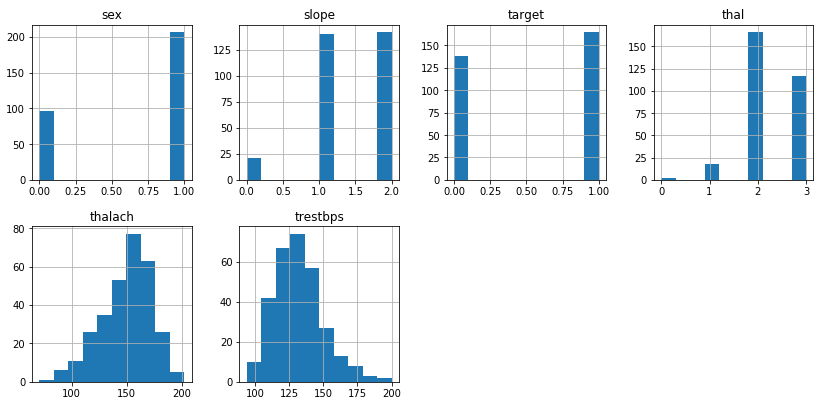
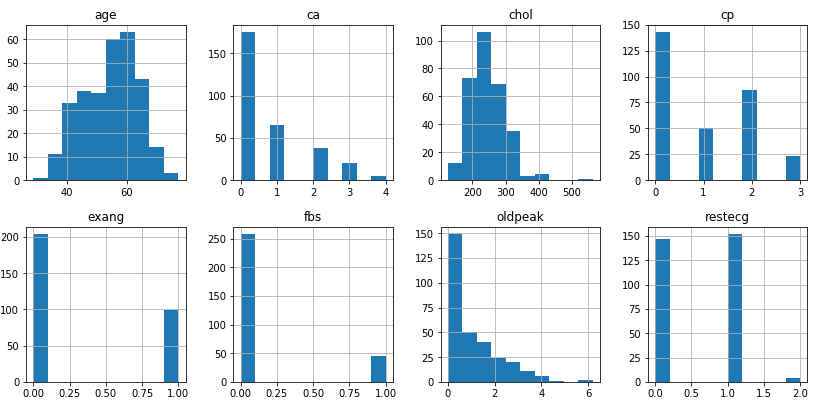
* heart.info()

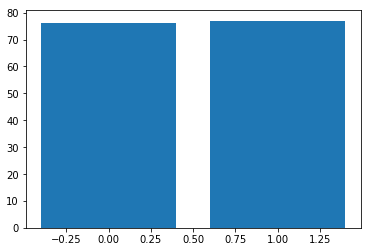


* heart['target'].unique()

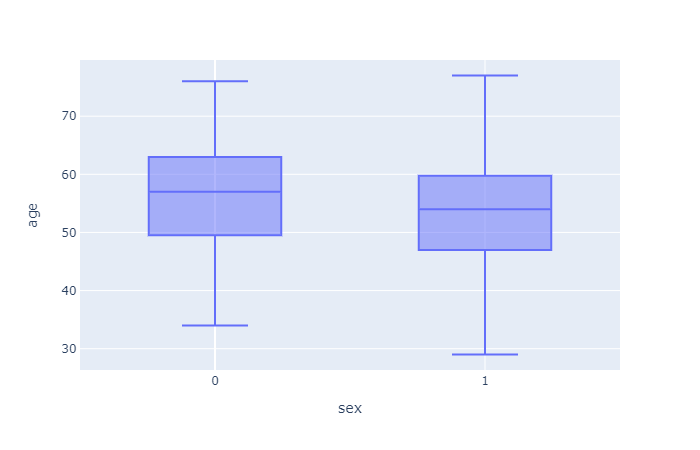
array([1, 0], dtype=int64)

**Visualization**

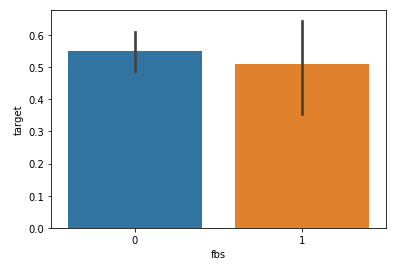
* heart.hist(figsize=(14,14))
* plt.show() 
* plt.bar(x=heart['sex'],height=heart['age'])
* plt.show()



