Data Mining and Warehouse

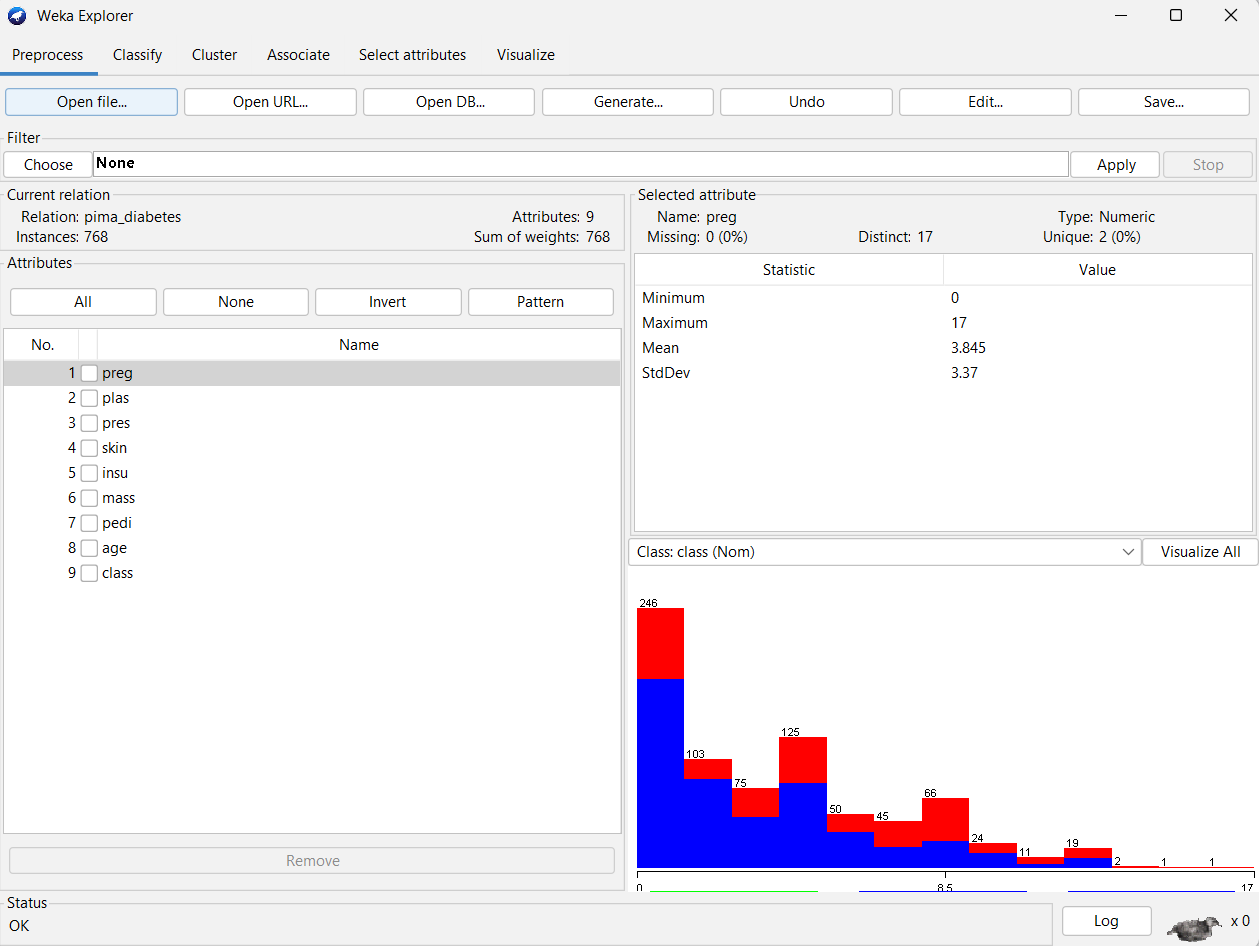
practical 1

Name: Jigar Siddhpura  
SAP: 60004210155  
Division/Batch: B2  
Branch: Computer Engineering

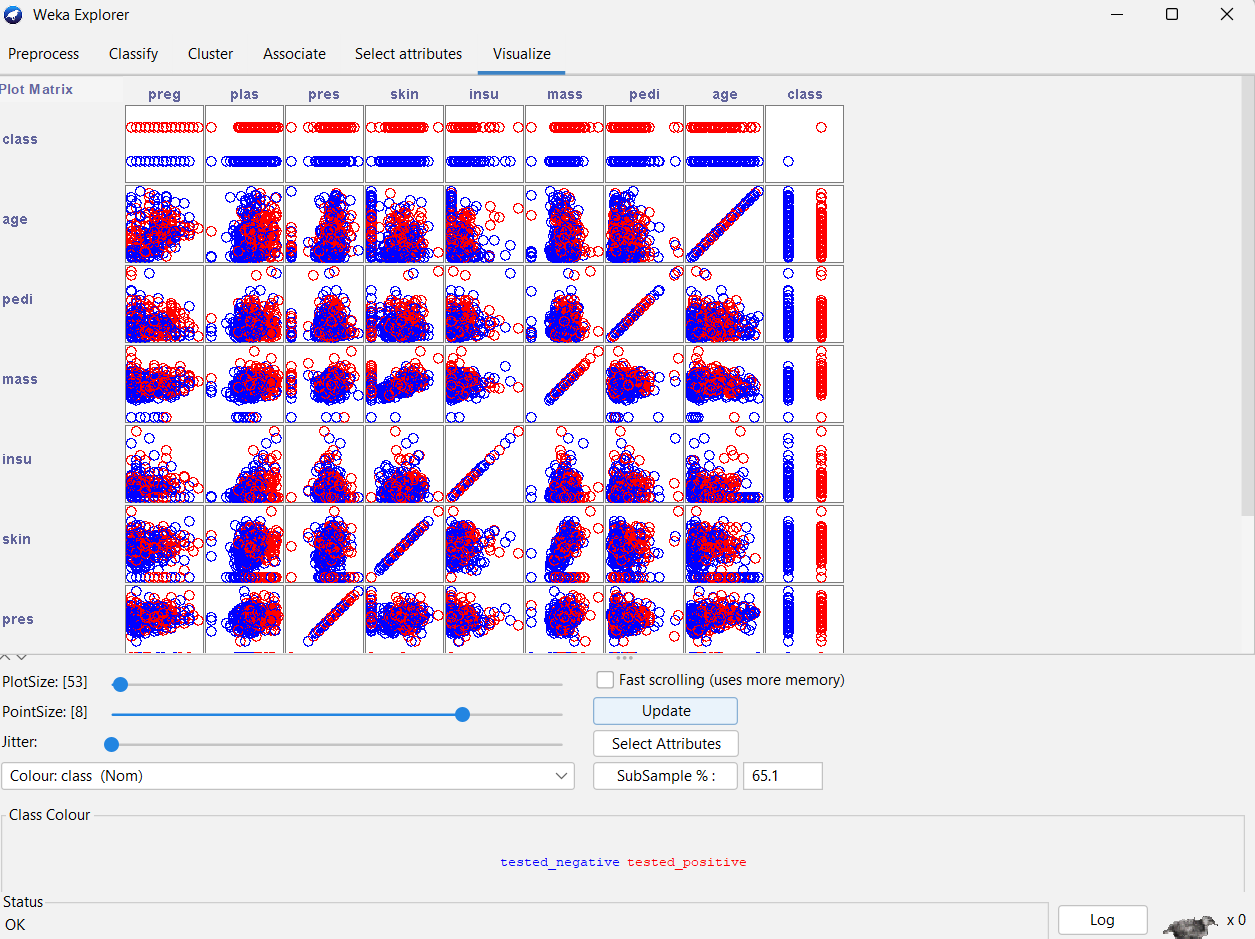


Pre-processing activities to be observed in Weka

1. **Visualization:** Visualize scatter plot for all the attributes from the dataset selected from Weka. Determine correlation if any using these plots for different datasets



We have loaded the ***Diabetes*** dataset into Weka and have visualized the various attributes. The histograms for each attribute display the variation in values within that attribute and can be viewed for each attribute.



Here we can visualise the scatter plots for each of the attributes of the dataset taken in pairs. By observing the scatter plot, we can infer the following:

* 1. Body mass index (weight in kg/ (height in m) ^2) vs Triceps skin fold thickness (mm) – There is a positive correlation between these attributes
  2. 2-Hour serum insulin (mu U/ml) vs Plasma glucose concentration 2 hours in an oral glucose tolerance test – There is a positive correlation between these attributes
  3. Diabetes pedigree function vs Triceps skin fold thickness (mm) – There is no correlation between these attributes and the plot is sparsely distributed

1. **Select Attributes**: Apply suitable feature selection filters like Gain Ratio etc to choose relevant attributes from the list of attributes. Observe the ranks / priority provided by the filter.

Graphical user interface, text, application, email

Description automatically generated

We now switch to the Select Attribute tab and have used the InfoGainAttributeEval method with the attribute Ranker. From the above results, the attribute ‘plas’ which stands for ‘Plasma glucose concentration 2 hours in an oral glucose tolerance test’ is the highest ranked with a score of 0.1901. This shows that the plasma attribute is the most important attribute for further processing.

Graphical user interface

Description automatically generated with low confidence

The data can be seen influenced by the attribute ‘plas’ and a comparably clear separation in classes correlating with its values.

1. **Pre-processing**:
   1. **Visualize All**: Select this button to visualize histograms of all attributes.

Chart, waterfall chart

Description automatically generated

Here we have visualized all of the attributes and their histograms. We can infer that some attributes like skin, pedi, preg, insu, age are left skewed while pres, plas, mass are normally distributed.

* 1. **Filter**: Choose Discretization under Unsupervised and Supervised methods. Observe the discretization and the outliers.

*Before*:

Graphical user interface

Description automatically generated

The attribute value before discretization shows a continuous histogram with varying values and no of instances.

*After*:

Graphical user interface

Description automatically generated

Here we have applied unsupervised discrete filtering to the dataset. For each attribute discrete buckets are created, and the instances are separated into their appropriate bucket. The attribute here is nominal and hence the bucket is split based on their values and extends from -infinity to 6.5 and from 6.5 to infinity respectively.

* 1. **IQR**: Observe the IQR values for a selected attribute. Observe the outlier and extreme values

Graphical user interface, text, application, email

Description automatically generated

These are the settings that have been applied for IQR.

A picture containing chart

Description automatically generated

When we apply the Inter Quartile Range (IQR) per attribute, we are able to see the outliers present in each of the attributes separately along with the extreme values that are present. Outliers are reported at less than Q1-1.5\*IQR and greater than Q3+1.5\*IQR.

* 1. **Remove the value**: Remove instances with outlier values and show the screenshots of dataset before and after the removal.

*Before:*

A picture containing chart

Description automatically generated

From the above IQR plot, we can determine the attributes that have the outliers. From this dataset, the attributes ‘predi’, ‘insu’ and ‘pres’ have outliers with a combined total of 49 instances.

*After:*

Chart

Description automatically generated with medium confidence

We can then use the attributes menu to remove those instances from the respective attributes and again visualising the attributes graphs shows that the outliers in the dataset have been removed.

1. **Classification**: Perform NB, kNN and DT/rule-based classification

Graphical user interface, text

Description automatically generated

We have performed Naïve Bayes classification on the dataset. The Naïve bayes classifier correctly classified 77.86% of the instances while 22.13% were incorrectly classified.

Text

Description automatically generated

We have performed Random Forest (Decision Tree/Rule Based) classification on the dataset. The Random Forest classifier correctly classified 76.95% of the instances while 23.04% were incorrectly classified.

1. **Clustering**: Perform kmeans, hierarchical clustering and explain the output

Graphical user interface, application

Description automatically generated

We have performed K-Means Clustering on the dataset to form groups of similar instances from the data. It incorrectly clustered 45.83% of the instances.

Graphical user interface, text, application, email

Description automatically generated

We have performed Hierarchical Clustering on the dataset to form groups of similar instances from the data. It incorrectly clustered 34.51% of the instances, thus being comparatively better than the K-Means clustering algorithm for this dataset.

1. **Association rule mining**: Perform apriori algo and show the rules created

Graphical user interface, application

Description automatically generated

Associative rule mining is used to find associations in the data. Weka has an associate tab that allows us to perform associative rule mining using algorithms like apriori. Here we have applied apriori to the dataset. From the results, we can infer that:

* 1. The attributes skin, mass and pres are associative
  2. The attributes mass and pres are associative
  3. The attributes preg and skin are associative

Conclusion

Weka is a very intuitive tool for exploring datasets and performing data mining operations. It provides extensive support for data visualisations through histograms to visualise variation in the instances within the data and through scatter plots for correlating various attributes. It also provides IQR which allows us to detect and remove outliers. We can also filter the data through discretisation and other processes and visualise the state of the data while doing so. It also allows us to select attributes from the dataset and then perform classification and clustering on the data to identify similar groups. Weka also allows us to perform associative mining on the data to find associated attributes for basket analysis.

**Name: Jigar Siddhpura SAPID:** 60004200155

**DIV: C/C2 Branch:** Computer Engineering

DMW Exp 2

# Interview Questions:

Q. Where does your data come from?

A. We get our data from CRM, Facebook, Google Analytics and BI software.

Q. How do you measure profit margin?

A. Our main formula for measuring the profit accumulated is Net income / revenue. Our profit margin deteriorated from the financial year 2019-2020 but increased from the financial year 2020-2021.

Q. Which product is your biggest source of revenue?

A. Our iPhones produce the biggest source of revenue.

Q. What are the business segments of Apple?

A. Countries like Japan, America, China etc. are our business segment nations. Currently the US is leading but Asia is catching up. America is contributing 46B dollars in the first quarter of 2021. 31% of Macbook users in the US are between ages 25-34. 53% of all MacBook users are from small towns. 1/10 people own an Apple Watch.

