```
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'].sum()

heart['target'].unique()

heart.isnull()

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)
```