# DATA MINING AND WAREHOUSE

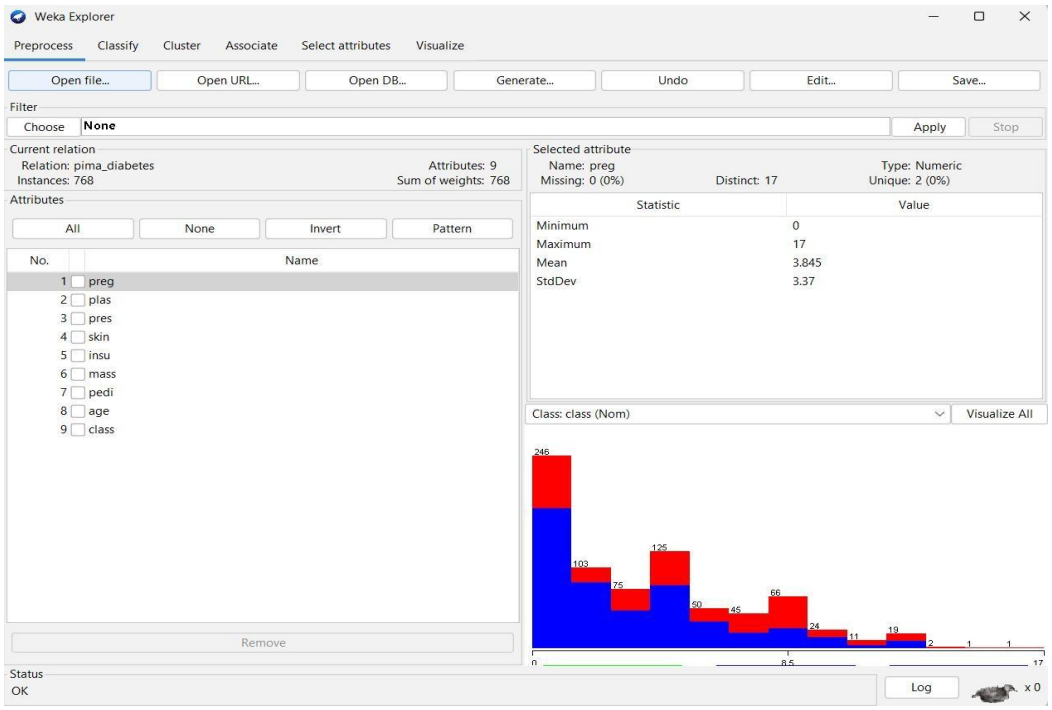
## PRACTICAL 1

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

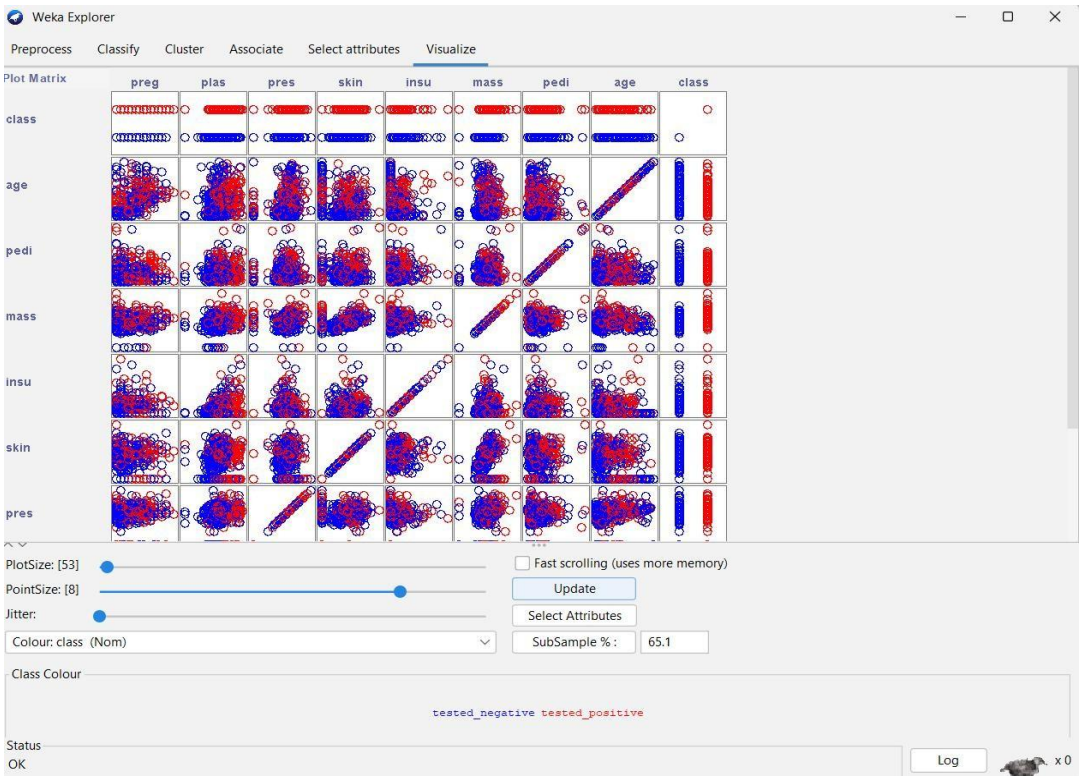

Program
Visualization
Tools
Help
Weka GUI Chooser
Applications
Explorer
Experimenter
KnowledgeFlow
Workbench
Simple CLI
WEKA
THE UNIVERSITY OF
WAIKATO
NEW ZEALAND
Waikato Environment for Knowledge Analysis
Version 3.8.6
(c) 1999 - 2022
The University of Waikato
Hamilton, New Zealand

# 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



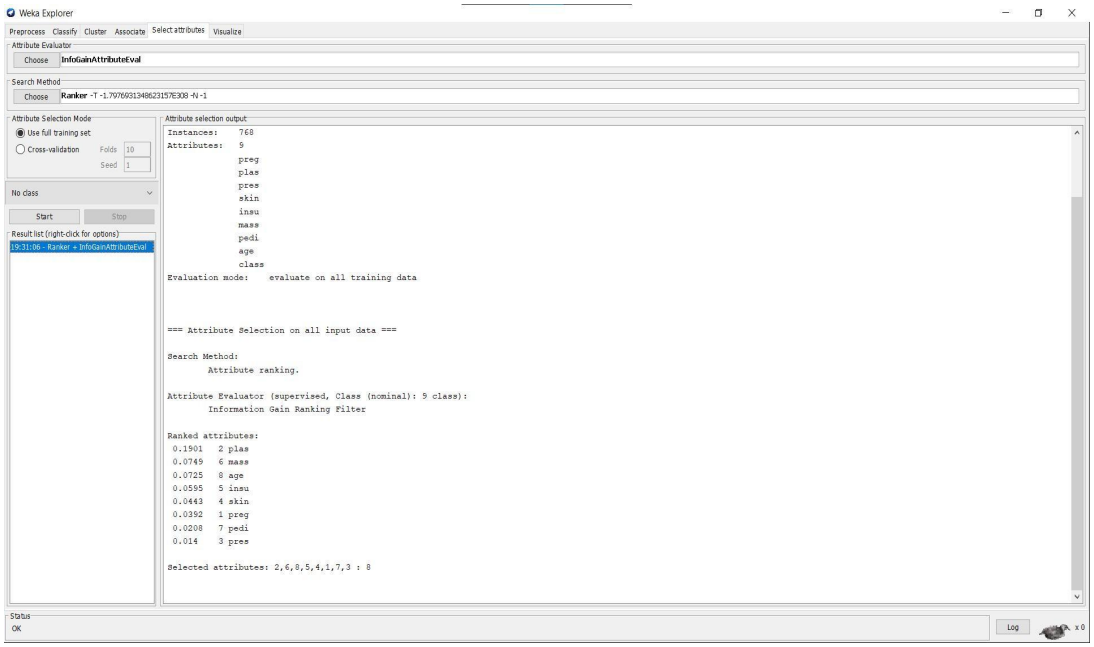

We have loaded the ***Diabetes*** dataset into Weka and have visualized the various attributes. The histograms for each attribute display the variation in values within that attribute and can be viewed for each attribute.



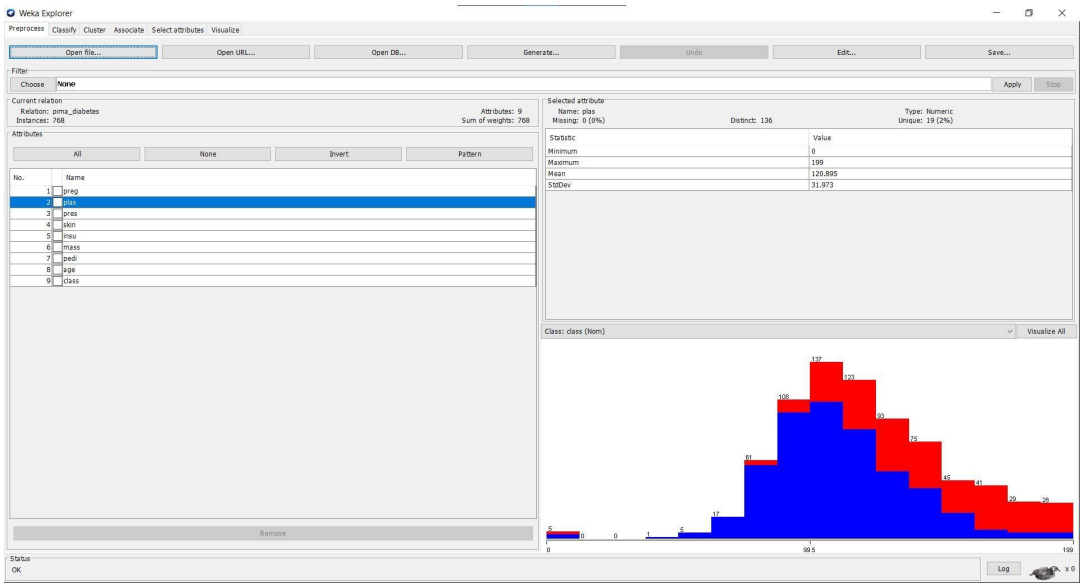

Here we can visualise the scatter plots for each of the attributes of the dataset taken in pairs. By observing the scatter plot, we can infer the following:

a. Body mass index (weight in kg/ (height in m) ^2) vs Triceps skin fold thickness (mm) – There is a positive correlation between these attributes
b. 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
c. Diabetes pedigree function vs Triceps skin fold thickness (mm) – There is no correlation between these attributes and the plot is sparsely distributed

2. **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.



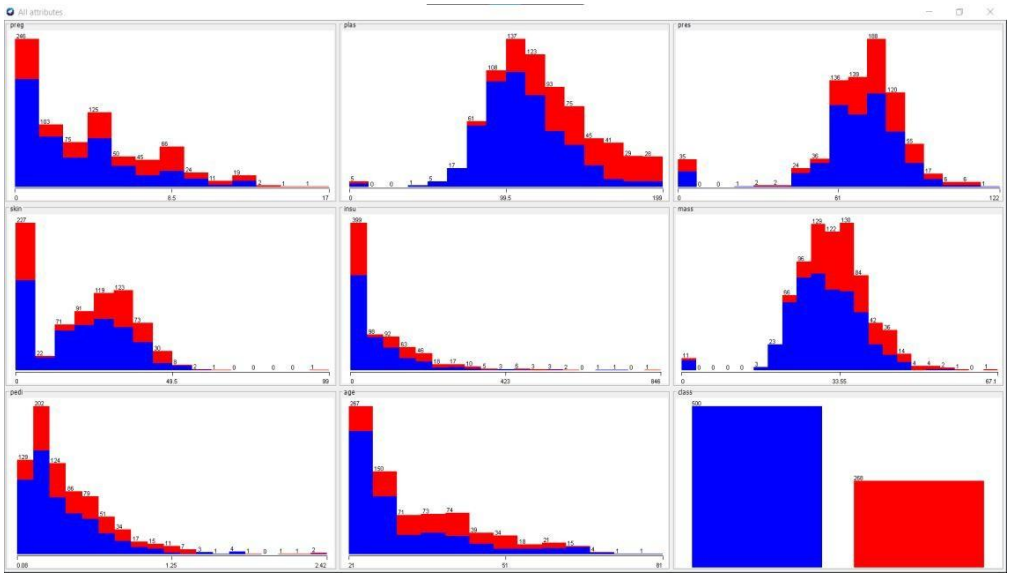

We now switch to the Select Attribute tab and have used the InfoGainAttributeEval method with the attribute Ranker. From the above results, the attribute ‘plas’ which stands for ‘Plasma glucose concentration 2 hours in an oral glucose tolerance test’ is the highest ranked with a score of 0.1901. This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




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

3. **Pre-processing**:

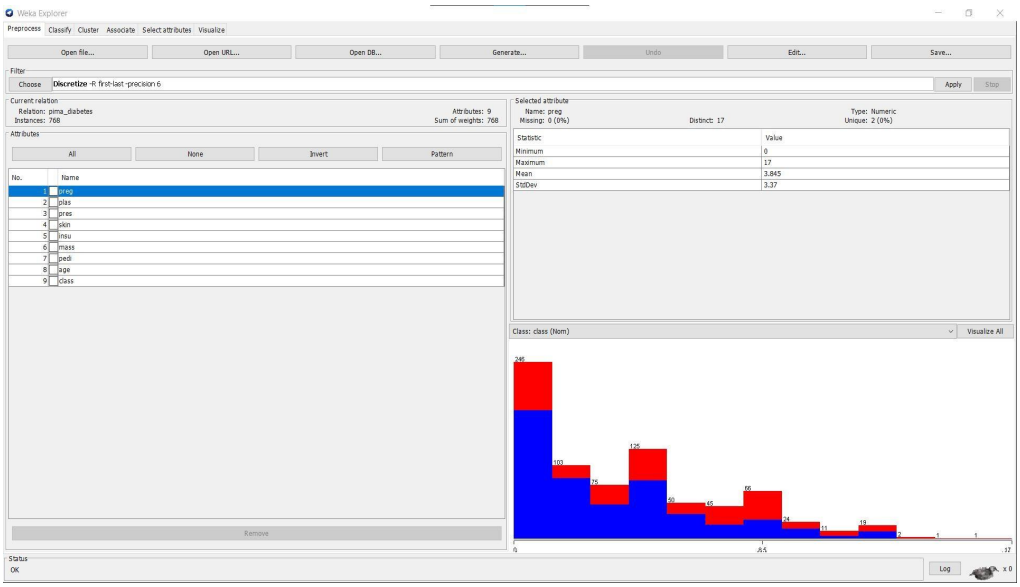
   a. **Visualize All**: Select this button to visualize histograms of all attributes.



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.

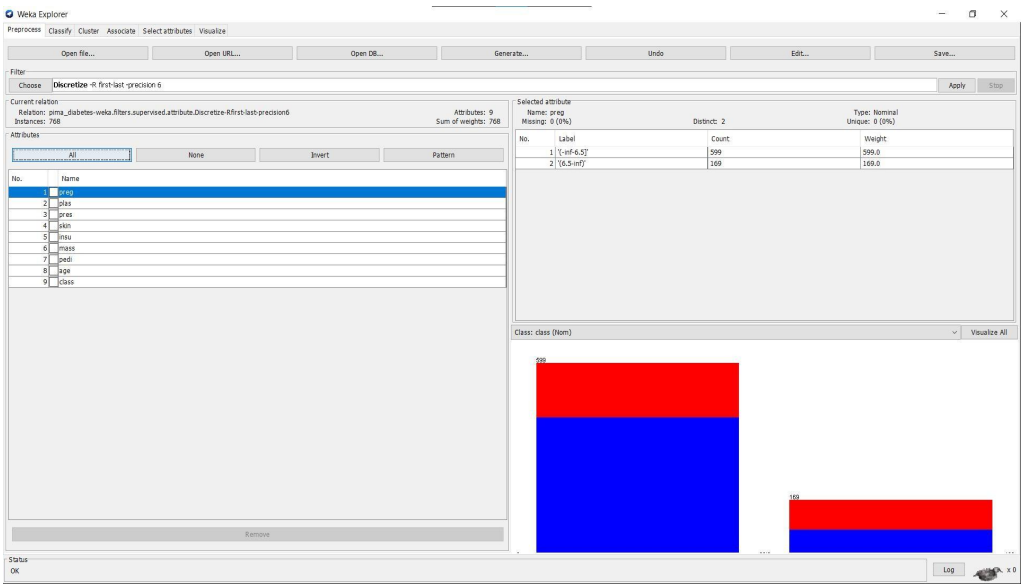
b. **Filter**: Choose Discretization under Unsupervised and Supervised methods. Observe the discretization and the outliers.

*Before*:



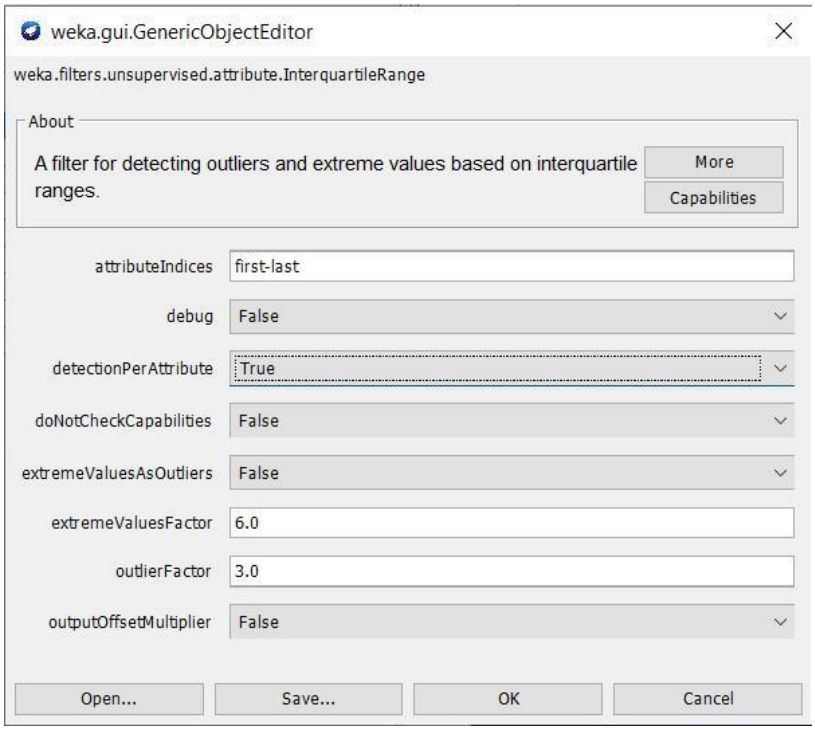
The attribute value before discretization shows a continuous histogram with varying values and no of instances.