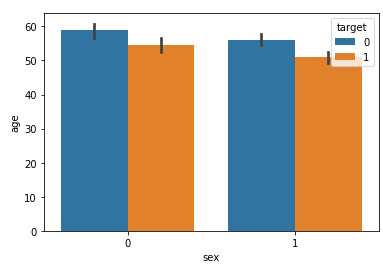
* px.box(heart,heart['sex'],heart['age'])



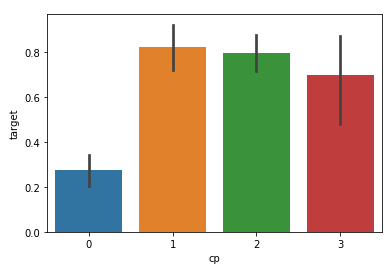
* sns.barplot(x="fbs", y="target", data=heart)
* plt.show()



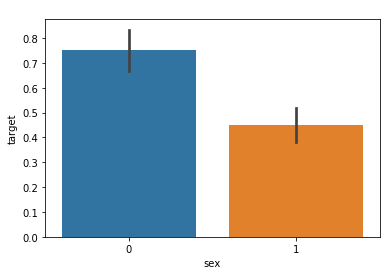
* sns.barplot(x=heart['sex'],y=heart['age'],hue=heart['target'])



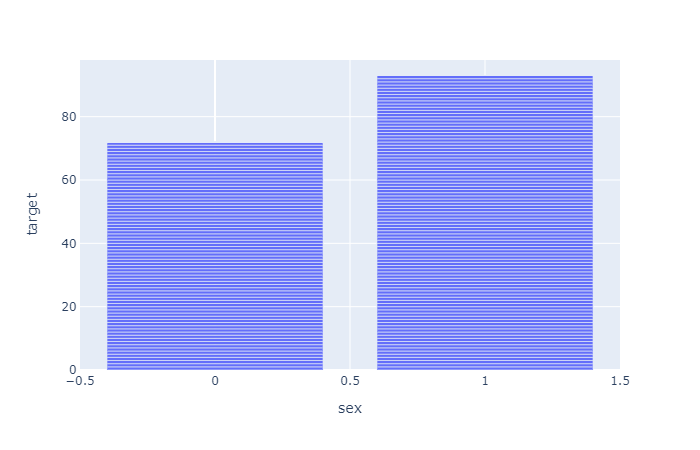
* sns.barplot(heart["cp"],heart['target'])



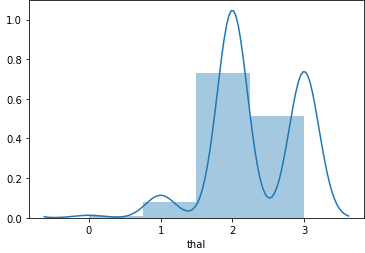
* sns.barplot(heart["sex"],heart['target'])



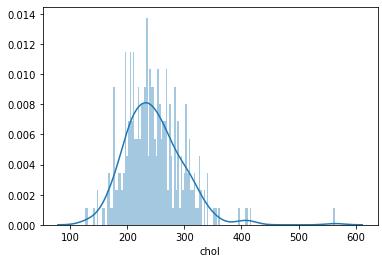
* px.bar(heart,heart['sex'],heart['target'])



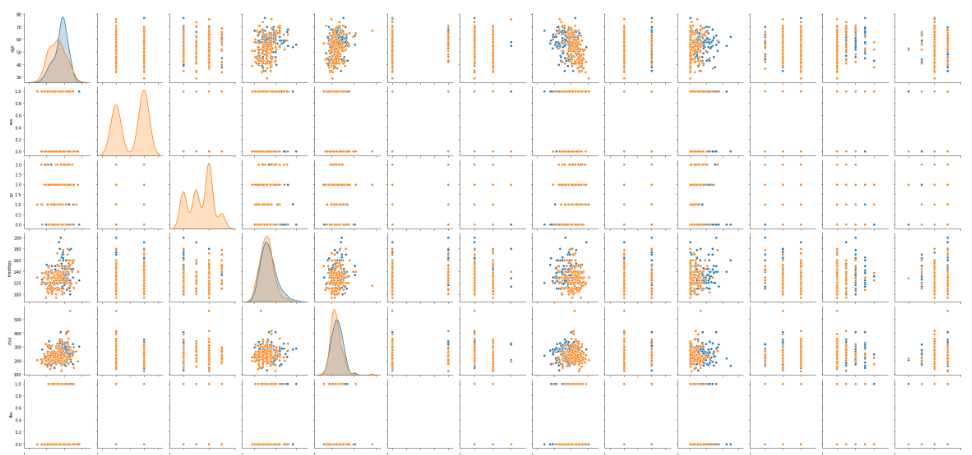
* sns.distplot(heart["thal"],bins=4)