Q. Customer demographics of your company?

A. More than 1.6B active apple devices are currently in use across the world out of which 1B are iPhones. iOS users account for 26.99% of all mobile users. As of 2019, 51% of all iOS users were female and 43% were male. It’s the only major mobile vendor with max female users.

Q. Which kind of analysis helps you the most in making strategic decisions?

A. Correlation analysis. It measures the strength of the linear relationship between two variables and computes their association. Simply put - correlation analysis calculates the level of change in one variable due to the change in the other. Q. Relation b/n customer satisfaction and product sales?

A. According to our research, we have seen that the more satisfied the customers are, better are the sales.

Q. How are apple services growing?

A.

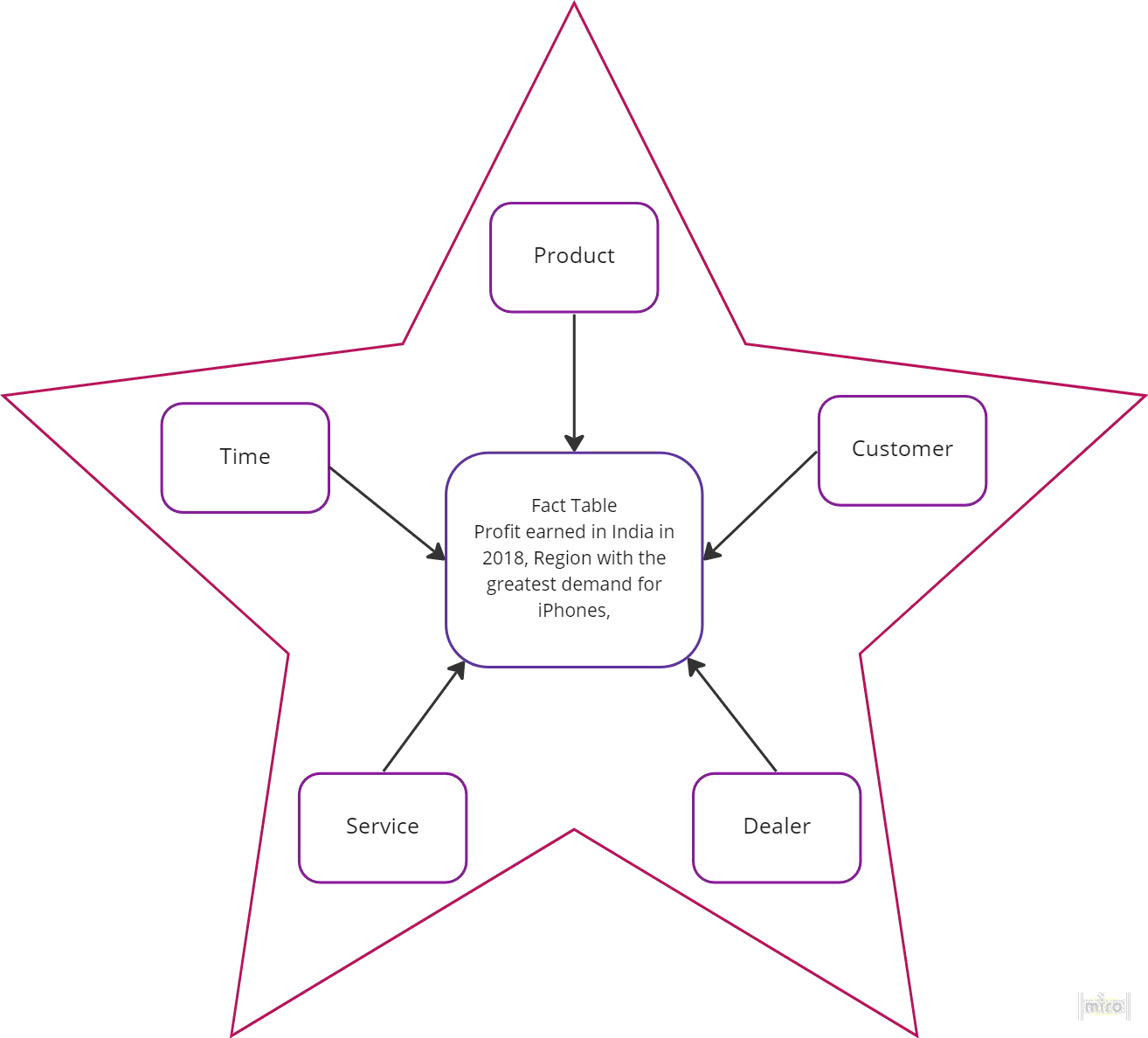
Apple Music had 98M users by 2021.

# IP:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time** | **Customer** | **Product** | **Service** | **Dealer** |
| Yearly | Name | Model Name | Name | Name |
| Quarterly | DOB | Model Year | Launch Year | State |
| Monthly | Gender | Price | Subscribers | Country |
| Weekly | State | Category | Revenue | Region |
| Daily | Country | Colour | Subscription Type | Contact No. |
| Season | Region | Specifications | SubscriptionPeri od | Email |
|  | Contact No. | Warranty | Price Plans | Sales |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Email | Storage | Region | Revenue |
|  |  | Place of Assembly | Country | Date First Operation |
|  |  | Revenue |  | Payment Methods |
|  |  | Sales |  |  |
|  |  | Cost of Manufacture |  |  |

Facts: Profit earned in India in 2018, Region with the greatest demand for iPhones, Most popular product among the ages 18-25 years, Service that generates the most revenue in Australia, the most common nationality among employees ***Data model:***



# Conclusion:

Hence, we have successfully understood the process of designing a data warehouse & the resources required to do so. We modelled an information package along with a star schema & fact table.

**Data Mining and Warehouse Experiment 3**

**Name: Jigar Siddhpura SAP: 60004210155**

**Batch: B2**

**Branch: Computer Engineering**

**Initialisation :**

from google.colab import drive drive.mount('/content/gdrive')

!pip install scikit-plot import pandas as pd import numpy as np import seaborn as sns import re

import matplotlib.pyplot as plt

from scikitplot.metrics import plot\_confusion\_matrix from sklearn.multiclass import OneVsRestClassifier

from sklearn.metrics import roc\_curve, roc\_auc\_score, accuracy\_score, classification\_report, confusion\_matrix titanic\_train = "/content/gdrive/MyDrive/Synapse-Task/synapse\_w1/train.csv"

titanic\_test = "/content/gdrive/MyDrive/Synapse-Task/synapse\_w1/test.csv" penguin\_df = sns.load\_dataset("penguins")

iris\_df = pd.read\_csv("https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv") spam\_df = pd.read\_csv("/content/gdrive/MyDrive/DMW/datasets/spam.csv")

wine\_df = pd.read\_csv("/content/gdrive/MyDrive/DMW/datasets/WineQT.csv") gaussian = []

decision = []

**Part A: Dataset(Titanic) :**



""" Titanic"""

titanic\_train\_df = pd.read\_csv(titanic\_train) titanic\_test\_df = pd.read\_csv(titanic\_test) import re

titles = []

for nm in titanic\_train\_df. ame: title\_search = re.search('(\w+)\.', nm) title = title\_search.group(1) titles.append(title)

titanic\_train\_df['Title'] = titles titanic\_train\_df.columns

titanic\_train\_df.drop(['PassengerId', 'Ticket', ' ame'], axis=1, inplace=True) nullPercent = {}

for i in titanic\_train\_df:

null\_count\_i = titanic\_train\_df.isnull().sum()[i] per = null\_count\_i\*100/titanic\_train\_df.shape[0] nullPercent[i] = per

for i in nullPercent:

if(nullPercent[i] > 50) : titanic\_train\_df.drop([i], axis=1, inplace=True) titanic\_train\_df.info()

mean = np.mean(titanic\_train\_df.Age) titanic\_train\_df['Age'].fillna(value=mean, inplace=True)

train\_df = titanic\_train\_df.assign(Family=lambda x: x.SibSp + x.Parch) def zscore\_norm(x):

mean = np.mean(x) std = np.std(x) return (x-mean)/std

train\_df = train\_df.assign(Age=lambda x: zscore\_norm(x.Age)) train\_df = train\_df.assign(Fare=lambda x: zscore\_norm(x.Fare)) train\_df = train\_df.assign(Family=lambda x: zscore\_norm(x.Family))

train\_df = pd.get\_dummies(train\_df , columns=['Pclass', 'Sex', 'Title', 'Embarked']) train\_df

y = train\_df.pop("Survived")

x = train\_df

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

x\_train , x\_valid , y\_train , y\_valid = train\_test\_split(x ,y, random\_state=10, stratify=y, test\_size=0.25 ) y\_train.value\_counts(normalize=True)

y\_valid.value\_counts(normalize=True)



***using naive bayes***

from sklearn.naive\_bayes import Gaussian B nb\_model = Gaussian B() nb\_model.fit(x\_train, y\_train)

nb\_accuracy = nb\_model.score(x\_valid, y\_valid)

print(nb\_accuracy)

***using decision tree***



from sklearn.tree import DecisionTreeClassifier dt\_model = DecisionTreeClassifier() dt\_model.fit(x\_train, y\_train)

dt\_accuracy = dt\_model.score(x\_valid, y\_valid) gaussian.append(nb\_accuracy) decision.append(dt\_accuracy)

accuracy = [nb\_accuracy, dt\_accuracy] Models = [' aiveBayes','DecisionTree']

sns.barplot(x=Models,y=accuracy).set(title="Titanic Dataset") y\_score\_gnb = nb\_model.predict\_proba(x\_valid)[:, 1]

fpr\_gnb, tpr\_gnb, thresholds\_gnb = roc\_curve(y\_valid, y\_score\_gnb) roc\_auc\_gnb = roc\_auc\_score(y\_valid, y\_score\_gnb)

y\_score\_dtc = dt\_model.predict\_proba(x\_valid)[:, 1]

fpr\_dtc, tpr\_dtc, thresholds\_dtc = roc\_curve(y\_valid, y\_score\_dtc) roc\_auc\_dtc = roc\_auc\_score(y\_valid, y\_score\_dtc) plt.figure(figsize=(8, 6))

plt.plot(fpr\_gnb, tpr\_gnb, color='blue', lw=2, label=' aive Bayes ROC curve (AUC = {:.2f})'.format(roc\_auc\_gnb)) plt.plot(fpr\_dtc, tpr\_dtc, color='green', lw=2, label='Decision Tree ROC curve (AUC =

{:.2f})'.format(roc\_auc\_dtc))

plt.plot([0, 1], [0, 1], color='gray', linestyle='--') plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate') plt.title('ROC-AUC Curve') plt.legend(loc='lower right') plt.show()

q = y\_valid

pred\_test = dt\_model.predict(x\_valid) pred\_test = pd.DataFrame(pred\_test) y\_valid = pd.DataFrame(y\_valid)

plot\_confusion\_matrix(y\_valid, pred\_test, figsize=(7,4), title='DecisionTree Confusion Matrix', cmap='BuGn') y\_valid = q

q = y\_valid

pred\_test = nb\_model.predict(x\_valid) pred\_test = pd.DataFrame(pred\_test) y\_valid = pd.DataFrame(y\_valid)

plot\_confusion\_matrix(y\_valid, pred\_test, figsize=(7,4), title='Gaussian B Confusion Matrix', cmap='BuGn') y\_test = q

from sklearn.model\_selection import KFold, cross\_val\_score kf = KFold(n\_splits=7,shuffle=True,random\_state=10)

cv\_score = cross\_val\_score(estimator=nb\_model,X=x\_train,y=y\_train,cv=kf,scoring='accuracy') mean\_cv\_score = np.mean(cv\_score)

print(mean\_cv\_score)



***ensemble tech - RandomForest***

from sklearn.ensemble import RandomForestClassifier rf\_model = RandomForestClassifier() rf\_model.fit(x\_train, y\_train)

rf\_accuracy = rf\_model.score(x\_valid, y\_valid)

print(rf\_accuracy)



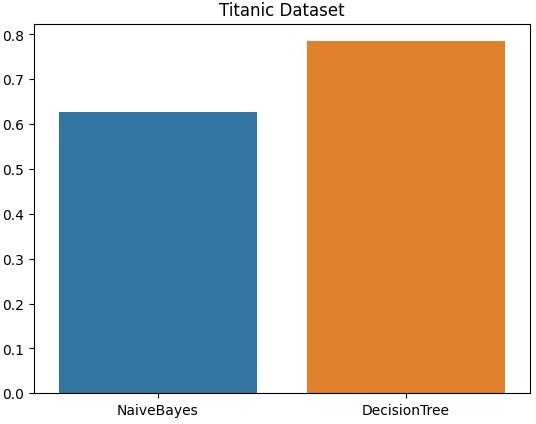
accuracies=[nb\_accuracy,dt\_accuracy,rf\_accuracy,mean\_cv\_score] plt.figure(figsize=(10,5))

models=[' aive Bayes','Decision Tree','Random Forest',' aive Bayes + Cross validation'] sns.barplot(x=models,y=accuracies,).set(title='Part C')

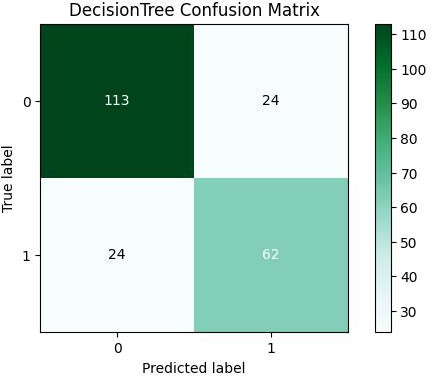
plt.xlabel('Models')

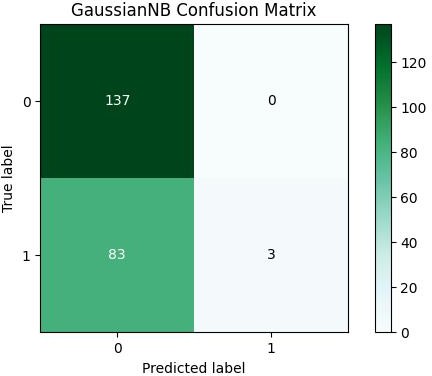
plt.ylabel('Accuracy') plt.title('Accuracy of Different Models')