*After*:

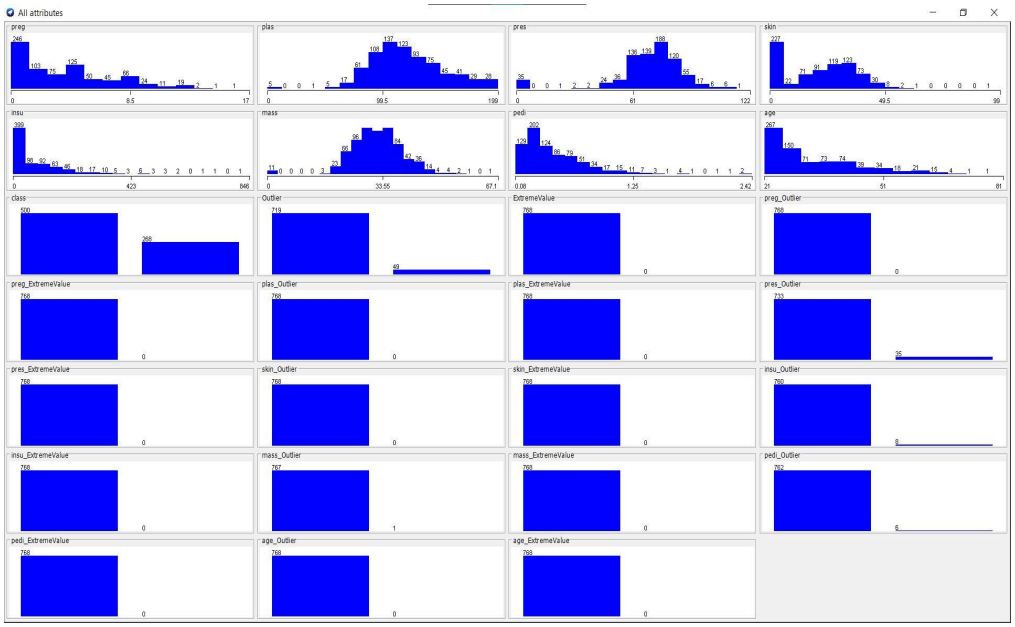


Here we have applied unsupervised discrete filtering to the dataset. For each attribute discrete buckets are created, and the instances are separated into their appropriate bucket. The attribute here is nominal and hence the bucket is split based on their values and extends from -infinity to 6.5 and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

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



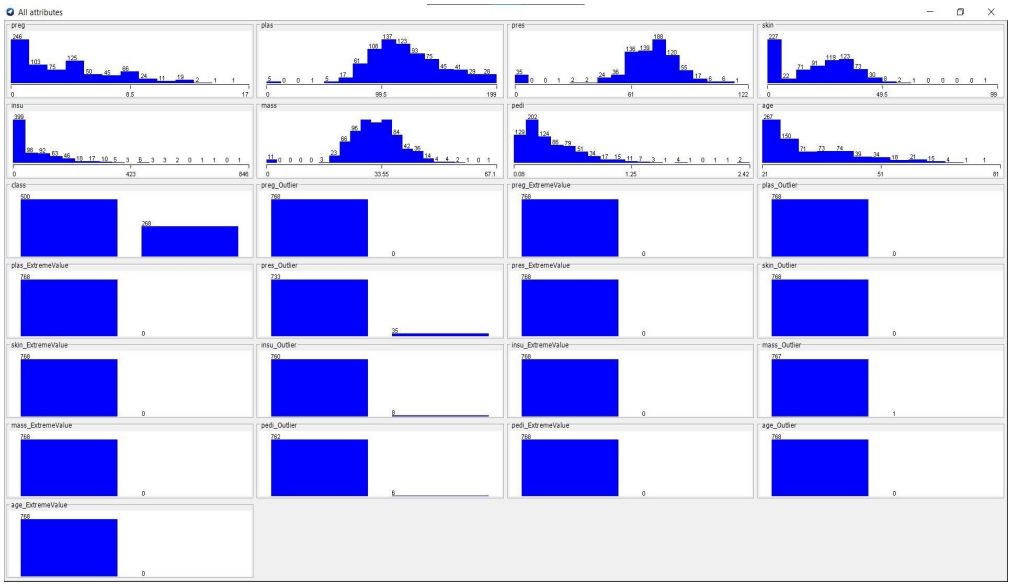

These are the settings that have been applied for IQR.




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.

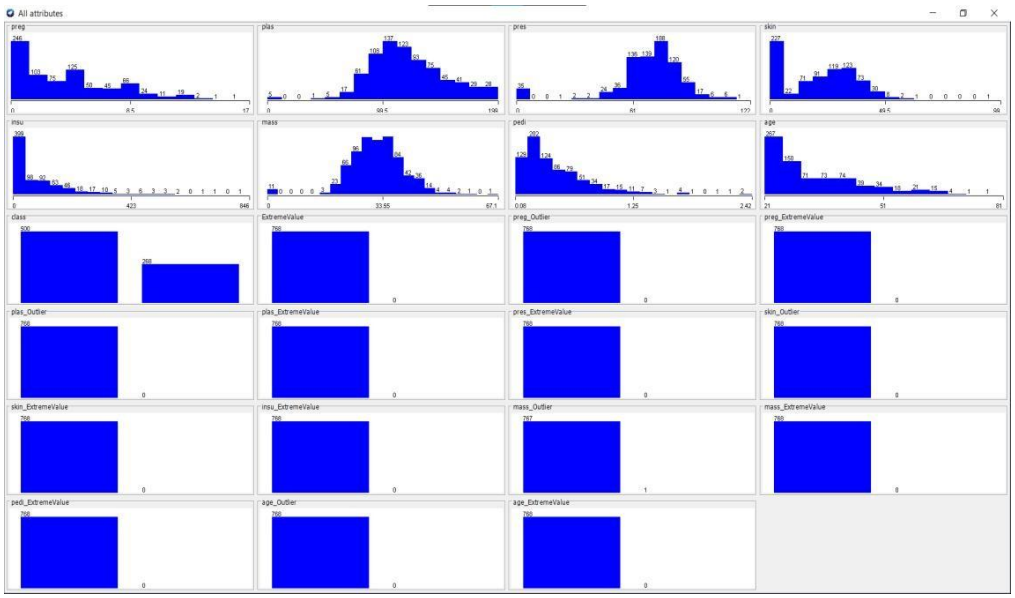
d. **Remove the value**: Remove instances with outlier values and show the screenshots of dataset before and after the removal.

*Before:*



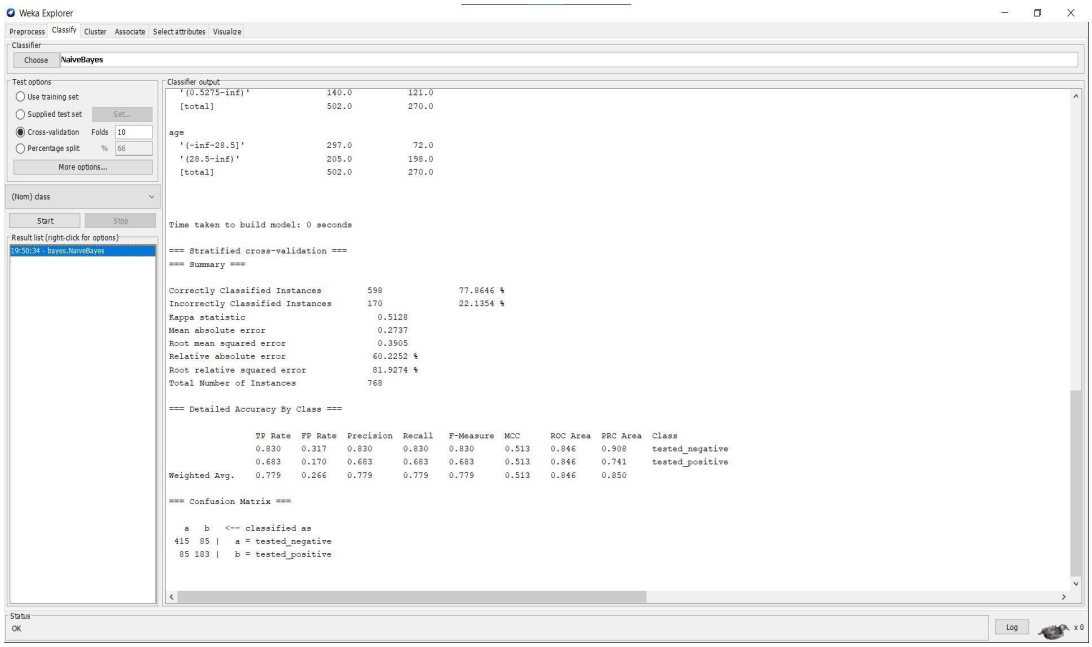

From the above IQR plot, we can determine the attributes that have the outliers. From this dataset, the attributes 'predi', 'insu' and 'pres' have outliers with a combined total of 49 instances.

*After:*

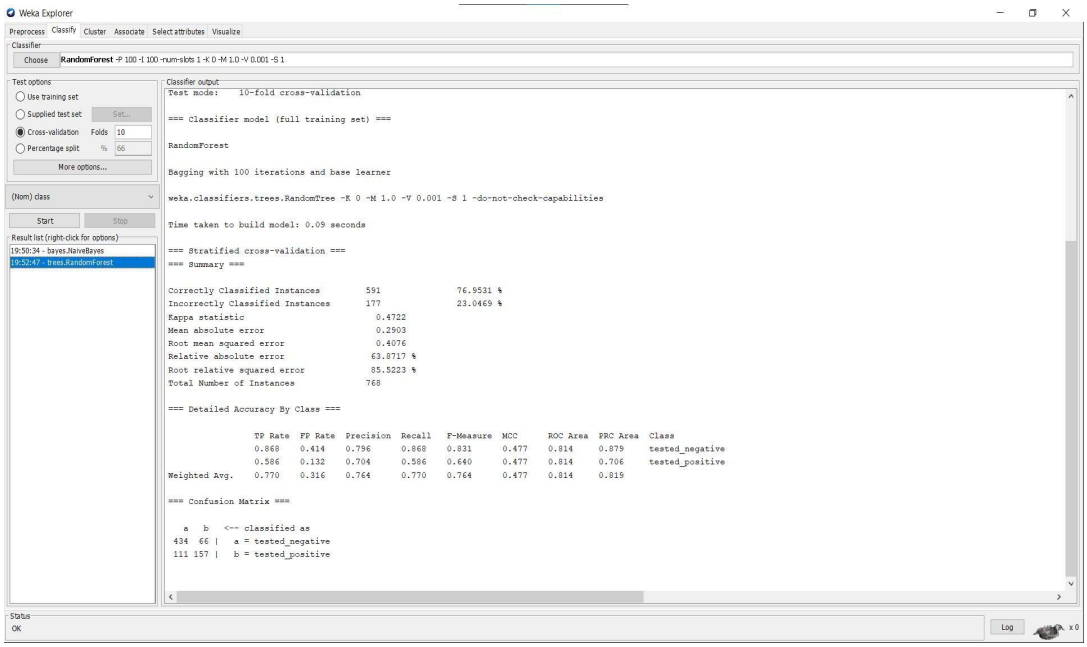



We can then use the attributes menu to remove those instances from the respective attributes and again visualising the attributes graphs shows that the outliers in the dataset have been removed.

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

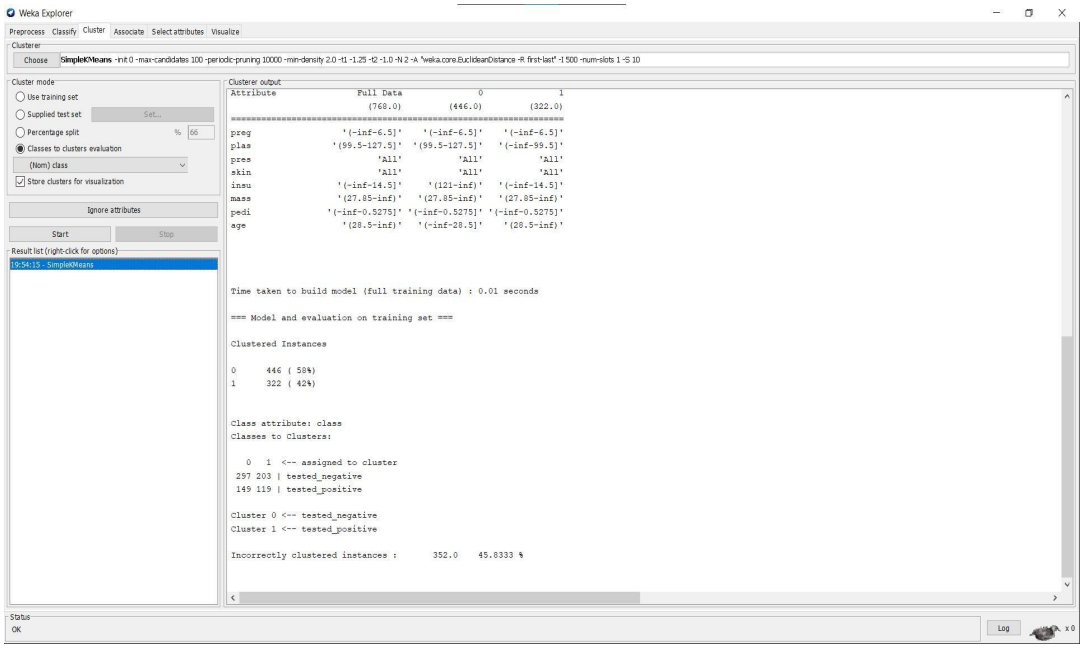



We have performed Naïve Bayes classification on the dataset. The Naïve bayes classifier correctly classified 77.86% of the instances while 22.13% were incorrectly classified.

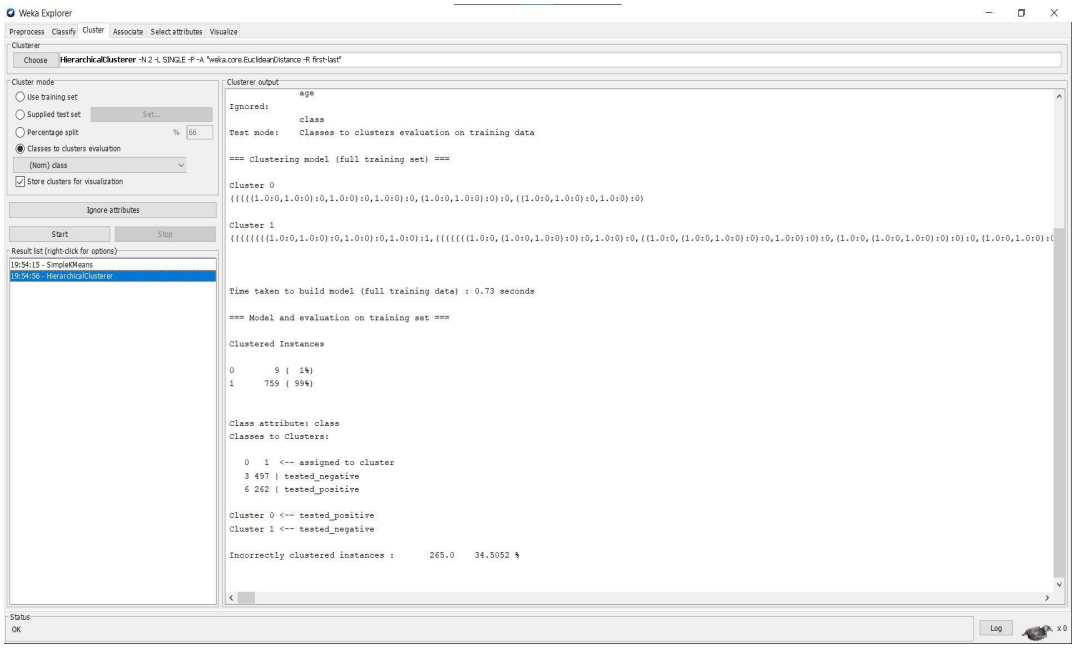



We have performed Random Forest (Decision Tree/Rule Based) classification on the dataset. The Random Forest classifier correctly classified 76.95% of the instances while 23.04% were incorrectly classified.

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

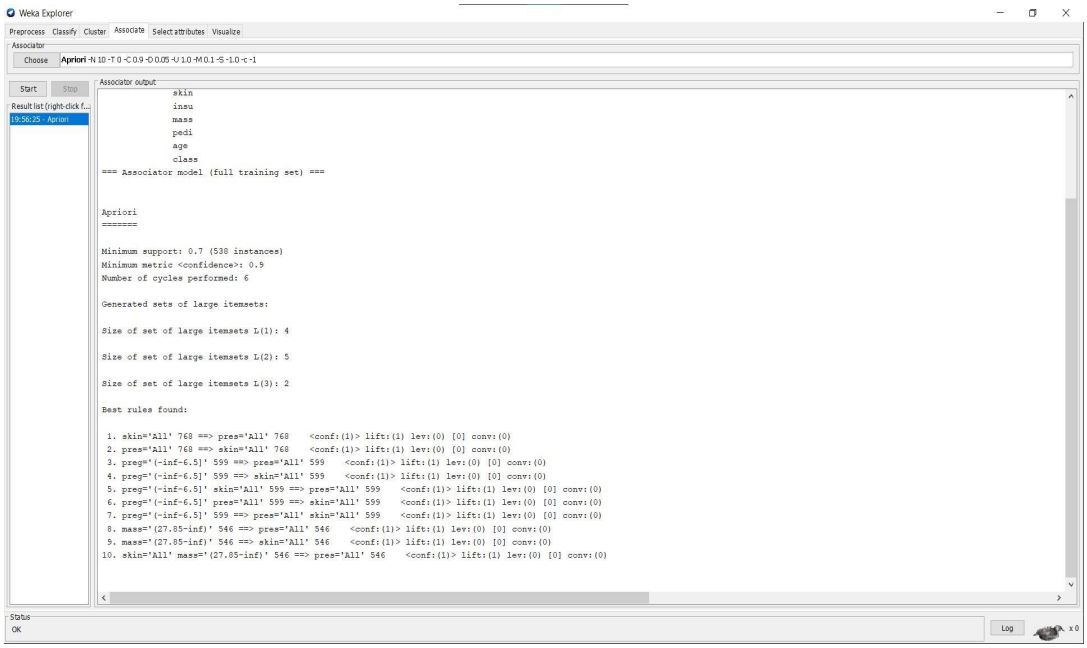



We have performed K-Means Clustering on the dataset to form groups of similar instances from the data. It incorrectly clustered 45.83% of the instances.




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.

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




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:

a. The attributes skin, mass and pres are associative
b. The attributes mass and pres are associative
c. 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.

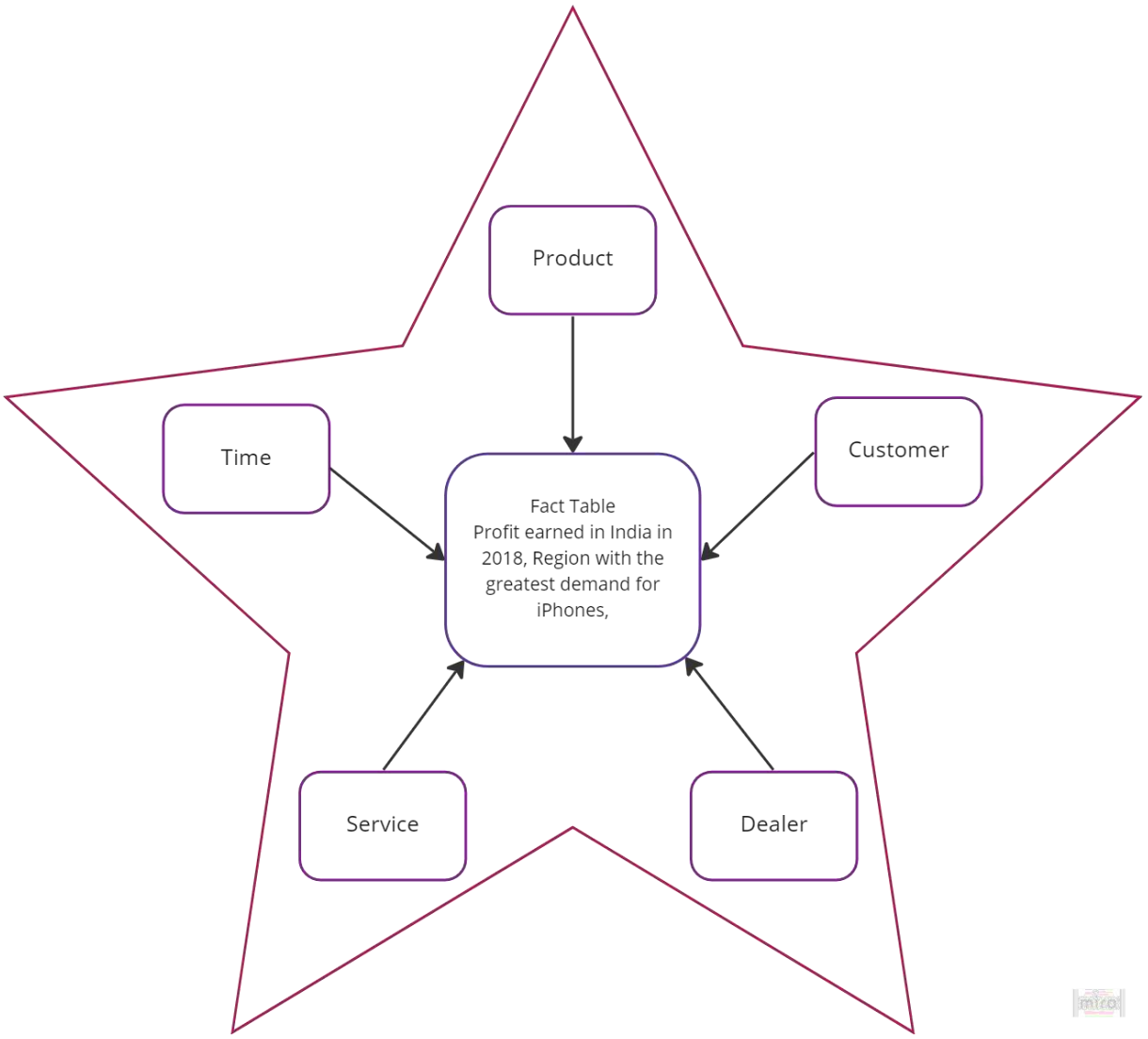
Q. How are apple services growing?
A.
Apple Music had 98M users by 2021.