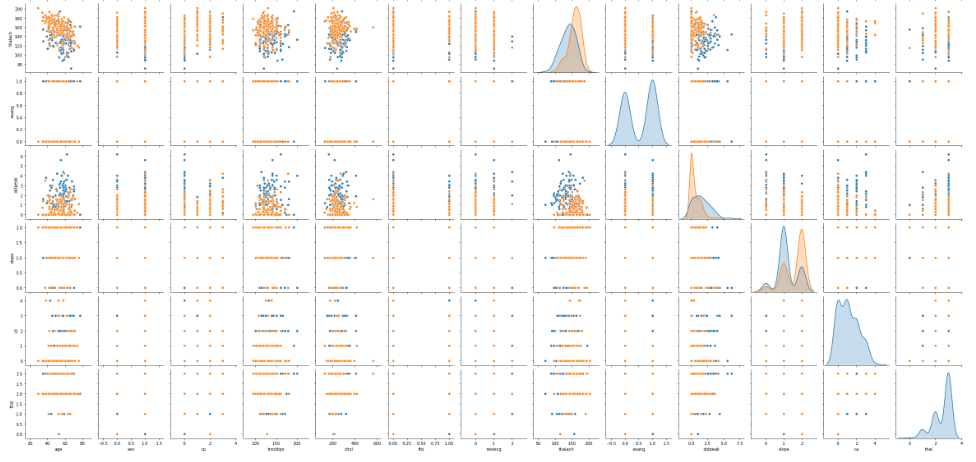


* sns.distplot(heart["chol"],bins=152)

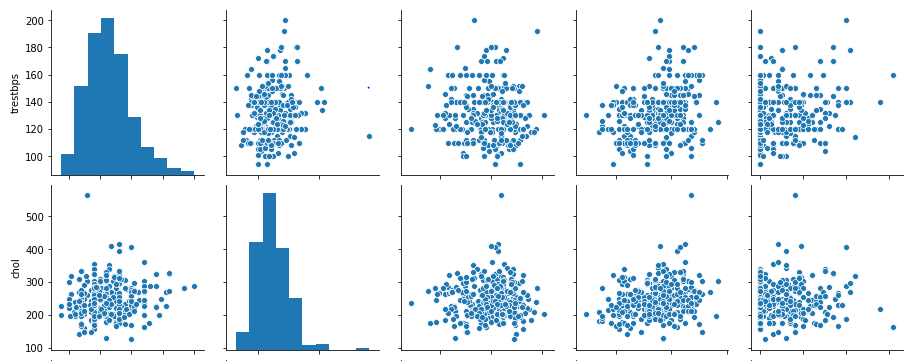


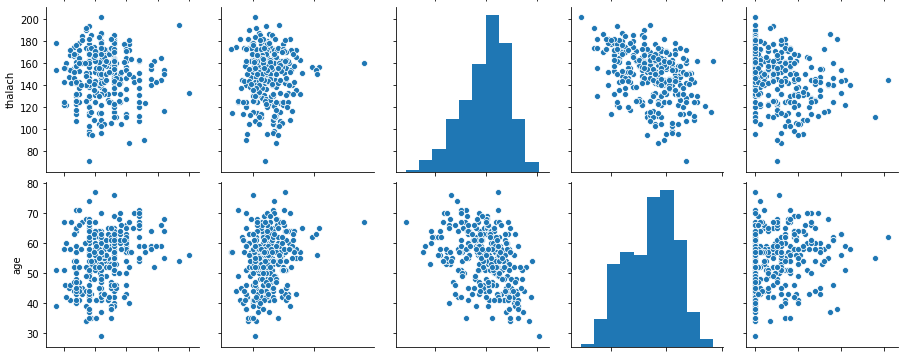
* sns.pairplot(heart,hue='target')