**Accuracies of both models :**

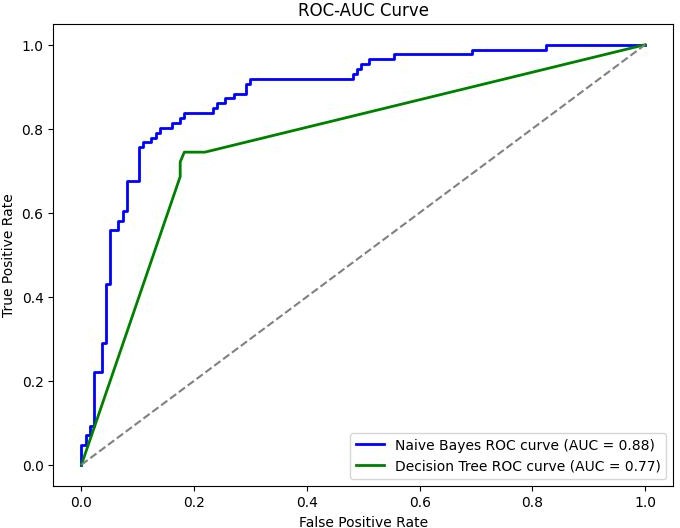


**Confusion Matrix :**





**AUROC:**



**Dataset 2 (penguins):**



""" Penguin"""

pg\_df = penguin\_df pg\_df.head() pg\_df.shape pg\_df.species.unique() pg\_df.island.unique()

pg\_df.dropna(inplace=True)

pg\_df = pd.get\_dummies(pg\_df, columns=['island','sex']) y = pg\_df.pop('species')

X = pg\_df

x\_train, x\_valid, y\_train, y\_valid = train\_test\_split(X,y,random\_state=10,stratify=y, test\_size=0.25) y\_train.value\_counts(normalize=True)

y\_valid.value\_counts(normalize=True)



***using naive bayes***

from sklearn.naive\_bayes import Gaussian B nb\_model = OneVsRestClassifier(Gaussian B()) nb\_model.fit(x\_train, y\_train)

nb\_accuracy = nb\_model.score(x\_valid, y\_valid)

print(nb\_accuracy)



***using decision tree***

from sklearn.tree import DecisionTreeClassifier

dt\_model = OneVsRestClassifier(DecisionTreeClassifier()) dt\_model.fit(x\_train, y\_train)

dt\_accuracy = dt\_model.score(x\_valid, y\_valid) print(dt\_accuracy) gaussian.append(nb\_accuracy)

decision.append(dt\_accuracy)



accuracy = [nb\_accuracy, dt\_accuracy] Models = [' aiveBayes','DecisionTree']

sns.barplot(x=Models,y=accuracy).set(title="Penguin Dataset")

y\_score\_gnb = nb\_model.predict\_proba(x\_valid)

fpr\_gnb, tpr\_gnb, thresholds\_gnb = roc\_curve(y\_valid.values, y\_score\_gnb.values) roc\_auc\_gnb = roc\_auc\_score(y\_valid, y\_score\_gnb)

y\_score\_dtc = dt\_model.predict\_proba(x\_valid)

fpr\_dtc, tpr\_dtc, thresholds\_dtc = roc\_curve(y\_valid.values, y\_score\_dtc.values) roc\_auc\_dtc = roc\_auc\_score(y\_valid, y\_score\_dtc)



plt.figure(figsize=(8, 6))

plt.plot(fpr\_gnb, tpr\_gnb, color='blue', lw=2, label=' aive Bayes ROC curve (AUC = {:.2f})'.format(roc\_auc\_gnb)) plt.plot(fpr\_dtc, tpr\_dtc, color='green', lw=2, label='Decision Tree ROC curve (AUC =

{:.2f})'.format(roc\_auc\_dtc))

plt.plot([0, 1], [0, 1], color='gray', linestyle='--') plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate') plt.title('ROC-AUC Curve') plt.legend(loc='lower right')

plt.show()



***confiusion matrix***

q = y\_valid

pred\_test = dt\_model.predict(x\_valid) pred\_test = pd.DataFrame(pred\_test) y\_valid = pd.DataFrame(y\_valid)

plot\_confusion\_matrix(y\_valid, pred\_test, figsize=(7,4), title='DecisionTree Confusion Matrix', cmap='BuGn')

y\_test = q



q = y\_valid

pred\_test = nb\_model.predict(x\_valid) pred\_test = pd.DataFrame(pred\_test) y\_valid = pd.DataFrame(y\_valid)

plot\_confusion\_matrix(y\_valid, pred\_test, figsize=(7,4), title='Gaussian B Confusion Matrix', cmap='BuGn')

y\_test = q



***auc - roc***

y\_score\_gnb = nb\_model.predict\_proba(x\_valid)

fpr\_gnb, tpr\_gnb, thresholds\_gnb = roc\_curve(y\_valid.values, y\_score\_gnb.values) roc\_auc\_gnb = roc\_auc\_score(y\_valid, y\_score\_gnb)

y\_score\_dtc = dt\_model.predict\_proba(x\_valid)

fpr\_dtc, tpr\_dtc, thresholds\_dtc = roc\_curve(y\_valid.values, y\_score\_dtc.values) roc\_auc\_dtc = roc\_auc\_score(y\_valid, y\_score\_dtc)

plt.figure(figsize=(8, 6))

plt.plot(fpr\_gnb, tpr\_gnb, color='blue', lw=2, label=' aive Bayes ROC curve (AUC = {:.2f})'.format(roc\_auc\_gnb)) plt.plot(fpr\_dtc, tpr\_dtc, color='green', lw=2, label='Decision Tree ROC curve (AUC =

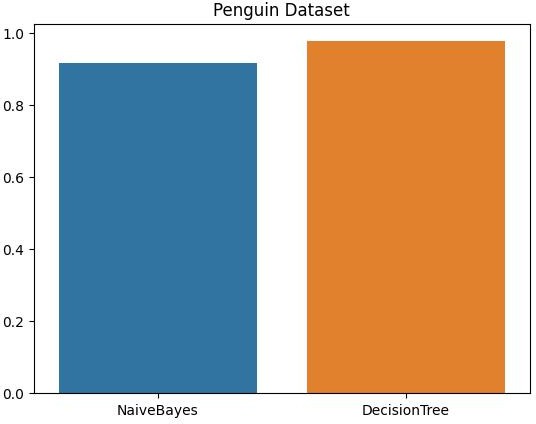
{:.2f})'.format(roc\_auc\_dtc))

plt.plot([0, 1], [0, 1], color='gray', linestyle='--') plt.xlabel('False Positive Rate')

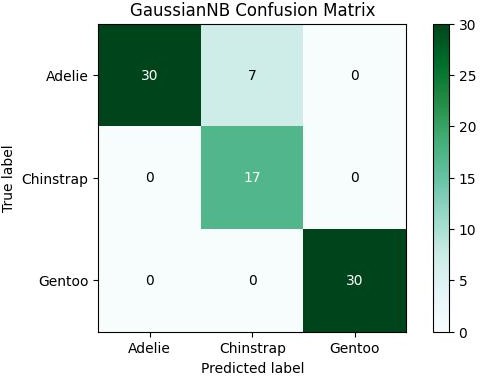
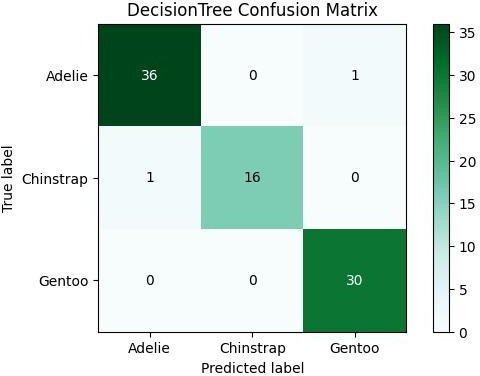
plt.ylabel('True Positive Rate') plt.title('ROC-AUC Curve') plt.legend(loc='lower right')

plt.show()

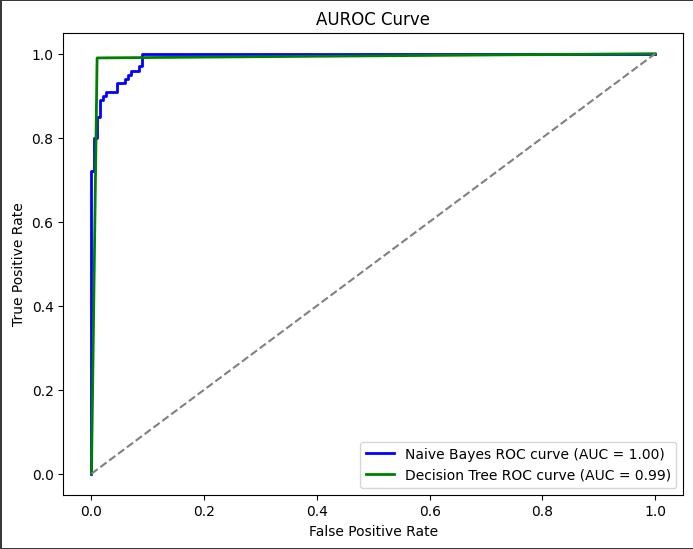
**Accuracies of both models :**



**Confusion Matrix :**



**AUROC:**



**Dataset 3 (Iris):**



""" Iris""" iris\_df.head() iris\_df.species.unique()

y = iris\_df.pop('species')

X = iris\_df

x\_train, x\_valid, y\_train, y\_valid = train\_test\_split(X,y,random\_state=10,stratify=y, test\_size=0.25) y\_train.value\_counts(normalize=True)



***using naive bayes***

from sklearn.naive\_bayes import Gaussian B nb\_model = Gaussian B() nb\_model.fit(x\_train, y\_train)

nb\_accuracy = nb\_model.score(x\_valid, y\_valid)

print(nb\_accuracy)



***using decision tree***

from sklearn.tree import DecisionTreeClassifier dt\_model = DecisionTreeClassifier() dt\_model.fit(x\_train, y\_train)

dt\_accuracy = dt\_model.score(x\_valid, y\_valid) print(dt\_accuracy) gaussian.append(nb\_accuracy) decision.append(dt\_accuracy)

accuracy = [nb\_accuracy, dt\_accuracy] Models = [' aiveBayes','DecisionTree']

sns.barplot(x=Models,y=accuracy).set(title="Iris Dataset")



q = y\_valid

pred\_test = dt\_model.predict(x\_valid) pred\_test = pd.DataFrame(pred\_test) y\_valid = pd.DataFrame(y\_valid)

plot\_confusion\_matrix(y\_valid, pred\_test, figsize=(7,4), title='DecisionTree Confusion Matrix', cmap='BuGn') y\_test = q

q = y\_valid

pred\_test = nb\_model.predict(x\_valid) pred\_test = pd.DataFrame(pred\_test) y\_valid = pd.DataFrame(y\_valid)

plot\_confusion\_matrix(y\_valid, pred\_test, figsize=(7,4), title='Gaussian B Confusion Matrix', cmap='BuGn')

y\_test = q



***auc - roc***

y\_score\_gnb = nb\_model.predict\_proba(x\_valid)

fpr\_gnb, tpr\_gnb, thresholds\_gnb = roc\_curve(y\_valid.values, y\_score\_gnb.values) roc\_auc\_gnb = roc\_auc\_score(y\_valid, y\_score\_gnb)

y\_score\_dtc = dt\_model.predict\_proba(x\_valid)

fpr\_dtc, tpr\_dtc, thresholds\_dtc = roc\_curve(y\_valid.values, y\_score\_dtc.values) roc\_auc\_dtc = roc\_auc\_score(y\_valid, y\_score\_dtc)

plt.figure(figsize=(8, 6))

plt.plot(fpr\_gnb, tpr\_gnb, color='blue', lw=2, label=' aive Bayes ROC curve (AUC = {:.2f})'.format(roc\_auc\_gnb)) plt.plot(fpr\_dtc, tpr\_dtc, color='green', lw=2, label='Decision Tree ROC curve (AUC =

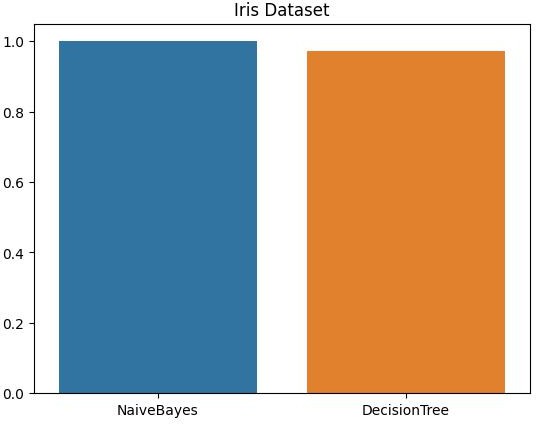
{:.2f})'.format(roc\_auc\_dtc))

plt.plot([0, 1], [0, 1], color='gray', linestyle='--') plt.xlabel('False Positive Rate')

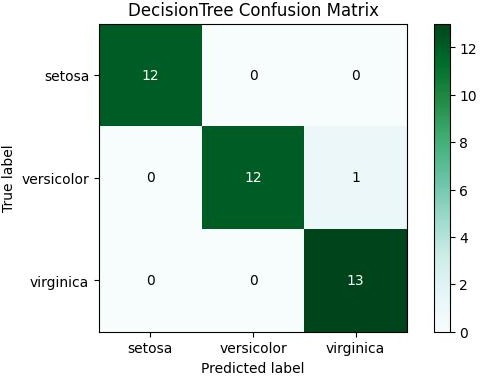
plt.ylabel('True Positive Rate') plt.title('ROC-AUC Curve') plt.legend(loc='lower right')