***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 | SubscriptionPeriod | 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,
```

## 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.name:
  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', 'Name'], 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)
```

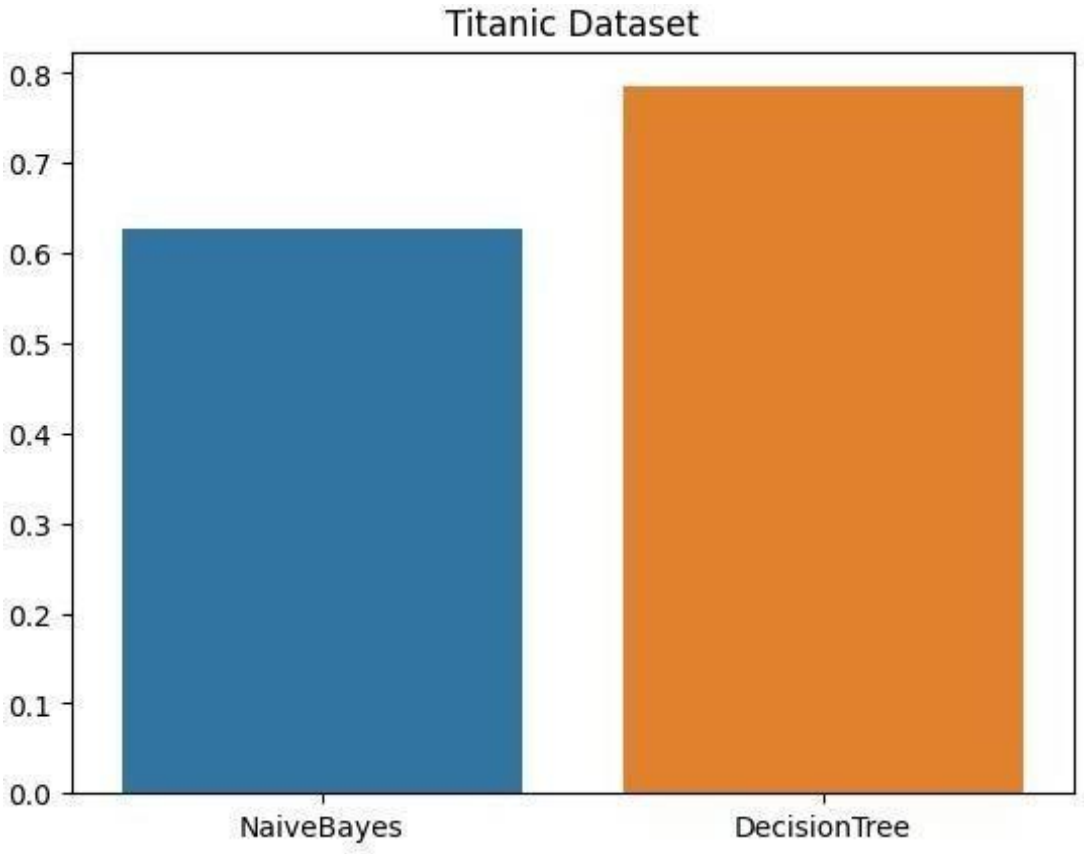
```
# using naive bayes
from sklearn.naive_bayes import GaussianNB
nb_model = GaussianNB()
nb_model.fit(x_train, y_train)
nb_accuracy = nb_model.score(x_valid, y_valid)
print(nb_accuracy)
```

```
# 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 = ['NaiveBayes','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='Naive 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="GaussianNB 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)
```

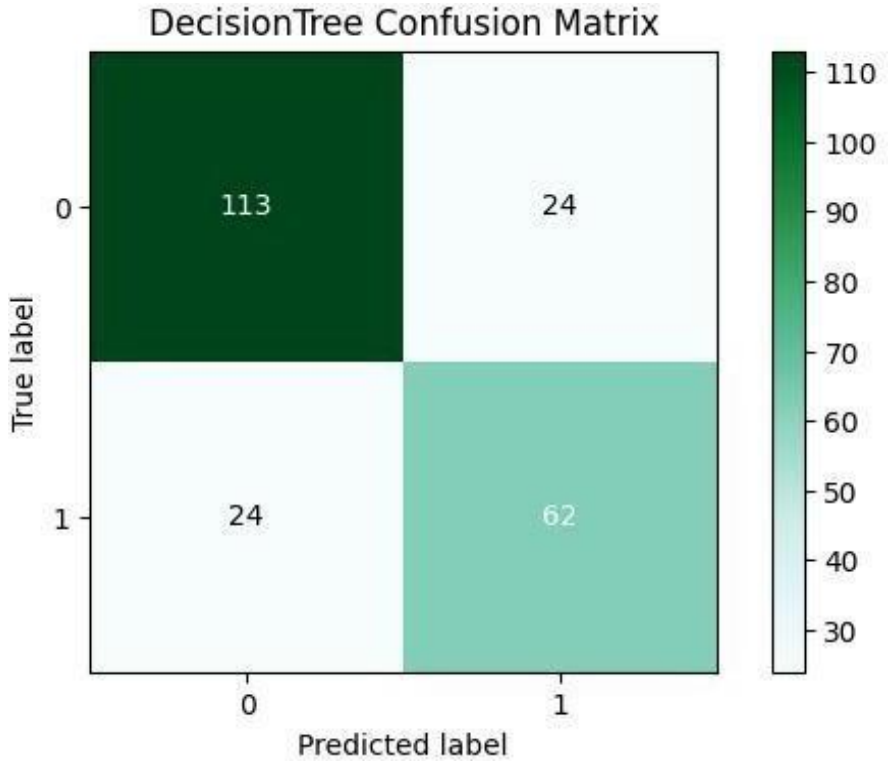
```
# ensemble tech - RandomForest
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier()
rf_model.fit(x_train, y_train)
rf_accuracy = rf_model.score(x_valid, y_valid)
print(rf_accuracy)
```

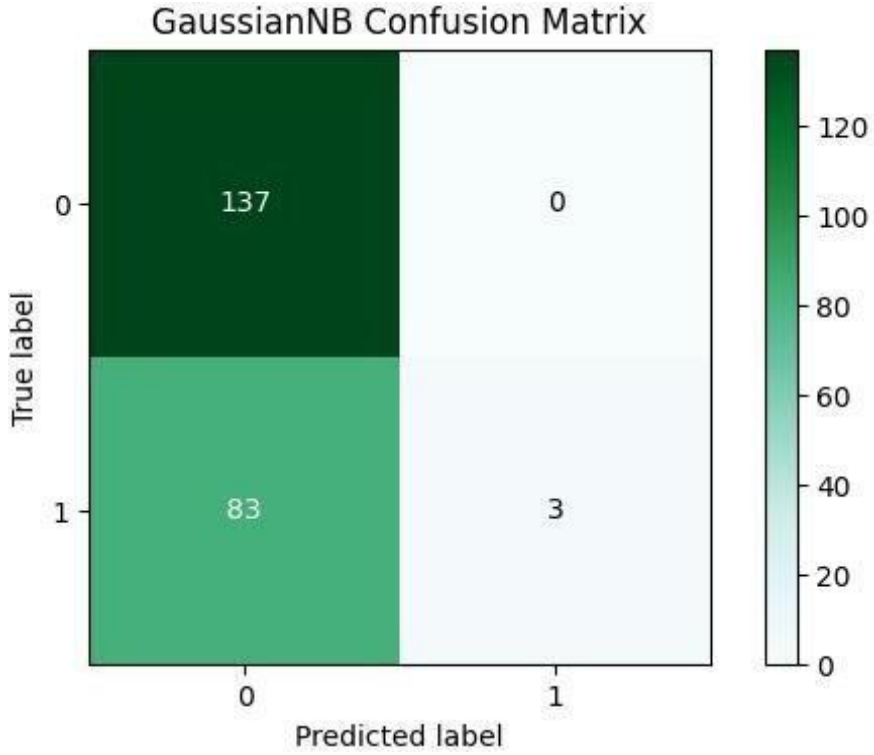
```
accuracies=[nb_accuracy,dt_accuracy,rf_accuracy,mean_cv_score]
plt.figure(figsize=(10,5))
models=[" aive Bayes','Decision Tree","Random Forest',' aive Bayes + Cross validation']
sns.barplot(x=models,y=accuracies,).set(title="Part C')
plt.xlabel("Models")
plt.ylabel('Accuracy') plt.title('Accuracy
of Different Models")
```

## Accuracies of both models :

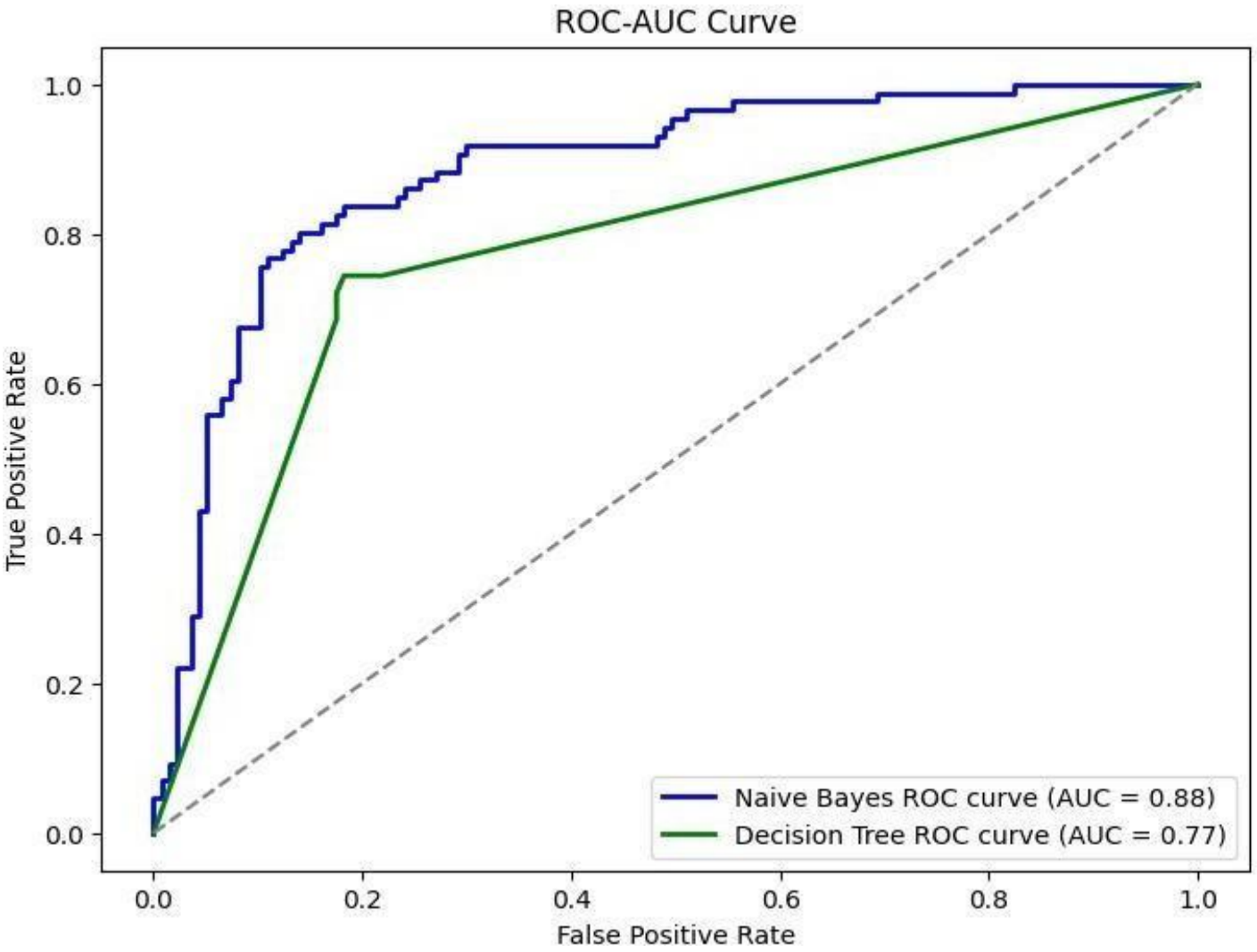


**Confusion Matrix :**



**AUROC:**


ROC-AUC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Naive Bayes ROC curve (AUC = 0.88)
Decision Tree ROC curve (AUC = 0.77)

## 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 GaussianNB
nb_model = OneVsRestClassifier(GaussianN
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 = [NaiveBayes','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_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,
```

```
plt.figure(figsize=(8, 6))
plt.plot(fpr_gnb, tpr_gnb, color='blue', lw=2, label=' aiNe 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()
```

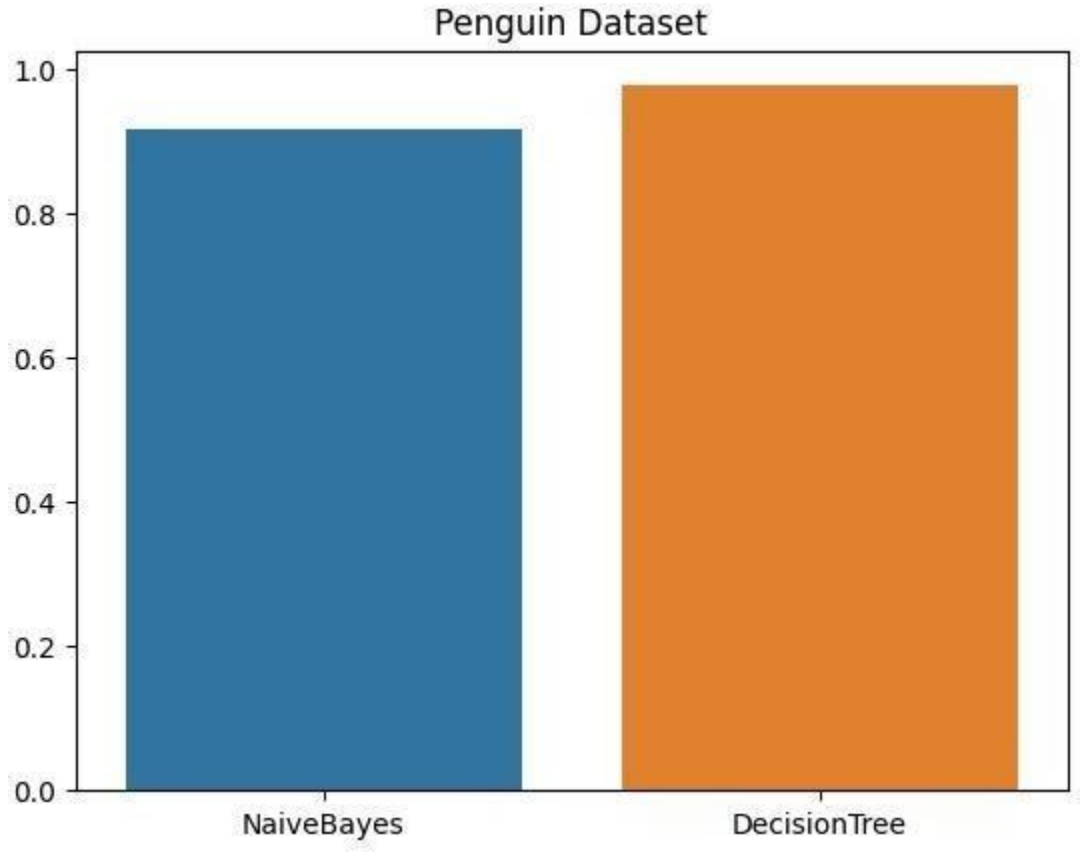
```
# confiusion matrix
q = y_valid
pred_test = dt_model.predict(x_valid)
pred_test = pd.DataFrame(pred_test)
y_valid = pd.DataFrame(y_valid)
plot_confusion_matrix(y_valid, pred_test, figsize=(7,4), title='DecisionTree Confusion Matrix", cmap="BuGn")
```

```
y_test = q
```

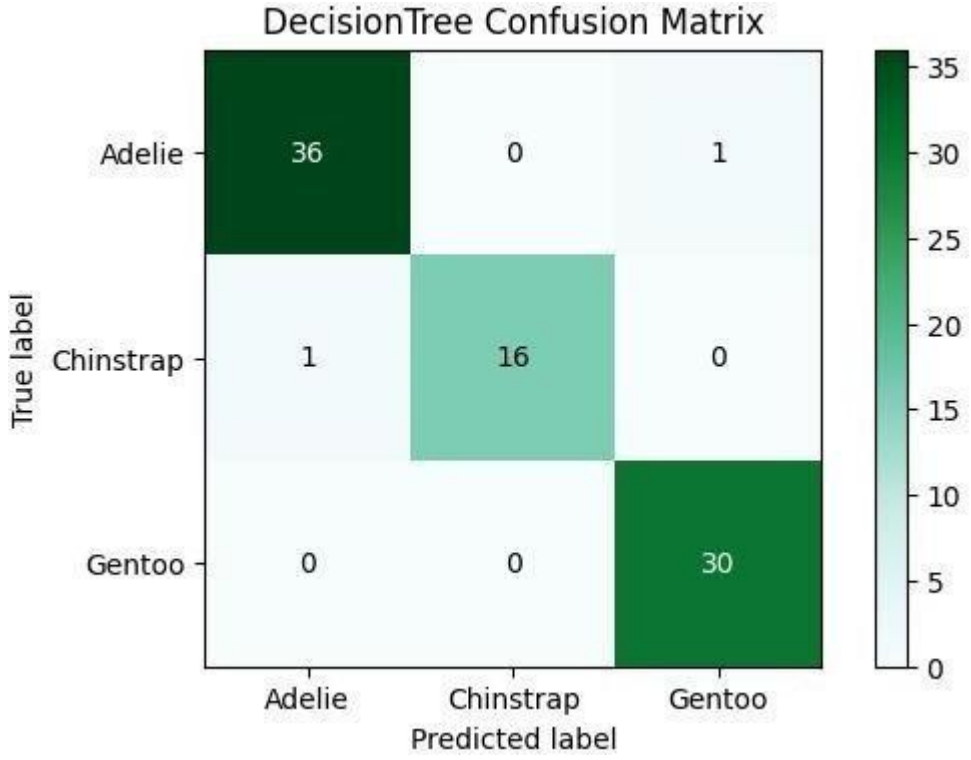
```
q = y_valid
pred_test = nb_model.predict(x_valid)
pred_test = pd.DataFrame(pred_test)
y_valid = pd.DataFrame(y_valid)
plot_confusion_matrix(y_valid, pred_test, figsize=(7,4), title="Gaussian B Confusion Matrix", cmap="BuGn")
y_test = q
```

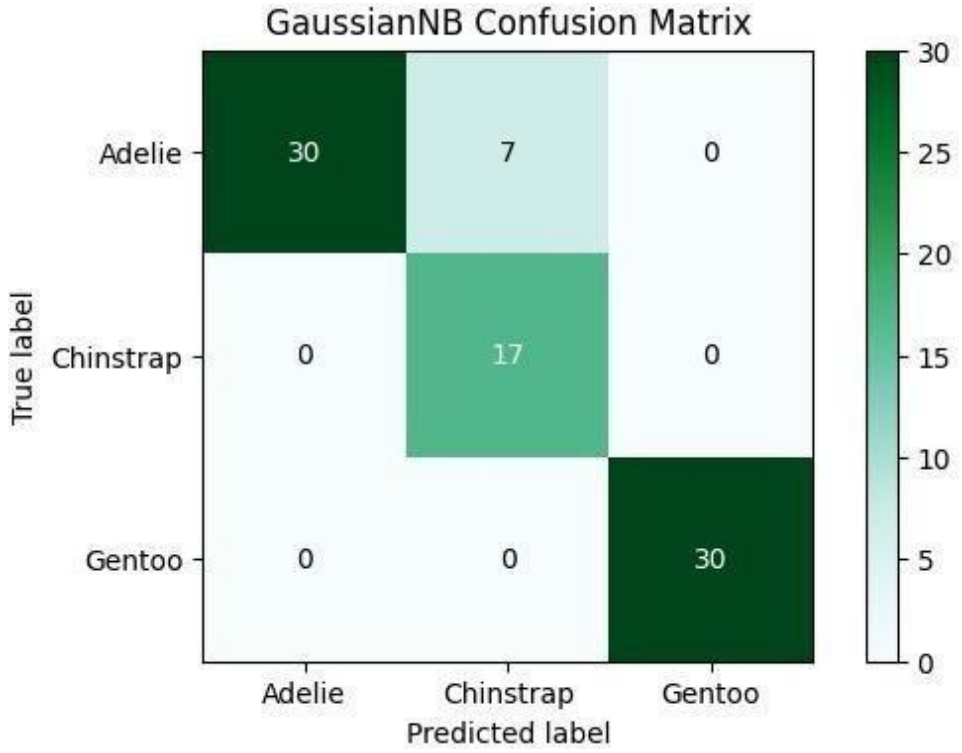
```
  auc - roc
y_score_gnb = nb_model.predict_proba(x_valid)
fpr_gnb, tpr_gnb, thresholds_gnb = roc_curve(y_valid.values, y_score_gnb.values)
roc_auc_gnb = roc_auc_score(y_valid, y_score_gnb)
y_score_dtc = dt_model.predict_proba(x_valid)
fpr_dtc, tpr_dtc, thresholds_dtc = roc_curve(y_valid.values, y_score_dtc.values)
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 :