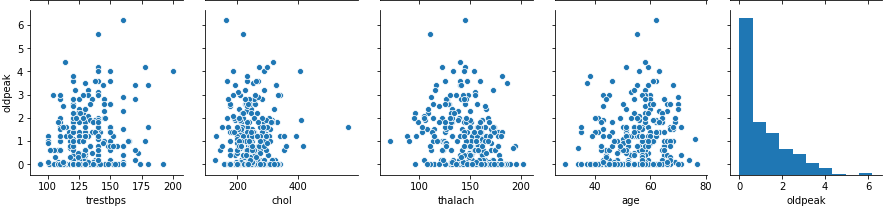




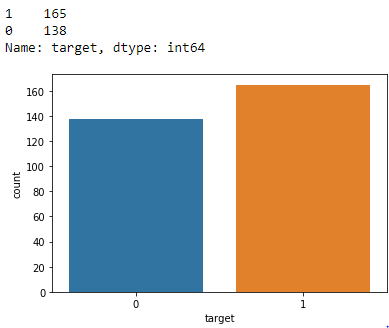
* numeric\_columns=['trestbps','chol','thalach','age','oldpeak']
* sns.pairplot(heart[numeric\_columns])



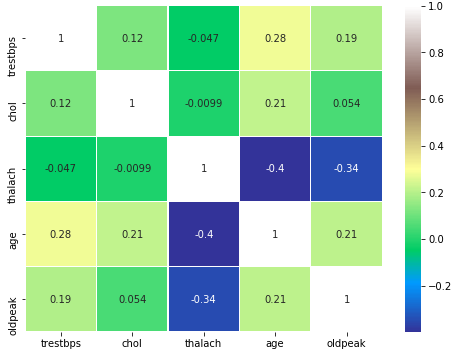




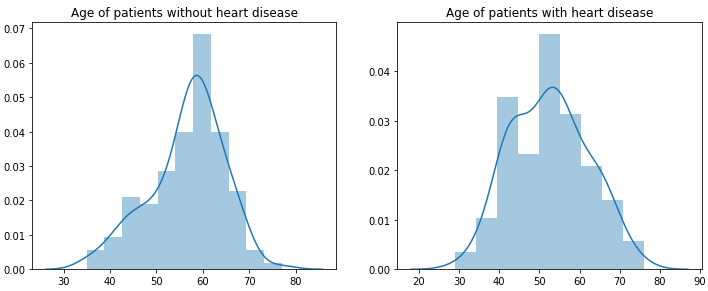
* y = heart["target"]
* sns.countplot(y)
* target\_temp = heart.target.value\_counts()
* print(target\_temp)

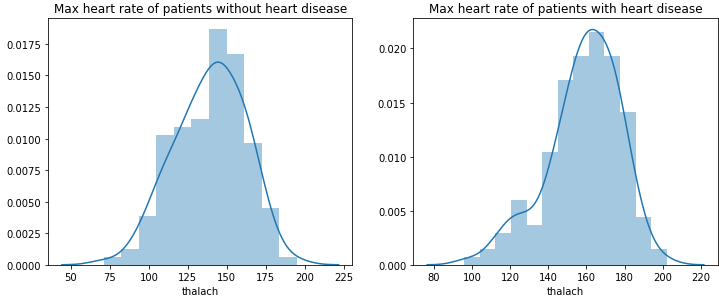


* sns.heatmap(heart[numeric\_columns].corr(),annot=True, cmap='terrain', linewidths=0.1)
* fig=plt.gcf()
* fig.set\_size\_inches(8,6)
* plt.show()

