**Iris csv(9-13)**

from google.colab import drive

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

import pandas as pd

import numpy as np

df=pd.read\_csv("/content/drive/MyDrive/Dataset/Iris.csv")

X = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm','PetalWidthCm']]

y = df['Species']

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=42)

#from sklearn.linear\_model import LogisticRegression

#LogRegressor = LogisticRegression()

#LogRegressor.fit(X\_train, y\_train)

#from sklearn.neighbors import KNeighborsClassifier

# Create and train the KNN model

#knn = KNeighborsClassifier()

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

#from sklearn.svm import SVC

#svc=SVC(probability=True)

#svc.fit(X\_train, y\_train)

#from sklearn.naive\_bayes import GaussianNB

#naive\_baiyes=GaussianNB()

#naive\_baiyes.fit(X\_train,y\_train)

from sklearn.tree import DecisionTreeClassifier

dt=DecisionTreeClassifier()

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

#predictions = LogRegressor.predict(X\_test)

#pred\_prob=LogRegressor.predict\_proba(X\_test)

#print("The predicted probabilities are",pred\_prob)

#predictions = knn.predict(X\_test)

#pre=knn.predict\_proba(X\_test)

#print("The pridected output is",pred\_prob)

#predictions = svc.predict(X\_test)

#pre=svc.predict\_proba(X\_test)

#print("The pridected output is",pred\_prob)

#predictions = naive\_baiyes.predict(X\_test)

#pre=naive\_baiyes.predict\_proba(X\_test)

#print("The pridected output is",pred\_prob)

predictions = dt.predict(X\_test)

pre=dt.predict\_proba(X\_test)

print("The pridected output is",pred\_prob)

sl=float(input("Enter sepal length"))

sw=float(input("Enter sepal width"))

pl=float(input("Enter petal length"))

pw=float(input("Enter petal width"))

y\_pred1=dt.predict([[sl,sw,pl,pw]])

print("The species is",y\_pred1)

# Evaluate the performance of the model

from sklearn.metrics import accuracy\_score,confusion\_matrix,classification\_report,roc\_auc\_score

accuracy = accuracy\_score(y\_test, predictions)

classification = classification\_report(y\_test, predictions,)

confusion = confusion\_matrix(y\_test, predictions)

auc\_roc = roc\_auc\_score(y\_test, pred\_prob,multi\_class='ovr')

# Print the evaluation metrics

print(f"Accuracy Score: {accuracy}")

print(f"\nClassification Report:\n{classification}")

print(f"\nConfusion Matrix:\n{confusion}")

print(f"\nAUC ROC Score: {auc\_roc}")

**Apple orange csv(6,7,8)**

**Logistic regression**

from google.colab import drive

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

import pandas as pd

import numpy as np

df=pd.read\_csv("/content/drive/MyDrive/Dataset/apples\_and\_oranges (1).csv")

x=df.iloc[:,0:2].values

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

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

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y, test\_size

=.2, random\_state = 0)

from sklearn.linear\_model import LogisticRegression

LogRegressor = LogisticRegression()

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

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

pred\_prob=LogRegressor.predict\_proba(x\_test)

print("The predicted probabilities are",pred\_prob)

w=int(input("Enter weight of fruit"))

s=float(input("Enter size of fruit"))

y\_pred1=LogRegressor.predict([[w,s]])

print("The fruit is",y\_pred1)

from sklearn.metrics import accuracy\_score,confusion\_matrix,classification\_report,roc\_auc\_score

print("Accuracy score:",accuracy\_score(y\_test,y\_pred))

print("classification Report:\n",classification\_report(y\_test,y\_pred))

print("Confusion Matrics:\n",confusion\_matrix(y\_test,y\_pred))

print("AUC Roc score is:\n",roc\_auc\_score(y\_test,pred\_prob[:,1]))

**KNN**

from google.colab import drive

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

import pandas as pd

import numpy as np

df=pd.read\_csv("/content/drive/MyDrive/Dataset/apples\_and\_oranges (1).csv")

x=df.iloc[:,0:2].values

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

import pandas as pd

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix, roc\_auc\_score

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, random\_state=42)

# Create and train the KNN model

knn = KNeighborsClassifier()