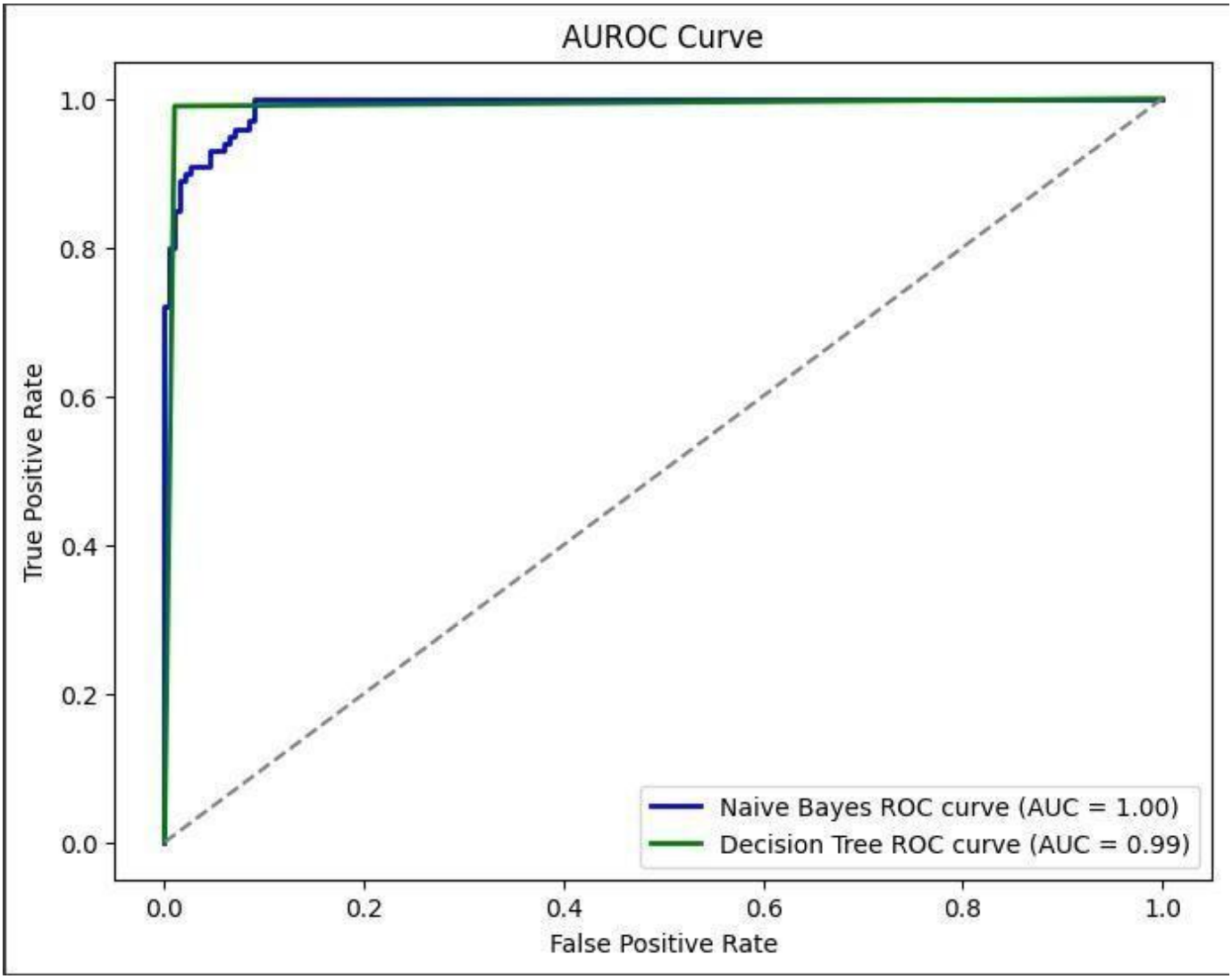


**Confusion Matrix :**



**AUROC:**


AUROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Naive Bayes ROC curve (AUC = 1.00)
Decision Tree ROC curve (AUC = 0.99)

## 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)
```

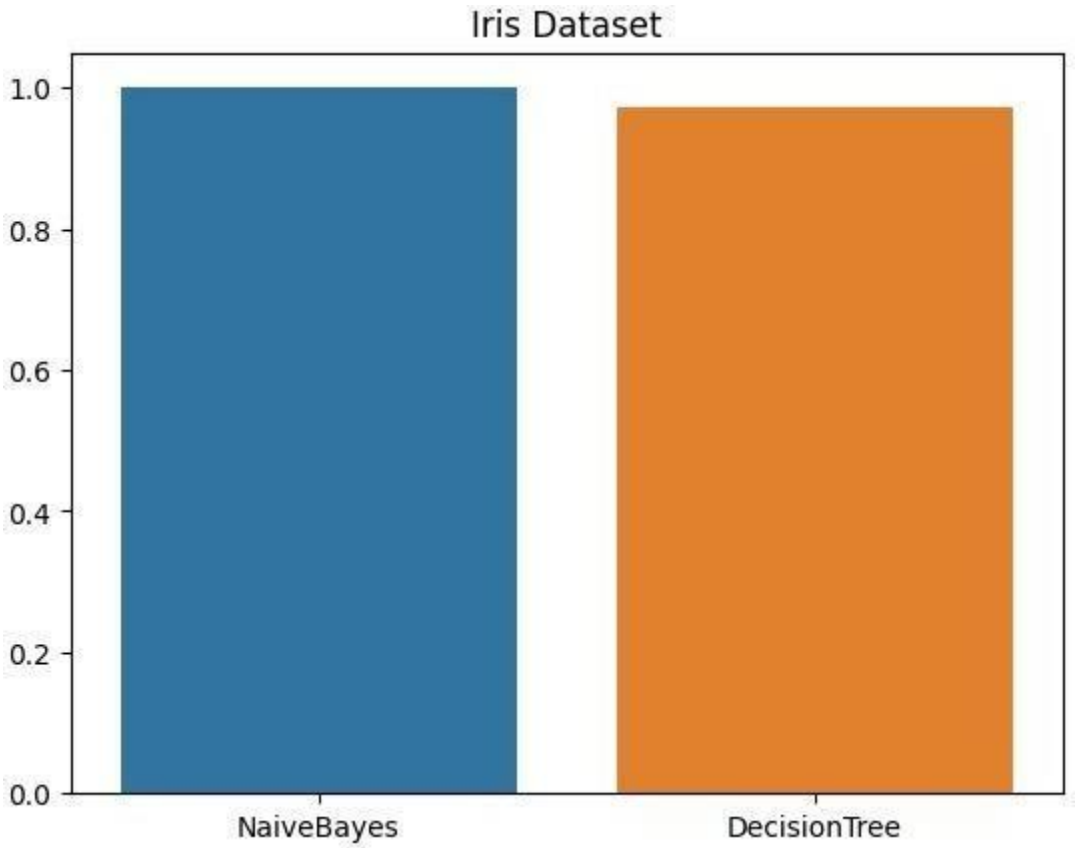
```
# using naive bayes
from sklearn.naive_bayes import GaussianNB
nb_model = GaussianNB()
nb_model.fit(x_train, y_train)
nb_accuracy = nb_model.score(x_valid, y_valid)
print(nb_accuracy)
```

```
# 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 = [NaiveBayes','DecisionTree']
sns.barplot(x=Models,y=accuracy).set(title="Iris Dataset")
```

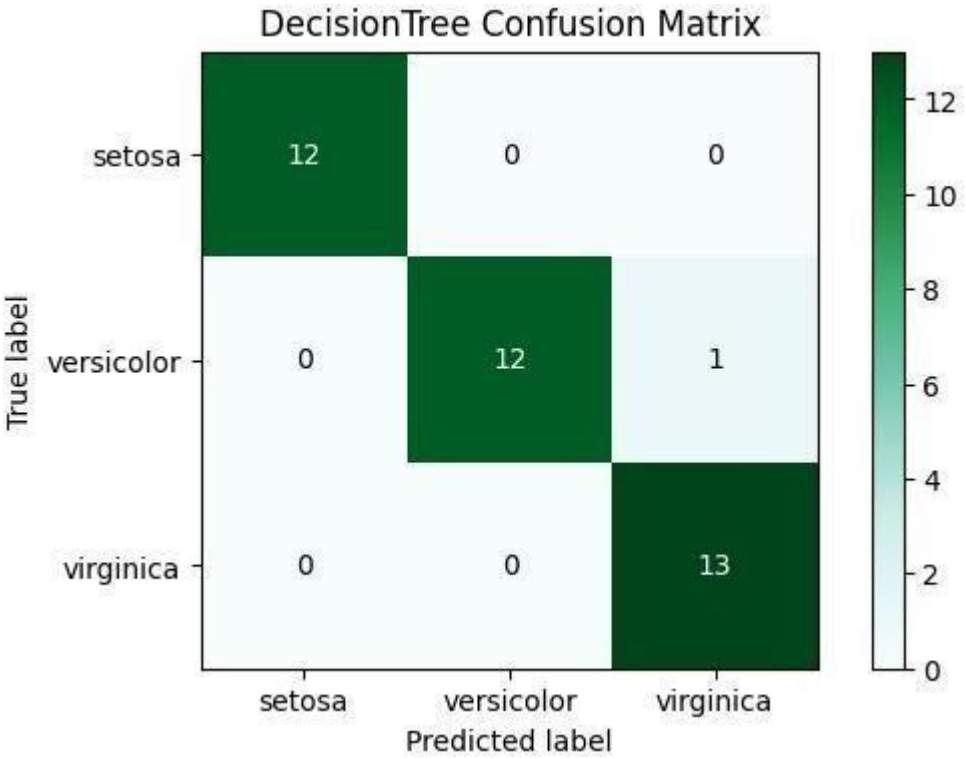
```
q = y_valid
pred_test = dt_model.predict(x_valid)
pred_test = pd.DataFrame(pred_test)
y_valid = pd.DataFrame(y_valid)
plot_confusion_matrix(y_valid, pred_test, figsize=(7,4), title='DecisionTree Confusion Matrix", cmap="BuGn") y_test
= q
q = y_valid
pred_test = nb_model.predict(x_valid)
pred_test = pd.DataFrame(pred_test)
y_valid = pd.DataFrame(y_valid)
plot_confusion_matrix(y_valid, pred_test, figsize=(7,4), title="Gaussian BN Confusion Matrix", cmap="BuGn")
y_test = q
```

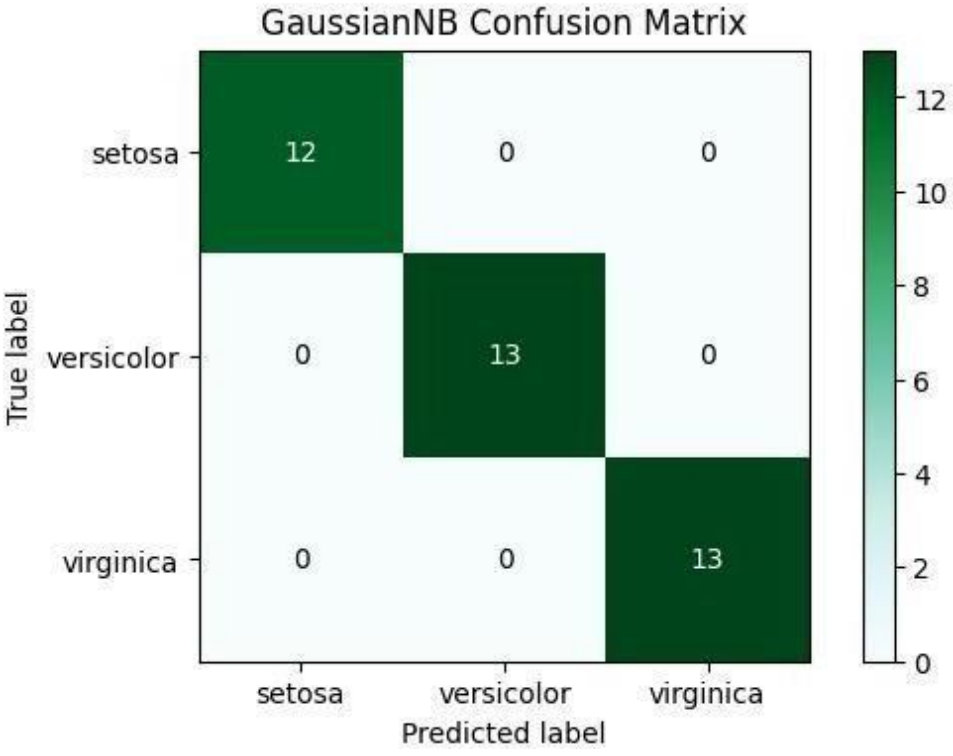
```
# auc - roc
y_score_gnb = nb_model.predict_proba(x_valid)
fpr_gnb, tpr_gnb, thresholds_gnb = roc_curve(y_valid.values, y_score_gnb.values)
roc_auc_gnb = roc_auc_score(y_valid, y_score_gnb)
y_score_dtc = dt_model.predict_proba(x_valid)
fpr_dtc, tpr_dtc, thresholds_dtc = roc_curve(y_valid.values, y_score_dtc.values)
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 :**

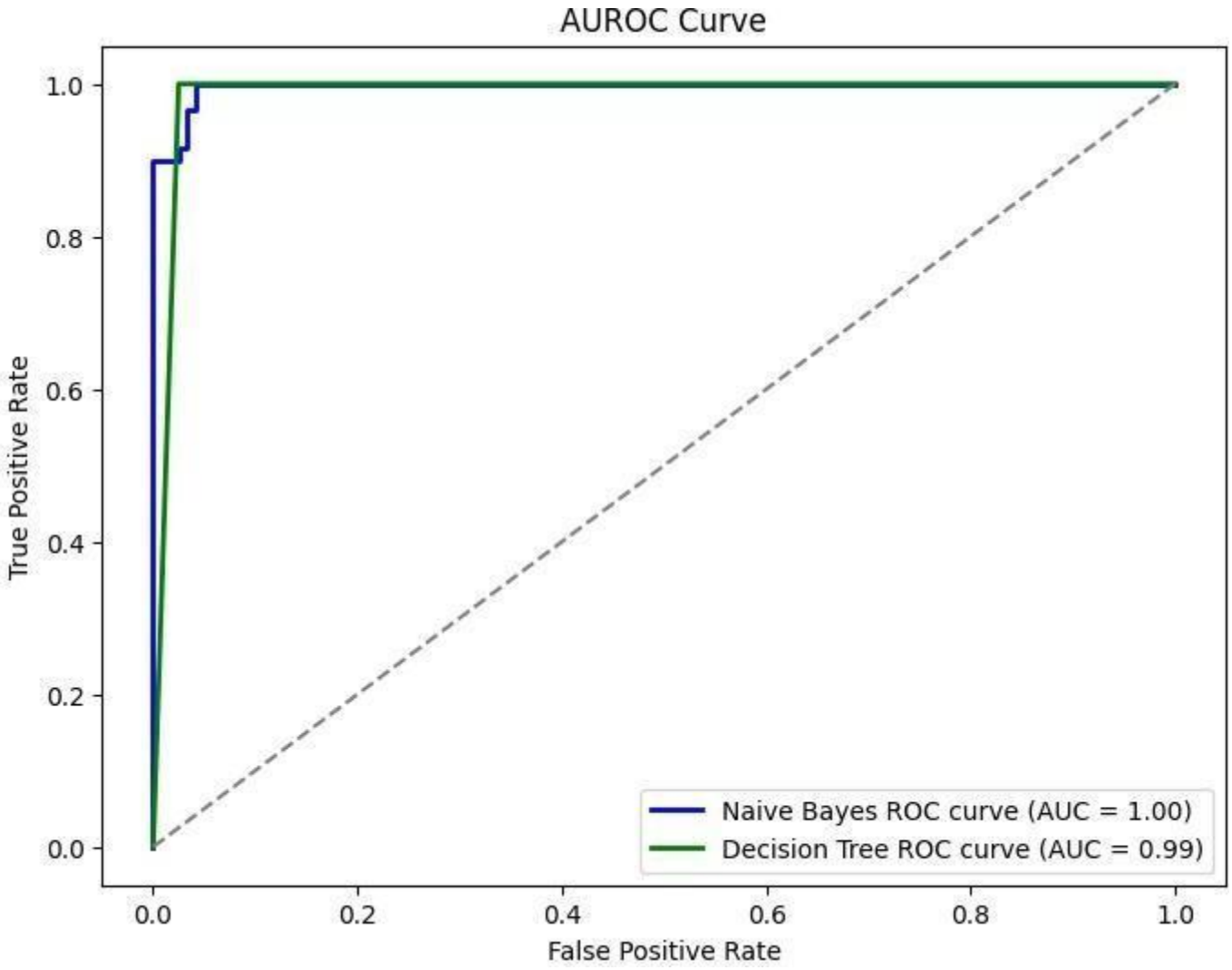


**Confusion Matrix :**



**AUROC:**


AUROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Naive Bayes ROC curve (AUC = 1.00)
Decision Tree ROC curve (AUC = 0.99)

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

```
"""###
        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                                   N
from nltk.corpus import stopwords
from nltk.stem.wordnet import Word etemmatizer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import ogisticRegression
from sklearn.tree import DecisionTreeClassifier from
sklearn.ensemble import RandomForestClassifier
                                            N
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")
```

```
spam_df.head()

def cleaning (text):
def cleanIng (text):
    # text = text.lower()
    text =  re.sub(r"@\S+", "",text)
    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)
    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
```

```
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
```