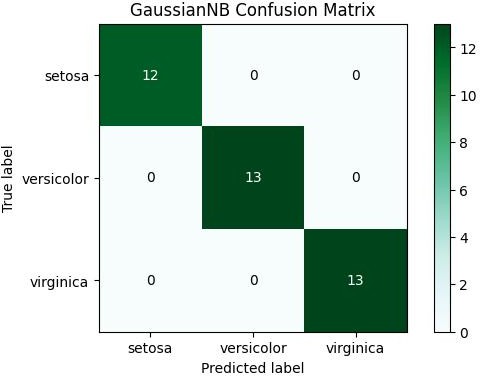
plt.show()

**Accuracies of both models :**

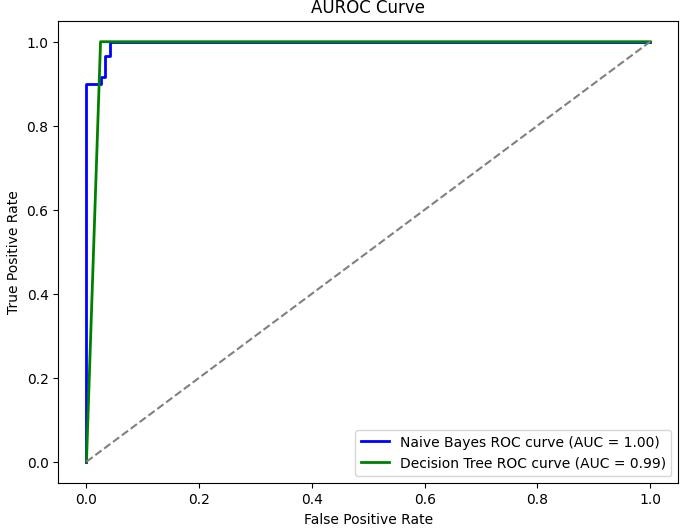


**Confusion Matrix :**





**AUROC:**



**Dataset 4 (Email spam-ham dataset):**

spam\_df.head()

def cleaning (text):

***text = text.lower()***

text = text = text = text =

text =

re.sub(r'@\S+', '',text)

re.sub(r'http\S+', '',text)

re.sub(r'pic.\S+', '',text)

***remove urls***

re.sub(r"[^a-zA-ZáéíóúÁÉÍÓÚ']", ' ',text)

***only keeps characters***

re.sub(r'\s+[a-zA-ZáéíóúÁÉÍÓÚ]\s+', ' ', text+' ')

***keep words with length>1 only***

spam\_df.head()

def cleaning (text):

*# text = text.lower()*

    text =  re.sub(r'@\S+', '',text)

    text =  re.sub(r'http\S+', '',text) *# remove urls*

    text =  re.sub(r'pic.\S+', '',text)

    text =  re.sub(r"[^a-zA-ZáéíóúÁÉÍÓÚ']", ' ',text) *# only keeps characters*

    text =  re.sub(r'\s+[a-zA-ZáéíóúÁÉÍÓÚ]\s+', ' ', text+' ')  *# keep words with length>1 only*

    text = "".join([i for i in text if i not in string.punctuation])

    words = word\_tokenize(text)

    stopwords = nltk.corpus.stopwords.words('english')   *# remove stopwords*

    text = " ".join([i for i in words if i not in stopwords])

    text= re.sub("\s[\s]+", " ",text).strip()

    text= re.sub("\s[\s]+", " ",text).strip() *# remove repeated/leading/trailing spaces*

    return text



""" spam""" import warnings

warnings.filterwarnings("ignore") import numpy as np

import pandas as pd import seaborn as sns

import matplotlib.pyplot as plt import nltk

import string

from nltk.tokenize import word\_tokenize import re

from nltk.corpus import stopwords

from nltk.stem.wordnet import Word etĮemmatizer

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

from sklearn.feature\_extraction.text import TfidfVectorizer from sklearn.linear\_model import ĮogisticRegression

from sklearn.tree import DecisionTreeClassifier from sklearn.ensemble import RandomForestClassifier import lightgbm as ltb

from sklearn.naive\_bayes import Gaussian B from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score import nltk

nltk.download('punkt') nltk.download('stopwords')

nltk.download('wordnet')

def lemmatize(data):

    wordnet = WordNetLemmatizer()

    lemmanized = []

    for i in range(len(data)):

        lemmed = []

        words = word\_tokenize(data['Message'].iloc[i])

        for w in words:

            lemmed.append(wordnet.lemmatize(w))

        lemmanized.append(lemmed)

    data['lemmanized'] = lemmanized

    data['text'] = data['lemmanized'].apply(' '.join)

    data=data.drop("lemmanized",axis=1)

    data=data.drop("Message",axis=1)

    return data

spam\_df = lemmatize(spam\_df)

obj = {"ham":0,"spam":1} spam\_df["Category"]=spam\_df["Category"].map(obj)

X = spam\_df['text']

y = spam\_df['Category']

x\_train, x\_valid, y\_train, y\_valid = train\_test\_split(X,y,random\_state=10,stratify=y, test\_size=0.25) y\_train.value\_counts(normalize=True)

from sklearn.feature\_extraction.text import TfidfVectorizer x\_train.shape

tfidf = TfidfVectorizer()

X\_train = tfidf.fit\_transform(x\_train) x\_valid = tfidf.transform(x\_valid)

dt\_model = DecisionTreeClassifier() dt\_model.fit(X\_train, y\_train)

pred = dt\_model.predict(x\_valid) dt\_accuracy = accuracy\_score(pred, y\_valid)

print(dt\_accuracy)



nb\_model = Gaussian B() nb\_model.fit(X\_train.toarray(), y\_train) pred = nb\_model.predict(x\_valid.toarray()) nb\_accuracy = accuracy\_score(pred, y\_valid) print(nb\_accuracy) gaussian.append(nb\_accuracy) decision.append(dt\_accuracy)

accuracy = [nb\_accuracy, dt\_accuracy] Models = [' aiveBayes','DecisionTree']

sns.barplot(x=Models,y=accuracy).set(title="Spam Text Message classification")



***auc - roc***

y\_score\_gnb = nb\_model.predict\_proba(x\_valid.toarray())[:, 1] fpr\_gnb, tpr\_gnb, thresholds\_gnb = roc\_curve(y\_valid, y\_score\_gnb) roc\_auc\_gnb = roc\_auc\_score(y\_valid, y\_score\_gnb)

y\_score\_dtc = dt\_model.predict\_proba(x\_valid)[:, 1]

fpr\_dtc, tpr\_dtc, thresholds\_dtc = roc\_curve(y\_valid, y\_score\_dtc) roc\_auc\_dtc = roc\_auc\_score(y\_valid, y\_score\_dtc) plt.figure(figsize=(8, 6))

plt.plot(fpr\_gnb, tpr\_gnb, color='blue', lw=2, label=' aive Bayes ROC curve (AUC = {:.2f})'.format(roc\_auc\_gnb)) plt.plot(fpr\_dtc, tpr\_dtc, color='green', lw=2, label='Decision Tree ROC curve (AUC =

{:.2f})'.format(roc\_auc\_dtc))

plt.plot([0, 1], [0, 1], color='gray', linestyle='--') plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate') plt.title('ROC-AUC Curve') plt.legend(loc='lower right')

plt.show()



***confiusion matrix***

q = y\_valid

pred\_test = dt\_model.predict(x\_valid) pred\_test = pd.DataFrame(pred\_test) y\_valid = pd.DataFrame(y\_valid)

plot\_confusion\_matrix(y\_valid, pred\_test, figsize=(7,4), title='DecisionTree Confusion Matrix', cmap='BuGn') y\_valid = q

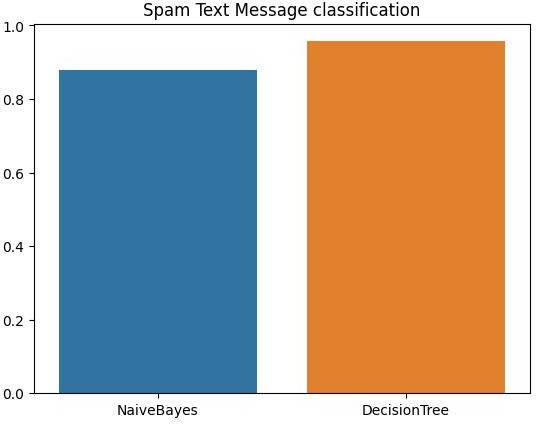
q = y\_valid

pred\_test = nb\_model.predict(x\_valid.toarray()) pred\_test = pd.DataFrame(pred\_test)

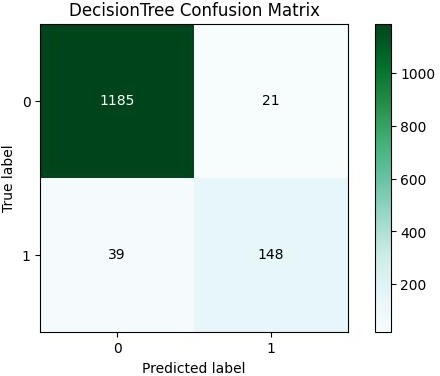
y\_valid = pd.DataFrame(y\_valid)

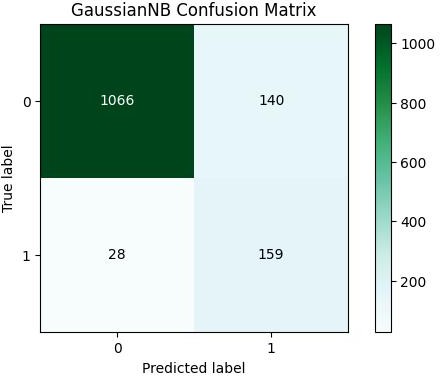
plot\_confusion\_matrix(y\_valid, pred\_test, figsize=(7,4), title='Gaussian B Confusion Matrix', cmap='BuGn') y\_valid = q

**Accuracies of both models :**

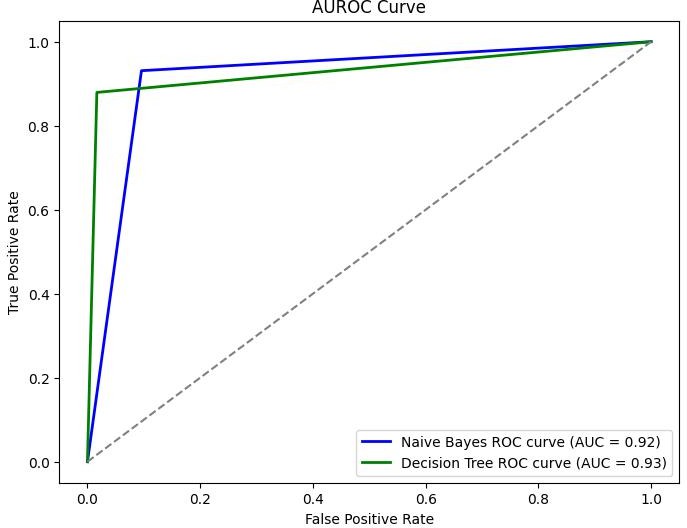


**Confusion Matrix :**





**AUROC:**



**Dataset 5 (Wine Quality Dataset):**



""" wine dataset""" wine\_df.head()

y = wine\_df.pop('quality')

X = wine\_df

x\_train, x\_valid, y\_train, y\_valid = train\_test\_split(X,y,random\_state=10,stratify=y, test\_size=0.25) y\_train.value\_counts(normalize=True)



***using naive bayes***

nb\_model = Gaussian B() nb\_model.fit(x\_train, y\_train)

nb\_accuracy = nb\_model.score(x\_valid, y\_valid)

print(nb\_accuracy)



***using decision tree***

dt\_model = DecisionTreeClassifier() dt\_model.fit(x\_train, y\_train)

dt\_accuracy = dt\_model.score(x\_valid, y\_valid) print(dt\_accuracy) gaussian.append(nb\_accuracy) decision.append(dt\_accuracy)

accuracy = [nb\_accuracy, dt\_accuracy] Models = [' aiveBayes','DecisionTree']

sns.barplot(x=Models,y=accuracy).set(title="Wine Quality Classifier")



***roc - auc***

y\_score\_gnb = nb\_model.predict\_proba(x\_valid)

fpr\_gnb, tpr\_gnb, thresholds\_gnb = roc\_curve(y\_valid.values, y\_score\_gnb.values) roc\_auc\_gnb = roc\_auc\_score(y\_valid, y\_score\_gnb)

y\_score\_dtc = dt\_model.predict\_proba(x\_valid)

fpr\_dtc, tpr\_dtc, thresholds\_dtc = roc\_curve(y\_valid.values, y\_score\_dtc.values) roc\_auc\_dtc = roc\_auc\_score(y\_valid, y\_score\_dtc)

plt.figure(figsize=(8, 6))

plt.plot(fpr\_gnb, tpr\_gnb, color='blue', lw=2, label=' aive Bayes ROC curve (AUC = {:.2f})'.format(roc\_auc\_gnb)) plt.plot(fpr\_dtc, tpr\_dtc, color='green', lw=2, label='Decision Tree ROC curve (AUC =

{:.2f})'.format(roc\_auc\_dtc))

plt.plot([0, 1], [0, 1], color='gray', linestyle='--') plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate') plt.title('ROC-AUC Curve') plt.legend(loc='lower right')

plt.show()



***confiusion matrix***

q = y\_valid

pred\_test = nb\_model.predict(x\_valid) pred\_test = pd.DataFrame(pred\_test) y\_valid = pd.DataFrame(y\_valid)

plot\_confusion\_matrix(y\_valid, pred\_test, figsize=(7,4), title='Gaussian B Confusion Matrix', cmap='BuGn') y\_test= q

q = y\_valid

pred\_test = dt\_model.predict(x\_valid) pred\_test = pd.DataFrame(pred\_test) y\_valid = pd.DataFrame(y\_valid)