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

# Make predictions on the testing data

predictions = knn.predict(x\_test)

pre=knn.predict\_proba(x\_test)

print("The pridected output is",pre)

w=int(input("Enter weight of fruit"))

s=float(input("Enter size of fruit"))

y\_pred1=knn.predict([[w,s]])

print("The fruit is",y\_pred1)

# Evaluate the performance of the model

accuracy = accuracy\_score(y\_test, predictions)

classification = classification\_report(y\_test, predictions)

confusion = confusion\_matrix(y\_test, predictions)

auc\_roc = roc\_auc\_score(y\_test, pre[:,1])

# Print the evaluation metrics

print(f"Accuracy Score: {accuracy}")

print(f"\nClassification Report:\n{classification}")

print(f"\nConfusion Matrix:\n{confusion}")

print(f"\nAUC ROC Score: {auc\_roc}")

**SVM**

import pandas as pd

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

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix, roc\_auc\_score

df=pd.read\_csv('/content/drive/My Drive/Dataset2/apples\_and\_oranges.csv')

X = df[['Size', 'Weight']]

y = df['Class']

# Split the dataset into training and testing data (80% training, 20% testing)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create and train the KNN model

svc = SVC(probability=True)

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

# Make predictions on the testing data

predictions = svc.predict(X\_test)

svc=svc.predict\_proba(X\_test)

print("The pridected output is",pre)

accuracy = accuracy\_score(y\_test, predictions)

classification = classification\_report(y\_test, predictions)

confusion = confusion\_matrix(y\_test, predictions)

auc\_roc = roc\_auc\_score(y\_test, pre[:,1])

# Print the evaluation metrics

print(f"Accuracy Score: {accuracy}")

print(f"\nClassification Report:\n{classification}")

print(f"\nConfusion Matrix:\n{confusion}")

print(f"\nAUC ROC Score: {auc\_roc}")

Simple Linear Regression

from google.colab import drive

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

import pandas as pd

import numpy as np

df=pd.read\_csv("/content/drive/My Drive/Dataset/mtcars.csv")

print(df.head(5))

x=df.iloc[:,[6]].values

y=df.iloc[:,1].values

print(x,y)

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

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=.3,random\_state=3)

from sklearn.linear\_model import LinearRegression

LinRegressor=LinearRegression();

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

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

from math import sqrt

from sklearn.metrics import mean\_absolute\_error,mean\_squared\_error,r2\_score

print('Mean absolute error:%.2f' % mean\_absolute\_error(y\_test,y\_pred))

print('Mean squared error:%.2f' % mean\_squared\_error(y\_test,y\_pred))

print('Mean squared error:%.2f' % sqrt(mean\_squared\_error(y\_test,y\_pred)))

print('R2 score:%.2f' % r2\_score(y\_test,y\_pred))

MultiLinear Regression

from google.colab import drive

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

import pandas as pd

import numpy as np

df=pd.read\_csv("/content/drive/My Drive/Dataset/mtcars.csv")

print(df.head(5))

#wt cyl and disp

#x=df.iloc[:,[2,3,6]].values

#removing disp

x=df.iloc[:,[2,6]].values

y=df.iloc[:,1].values

print(x,y)

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

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=.3,random\_state=3)

from sklearn.linear\_model import LinearRegression

LinRegressor=LinearRegression();

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

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

from math import sqrt

from sklearn.metrics import mean\_absolute\_error,mean\_squared\_error,r2\_score

print('Mean absolute error:%.2f' % mean\_absolute\_error(y\_test,y\_pred))

print('Mean squared error:%.2f' % mean\_squared\_error(y\_test,y\_pred))

print('Mean squared error:%.2f' % sqrt(mean\_squared\_error(y\_test,y\_pred)))

print('R2 score:%.2f' % r2\_score(y\_test,y\_pred))

Q1 Correlation

import numpy as np

import matplotlib.pyplot as plt

from scipy.stats import pearsonr

x=7,8,9,11,15,20

y=130,135,140,142,147,156

corr,\_=pearsonr(x,y)

print('Pearsons Correlation: %.3f'%corr)

plt.scatter(x,y)

plt.plot(np.unique (x), np.poly1d (np.polyfit(x,y,1))(np.unique(x)), color ='blue')