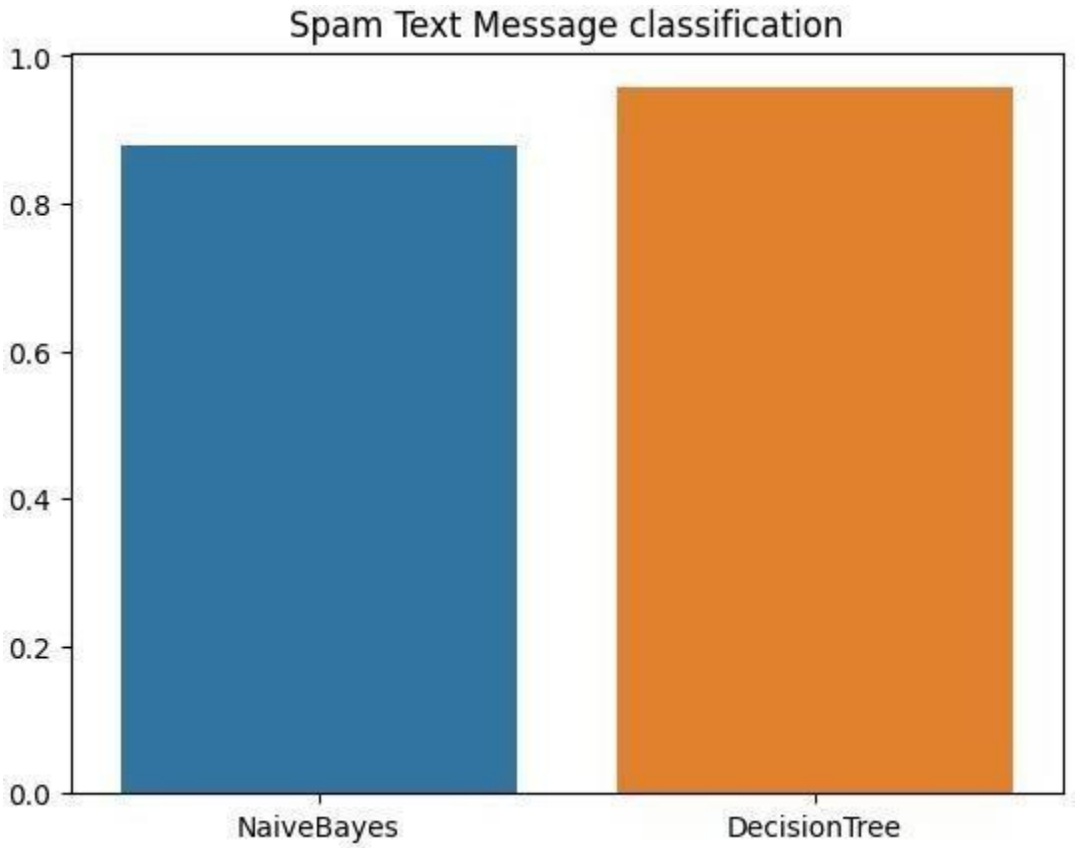
```
dt_model = DecisionTreeClassifier()
dt_model.fit(X_train, y_train)
pred = dt_model.predict(x_valid)
dt_accuracy =
```

```
nb_model = GaussianNB()
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 = ['naiveBayes','DecisionTree']
sns.barplot(x=Models,y=accuracy).set(title="Spam Text Message classification")
```

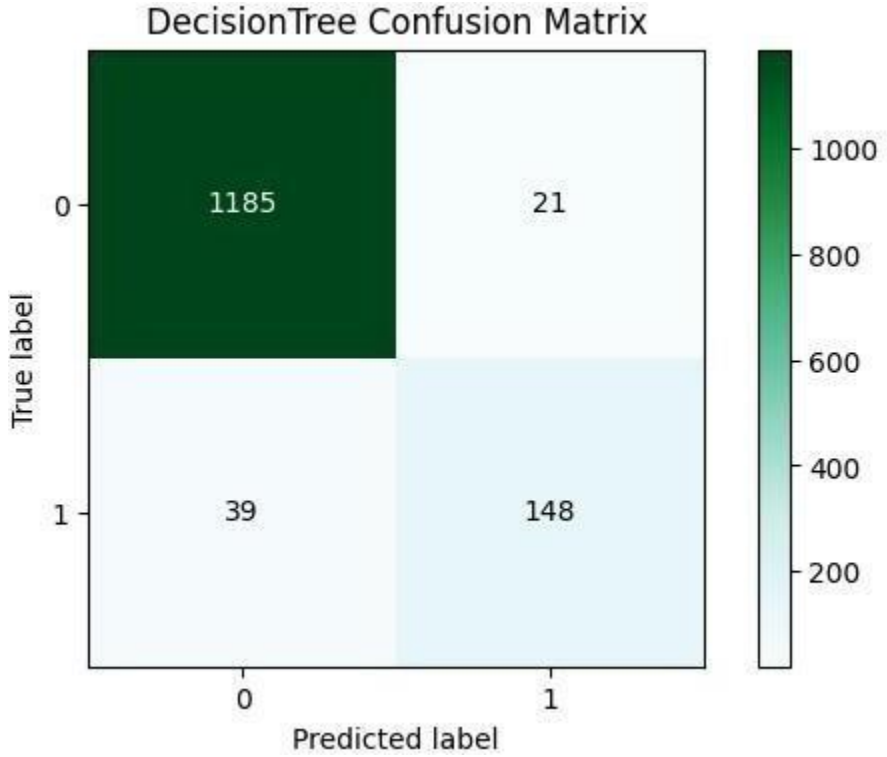
```
# auc - roc
y_score_gnb = nb_model.predict_proba(x_valid.toarray())[:, 1] fpr_gnb,
tpr_gnb, thresholds_gnb = roc_curve(y_valid, y_score_gnb)
roc_auc_gnb = roc_auc_score(y_valid, y_score_gnb)
y_score_dtc = dt_model.predict_proba(x_valid)[:, 1]
fpr_dtc, tpr_dtc, thresholds_dtc = roc_curve(y_valid, 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=' naive 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()
```

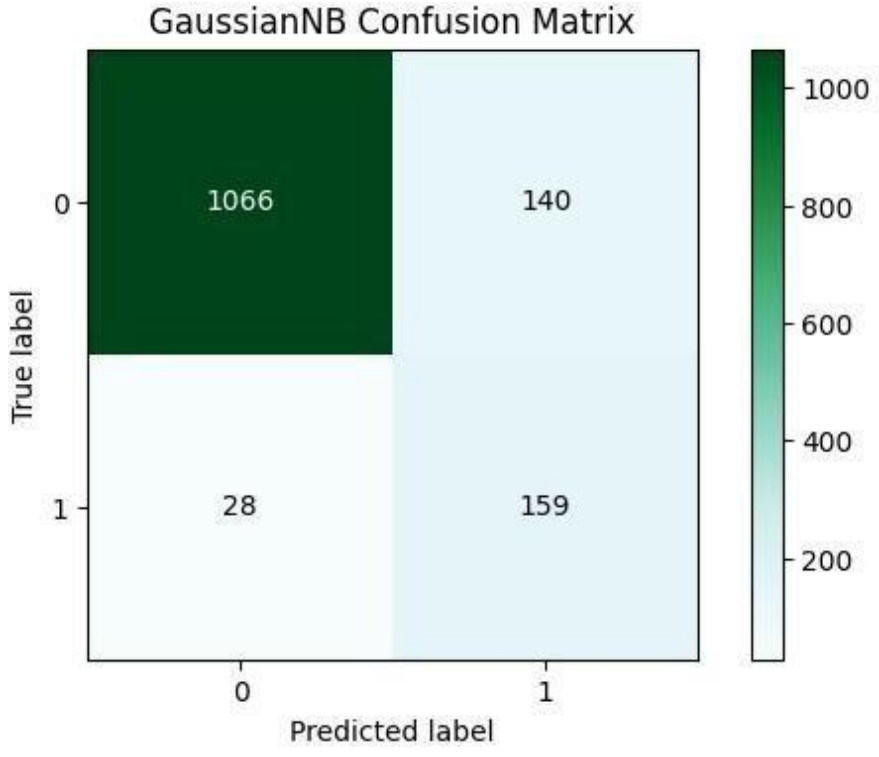
```
# confiusion matrix
q = y_valid
pred_test = dt_model.predict(x_valid)
pred_test = pd.DataFrame(pred_test)
y_valid = pd.DataFrame(y_valid)
plot_confusion_matrix(y_valid, pred_test, figsize=(7,4), title='DecisionTree Confusion Matrix', cmap='BuGn')
y_valid = q
q = y_valid
pred_test = nb_model.predict(x_valid.toarray()) pred_test
= pd.DataFrame(pred_test)
y_valid = pd.DataFrame(y_valid)
plot_confusion_matrix(y_valid, pred_test, figsize=(7,4), title='Gaussian NB Confusion Matrix', cmap='BuGn')
y_valid = q
```

**Accuracies of both models :**

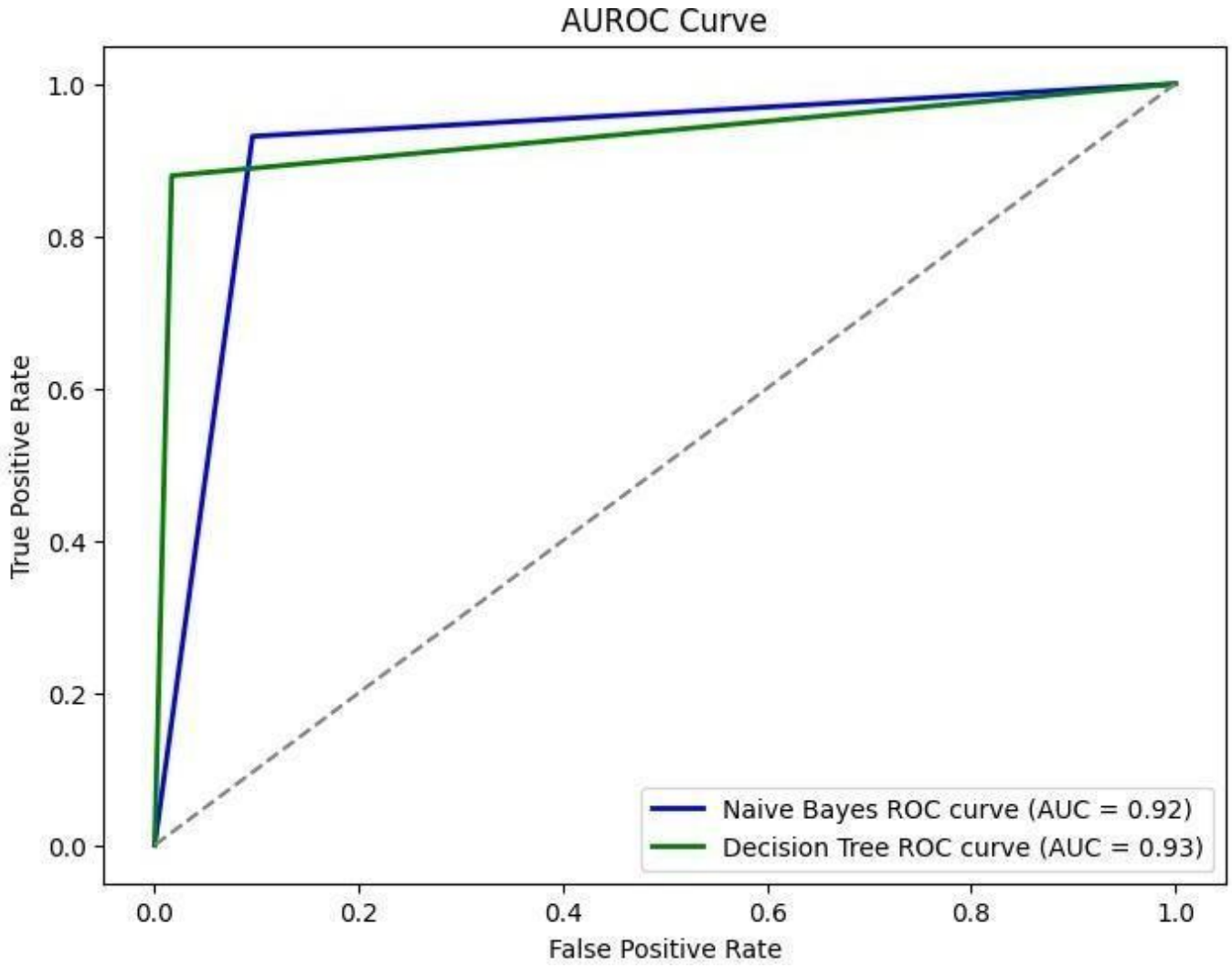


**Confusion Matrix :**



**AUROC:**


AUROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Naive Bayes ROC curve (AUC = 0.92)
Decision Tree ROC curve (AUC = 0.93)

## 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 = GaussianNB()
nb_model.fit(x_train, y_train)
nb_accuracy = nb_model.score(x_valid, y_valid)
print(nb_accuracy)
```

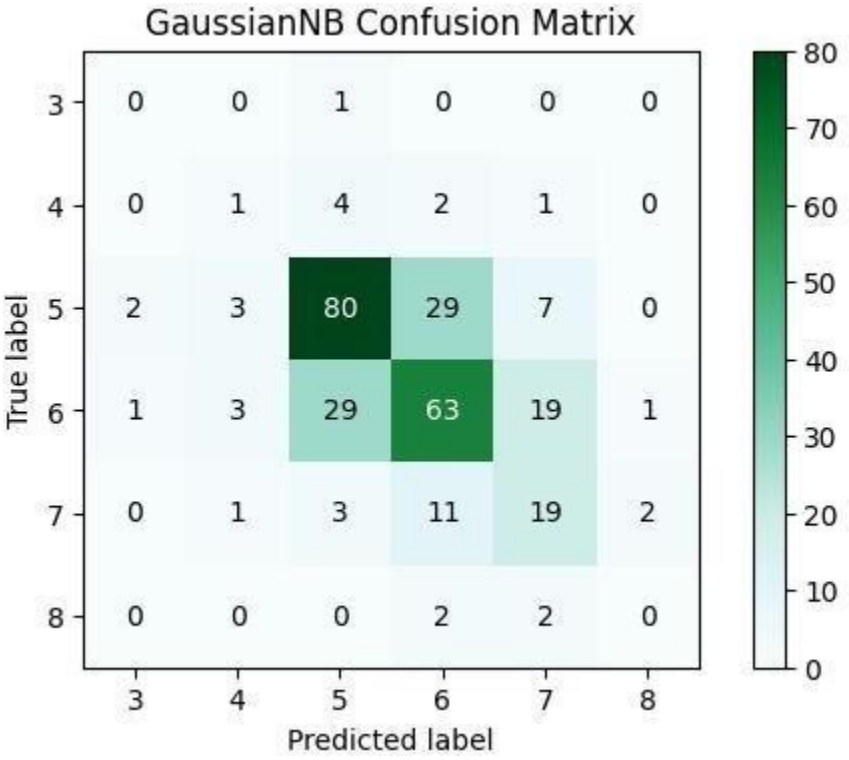
```
# using decision tree
dt_model = DecisionTreeClassifier()
dt_model.fit(x_train, y_train)
dt_accuracy = dt_model.score(x_valid, y_valid)
print(dt_accuracy)
gaussian.append(nb_accuracy)
decision.append(dt_accuracy)
accuracy = [nb_accuracy, dt_accuracy]
Models = ['NaiveBayes','DecisionTree']
sns.barplot(x=Models,y=accuracy).set(title="Wine Quality Classifier")
```

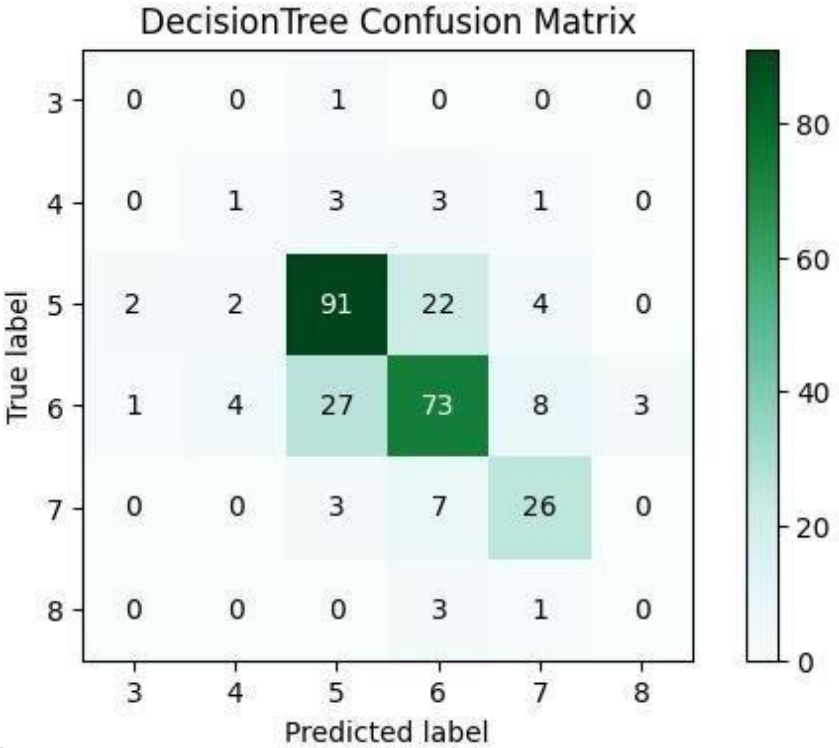
```
# roc - auc
y_score_gnb = nb_model.predict_proba(x_valid)
fpr_gnb, tpr_gnb, thresholds_gnb = roc_curve(y_valid.values, y_score_gnb.values)
roc_auc_gnb = roc_auc_score(y_valid, y_score_gnb)
y_score_dtc = dt_model.predict_proba(x_valid)
fpr_dtc, tpr_dtc, thresholds_dtc = roc_curve(y_valid.values, y_score_dtc.values)
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 NB 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
```

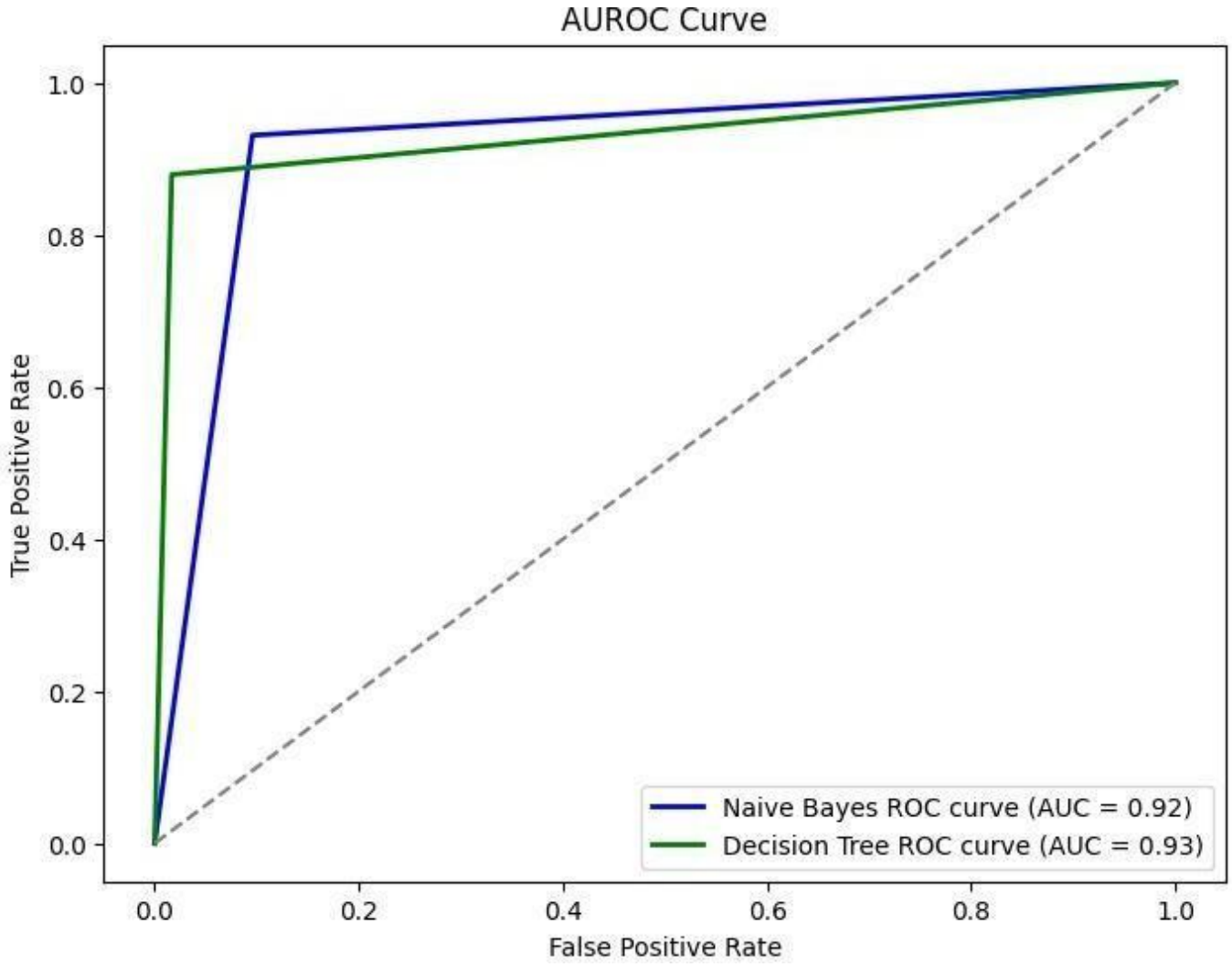
**Accuracies of both models :**

**Confusion Matrix :**



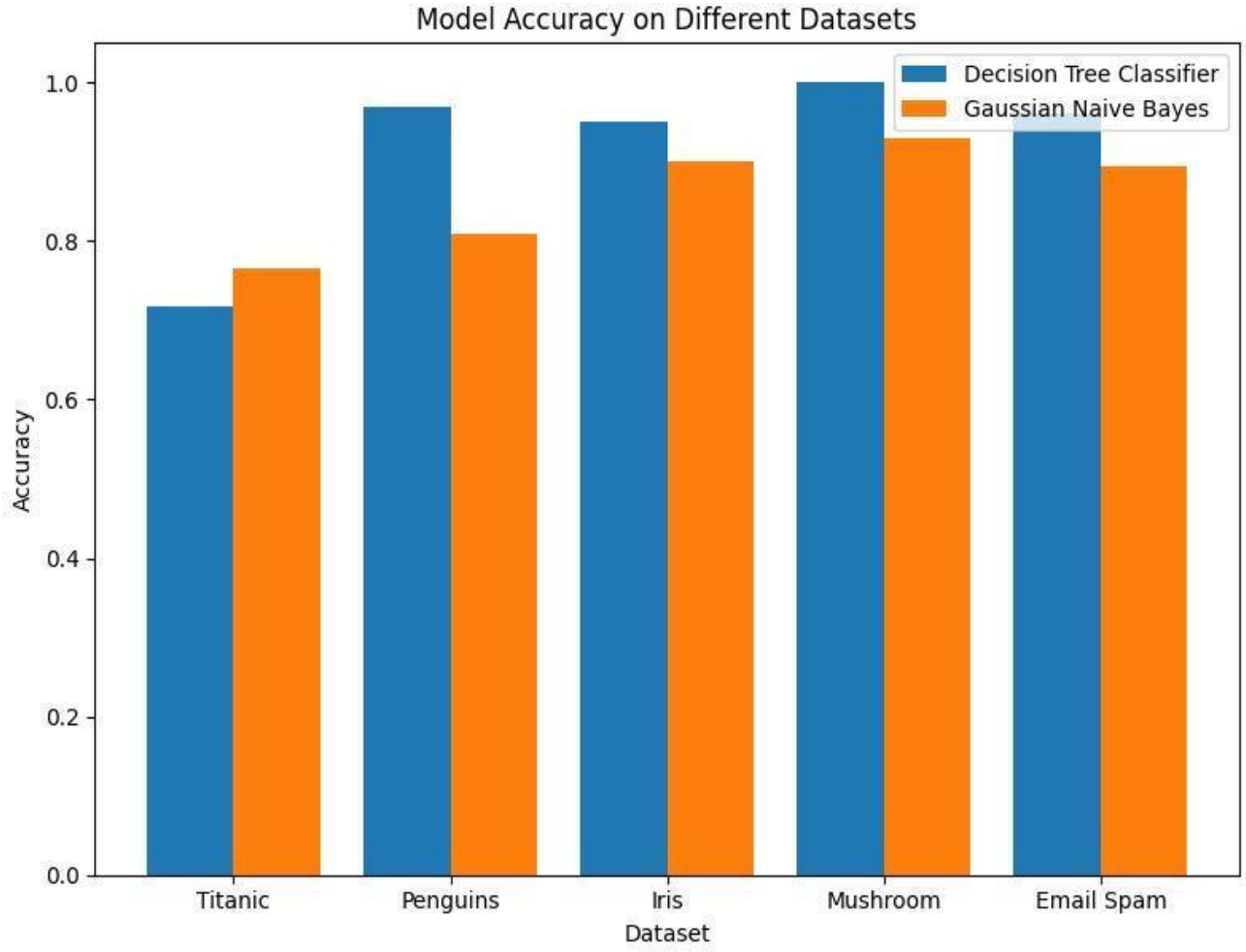
**AUROC:**


AUROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Naive Bayes ROC curve (AUC = 0.92)
Decision Tree ROC curve (AUC = 0.93)

## 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:


Model Accuracy on Different Datasets
Decision Tree Classifier
Gaussian Naive Bayes
Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
Titanic
Penguins
Iris
Mushroom
Email Spam
Dataset

## Part C:

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

## Modification Titanic Dataset :

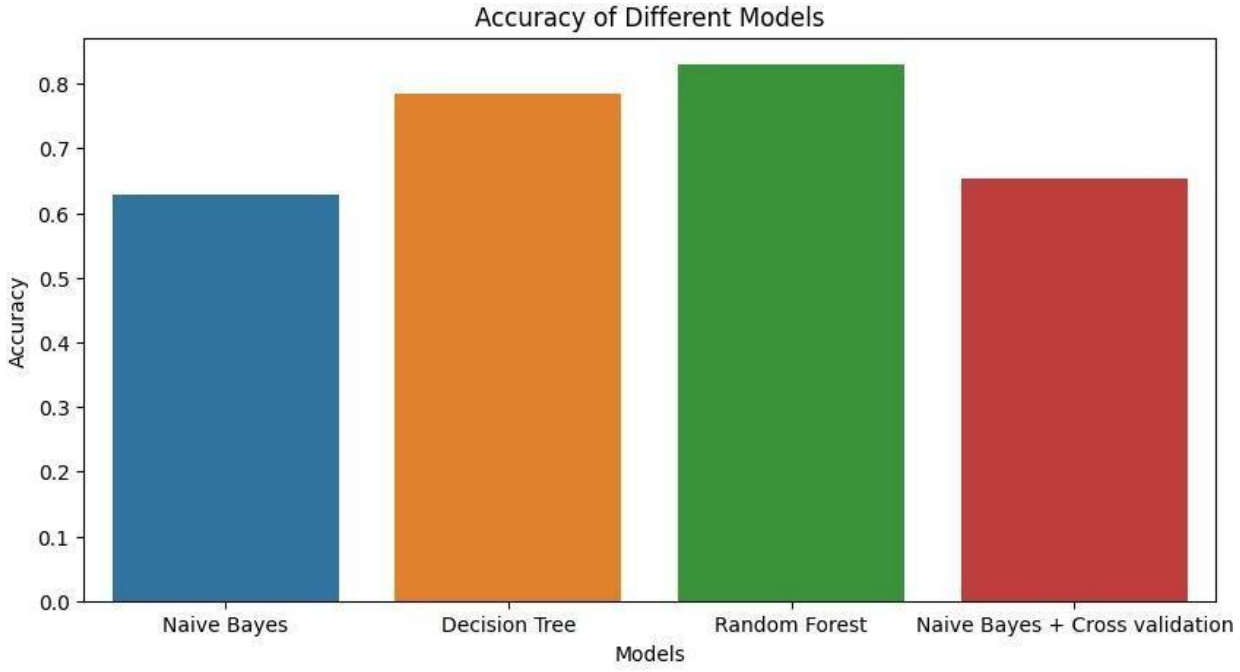
```
from sklearn.model_selection import KFold,