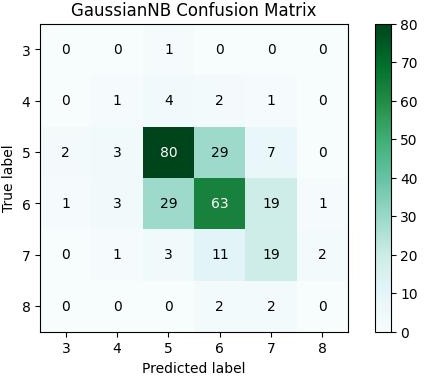
plot\_confusion\_matrix(y\_valid, pred\_test, figsize=(7,4), title='DecisionTree Confusion Matrix', cmap='BuGn')

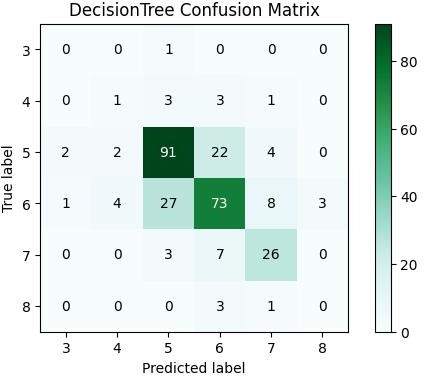
ytest = q

**Accuracies of both models :**

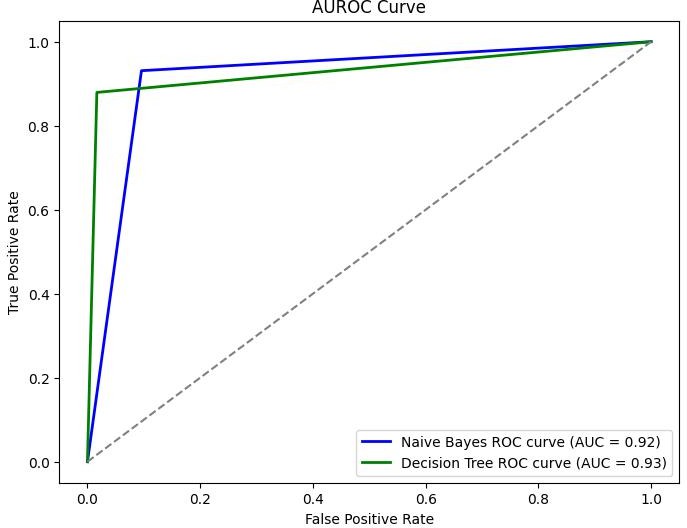


**Confusion Matrix :**





**AUROC:**



**Plot comparison graphs using the results of DT and NB**



def grouped\_barplot(data1, data2, labels, xticklabels= one, title="Grouped Barplot"): positions = np.arange(len(labels))

width = 0.35

***Create the grouped barplot***

plt.figure(figsize=(8, 6))

plt.bar(positions - width/2, data1, width, label='Group 1') plt.bar(positions + width/2, data2, width, label='Group 2') plt.xlabel('Dataset')

plt.ylabel('Acurracy') plt.title(title)

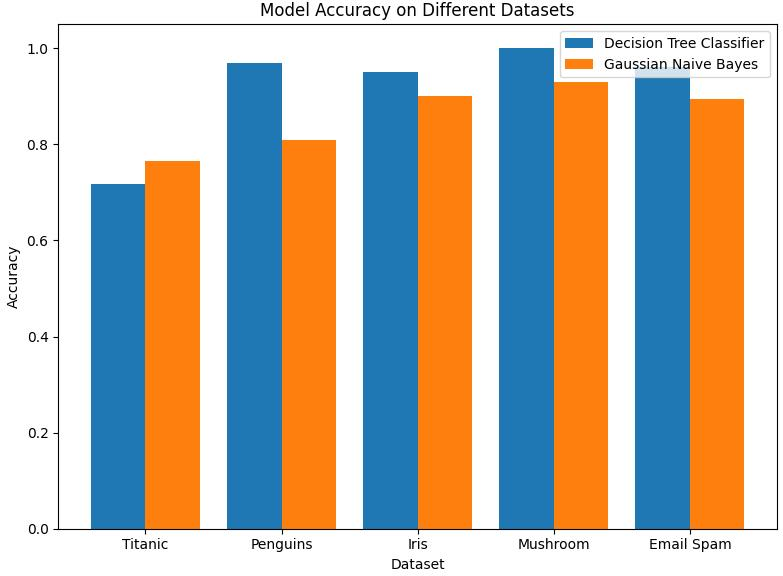
***Set x-axis ticks and labels*** plt.xticks(positions, labels) if xticklabels:

plt.xticks(positions, xticklabels) plt.legend()

plt.show()

category\_labels = ['Titanic', 'Penguin', 'Iris', 'Email spam', 'Wine'] grouped\_barplot(gaussian, decision, category\_labels, title="Model Comparison")

**comparison graphs:**



**Part C:**

**Modify DT/NB to use k-fold cross validation and ensemble models :Kfolds cross validation of 7 folds :**

**Modification Titanic Dataset :**

from sklearn.model\_selection import KFold, cross\_val\_score kf = KFold(n\_splits=7,shuffle=True,random\_state=10)

cv\_score = cross\_val\_score(estimator=nb\_model,X=x\_train,y=y\_train,cv=kf,scoring='accuracy') mean\_cv\_score = np.mean(cv\_score)

print(mean\_cv\_score)



***ensemble tech - RandomForest***

from sklearn.ensemble import RandomForestClassifier

rf\_model = RandomForestClassifier() rf\_model.fit(x\_train, y\_train)

rf\_accuracy = rf\_model.score(x\_valid, y\_valid)

print(rf\_accuracy)



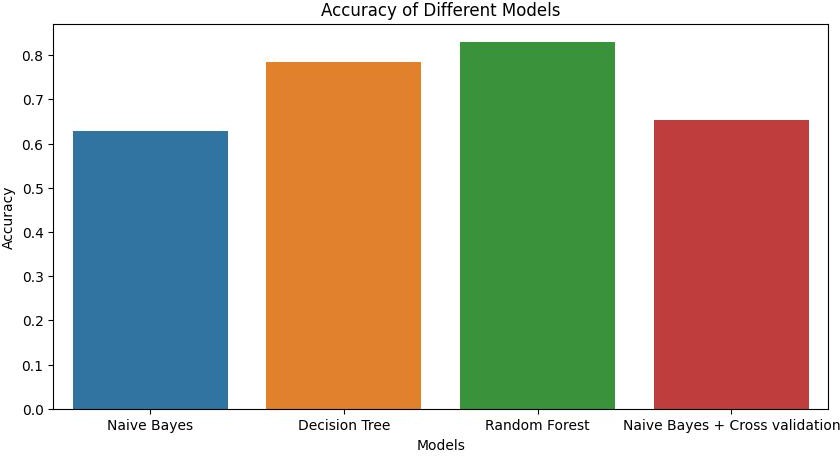
accuracies=[nb\_accuracy,dt\_accuracy,rf\_accuracy,mean\_cv\_score] plt.figure(figsize=(10,5))

models=[' aive Bayes','Decision Tree','Random Forest',' aive Bayes + Cross validation'] sns.barplot(x=models,y=accuracies,).set(title='Part C')

plt.xlabel('Models')

plt.ylabel('Accuracy') plt.title('Accuracy of Different Models')

**Comparison k-fold cross validation and ensemble models :**



**Conclusion:**

**Thus, we have successfully implemented Classification algorithm using Decision Tree ID3 and Naïve Bayes algorithm and performed all the parts**

**Name: Jigar Siddhpura SAPID:** 60004200155

**DIV: C/C2 Branch:** Computer Engineering

**DMW EXP 4**

**CODE:**

from google.colab import drive drive.mount('/content/gdrive')



import numpy as np import pandas as pd

import matplotlib.pyplot as plt import seaborn as sns

from sklearn.model\_selection import train\_test\_split from sklearn.linear\_model import ĮinearRegression from sklearn.preprocessing import ĮabelEncoder

from sklearn.metrics import mean\_squared\_error

df = pd.read\_csv('/content/gdrive/MyDrive/DMW/datasets/StudentsPerformance.csv') df.head()

df['final\_score'] = df.apply(lambda x: (x['math score'] + x['reading score']

+ x['writing score'])/3, axis=1)

df2 = pd.get\_dummies(df, columns=['gender','lunch','parental level of education','race/ethnicity','test preparation course'])

df2 = df2.drop(['math score','reading score','writing score'],axis=1)

*multi-variate*

y = df2['final\_score']

X = df2.drop(['final\_score'],axis=1) xtrain, xtest, ytrain, ytest =

train\_test\_split(X,y,test\_size=0.25,random\_state=10) sns.boxplot(data=df2['final\_score'],orient='h')



model = ĮinearRegression() model.fit(xtrain,ytrain)

score = model.score(xtest,ytest) print(score)

ypred = model.predict(xtest)

sns.scatterplot(data=df,x=df['reading score'],y=df['final\_score'])

*regression line*

m,b = np.polyfit(x=df['reading score'],y=df['final\_score'],deg=1)

X = df['reading score'] plt.plot(X, m\*X+b)

*univariate*

X\_uni = df['reading score'] y\_uni = df['final\_score']

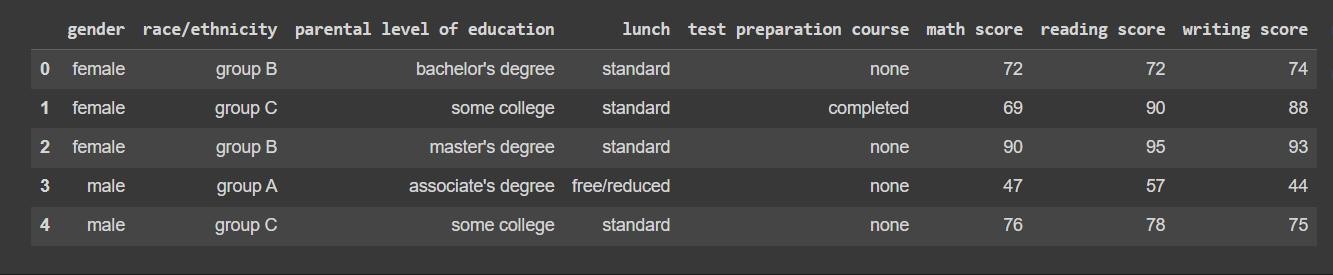
x\_uni\_train, x\_uni\_test, y\_uni\_train, y\_uni\_test = train\_test\_split(X\_uni,y\_uni,test\_size=0.25,random\_state=10) x\_uni\_train = x\_uni\_train.values.reshape(-1,1)

x\_uni\_test = x\_uni\_test.values.reshape(-1,1) uni\_model = ĮinearRegression() uni\_model.fit(x\_uni\_train,y\_uni\_train)

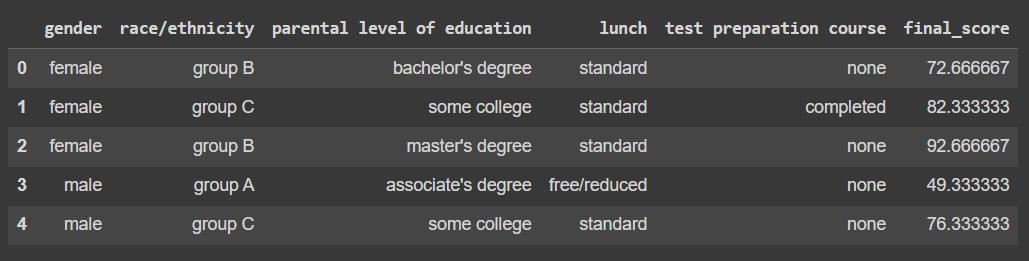
uni\_score = uni\_model.score(x\_uni\_test,y\_uni\_test) print(uni\_score)

y\_uni\_pred = uni\_model.predict(x\_uni\_test)

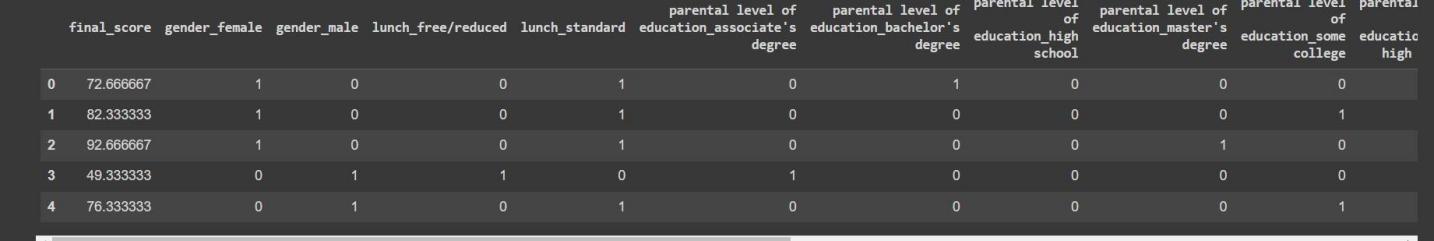
OUTPUT:



head() of the database



df.head() after adding a final score column

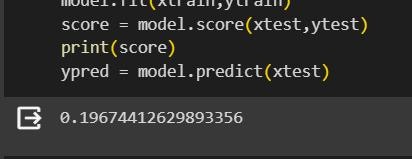


df.head() after applying One-hot encoding to the dataset



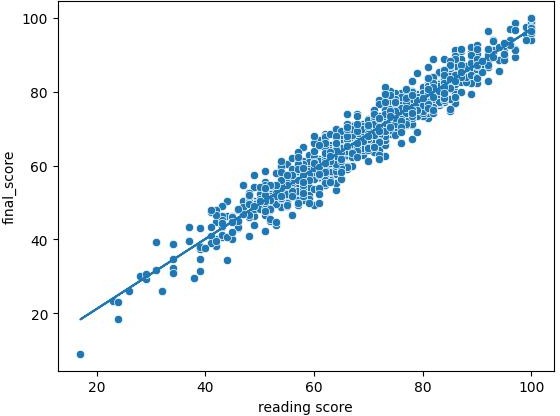
**Considering Multivariate Linear Regression**

**Prediction Score**

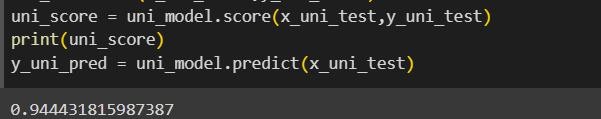


**Now considering Univariate Linear Regression with Reading Score as the feature**

*Scatter plot*



Prediction Score of Univariate LR



**CONCLUSION**:

We have implemented Multivariate and Univariate Linear

Regression on a dataset and have observed the differences in their AccuracyScore and Mean Squared Errors. We observe 19.67% accuracy in the case of

Multivariate whereas in the case ofUnivariate, the accuracy score is 94.43%

94.21% and the Mean Squared Error is 109.48. Therefore we can conclude that using Multivariate Linear Regression is better than using Univariate but nevertheless the efciency of Univariate is still great.

