**7) Demonstrate Decision tree classification model and Evaluate the performance of classifier on Iris dataset**

import pandas as pd

import numpy as np

from sklearn.datasets import load\_iris

from sklearn.metrics import accuracy\_score

# Reading the Iris.csv file

data = load\_iris()

# Extracting Attributes / Features

X = data.data

# Extracting Target / Class Labels

y = data.target

# Import Library for splitting data

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

# Creating Train and Test datasets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y, random\_state = 50, test\_size = 0.25)

# Creating Decision Tree Classifier

from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier()

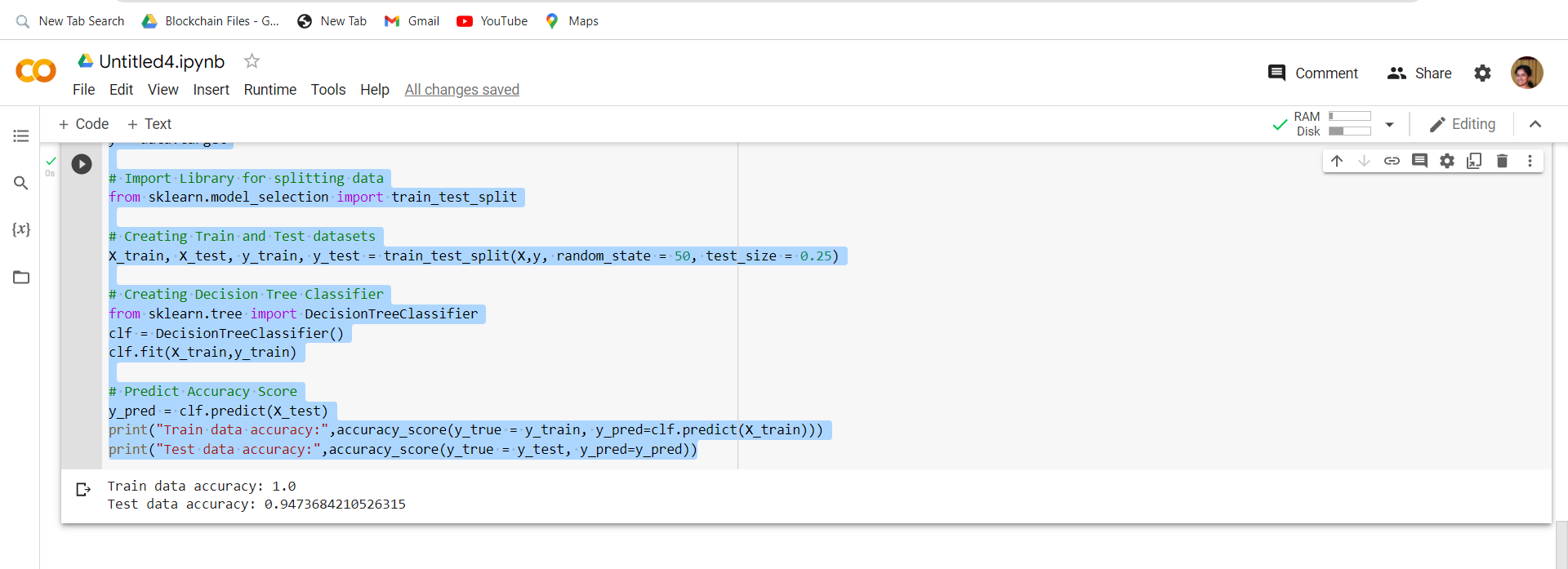
clf.fit(X\_train,y\_train)

# Predict Accuracy Score

y\_pred = clf.predict(X\_test)

print("Train data accuracy:",accuracy\_score(y\_true = y\_train, y\_pred=clf.predict(X\_train)))

print("Test data accuracy:",accuracy\_score(y\_true = y\_test, y\_pred=y\_pred))



8) Demonstrate any of the Clustering model and evaluate the performance on **Iris dataset**.

https://medium.com/pursuitnotes/k-means-clustering-model-in-6-steps-with-python-35b532cfa8ad

#importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Iris.csv')

x = dataset.iloc[:, [1, 2, 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\_)

#Plotting the results onto a line graph, allowing us to observe 'The elbow'

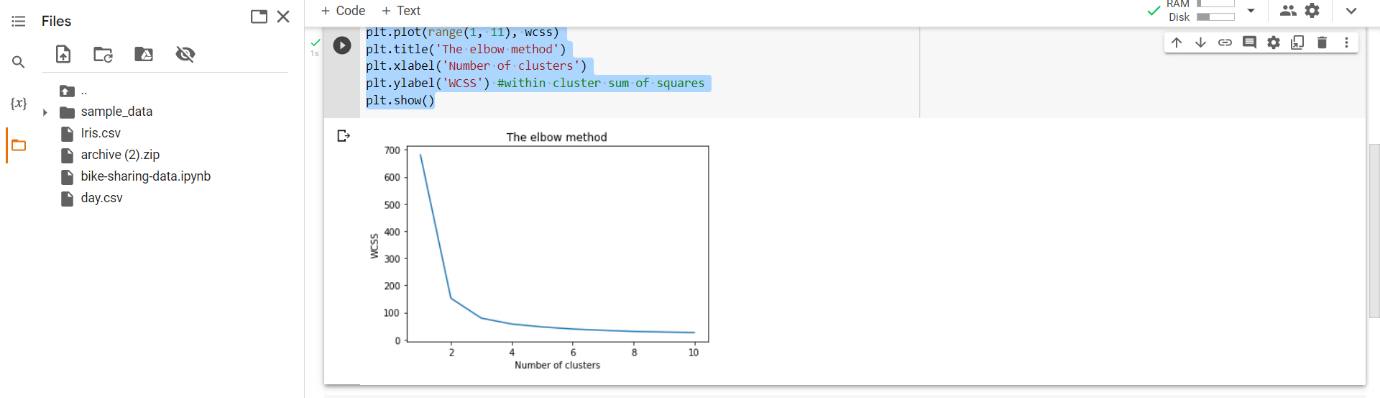
plt.plot(range(1, 11), wcss)

plt.title('The elbow method')

plt.xlabel('Number of clusters')

plt.ylabel('WCSS') #within cluster sum of squares

plt.show()



#Applying kmeans to the dataset / Creating the kmeans classifier

kmeans = KMeans(n\_clusters = 3, init = 'k-means++', 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 = 'Iris-setosa')

plt.scatter(x[y\_kmeans == 1, 0], x[y\_kmeans == 1, 1], s = 100, c = 'blue', label = 'Iris-versicolour')

plt.scatter(x[y\_kmeans == 2, 0], x[y\_kmeans == 2, 1], s = 100, c = 'green', label = 'Iris-virginica')

#Plotting the centroids of the clusters

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:,1], s = 100, c = 'yellow', label = 'Centroids')

plt.legend()