Q2 iris

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

import pandas as pd

# Load iris dataset

iris = load\_iris()

iris\_df = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

# Create a correlation matrix

correlation\_matrix = iris\_df.corr()

# Plot the correlation matrix using a heatmap

sns.heatmap(correlation\_matrix, annot=True, cmap="coolwarm", linewidths=.5)

plt.title("Correlation Plot of Iris Dataset")

plt.show()

Q11

K means algorithm

# importing the necessary libraries

from sklearn.preprocessing import StandardScaler

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

import os

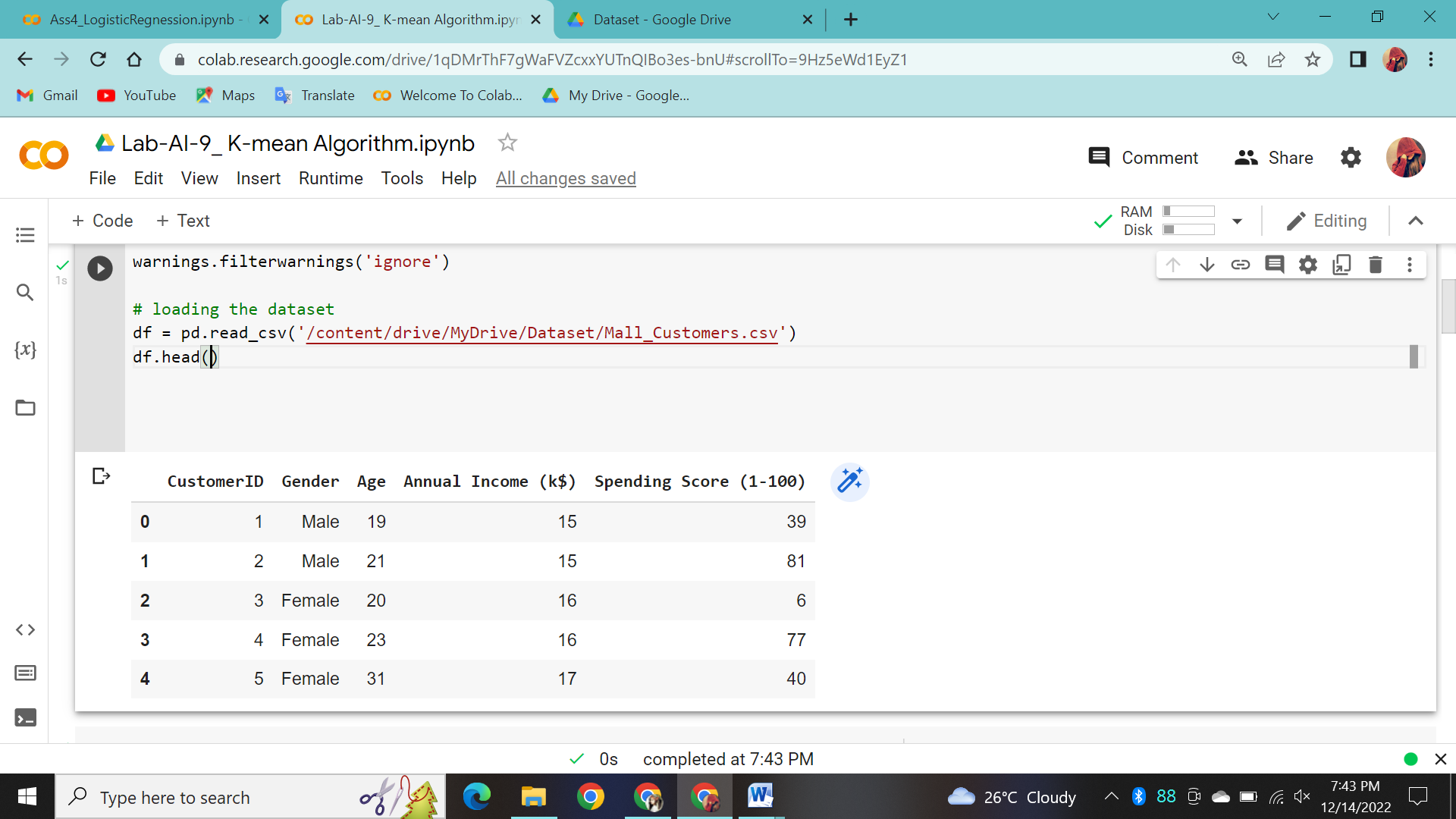
import warnings

warnings.filterwarnings('ignore')