cross_val_score kf =

KFold(n_splits=7,shuffle=True,random_state

=10)
```

```
# 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)
```

```
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 LinearRegression
from  sklearn.preprocessing  import LabelEncoder
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 = linearRegression()
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 = linearRegression()
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:

| | gender | race/ethnicity | parental level of education | lunch | test preparation course | math score | reading score | writing score |
|---|---|---|---|---|---|---|---|---|
| 0 | female | group B | bachelor's degree | standard | none | 72 | 72 | 74 |
| 1 | female | group C | some college | standard | completed | 69 | 90 | 88 |
| 2 | female | group B | master's degree | standard | none | 90 | 95 | 93 |
| 3 | male | group A | associate's degree | free/reduced | none | 47 | 57 | 44 |
| 4 | male | group C | some college | standard | none | 76 | 78 | 75 |

head() of the database

| | gender | race/ethnicity | parental level of education | lunch | test preparation course | final_score |
|---|---|---|---|---|---|---|
| 0 | female | group B | bachelor's degree | standard | none | 72.666667 |
| 1 | female | group C | some college | standard | completed | 82.333333 |
| 2 | female | group B | master's degree | standard | none | 92.666667 |
| 3 | male | group A | associate's degree | free/reduced | none | 49.333333 |
| 4 | male | group C | some college | standard | none | 76.333333 |

df.head() after adding a final score column

| | final_score | gender_female | gender_male | lunch_free/reduced | lunch_standard | parental level of education_associate's degree | parental level of education_bachelor's degree | parental level of education_high school | parental level of education_master's degree | parental level of education_some college | parental educatic high |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 72.666667 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | |
| 1 | 82.333333 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | |
| 2 | 92.666667 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | |
| 3 | 49.333333 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | |
| 4 | 76.333333 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | |

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

---

## Considering Multivariate Linear Regression

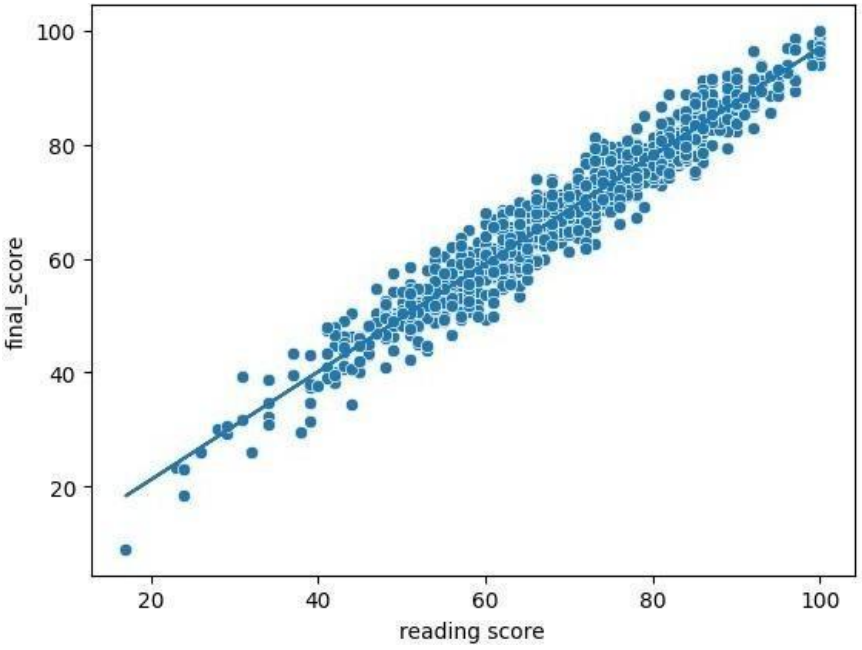
### Prediction Score

```
score = model.score(xtest,ytest)
print(score)
ypred = model.predict(xtest)
```

```
0.19674412629893356
```

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

*Scatter plot*



Prediction Score of Univariate LR

```
uni_score = uni_model.score(x_uni_test,y_uni_test)
print(uni_score)
y_uni_pred = uni_model.predict(x_uni_test)
```

```
0.944431815987387
```

**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.

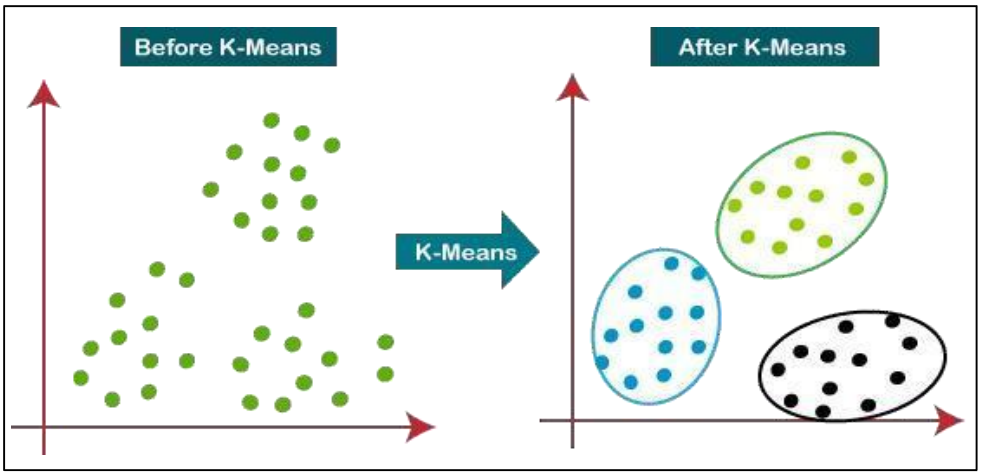
**Name: Jigar Siddhpura** **SAPID:** 60004200155

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

# DMW EXPERIMENT 5

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

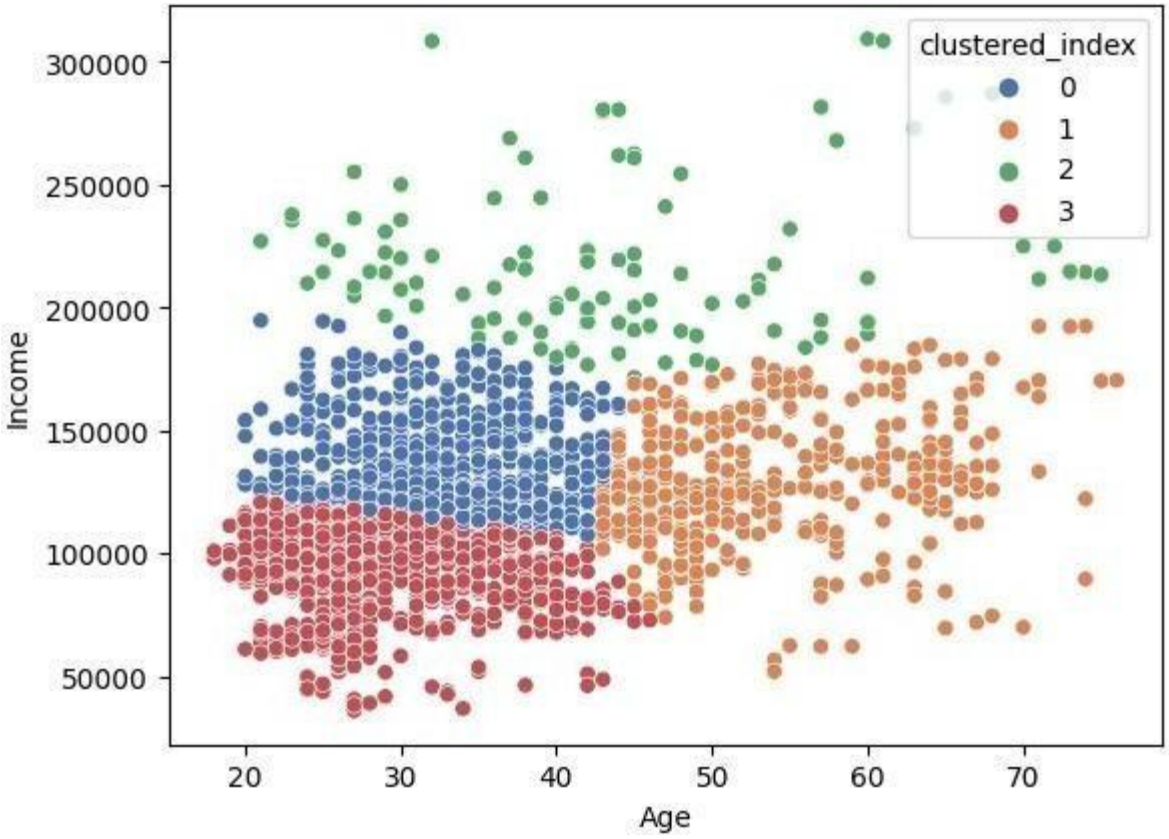
The below diagram explains the working of the K-means Clustering Algorithm:



Program:

```
from google.colab import drive
drive.mount("/content/gdr ive")
import numpy as np
import pandas as pd
import matplotl ib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans, Agglomerat iveCluster ing
from scipy.cluster.h ierarchy import dendrogram
from sklearn.preprocess ing import StandardScaler
from sklearn.metr ics import s lhouette_score
df =
pd.read_csv("/content/gdr ive/MyDr ive/DMW/datasets/customer_segmentat on.csv")
df.head()
df.drop(["ID"], inplace = True, ax is = 1)
features = df[df.columns]
scaler = StandardScaler()
scaled = scaler.f it_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 i in range(1,11):
  kmeans = KMeans(n_clusters= i,random_state=10)
  kmeans.f it(data) wcss["wcss_score"].append(kmeans.
  nert a_) wcss["no_of_clusters"].append(i)
plt.f gure(f gs ze=(7,5))        i
plt.plot(wcss i["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)") i
plt.ylabel("WCSS (W th ng Cluster Sum of Squared d stance )")
                  i  i                          i
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()    i
clustered_data["clustered_ ndex"] = pred ct on
sns.scatterplot(x=clustered_data.Age, y=clustered_data.Income,
hue=clustered_data.clustered_ ndex, palette="deep")
                           i
```

**Output:**




```
# checking the quality of clustering
score = silhouette_score(X=df,labels=clustered_data.clustered_index)
score
```

