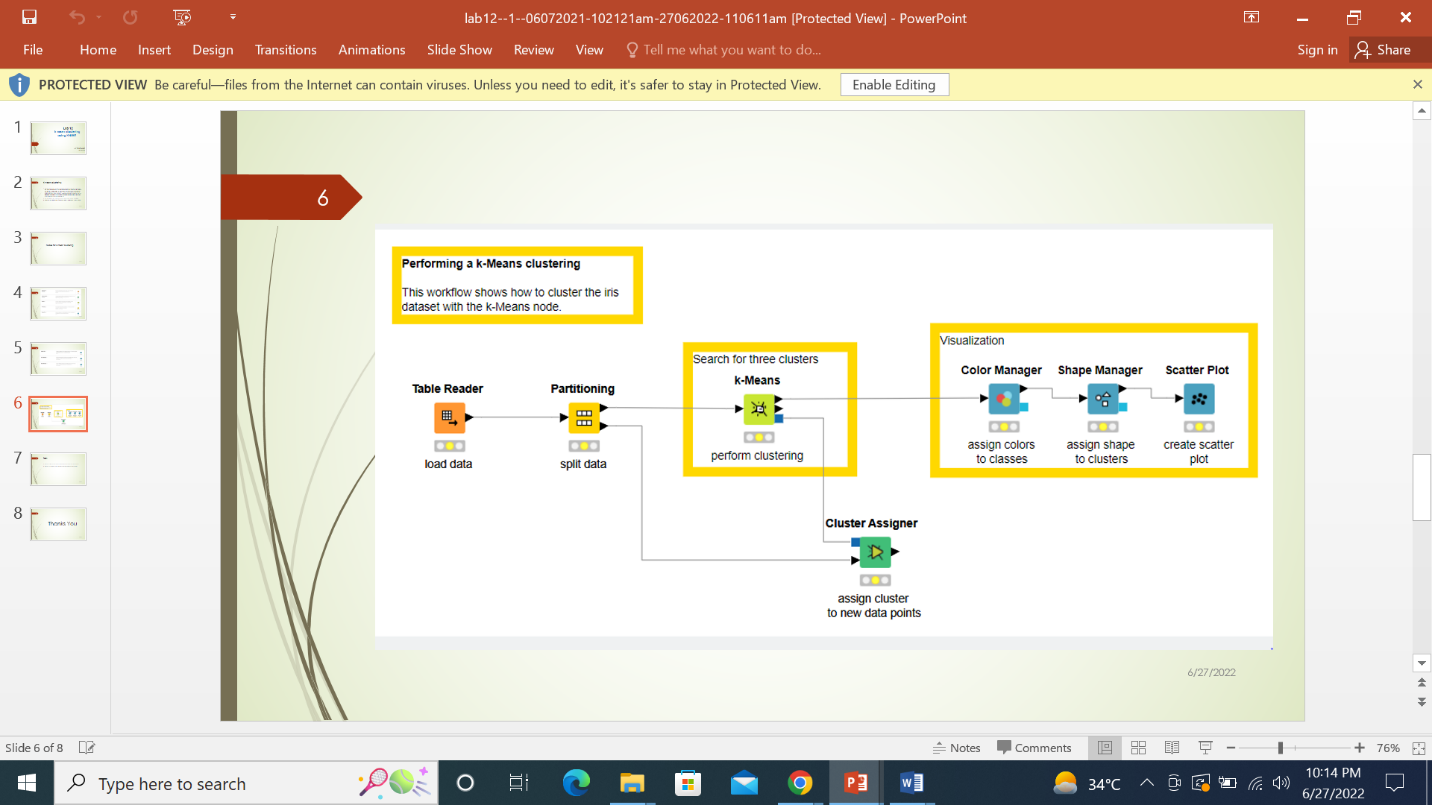