# loading the dataset

df = pd.read\_csv('/content/drive/MyDrive/Dataset/Mall\_Customers.csv')

df.head()



# renaming the heads

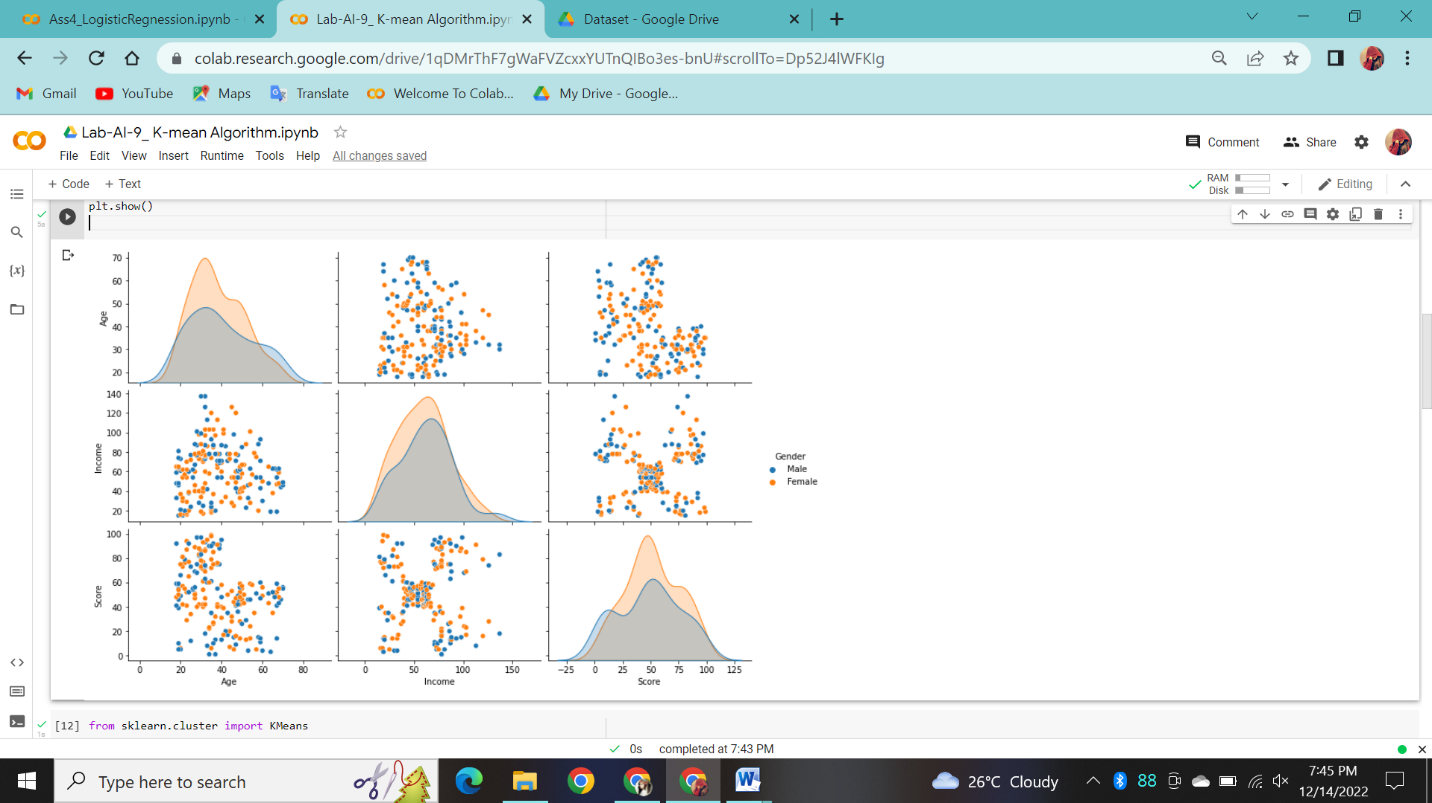
df.rename(index=str, columns={'Annual Income (k$)': 'Income','Spending Score (1-100)': 'Score'}, inplace=True)