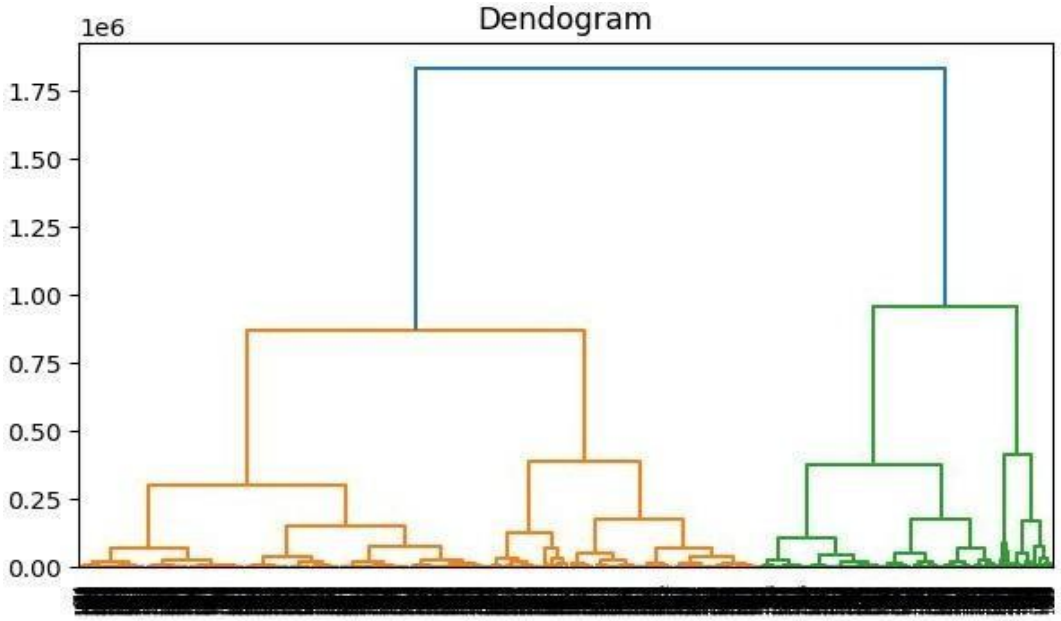
```
0.238448488332598
```

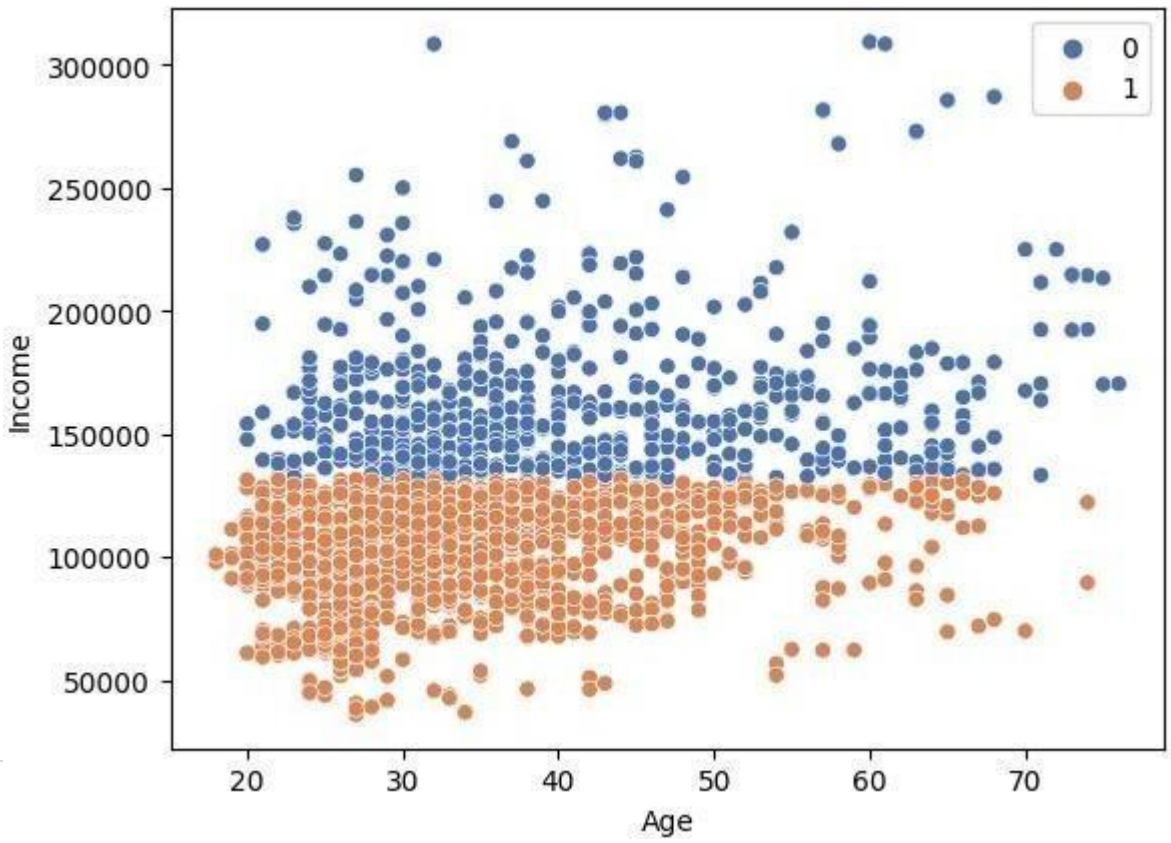
## Program :

```
# Hierarchichal clustering
from scipy.cluster.hierarchy import dendrogram,linkage
data = clustered_data[["Age","Income"]]

plt.figure(figsize=(10,7))
plt.title("Dendogram")
dend = dendrogram(linkage(data,method="ward"))
cluster =
AgglomerativeClustering(n_clusters=2,affinity="euclidean",linkage="ward")
labels_ = cluster.fit_predict(data)

sns.scatterplot(x=data.Age, y=data.Income, hue=labels_, palette="deep
```

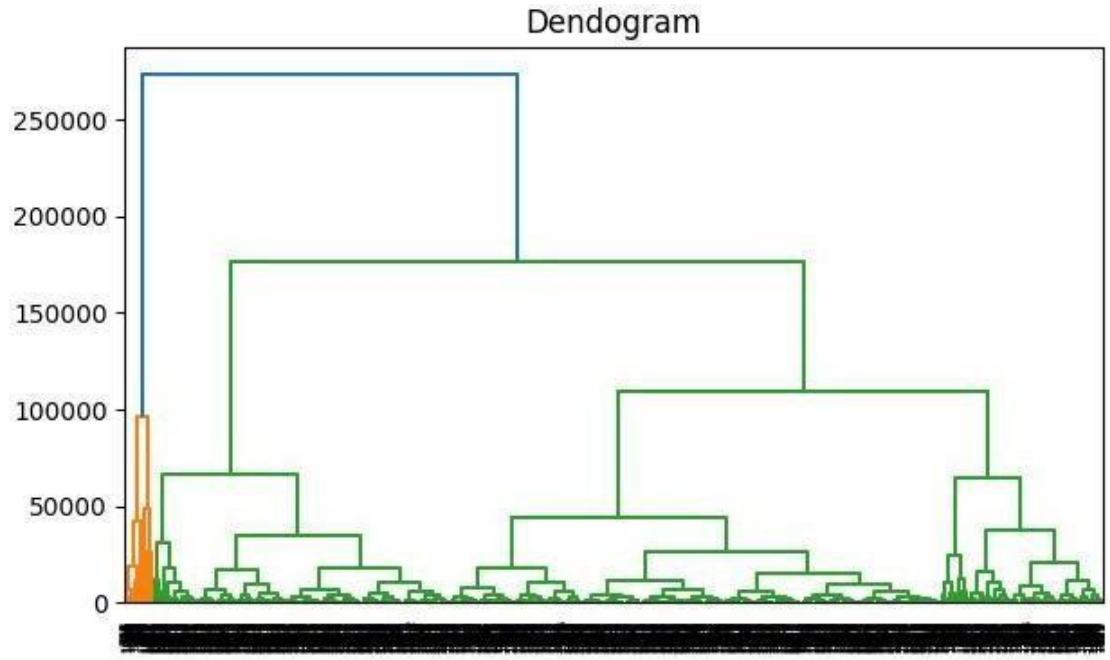
**Output:**

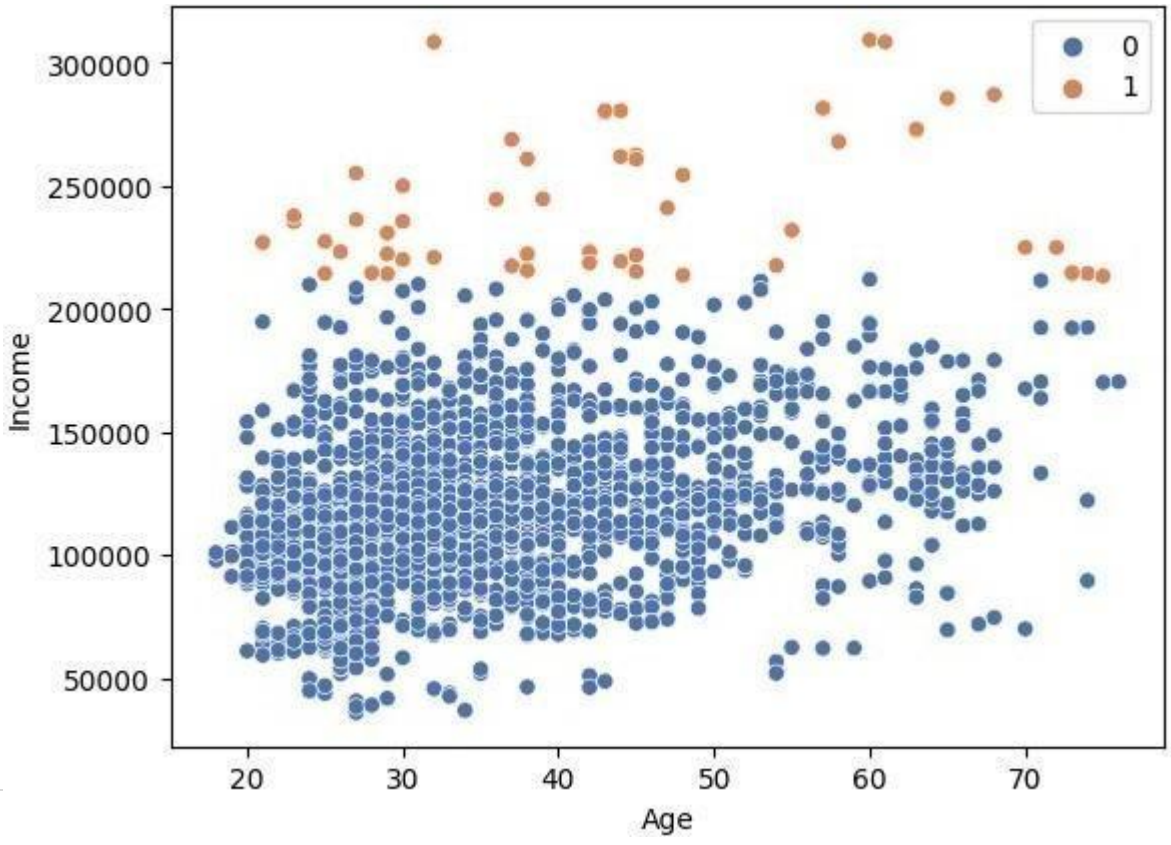
- ***Ward Hierarchical Clustering***



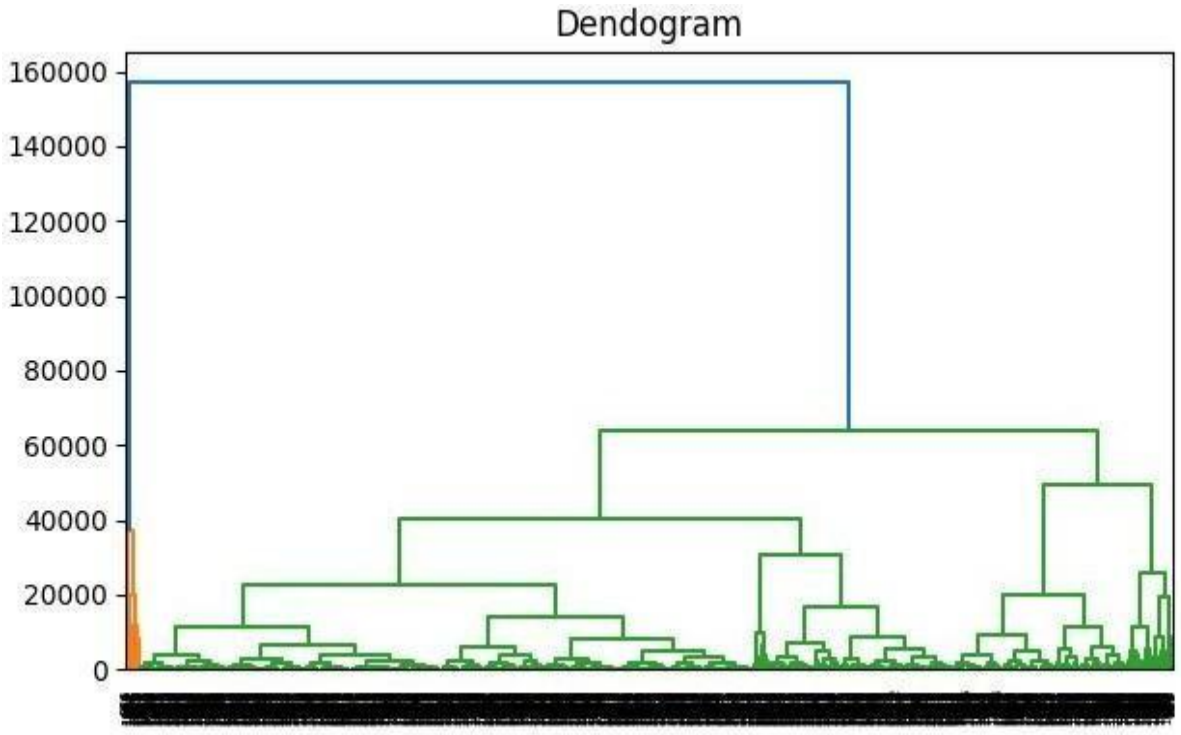



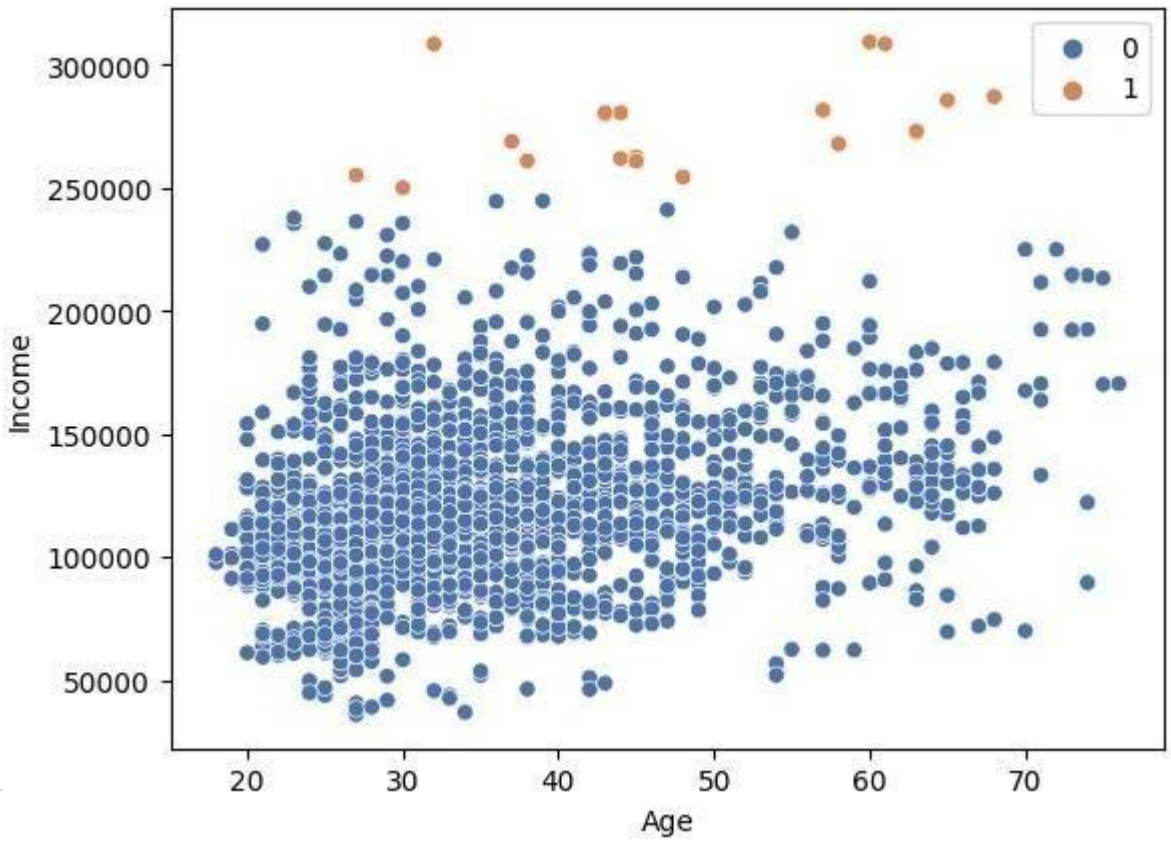
- ***Complete Hierarchical Clustering***



- ***Average Hierarchical Clustering***



**Part B:**

1. Plot Elbow Method and suggest optimal number of clusters

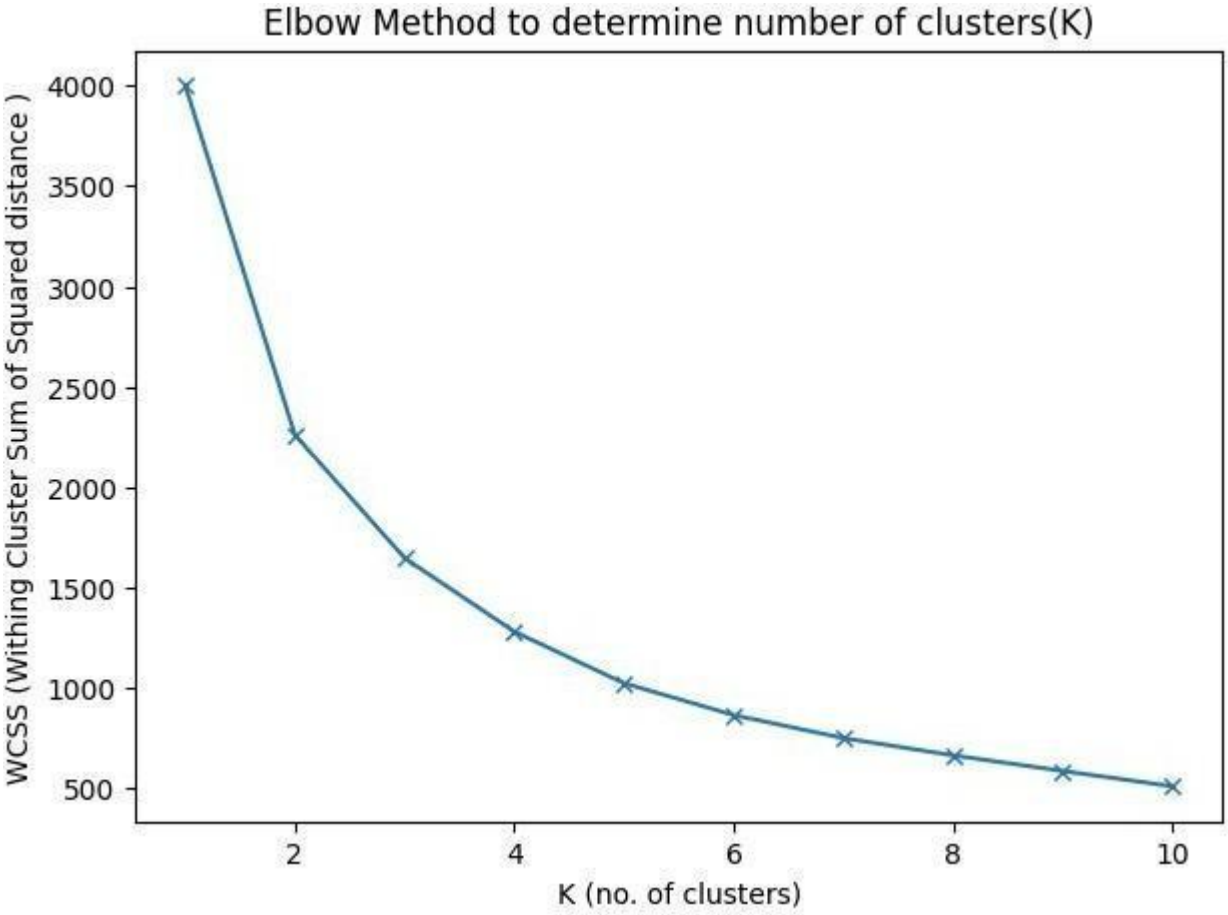
A fundamental step for any unsupervised algorithm is to determine the optimal number of clusters into which the data may be clustered. The **Elbow Method** is one of the most popular methods to determine this optimal value of k.