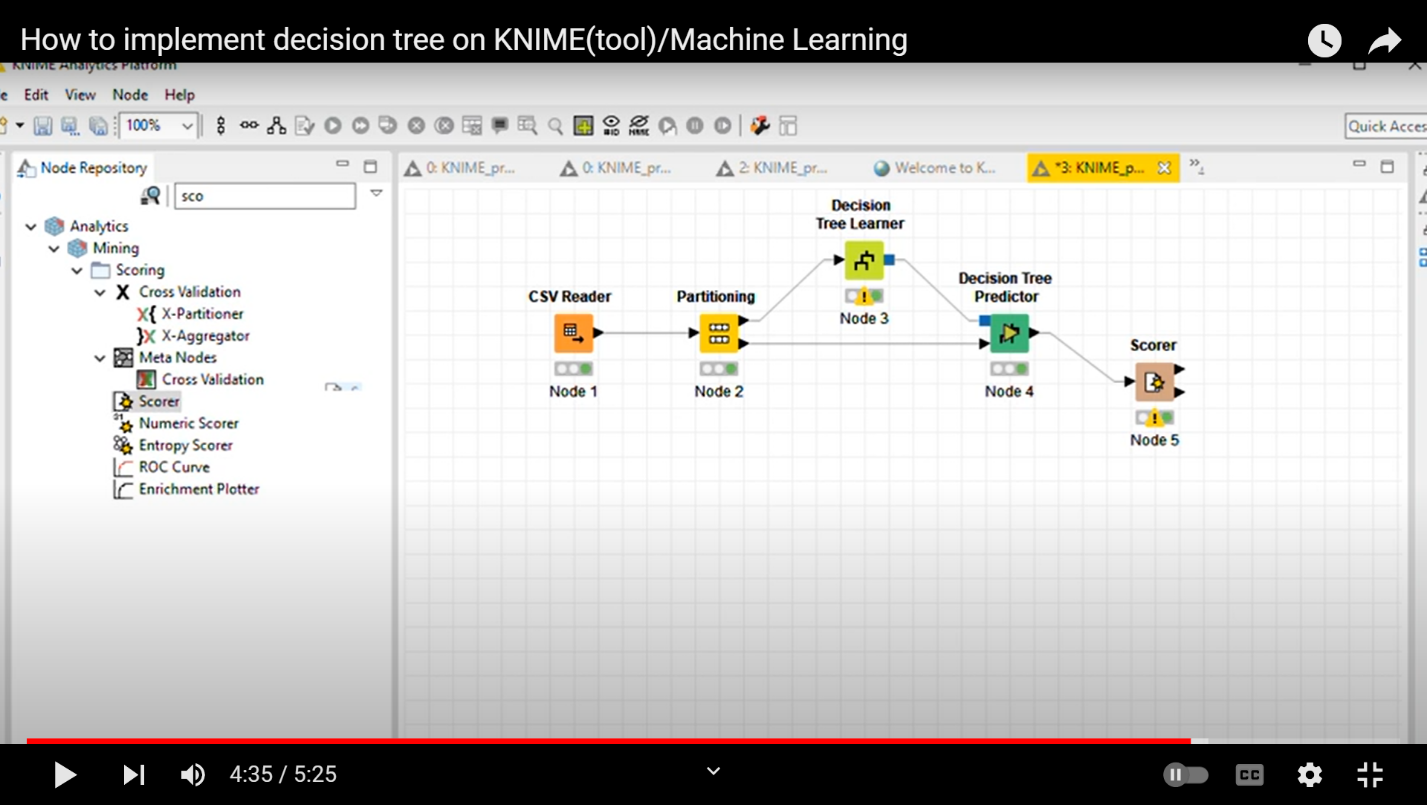