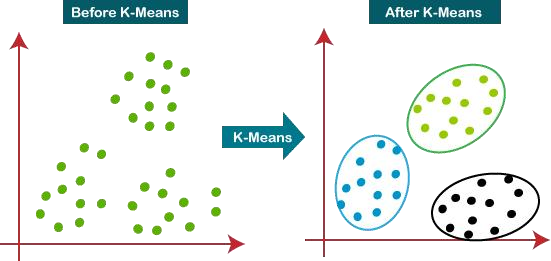
**Name: Jigar Siddhpura SAPID:** 60004200155

**DIV: C/C2 Branch:** Computer Engineering

**DMW EXPERIMENT 5**

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

The below diagram explains the working of the K-means Clustering Algorithm:



Program:

from google.colab mport dr ve dr ve.mount('/content/gdr ve')



mport numpy as np mport pandas as pd

mport matplotl b.pyplot as plt mport seaborn as sns

from sklearn.cluster mport KMeans, Agglomerat veCluster ng from sc py.cluster.h erarchy mport dendrogram

from sklearn.preprocess ng mport StandardScaler from sklearn.metr cs mport s lhouette\_score

df =

pd.read\_csv('/content/gdr ve/MyDr ve/DMW/datasets/customer\_segmentat on.csv') df.head()

df.drop(['ID'], nplace = True, ax s = 1) features = df[df.columns]

scaler = StandardScaler()

scaled = scaler.f t\_transform(features.values) scaled = pd.DataFrame(scaled,columns=df.columns) scaled.head()

data = scaled[['Age','Income']]

*elbow curve*

wcss = {'wcss\_score':[],'no\_of\_clusters':[]} for n range(1,11):

kmeans = KMeans(n\_clusters= ,random\_state=10) kmeans.f t(data) wcss['wcss\_score'].append(kmeans. nert a\_) wcss['no\_of\_clusters'].append( )

plt.f gure(f gs ze=(7,5)) plt.plot(wcss['no\_of\_clusters'],wcss['wcss\_score'],marker='x') plt.t tle("Elbow Method to determ ne number of clusters(K)") plt.xlabel("K (no. of clusters)")

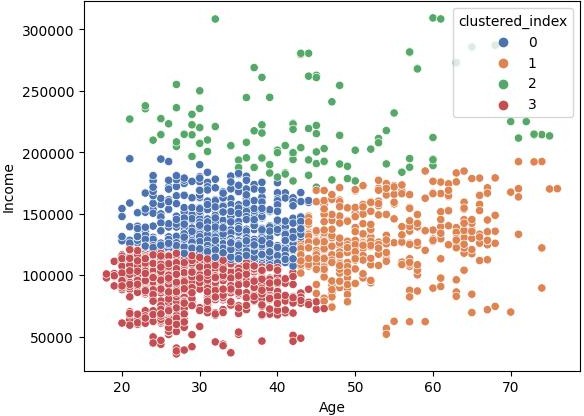
plt.ylabel("WCSS (W th ng Cluster Sum of Squared d stance )")

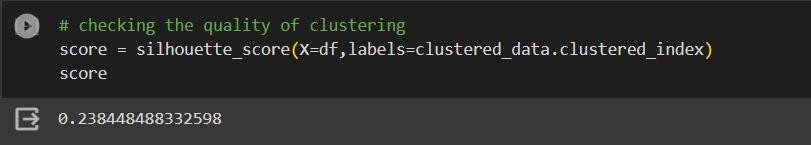
plt.show() kmeans=KMeans(n\_clusters=4,random\_state=42) kmeans.f t(data)

pred ct on = kmeans.f t\_pred ct(data) clustered\_data = df.copy() clustered\_data['clustered\_ ndex'] = pred ct on

sns.scatterplot(x=clustered\_data.Age, y=clustered\_data.Income, hue=clustered\_data.clustered\_ ndex, palette='deep')

**Output:**





**Program :**



*Hierarchichal clustering*

from sc py.cluster.h erarchy mport dendrogram,l nkage data = clustered\_data[['Age','Income']]

plt.f gure(f gs ze=(10,7)) plt.t tle("Dendogram")

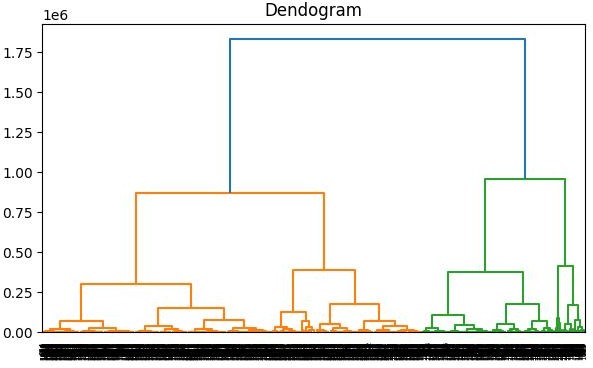
dend = dendrogram(l nkage(data,method='ward')) cluster =

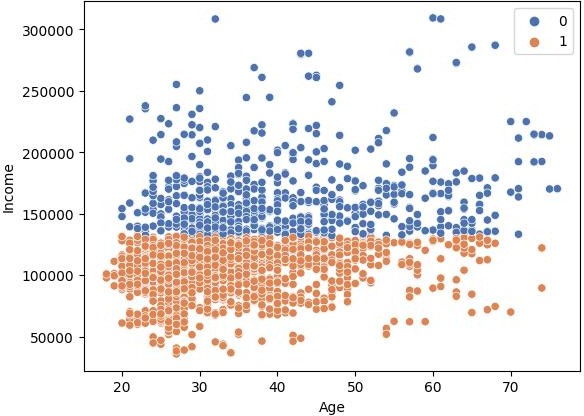
Agglomerat veCluster ng(n\_clusters=2,aff n ty='eucl dean',l nkage='ward') labels\_ = cluster.f t\_pred ct(data)

sns.scatterplot(x=data.Age, y=data.Income, hue=labels\_, palette='deep

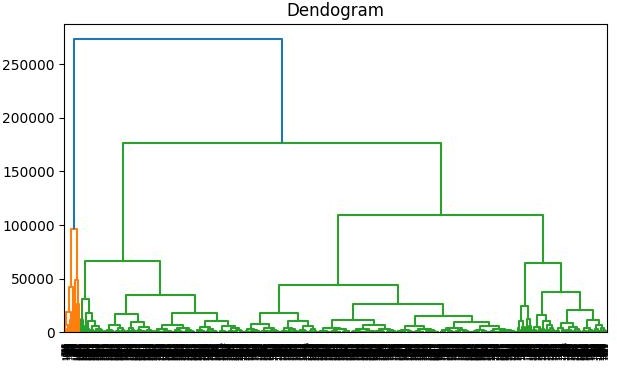
**Output:**

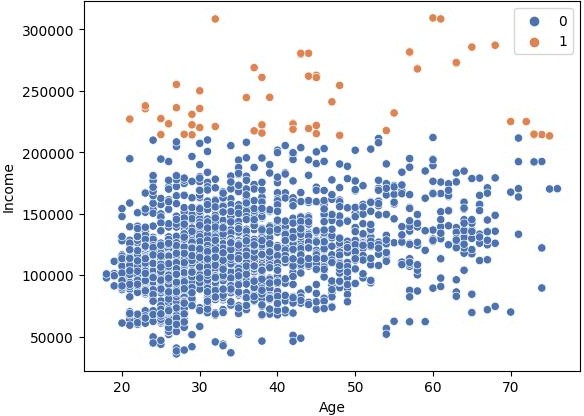
* ***Ward Hierarchical Clustering***



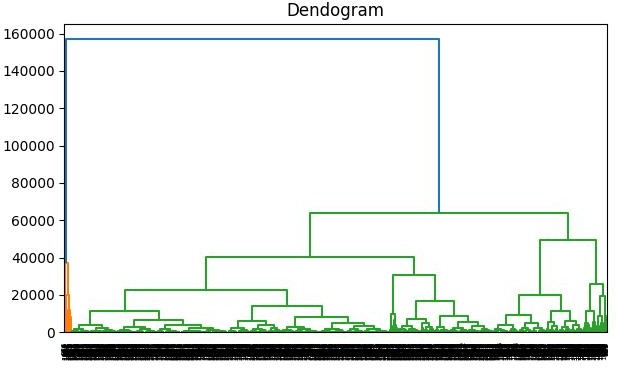


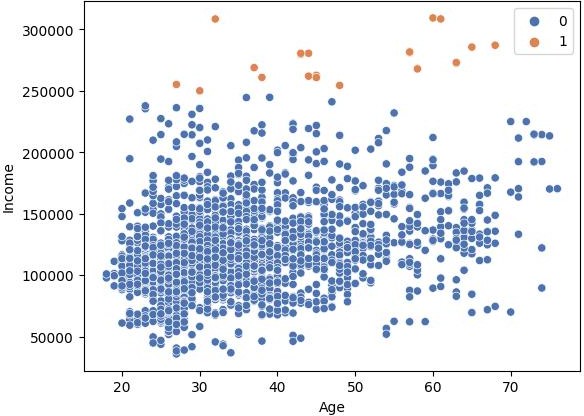
* ***Complete Hierarchical Clustering***





* ***Average Hierarchical Clustering***



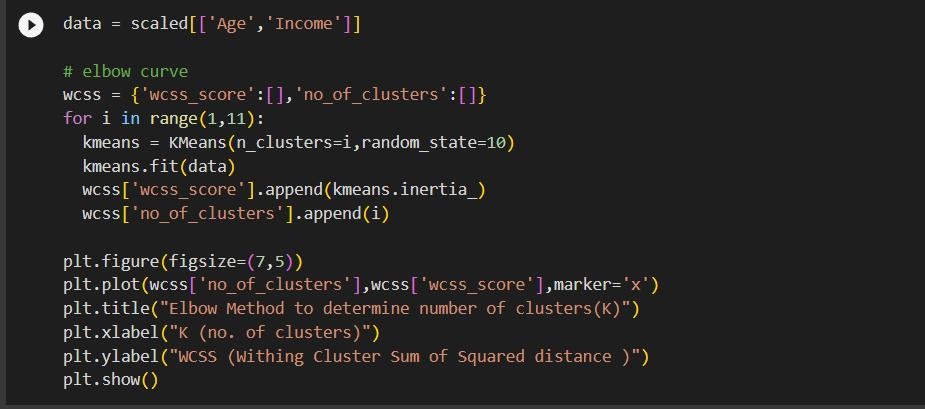


**Part B:**

* 1. Plot Elbow Method and suggest optimal number of clusters

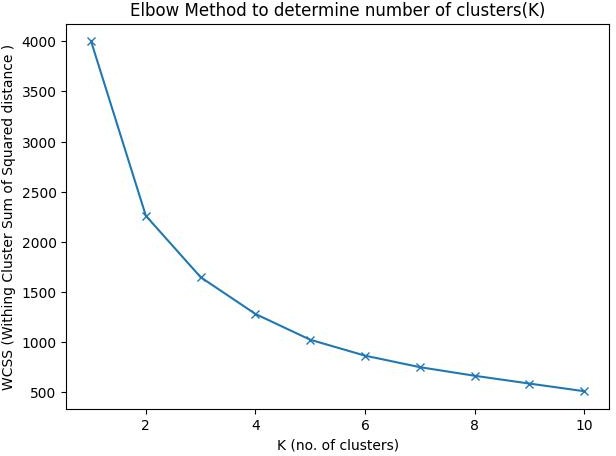
A fundamental step for any unsupervised algorithm is to determine the optimal number of clusters into which the data may be clustered. The **Elbow Method** is one of the most popular methods to determine this optimal value of k.

**Program:**



To determine the optimal number of clusters, we have to select the value of k at the “elbow” i.e. the point after which the distortion/inertia start decreasing in a linear fashion. Thus, for the given data, we conclude that the optimal number of clusters for the data is 3.