# data in a detailed way with pairplot

X = df.drop(['CustomerID', 'Gender'], axis=1)

sns.pairplot(df.drop('CustomerID', axis=1), hue='Gender', aspect=1.5)

plt.show()



from sklearn.cluster import KMeans

clusters = []

for i in range(1, 11):

    km = KMeans(n\_clusters=i).fit(X)

clusters.append(km.inertia\_)

fig, ax = plt.subplots(figsize=(12, 8))

sns.lineplot(x=list(range(1, 11)), y=clusters, ax=ax)

ax.set\_title('Searching for Elbow')

ax.set\_xlabel('Clusters')

ax.set\_ylabel('Inertia')

# Annotate arrow

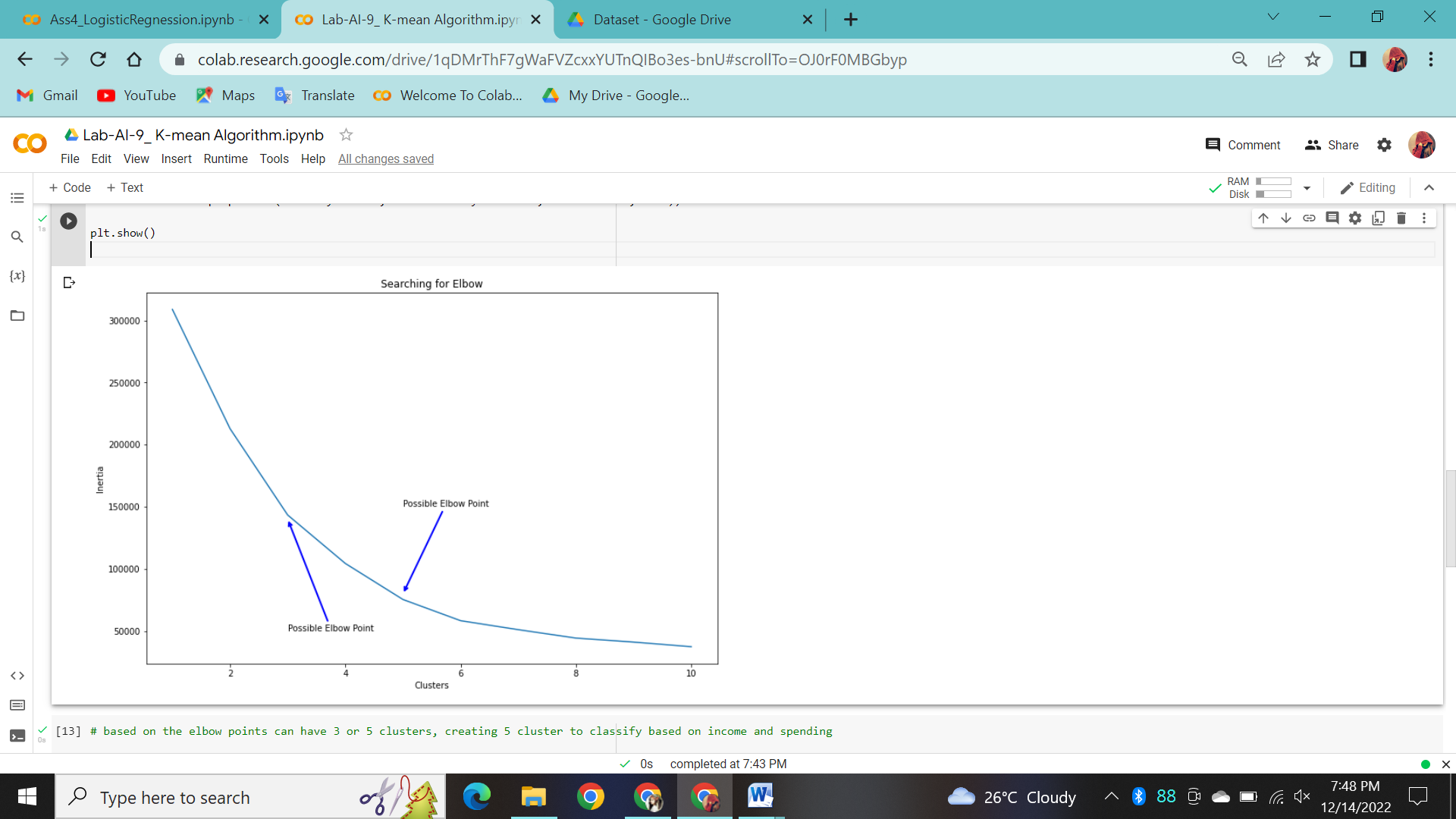
ax.annotate('Possible Elbow Point', xy=(3, 140000), xytext=(3, 50000), xycoords='data',