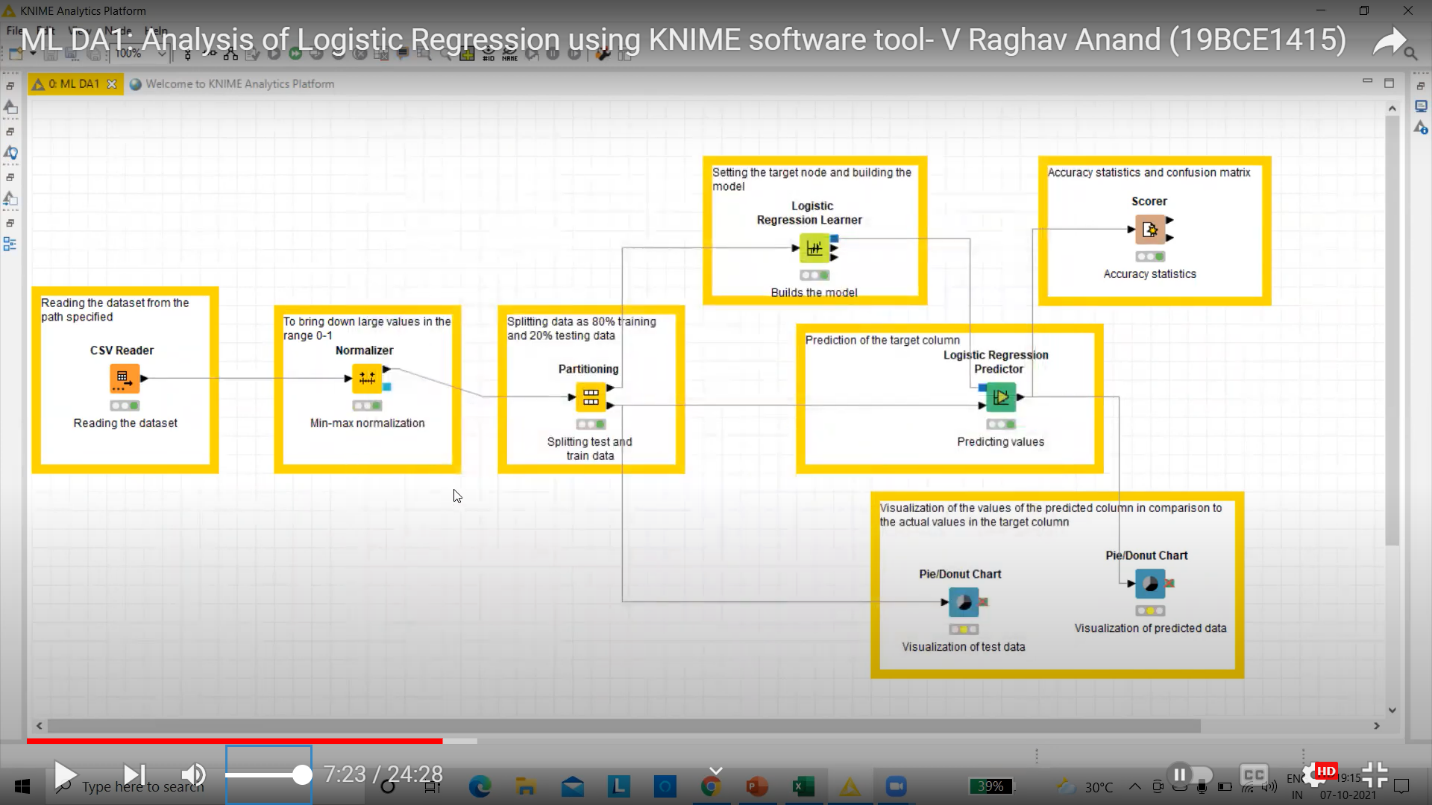
k-mean



