**ADMISSION PREDICT**

import pandas as pd

df=pd.read\_csv('Admission\_Predict.csv')

df.columns

df.columns=df.columns.str.rstrip()

df.loc[df['Chance of Admit']>=0.80,'Chance of Admit']=1

df.loc[df['Chance of Admit']<0.80,'Chance of Admit']=0

df['Chance of Admit']

df.columns

df=df.drop('Serial No.',axis=1)

df

x=df.iloc[:,0:7]

y=df.iloc[:,7]

y

from sklearn.model\_selection import train\_test\_split

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.25,random\_state=0)

from sklearn.tree import DecisionTreeClassifier

model=DecisionTreeClassifier(criterion="entropy",max\_depth=4)

model.fit(x\_train,y\_train)

y\_pred=model.predict(x\_test)

y\_pred

from sklearn.metrics import confusion\_matrix

matrix=confusion\_matrix(y\_test,y\_pred,labels=[0.0,1,0])

x\_train.shape

x\_test.shape

matrix

from sklearn.metrics import accuracy\_score

acc=accuracy\_score(y\_test,y\_pred)

print(acc)

from sklearn.metrics import classification\_report

cr=classification\_report(y\_test,y\_pred)

print(cr)

**HEART**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read\_csv('Heart.csv')

print(df)

df.shape #gives the number of rows and columns

df.head() #gives first 5 rows

df.dtypes #gives the datatypes of columns

df.isnull().sum() #counts the no. of missing values in each row or column in a dataframe

(df==0).sum() #count number of zero values in a dataframe

df['Age'].mean()

df.columns #display the nmaes of columns

df.info #gives overall info about the dataset

from sklearn.model\_selection import train\_test\_split

hd = df[['Age','Sex','ChestPain','RestBP','Chol']]

x=hd.drop('Chol', axis=1)

y=hd['Chol']

print(x)

print(y)

#Now lets split the dataset

xtrain,xtest,ytrain,ytest=train\_test\_split(x,y,test\_size=0.25)

xtrain.shape

#here xtest ytest xtrain and y train are 4 tuples in numpy arrays

#here xtrain and xtest stes of independent variables and

#here ytarain and ytest are the sets of dependent variables

ytest.shape

df['Sex']=df['Sex'].replace([1], 'Male')

df['Sex']=df['Sex'].replace([0], 'Female')

print(df)

dfl = df[df['AHD'].values == 'Yes']

print(dfl)

print(dfl.Age.max())

print(dfl.Age.min())

df.groupby('AHD').AHD.count().plot.bar(ylim=0)

plt.plot()

[]

df.plot(kind='box',subplots=True,layout=(2,7),sharex=False,

figsize=(20,10),color='blue')

df['Age'].hist(figsize=(10,13))

plt.title('Age Histogram')

Text(0.5, 1.0, 'Age Histogram')

df.hist(figsize=(12,15))

plt.show()

pd.crosstab(df.Age, df.AHD).plot(kind='bar',figsize=(12,15))

plt.title('Barchart for Age V AHD')

Text(0.5, 1.0, 'Barchart for Age V AHD')

**MALL CUSTOMERS**

import warnings

warnings.filterwarnings('ignore')

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

df=pd.read\_csv("Mall\_Customers.csv")

df.head()

df.info()

plt.style.use("fivethirtyeight")

plt.figure(1,(15,8))

n=0

for x in ['Age','Annual Income (k$)','Spending Score (1-100)']:

n+=1

plt.subplot(1,3,n)

plt.subplots\_adjust (hspace=0.5,wspace=0.5)

sns.distplot(df[x],bins=40)

plt.show()

plt.figure(1,(15,5))

sns.countplot(y="Gender",data=df)

plt.show()

plt.style.use("fivethirtyeight")

plt.figure(1,(15,7))

n=0

for x in['Age','Annual Income (k$)','Spending Score (1-100)']:

for y in['Age','Annual Income (k$)','Spending Score (1-100)']:

n+=1

plt.subplot(3,3,n)

plt.subplots\_adjust(hspace=0.5,wspace=0.5)

sns.regplot(x=x,y=y,data=df)

plt.show()

df.isnull().sum()

df.keys()

plt.scatter(df['Annual Income (k$)'], df['Spending Score (1-100)'])

X=df.iloc[:,[3,4]].values

from sklearn.cluster import KMeans

kmeans = KMeans(n\_clusters= 5,init= 'k-means++',random\_state = 42)

y\_kmeans = kmeans.fit\_predict(X)

cluster=[]

for k in range(1,11):

kmean=KMeans(n\_clusters=k).fit(X)

cluster.append(kmean.inertia\_)

plt.figure(figsize=(12,8))

plt.plot(range(1,11),cluster,'r-')

plt.xlabel('Inertia')

plt.ylabel('n\_cluster')

plt.show()

import matplotlib.pyplot as plt

import pandas as pd

dataset=pd.read\_csv("Mall\_Customers.csv")

dataset

X=dataset.iloc[:,[3,4]].values

import scipy.cluster.hierarchy as sch

dendrogram=sch.dendrogram(sch.linkage(X,method='ward'))

plt.title("Dendrogram")