             arrowprops=dict(arrowstyle='->', connectionstyle='arc3', color='blue', lw=2))

ax.annotate('Possible Elbow Point', xy=(5, 80000), xytext=(5, 150000), xycoords='data',

             arrowprops=dict(arrowstyle='->', connectionstyle='arc3', color='blue', lw=2))

plt.show()



# based on the elbow points can have 3 or 5 clusters, creating 5 cluster to classify based on income and spending

km5 = KMeans(n\_clusters=5).fit(X)

X['Labels'] = km5.labels\_

plt.figure(figsize=(12, 8))

sns.scatterplot(X['Income'], X['Score'], hue=X['Labels'], palette=sns.color\_palette('hls', 5))

plt.title('KMeans with 5 Clusters')

plt.show()

Q12

Agglomerative

# importing the necessary libraries

from sklearn.preprocessing import StandardScaler

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

import os

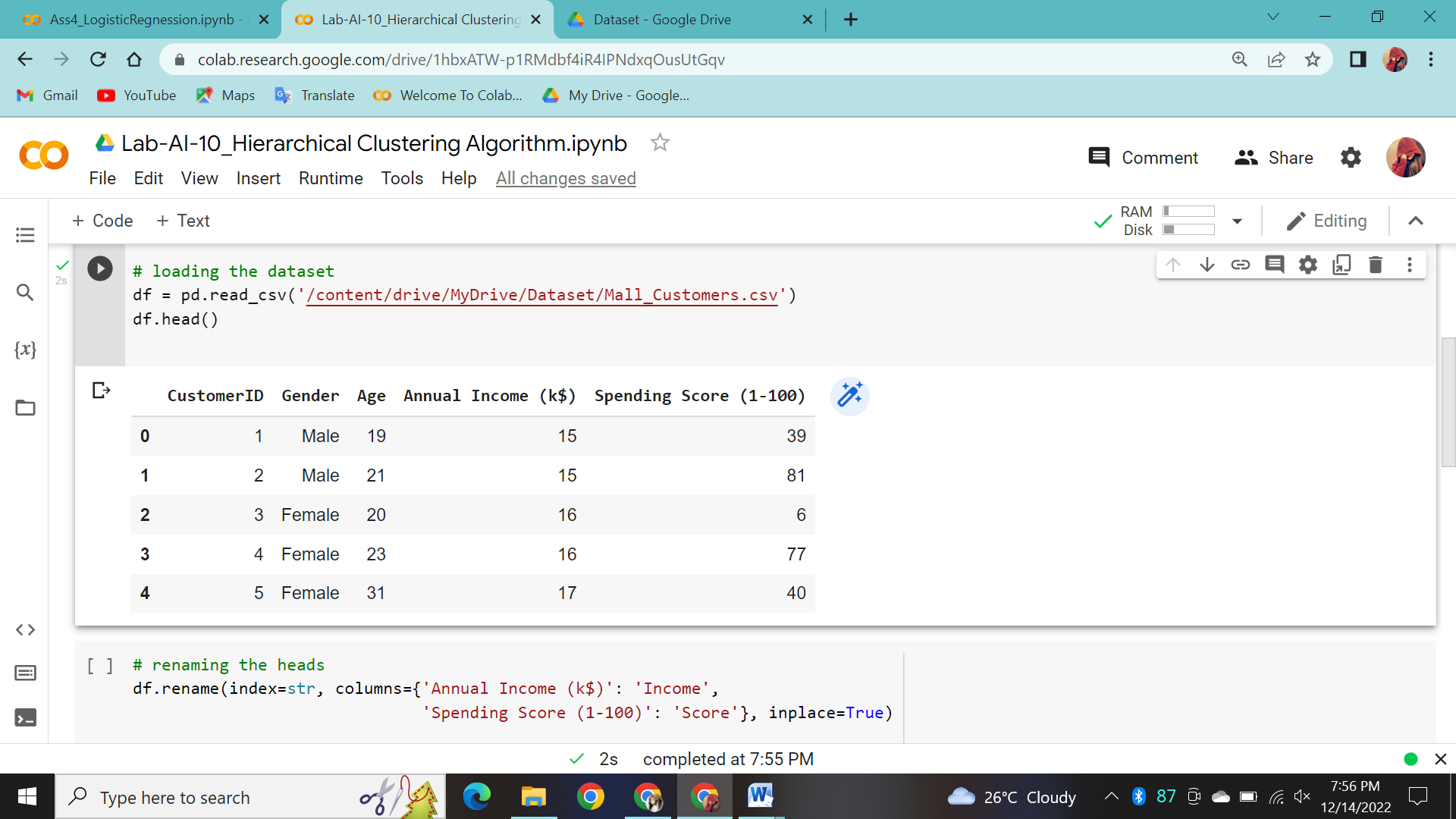
import warnings

warnings.filterwarnings('ignore')

# loading the dataset

df = pd.read\_csv('/content/drive/MyDrive/Dataset/Mall\_Customers.csv')

df.head()



# renaming the heads

df.rename(index=str, columns={'Annual Income (k$)': 'Income',

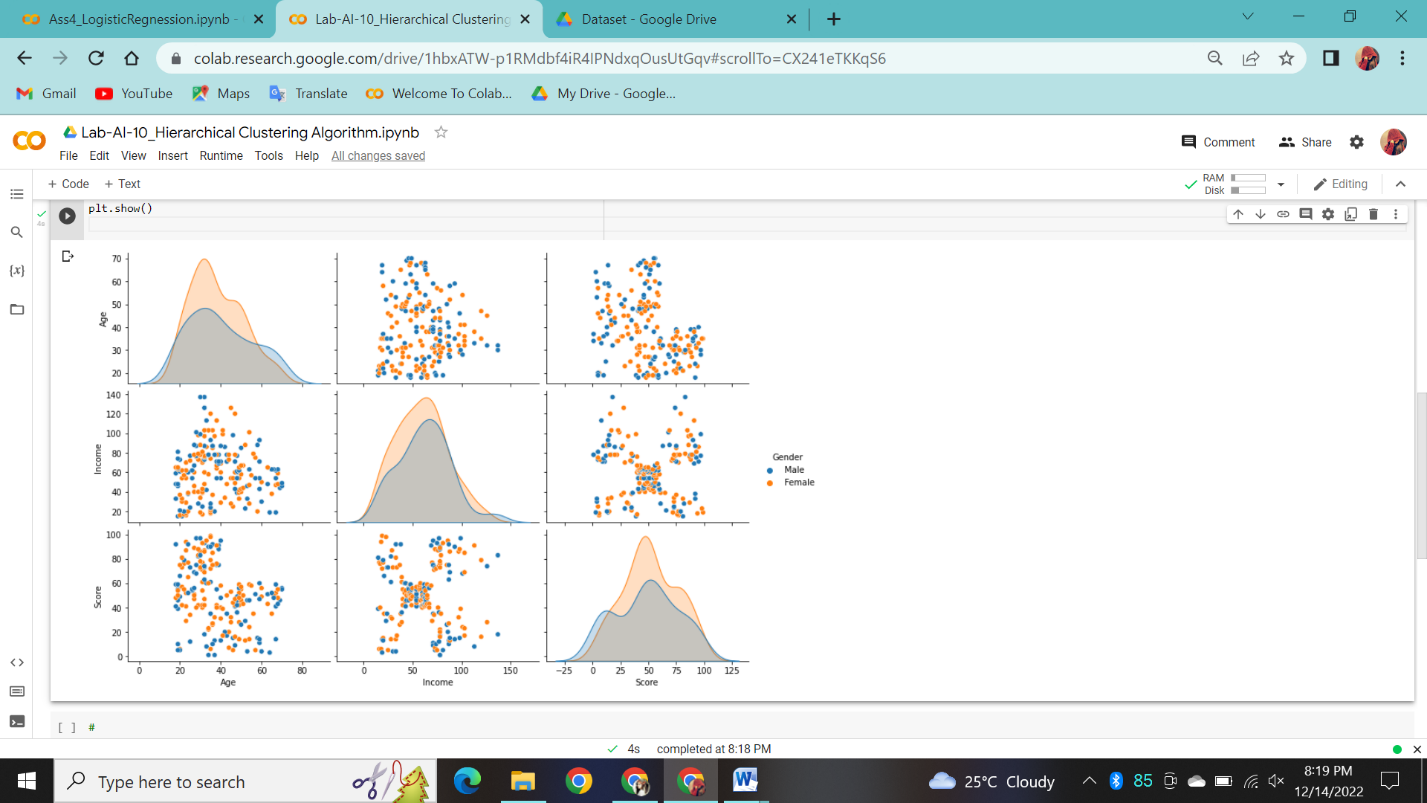
                              'Spending Score (1-100)': 'Score'}, inplace=True)