**Output :**



**Conclusion:** Thus, we have successfully implemented Clustering Algorithm Using

1. k-means 2. Hierarchical(ward/complete/average)

**Name: Jigar Siddhpura SAPID:** 60004210155

**DIV: C/C2 Branch:** Computer Engineering

**DMW - EXPERIMENT 6 - ASSOCIATION RULE MINING**

**Part A:**

Read min\_support and confidence from the user Program Apriori algorithm using inbuilt functions. Print the association rules

**Code:**

**import numpy as np import pandas as pd**

**from mlxtend.frequent\_patterns import apriori, association\_rules from mlxtend.preprocessing import TransactionEncoder**

**df = pd.read\_csv('/content/gdrive/MyDrive/DMW/datasets/GroceryStoreDataSet.csv',sep=',',names**

**=['products']) df.head()**

**#one hot encoding**

**data = list(df['products'].apply(lambda x:x.split(','))) encoder = TransactionEncoder()**

**encoded\_data = encoder.fit\_transform(data)**

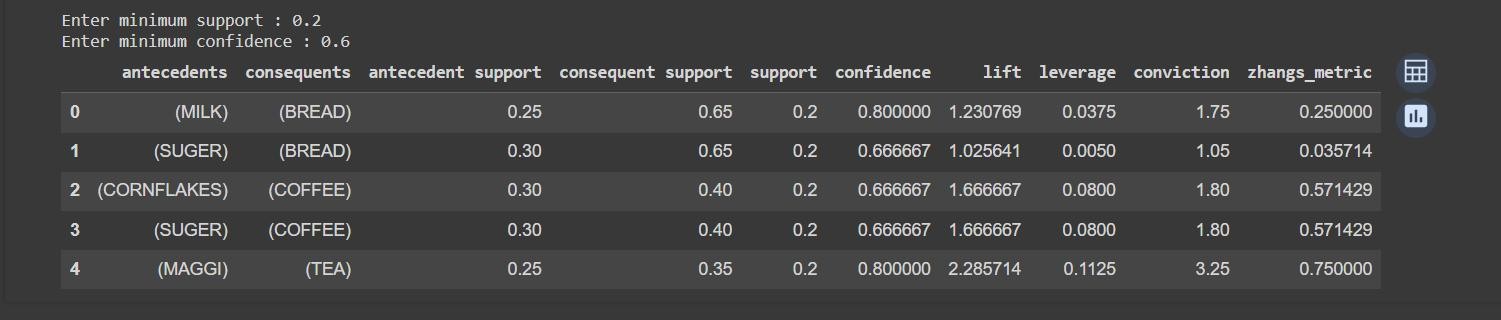
**df2 = pd.DataFrame(encoded\_data,columns=encoder.columns\_) df2.replace(True,1,inplace=True) df2.replace(False,0,inplace=True)**

**frq\_items = apriori(df2,min\_support=min\_support,use\_colnames=True)**

**rules = association\_rules(frq\_items,metric='confidence',min\_threshold=min\_conf) print(f"Enter minimum support : 0.2")**

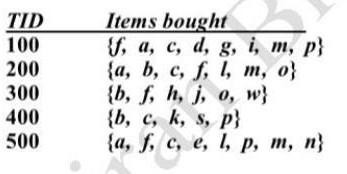
**print(f"Enter minimum confidence : 0.6") rules**

**Output:**



**Part B:**

Program FP tree using inbuilt functions for the following dataset



Print the frequent patterns generated.

**Code:**

from mlxtend.frequent\_patterns.fpgrowth import fpgrowth

dataset = [['f', 'a', 'c', 'd', 'g', 'i', 'm', 'p'],

['a', 'b', 'c', 'f', 'l', 'm', 'o'],

['b', 'f', 'h', 'j', 'o', 'w'],

['b', 'c', 'k', 's', 'p'],

['a', 'f', 'c', 'e', 'l', 'p', 'm', 'n']]

encoder = TransactionEncoder()

encoded\_data = encoder.fit\_transform(dataset)

fp\_df = pd.DataFrame(encoded\_data,columns=encoder.columns\_)

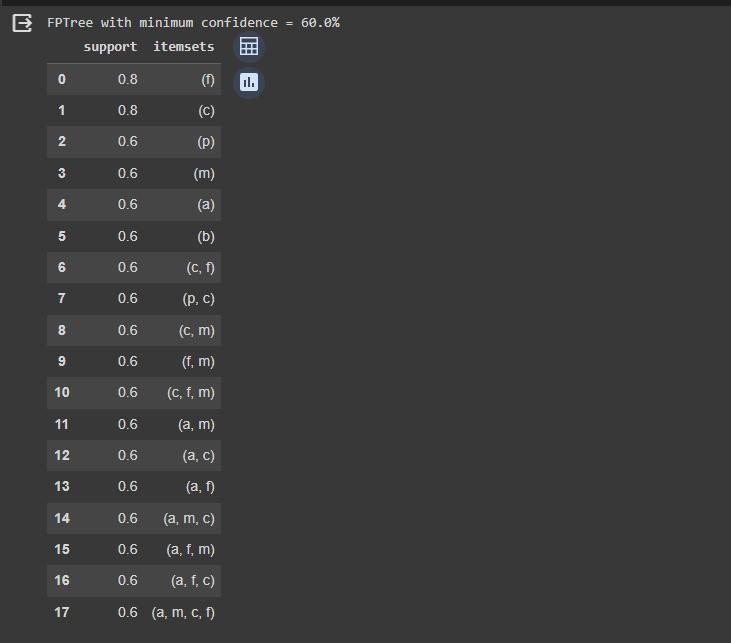
pattern = fpgrowth(fp\_df,min\_support=min\_conf\_fp,use\_colnames=True) print(f"FPTree with minimum confidence = {min\_conf\_fp\*100}%")

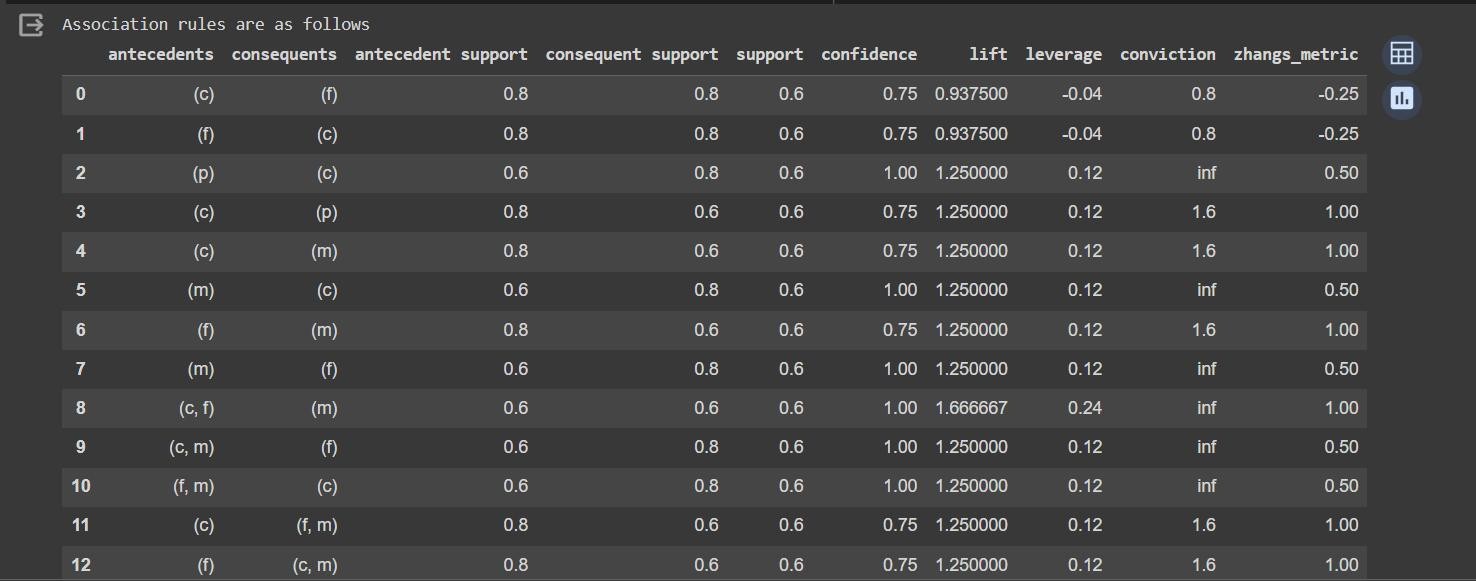
Pattern

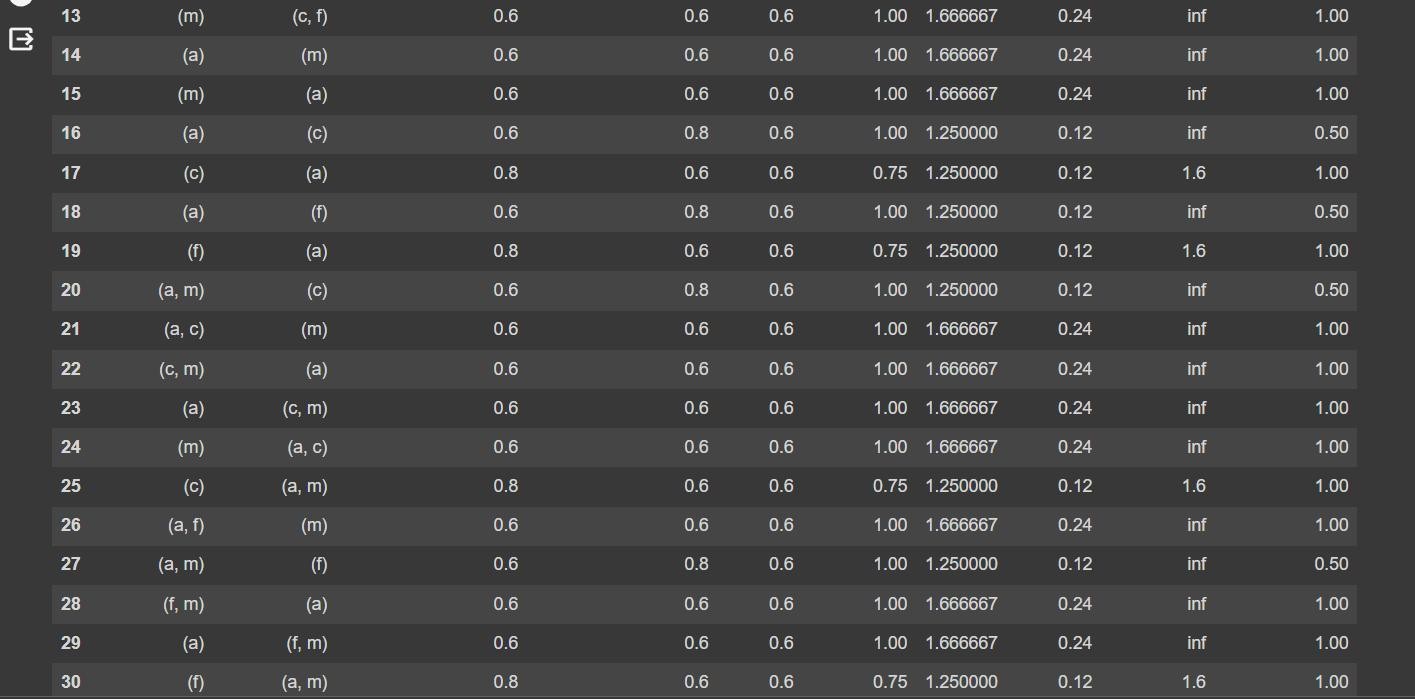
rules = association\_rules(pattern,metric='confidence',min\_threshold=min\_conf\_fp) print(f"Association rules are as follows")

rules

**Output:**







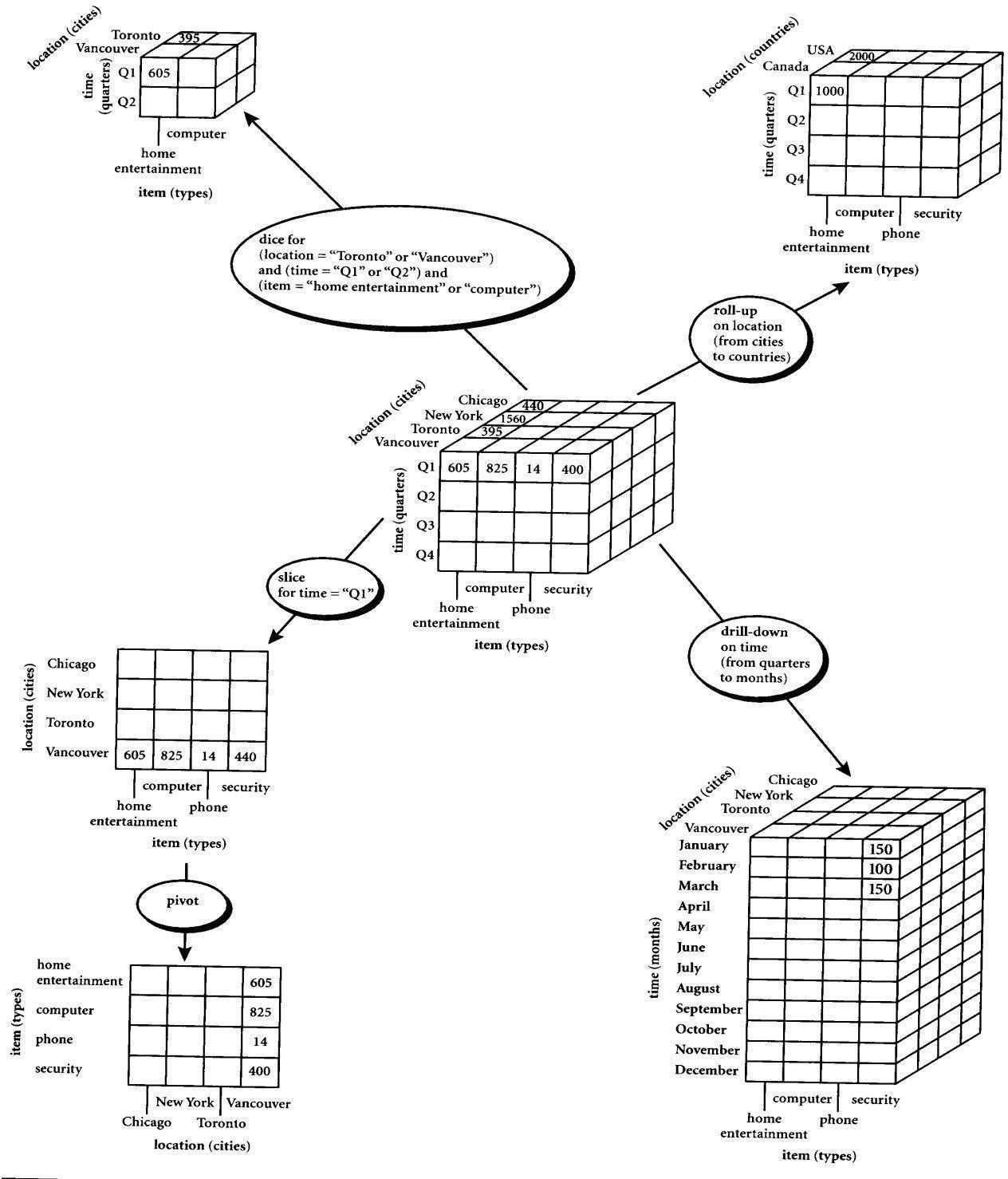
**Conclusion:**

Implemented Apriori and algorithm for a market basket analysis dataset and made and FP Tree for the given dataset. Apriori is a Join-Based algorithm and FP-Growth is Tree-Based algorithm for frequent itemset mining or frequent pattern mining for market basket analysis.