plt.xlabel("Customers")

plt.ylabel("Euclidean Distance")

plt.tick\_params(axis='x',labelbottom=False)

plt.show()

from sklearn.cluster import AgglomerativeClustering

hc=AgglomerativeClustering(n\_clusters=5,affinity="euclidean",linkage='ward')

y\_pred = hc.fit\_predict(X)

plt.scatter(X[y\_pred ==0,0],X[y\_pred==0,1],s=100,c='red',label='c1')

plt.scatter(X[y\_pred ==1,0],X[y\_pred==1,1],s=100,c='blue',label='c2')

plt.scatter(X[y\_pred ==2,0],X[y\_pred==2,1],s=100,c='green',label='c3')

plt.scatter(X[y\_pred ==3,0],X[y\_pred==3,1],s=100,c='yellow',label='c4')

plt.scatter(X[y\_pred ==4,0],X[y\_pred==4,1],s=100,c='black',label='c5')

plt.title('Clusters of cudtomers')

plt.xlabel('Annual Income')

plt.ylabel('Spending Score')

plt.legend()

plt.show()

plt.rcParams['figure.figsize']=(18,8)

plt.subplot(1,2,1)

sns.set(style='whitegrid')

sns.distplot(data['Annual Income (k$)'])

plt.title('Distribution of Annual Income',fontsize=20)

plt.xlabel('Range of Annual Income')

plt.ylabel('Count')

plt.subplot(1,2,2)

sns.set(style='whitegrid')

sns.distplot(data['Age'],color='red')

plt.title('Distribution of Age',fontsize=20)

plt.xlabel('Range of Age')

plt.ylabel('Count')

plt.show()

plt.rcParams['figure.figsize']=(15,8)

sns.countplot(data['Age'],palette='hsv')

plt.title('Distribution of Age',fontsize=20)

plt.show()

plt.rcParams['figure.figsize']=(20,8)

sns.countplot(data['Annual Income (k$)'],palette='rainbow')

plt.title('Distribution of Annual Income',fontsize=20)

plt.show()

plt.rcParams['figure.figsize']=(20,8)

sns.countplot(data['Spending Score (1-100)'],palette='copper')

plt.title('Distribution of Spending Score',fontsize=20)

plt.show()

sns.pairplot(data)

plt.title('Pairplot for the Data',fontsize=20)

plt.show()

x=data['Annual Income (k$)']

y=data['Age']

z=data['Spending Score (1-100)']

sns.lineplot(x,y,color='blue')

sns.lineplot(x,z,color='pink')

plt.title('Annual Income vs Age and Spending Score',fontsize=20)

plt.show()

**WEIGHT-HEIGHT**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from operator import mul

import math

def compute\_regcoef(x,y):

n = len(x)

sumxy = sum(list(map(mul,x,y)))

num = n\* sumxy - sum (x)\*sum(y)

sumxx = sum(list(map(mul,x,x)))

denom = n\*sumxx -sum(x)\*\*2

m = num/denom

c =(sum(y)-m\*sum(x))/n

return(c,m)

def plot\_regline(x,y,b):

plt.scatter(x,y,color='b',marker='o',s=80)

y\_pred = np.float\_(x)\*b[1] + b[0]

plt.plot(x,y\_pred,color='g')

plt.xlabel('X-variable')

plt.ylabel('Y-variable')

plt.show()

x= [1,2,3,4,5,6,7,8,9]

y= [1,2,3,4,5,6,7,8,9]

b= compute\_regcoef(x,y)

type(b)

print('intercept',b[0])

print('slope',b[1])

plot\_regline(x,y,b)

x= [1,2,3,4,5,6,7,8,9]

y= [1,2,3,7,7,8,9,9,9]

b= compute\_regcoef(x,y)

type(b)

print('intercept',b[0])

print('slope',b[1])

plot\_regline(x,y,b)

df=pd.read\_csv('weight-height.csv')

print(df)

df.columns

df.dtypes

x=df.iloc[:,1:2].values

y=df.iloc[:,2].values

b=compute\_regcoef(x,y)

type(b)

print('intercept',b[0])

print('slope',b[1])

plot\_regline(x,y,b)

from sklearn.model\_selection import train\_test\_split

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.25,random\_state=0)

from sklearn.linear\_model import LinearRegression

regression= LinearRegression()

regression.fit(x\_train,y\_train)

LinearRegression()

m=regression.coef\_

print('Regression Coefficient / slope of regression line',m)

c= regression.intercept\_

print('Intercept',c)

y\_pred = regression.predict(x\_test)

print(y\_pred)

x\_test[0]\*m+c

plt.scatter(x\_test, y\_test)

plt.plot(x\_test,y\_pred,color='r')

plt.show()

from sklearn import metrics

print('Mean Absolute Error',metrics.mean\_absolute\_error(y\_test,y\_pred))

print('Mean Squared Error',metrics.mean\_squared\_error(y\_test,y\_pred))

print('Root Mean Squared Error',np.sqrt(metrics.mean\_absolute\_error(y\_test,y\_pred)))

dfl = pd.DataFrame({'Actual value':y\_test.flatten(),'Predicted Value':y\_pred.flatten()})

dfl