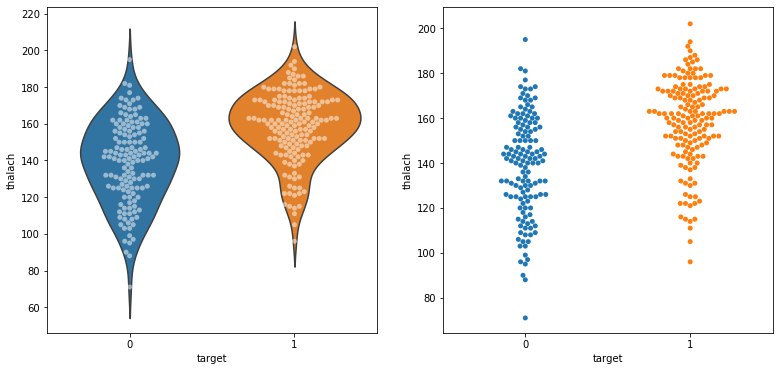


* # create four distplots
* plt.figure(figsize=(12,10))
* plt.subplot(221)
* sns.distplot(heart[heart['target']==0].age)
* plt.title('Age of patients without heart disease')
* plt.subplot(222)
* sns.distplot(heart[heart['target']==1].age)
* plt.title('Age of patients with heart disease')
* plt.subplot(223)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

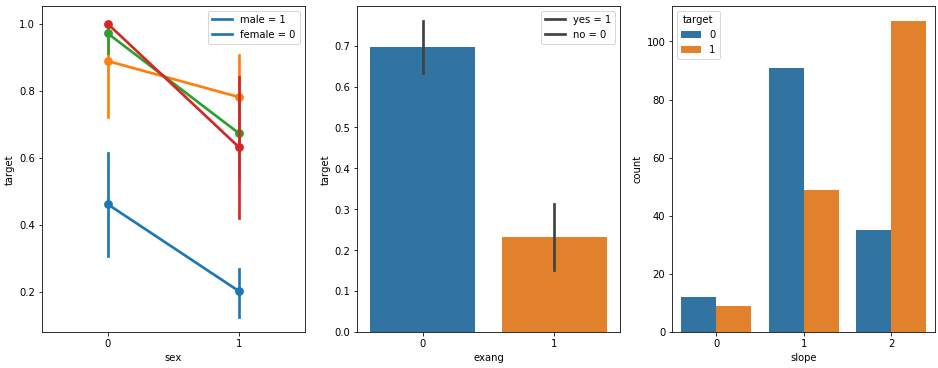




* plt.figure(figsize=(13,6))
* plt.subplot(121)
* sns.violinplot(x="target", y="thalach", data=heart, inner=None)
* sns.swarmplot(x="target", y="thalach", data=heart, color='w', alpha=0.5)
* plt.subplot(122)
* sns.swarmplot(x="target", y="thalach", data=heart)
* plt.show()



* # 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

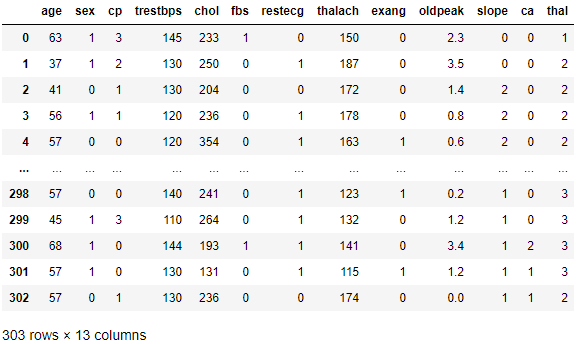
ca 0

thal 0

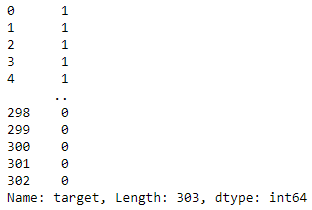
target 0

dtype: int64

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



* Y



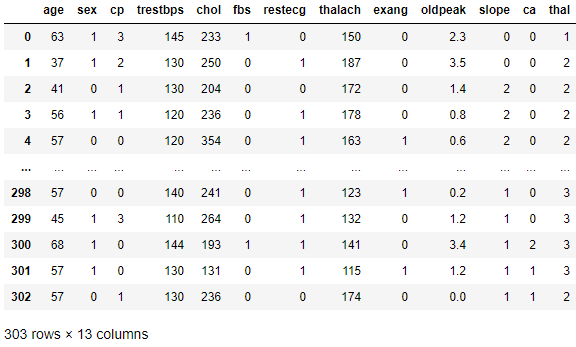
* X.shape

(303, 13)