# data in a detailed way with pairplot

X = df.drop(['CustomerID', 'Gender'], axis=1)

sns.pairplot(df.drop('CustomerID', axis=1), hue='Gender', aspect=1.5)

plt.show()



#

from sklearn.cluster import AgglomerativeClustering

agglom = AgglomerativeClustering(n\_clusters=5, linkage='average').fit(X)

X['Labels'] = agglom.labels\_

plt.figure(figsize=(12, 8))

sns.scatterplot(X['Income'], X['Score'], hue=X['Labels'], palette=sns.color\_palette('hls', 5))

plt.title('Agglomerative with 5 Clusters')

plt.show()



Webview

package nikam.soft.webview;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.webkit.WebView;

import android.webkit.WebViewClient;

import android.widget.Button;

import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

Button b1;

EditText ed1;

private WebView wv1;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

b1=(Button)findViewById(R.id.button);

ed1=(EditText)findViewById(R.id.editText);

wv1=(WebView)findViewById(R.id.webView);

wv1.setWebViewClient(new MyBrowser());

b1.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

String url = ed1.getText().toString();

wv1.getSettings().setLoadsImagesAutomatically(true);

wv1.getSettings().setJavaScriptEnabled(true);

wv1.setScrollBarStyle(View.SCROLLBARS\_INSIDE\_OVERLAY);

wv1.loadUrl(url);

}

});

}

private class MyBrowser extends WebViewClient {

@Override

public boolean shouldOverrideUrlLoading(WebView view, String url) {

view.loadUrl(url);

return true;

}

}

}

Following is the modified content of the xml res/layout/activity\_main.xml.

In the following code abc indicates the logo of indiraicem

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity">

<TextView android:text="WebView" android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/textview"

android:textSize="35dp"

android:layout\_alignParentTop="true"

android:layout\_centerHorizontal="true" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="ICEM"

android:id="@+id/textView"

android:layout\_below="@+id/textview"

android:layout\_centerHorizontal="true"

android:textColor="#ff7aff24"

android:textSize="35dp" />

<EditText

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/editText"

android:hint="Enter Text"

android:focusable="true"

android:textColorHighlight="#ff7eff15"

android:textColorHint="#ffff25e6"

android:layout\_marginTop="46dp"

android:layout\_below="@+id/imageView"

android:layout\_alignParentLeft="true"

android:layout\_alignParentStart="true"

android:layout\_alignRight="@+id/imageView"

android:layout\_alignEnd="@+id/imageView" />

<ImageView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/imageView"

android:src="@drawable/abc"

android:layout\_below="@+id/textView"

android:layout\_centerHorizontal="true" />

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Enter"

android:id="@+id/button"

android:layout\_alignTop="@+id/editText"

android:layout\_toRightOf="@+id/imageView"

android:layout\_toEndOf="@+id/imageView" />

<WebView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/webView"

android:layout\_below="@+id/button"

android:layout\_alignParentLeft="true"

android:layout\_alignParentStart="true"

android:layout\_alignParentRight="true"

android:layout\_alignParentEnd="true"

android:layout\_alignParentBottom="true" />

</RelativeLayout>