Decision-tree



logistic regressiojn



**NUERAL NETWORK:**

import numpy as np

import pandas as pd

import matplotlib.pyplot

from google.colab import drive

drive.mount("/content/gdrive",force\_remount=True)

data=pd.read\_csv("/content/gdrive/My Drive/dataset/heart.csv")

data.head()

dataset.count()

X = pd.DataFrame(dataset.iloc[:, 0:10].values)

y = dataset.iloc[:, 13].values

print(X)

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

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

X\_test = sc.transform(X\_test)

print(X\_train)

import keras

from keras.models import Sequential

from keras.layers import Dense

classifier = Sequential()

classifier.add(Dense(6, activation = 'relu', input\_dim = 10))

classifier.add(Dense(6, activation = 'relu'))

classifier.add(Dense(1, activation = 'sigmoid'))

classifier.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

classifier.fit(X\_train, y\_train, batch\_size = 10, epochs = 100)

y\_pred = classifier.predict(X\_test)

y\_pred = (y\_pred > 0.5)

from sklearn.metrics import confusion\_matrix, accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

print("CONFUSION MATRIX")

print(cm)

print("ACCURACY")

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

K MEAN ALGORITHM:

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset= pd.read\_csv('/content/Mall\_Customers.csv')

X=dataset.iloc[:, [3,4]].values

from sklearn.cluster import KMeans

wcss=[]

for i in range(1,11):

     kmeans = KMeans(n\_clusters=i, init ='k-means++', max\_iter=300,  n\_init=10,random\_state=0 )

kmeans.fit(X)

wcss.append(kmeans.inertia\_)

plt.plot(range(1,11),wcss)

plt.title('The Elbow Method Graph')

plt.xlabel('Number of clusters')

plt.ylabel('WCSS')

plt.show()

# According to the Elbow graph we deterrmine the clusters number as 5. Applying k-means algorithm to the X dataset.

kmeans = KMeans(n\_clusters=5, init ='kmeans++', max\_iter=300, n\_init=10,random\_state=0 )

y\_kmeans = kmeans.fit\_predict(X)

# Visualising the clusters

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

plt.scatter(X[y\_kmeans==1, 0], X[y\_kmeans==1, 1], s=100, c='blue', label ='Cluster 2')

plt.scatter(X[y\_kmeans==2, 0], X[y\_kmeans==2, 1], s=100, c='green', label ='Cluster 3')

plt.scatter(X[y\_kmeans==3, 0], X[y\_kmeans==3, 1], s=100, c='cyan', label ='Cluster 4')

plt.scatter(X[y\_kmeans==4, 0], X[y\_kmeans==4, 1], s=100, c='magenta', label ='Cluster 5')

#Plot the centroid. This time we're going to use the cluster centres  #attribute that returns here the coordinates of the centroid.

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s=300, c='yellow', label = 'Centroids')

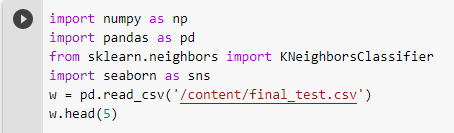
plt.title('Clusters of Customers')

plt.xlabel('Annual Income(k$)')

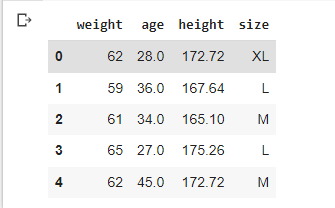
plt.ylabel('Spending Score(1-100')

plt.show()

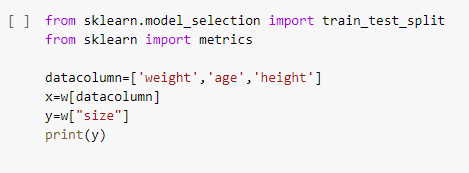
**Solution**



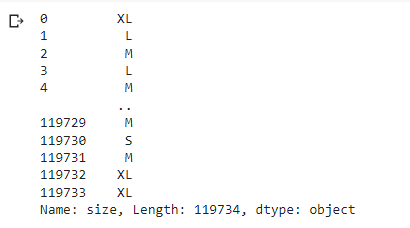
**Output**



**Solution:**

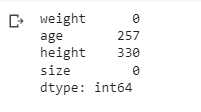


**Output:**



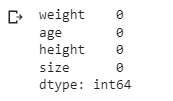
**Solution:**



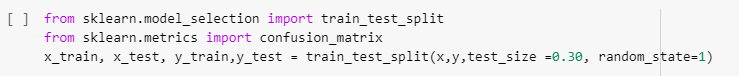


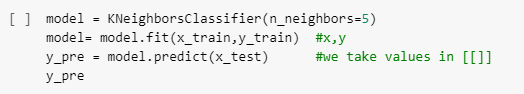
**Solution:**





**Solution:**





**Output:**

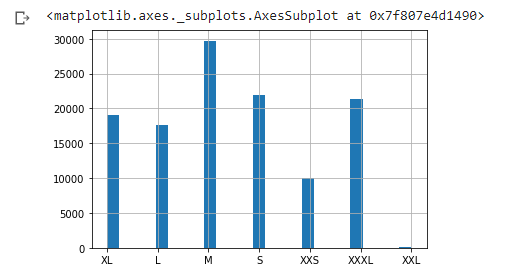


**Solution:**





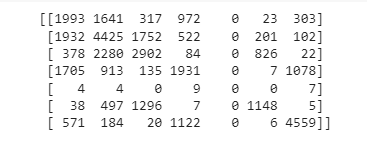
**Output:**



**Solution:**



**Output:**



**Solution:**



**Output:**