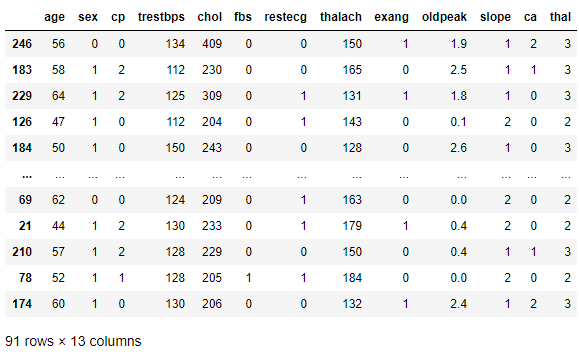
* y.shape

(303,)

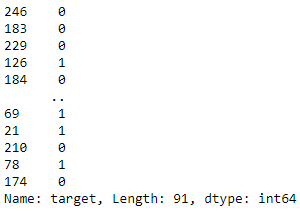
* from sklearn.model\_selection import train\_test\_split
* from sklearn.preprocessing import StandardScaler
* X=heart.drop(['target'],axis=1)
* X



* X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,random\_state=10,test\_size=0.3,shuffle=True)
* X\_test



* y\_test



* 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):

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)

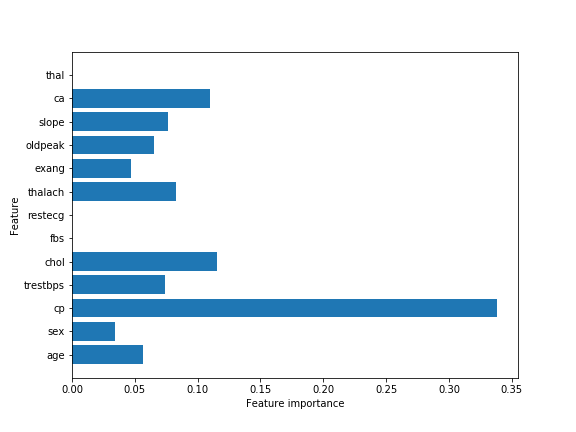
plt.xlabel("Feature importance")

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)

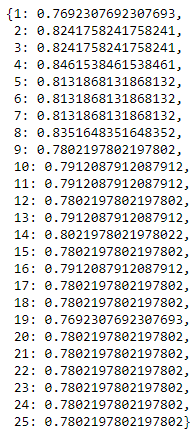
knn.fit(X\_train\_std,y\_train)

prediction\_knn=knn.predict(X\_test\_std)

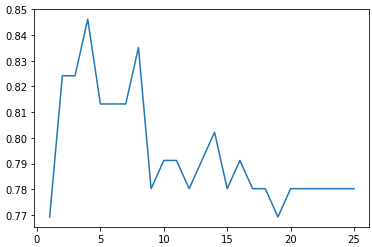
scores[k]=accuracy\_score(y\_test,prediction\_knn)

scores\_list.append(accuracy\_score(y\_test,prediction\_knn))

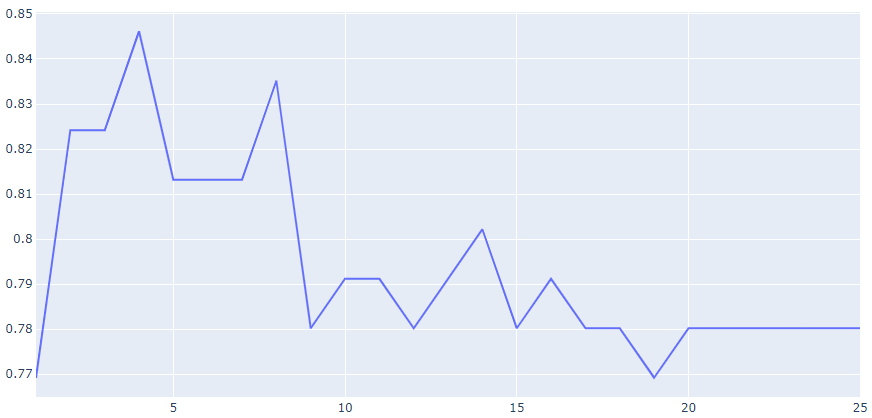
* scores



* 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



* (X\_knn\_prediction[0])

1

* print(Catagory[int(X\_knn\_prediction[0])])



* 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)