**Name: Jigar Siddhpura SAPID:** 60004210155

**DIV: C/C2 Branch:** Computer Engineering

**DMW - Experiment 7**



**EXERCISE 1**

Consider a data Warehouse for a hospital, where there are three dimensions:

1. Doctor
2. Patient
3. Time

With two measures

* 1. Count
  2. Charge

Where Charge is the fee that the Doctor charges a patient for a visit. Using the above example describe the following operations:

* + 1. Slice
    2. Dice
    3. Roll Up
    4. Drill Down (v) Pivot

**NOTE: Assume data according to the dimensions and measures and explore individual tasks diagrammatically.**

**EXERCISE 2**

**To create Pivot of Table using MS Excel**

**Follow these steps …**

* Start with M.S Excel.
* In excel sheet create 4 columns PRODUCT, ORIGIN, DAY OF SALE, SOLD UNITS (FACT COLUMN).
* Insert around fifty rows of data.
* Save the table data.
* Go to Insert Tab-> click on Pivot Table-> New work sheet-> Ok.
* Right side you will find pivot table fields.

It contains all columns of our table that we created. Select product in rows,

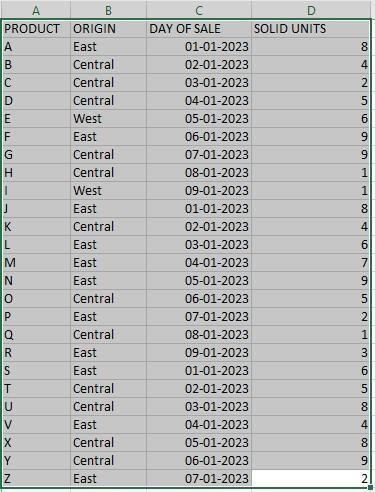
Days in column,

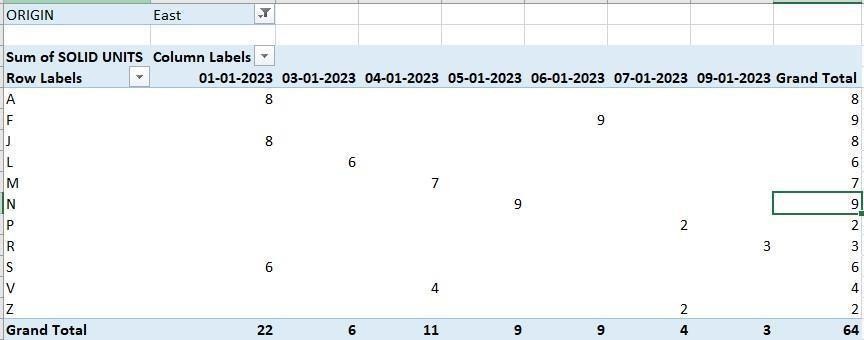
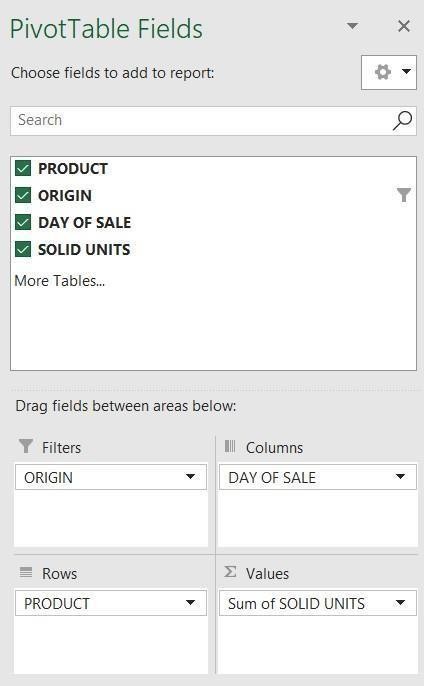
Unit sold in ∑ values.

Later apply filter using Origin.

Also, we can flip the rows & columns or combine together as rows only to see different views of same data.

Dataset:





**Conclusion:** Thus, we performed various OLAP instructions.

**Name: Jigar Siddhpura SAPID:** 60004210155

**DIV:** C/C2 **Branch:** Computer Engineering

**DMW - Experiment 8**

**Code:**

mport numpy as np



def page\_rank\_algor thm(graph,damp ng\_factor): outgo ng=d ct()

ncom ng\_nodes=d ct() coeff c ents= d ct()

for n range(len(graph)): outgo ng[ ]=0

for , node n enumerate(graph): for edge n node:

f edge:

outgo ng[ ] += 1

for n range(len(graph)): temp=[]

for node n graph: f node[ ]:

temp.append(node) ncom ng\_nodes[ ] = temp

coeff c ents\_l st = []

for ,node n enumerate(graph): temp = []

for j,other\_node n enumerate(graph):

f other\_node n ncom ng\_nodes[ ]: temp.append(damp ng\_factor\*(1.0/outgo ng[j]))

el f == j: temp.append(-1)

else:

temp.append(0) coeff c ents[ ]= temp

coeff c ents\_l st.append(temp)

constant\_matr x = []

for n range(len(graph)):

constant\_matr x.append(damp ng\_factor-1)

pageranks = np.l nalg.solve(np.array(coeff c ents\_l st),np.array(constant\_matr x))

*print()*

for ,rank n enumerate(pageranks):

pr nt('Page Rank of {} s {:.4f}'.format(chr(65+ ), rank))



f name ==" ma n ":

n = nt( nput('Enter the number of nodes : ')) d = float( nput('Enter the damp ng factor : '))

*graph repr connected points*

graph = []

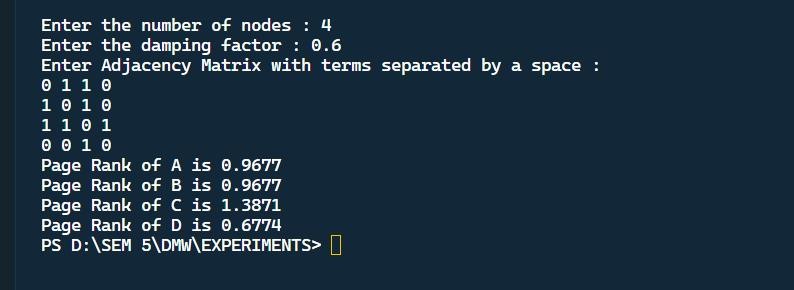
*graph = [[0, , ,0],[ ,0, ,0],[ , ,0, ],[0,0, ,0]]*

pr nt('Enter Adjacency Matr x w th terms separated by a space : ') for n range(n):

temp\_l st = nput().spl t(' ') graph.append(l st(map( nt,temp\_l st)))

page\_rank\_algor thm(graph,d)

**Output:**



**Conclusion:**

Learnt about page rank algorithm in web structure mining and implemented it in python for the graph:

**Name: Jigar Siddhpura SAPID:** 60004210155

**DIV:** C/C2 **Branch:** Computer Engineering

**DMW - Experiment 9**

**Code:**

import networkx as nx

import matplotlib.pyplot as plt

import numpy as np

graph\_matrix = np.array([

[0, 0, 0, 1, 0, 0, 0, 0], *# A -> D*

[0, 0, 1, 0, 1, 0, 0, 0], *# B -> E, C*

[1, 0, 0, 0, 0, 0, 0, 0], *# C -> A*

[0, 0, 1, 0, 0, 0, 0, 0], *# D -> C*

[0, 1, 1, 1, 0, 1, 0, 0], *# E -> B, C, D, F*

[0, 0, 1, 0, 0, 0, 0, 1], *# F -> C, H*

[1, 0, 1, 0, 0, 0, 0, 0], *# G -> A, C*

[1, 0, 0, 0, 0, 0, 0, 0], *# H -> A*

])

G = nx.DiGraph()

labels = {}

for i in range(len(graph\_matrix)):

    node\_label = chr(ord('A') + i)

    labels[i] = node\_label

    G.add\_node(i, label=node\_label)

    for j in range(len(graph\_matrix[i])):

        if graph\_matrix[i][j] == 1:

            G.add\_edge(i, j)

plt.figure(figsize=(10, 10))

pos = nx.spring\_layout(G)

nx.draw\_networkx(G, pos=pos, with\_labels=True, labels=labels)

hubs, authorities = nx.hits(G, max\_iter=50, normalized=True)

print("Hub Scores:")

for key, value in hubs.items():

    print(f'{labels[key]}: {value}')

print()

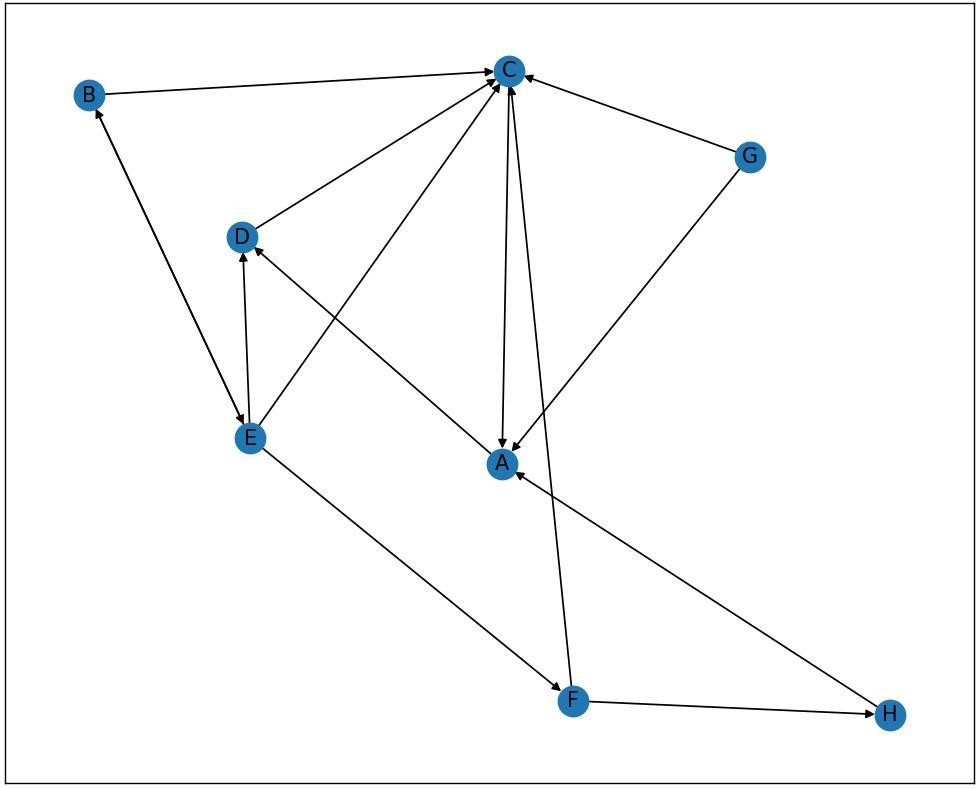
print("Authority Scores:")

for key, value in authorities.items():

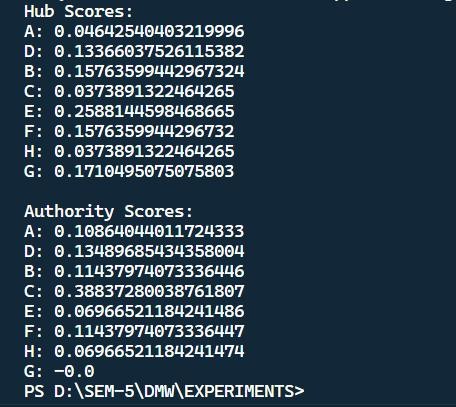
    print(f'{labels[key]}: {value}')

plt.show()

**Graph:**



**Output:**



**Name: Jigar Siddhpura SAPID:** 60004210155

**DIV:** C/C2 **Branch:** Computer Engineering

**DMW - Experiment 10**

**Code:**

import numpy as np

from sklearn\_extra.cluster import KMedoids

from sklearn.datasets import make\_blobs

import matplotlib.pyplot as plt

X, \_ = make\_blobs(n\_samples=3000, centers=3, cluster\_std=1, random\_state=42)

# above line of code generate some sample spatial clustered data

# n\_samples: The total number of points equally divided among clusters.

# centers: The number of centers to generate, or the fixed center locations.

# cluster\_std: The standard deviation of the clusters. Larger values spread out the clusters.

# random\_state: Seed for random number generation to ensure reproducibility.

def clarans(X, n\_clusters, num\_local, max\_neighbor):

    kmedoids = KMedoids(n\_clusters=n\_clusters, method='alternate', max\_iter=1)

    best\_cost = float('inf')

    best\_medoids = None

    for \_ in range(num\_local):

        kmedoids.fit(X)

        medoids = kmedoids.medoid\_indices\_

        cost = np.sum(np.min(X[medoids] - X[:, np.newaxis], axis=2), axis=1).mean()

        if cost < best\_cost:

            best\_cost = cost

            best\_medoids = medoids

return best\_medoids

k = 3 #<- no. of cluster

num\_local = 10

max\_neighbor = 10

medoids = clarans(X, k, num\_local, max\_neighbor)

# plotting the results

plt.scatter(X[:, 0], X[:, 1], c='blue', marker='o', s=30, label='Data Points')

plt.scatter(X[medoids, 0], X[medoids, 1], c='red', marker='o', s=100, label='Medoids')

plt.title('CLARANS Clustering')

plt.legend()

plt.show()

)

**Output :**

