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

pyo.init\_notebook\_mode(connected=True)

cf.go\_offline()

heart=pd.read\_csv(r'E:\DS\Heart-Disease\heart.csv')

heart

info = ["age","1: male, 0: female","chest pain type, 1: typical angina, 2: atypical angina, 3: non-anginal pain, 4: 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])

heart['target']

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

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

heart.shape

heart.size

heart.describe()

heart.info()

heart['target'].unique()

heart.hist(figsize=(14,14))

plt.show()

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

plt.show()

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

plt.show()

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

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

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

sns.distplot(heart["thal"])

sns.distplot(heart["chol"])

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

numeric\_columns=['trestbps','chol','thalach','age','oldpeak']

heart['target']

y = heart["target"]

sns.countplot(y)

target\_temp = heart.target.value\_counts()

print(target\_temp)

# create a correlation heatmap

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()

heart['target'].value\_counts()

heart['target'].isnull()

heart['target'].sum()

heart['target'].unique()

heart.isnull().sum()

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

X

X.shape

y.shape

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))

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)

prediction=dt.predict(X\_test)

accuracy\_dt=accuracy\_score(y\_test,prediction)\*100

accuracy\_dt

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)))

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]

print(Catagory[int(X\_DT\_prediction[0])])

print("Feature importances:\n{}".format(dt.feature\_importances\_))

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')

sc=StandardScaler().fit(X\_train)

X\_train\_std=sc.transform(X\_train)

X\_test\_std=sc.transform(X\_test)

X\_test\_std

from sklearn.neighbors import KNeighborsClassifier

knn=KNeighborsClassifier(n\_neighbors=4)

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

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

accuracy\_knn=accuracy\_score(y\_test,prediction\_knn)\*100

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)))

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])

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)