## Program:

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

# elbow curve
wcss = {'wcss_score':[],'no_of_clusters':[]}
for i in range(1,11):
  kmeans = KMeans(n_clusters=i,random_state=10)
  kmeans.fit(data)
  wcss['wcss_score'].append(kmeans.inertia_)
  wcss['no_of_clusters'].append(i)

plt.figure(figsize=(7,5))
plt.plot(wcss['no_of_clusters'],wcss['wcss_score'],marker='x')
plt.title("Elbow Method to determine number of clusters(K)")
plt.xlabel("K (no. of clusters)")
plt.ylabel("WCSS (Withing Cluster Sum of Squared distance )")
plt.show()
```

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:

```
Enter minimum support : 0.2
Enter minimum confidence : 0.6
```

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction | zhangs_metric |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | (MILK) | (BREAD) | 0.25 | 0.65 | 0.2 | 0.800000 | 1.230769 | 0.0375 | 1.75 | 0.250000 |
| 1 | (SUGER) | (BREAD) | 0.30 | 0.65 | 0.2 | 0.666667 | 1.025641 | 0.0050 | 1.05 | 0.035714 |
| 2 | (CORNFLAKES) | (COFFEE) | 0.30 | 0.40 | 0.2 | 0.666667 | 1.666667 | 0.0800 | 1.80 | 0.571429 |
| 3 | (SUGER) | (COFFEE) | 0.30 | 0.40 | 0.2 | 0.666667 | 1.666667 | 0.0800 | 1.80 | 0.571429 |
| 4 | (MAGGI) | (TEA) | 0.25 | 0.35 | 0.2 | 0.800000 | 2.285714 | 0.1125 | 3.25 | 0.750000 |

**Part B:**

Program FP tree using inbuilt functions for the following dataset

| TID | Items bought |
|---|---|
| 100 | {f, a, c, d, g, i, m, p} |
| 200 | {a, b, c, f, l, m, o} |
| 300 | {b, f, h, j, o, w} |
| 400 | {b, c, k, s, p} |
| 500 | {a, f, c, e, l, p, m, n} |

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:

FPTree with minimum confidence = 60.0%

| | support | itemsets |
|---|---|---|
| 0 | 0.8 | (f) |
| 1 | 0.8 | (c) |
| 2 | 0.6 | (p) |
| 3 | 0.6 | (m) |
| 4 | 0.6 | (a) |
| 5 | 0.6 | (b) |
| 6 | 0.6 | (c, f) |
| 7 | 0.6 | (p, c) |
| 8 | 0.6 | (c, m) |
| 9 | 0.6 | (f, m) |
| 10 | 0.6 | (c, f, m) |
| 11 | 0.6 | (a, m) |
| 12 | 0.6 | (a, c) |
| 13 | 0.6 | (a, f) |
| 14 | 0.6 | (a, m, c) |
| 15 | 0.6 | (a, f, m) |
| 16 | 0.6 | (a, f, c) |
| 17 | 0.6 | (a, m, c, f) |

Association rules are as follows

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction | zhangs_metric |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | (c) | (f) | 0.8 | 0.8 | 0.6 | 0.75 | 0.937500 | -0.04 | 0.8 | -0.25 |
| 1 | (f) | (c) | 0.8 | 0.8 | 0.6 | 0.75 | 0.937500 | -0.04 | 0.8 | -0.25 |
| 2 | (p) | (c) | 0.6 | 0.8 | 0.6 | 1.00 | 1.250000 | 0.12 | inf | 0.50 |
| 3 | (c) | (p) | 0.8 | 0.6 | 0.6 | 0.75 | 1.250000 | 0.12 | 1.6 | 1.00 |
| 4 | (c) | (m) | 0.8 | 0.6 | 0.6 | 0.75 | 1.250000 | 0.12 | 1.6 | 1.00 |
| 5 | (m) | (c) | 0.6 | 0.8 | 0.6 | 1.00 | 1.250000 | 0.12 | inf | 0.50 |
| 6 | (f) | (m) | 0.8 | 0.6 | 0.6 | 0.75 | 1.250000 | 0.12 | 1.6 | 1.00 |
| 7 | (m) | (f) | 0.6 | 0.8 | 0.6 | 1.00 | 1.250000 | 0.12 | inf | 0.50 |
| 8 | (c, f) | (m) | 0.6 | 0.6 | 0.6 | 1.00 | 1.666667 | 0.24 | inf | 1.00 |
| 9 | (c, m) | (f) | 0.6 | 0.8 | 0.6 | 1.00 | 1.250000 | 0.12 | inf | 0.50 |
| 10 | (f, m) | (c) | 0.6 | 0.8 | 0.6 | 1.00 | 1.250000 | 0.12 | inf | 0.50 |
| 11 | (c) | (f, m) | 0.8 | 0.6 | 0.6 | 0.75 | 1.250000 | 0.12 | 1.6 | 1.00 |
| 12 | (f) | (c, m) | 0.8 | 0.6 | 0.6 | 0.75 | 1.250000 | 0.12 | 1.6 | 1.00 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 13 | (m) | (c, f) | 0.6 | 0.6 | 0.6 | 1.00 | 1.666667 | 0.24 | inf | 1.00 |
| 14 | (a) | (m) | 0.6 | 0.6 | 0.6 | 1.00 | 1.666667 | 0.24 | inf | 1.00 |
| 15 | (m) | (a) | 0.6 | 0.6 | 0.6 | 1.00 | 1.666667 | 0.24 | inf | 1.00 |
| 16 | (a) | (c) | 0.6 | 0.8 | 0.6 | 1.00 | 1.250000 | 0.12 | inf | 0.50 |
| 17 | (c) | (a) | 0.8 | 0.6 | 0.6 | 0.75 | 1.250000 | 0.12 | 1.6 | 1.00 |
| 18 | (a) | (f) | 0.6 | 0.8 | 0.6 | 1.00 | 1.250000 | 0.12 | inf | 0.50 |
| 19 | (f) | (a) | 0.8 | 0.6 | 0.6 | 0.75 | 1.250000 | 0.12 | 1.6 | 1.00 |
| 20 | (a, m) | (c) | 0.6 | 0.8 | 0.6 | 1.00 | 1.250000 | 0.12 | inf | 0.50 |
| 21 | (a, c) | (m) | 0.6 | 0.6 | 0.6 | 1.00 | 1.666667 | 0.24 | inf | 1.00 |
| 22 | (c, m) | (a) | 0.6 | 0.6 | 0.6 | 1.00 | 1.666667 | 0.24 | inf | 1.00 |
| 23 | (a) | (c, m) | 0.6 | 0.6 | 0.6 | 1.00 | 1.666667 | 0.24 | inf | 1.00 |
| 24 | (m) | (a, c) | 0.6 | 0.6 | 0.6 | 1.00 | 1.666667 | 0.24 | inf | 1.00 |
| 25 | (c) | (a, m) | 0.8 | 0.6 | 0.6 | 0.75 | 1.250000 | 0.12 | 1.6 | 1.00 |
| 26 | (a, f) | (m) | 0.6 | 0.6 | 0.6 | 1.00 | 1.666667 | 0.24 | inf | 1.00 |
| 27 | (a, m) | (f) | 0.6 | 0.8 | 0.6 | 1.00 | 1.250000 | 0.12 | inf | 0.50 |
| 28 | (f, m) | (a) | 0.6 | 0.6 | 0.6 | 1.00 | 1.666667 | 0.24 | inf | 1.00 |
| 29 | (a) | (f, m) | 0.6 | 0.6 | 0.6 | 1.00 | 1.666667 | 0.24 | inf | 1.00 |
| 30 | (f) | (a, m) | 0.8 | 0.6 | 0.6 | 0.75 | 1.250000 | 0.12 | 1.6 | 1.00 |

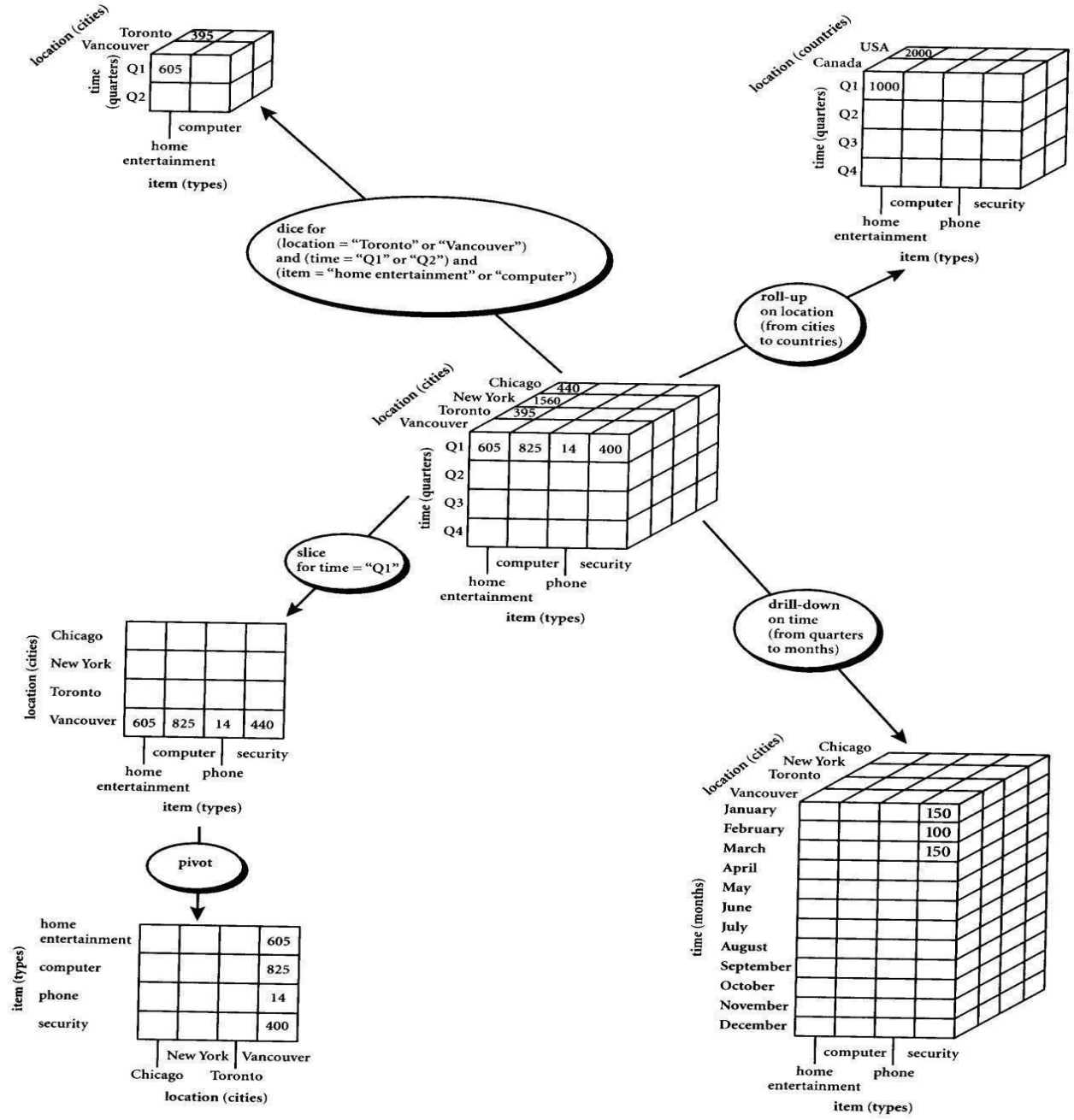
## Conclusion:

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:

(i) Doctor

(ii) Patient

(iii) Time

With two measures

(a) Count

(b) Charge

Where Charge is the fee that the Doctor charges a patient for a visit.

Using the above example describe the following operations:

(i) Slice

(ii) Dice

(iii) Roll Up

(iv) Drill Down (v) Pivot

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

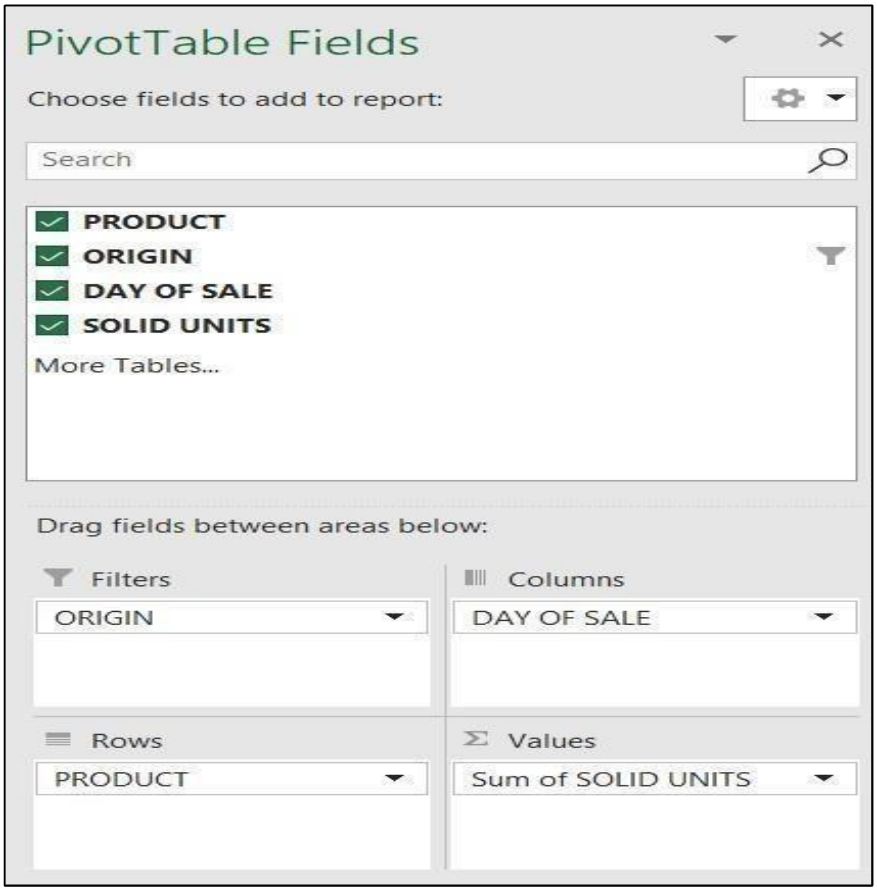
**EXERCISE 2**

## 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:

| A | B | C | D |
|---|---|---|---|
| PRODUCT | ORIGIN | DAY OF SALE | SOLID UNITS |
| A | East | 01-01-2023 | 8 |
| B | Central | 02-01-2023 | 4 |
| C | Central | 03-01-2023 | 2 |
| D | Central | 04-01-2023 | 5 |
| E | West | 05-01-2023 | 6 |
| F | East | 06-01-2023 | 9 |
| G | Central | 07-01-2023 | 9 |
| H | Central | 08-01-2023 | 1 |
| I | West | 09-01-2023 | 1 |
| J | East | 01-01-2023 | 8 |
| K | Central | 02-01-2023 | 4 |
| L | East | 03-01-2023 | 6 |
| M | East | 04-01-2023 | 7 |
| N | East | 05-01-2023 | 9 |
| O | Central | 06-01-2023 | 5 |
| P | East | 07-01-2023 | 2 |
| Q | Central | 08-01-2023 | 1 |
| R | East | 09-01-2023 | 3 |
| S | East | 01-01-2023 | 6 |
| T | Central | 02-01-2023 | 5 |
| U | Central | 03-01-2023 | 8 |
| V | East | 04-01-2023 | 4 |
| X | Central | 05-01-2023 | 8 |
| Y | Central | 06-01-2023 | 9 |
| Z | East | 07-01-2023 | 2 |




| ORIGIN | East | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Sum of SOLID UNITS | Column Labels | | | | | | | |
| Row Labels | 01-01-2023 | 03-01-2023 | 04-01-2023 | 05-01-2023 | 06-01-2023 | 07-01-2023 | 09-01-2023 | Grand Total |
| A | 8 | | | | | | | 8 |
| F | | | | | 9 | | | 9 |
| J | 8 | | | | | | | 8 |
| L | | 6 | | | | | | 6 |
| M | | | 7 | | | | | 7 |
| N | | | | 9 | | | | 9 |
| P | | | | | | 2 | | 2 |
| R | | | | | | | 3 | 3 |
| S | 6 | | | | | | | 6 |
| V | | | 4 | | | | | 4 |
| Z | | | | | | 2 | | 2 |
| Grand Total | 22 | 6 | 11 | 9 | 9 | 4 | 3 | 64 |

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

**Name: Jigar Siddhpura** **SAPID:** 60004210155

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

# DMW - Experiment 8

## Code:

```
import numpy as np

def page_rank_algorithm(graph,damping_factor):
    outgoing=dict()
    incoming_nodes=dict()
    coefficients= dict()

    for i in range(len(graph)):
        outgoing[i]=0

    for i, node in enumerate(graph):
        for edge in node:
            if edge:
                outgoing[i] += 1

    for i in range(len(graph)):
        temp=[]
        for node in graph:
            if node[i]:
                temp.append(node)
        incoming_nodes[i] = temp

    coefficients_list = []

    for i,node in enumerate(graph):
        temp = []
        for j,other_node in enumerate(graph):
            if other_node in incoming_nodes[i]:
                temp.append(damping_factor*(1.0/outgoing[j]))
            elif i == j:
                temp.append(-1)
            else:
                temp.append(0)
        coefficients[i]= temp
        coefficients_list.append(temp)

    constant_matrix = []
    for i in range(len(graph)):
        constant_matrix.append(damping_factor-1)

    pageranks = np.linalg.solve(np.array(coefficients_list),np.array(constant_matrix))
    # print()
    for i,rank in enumerate(pageranks):
        print("Page Rank of {} is {:.4f}".format(chr(65+i), rank))
```

```python
if __name__=="__main__":
    n = int(input("Enter the number of nodes : "))
    d = float(input("Enter the damping factor : "))
    # graph repr connected points

    graph = []
    # graph = [[0,1,1,0],[1,0,1,0],[1,1,0,1],[0,0,1,0]]
    print("Enter Adjacency Matrix with terms separated by a space : ")
    for i in range(n):
        temp_list = input().split(" ")
        graph.append(list(map(int,temp_list)))
    page_rank_algorithm(graph,d)
```

**Output:**

```
Enter the number of nodes : 4
Enter the damping factor : 0.6
Enter Adjacency Matrix with terms separated by a space :
0 1 1 0
1 0 1 0
1 1 0 1
0 0 1 0
Page Rank of A is 0.9677
Page Rank of B is 0.9677
Page Rank of C is 1.3871
Page Rank of D is 0.6774
PS D:\SEM 5\DMW\EXPERIMENTS>
```

**Conclusion:**

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

**Name: Jigar Siddhpura** **SAPID:** 60004210155

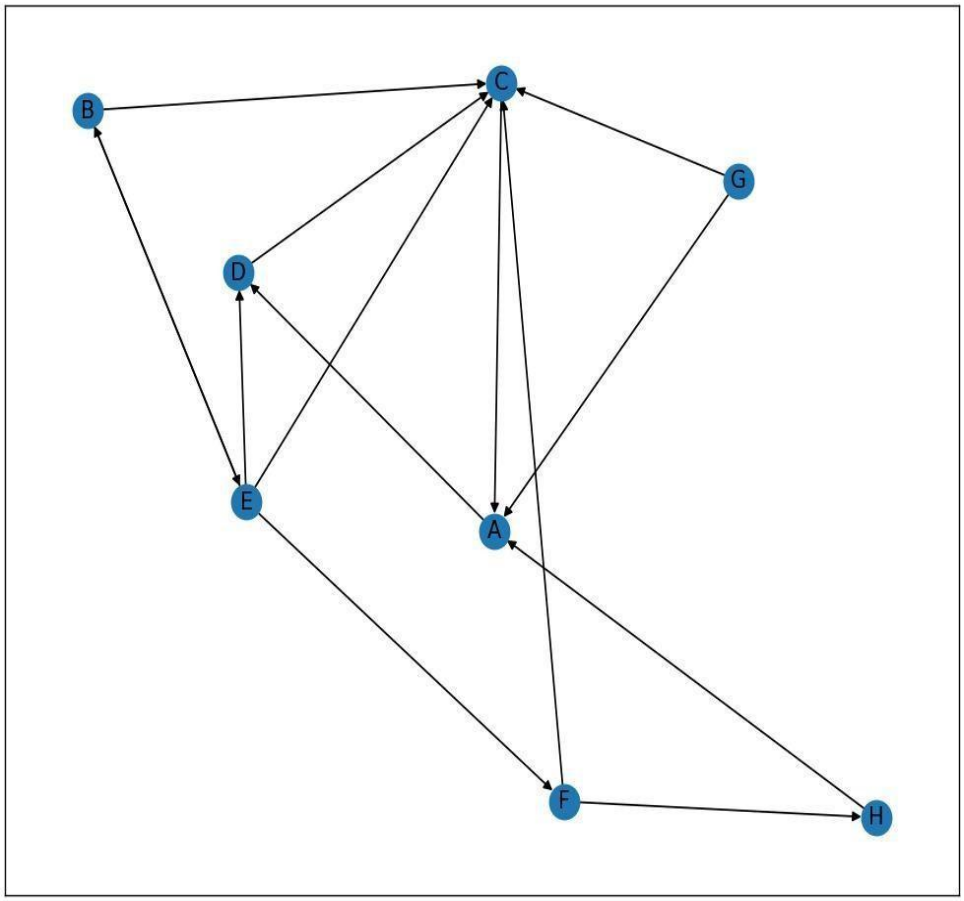
**DIV:** C/C2 **Branch:** Computer Engineering

# DMW - Experiment 9

## Code:

```
 mport networkx as nx
 mport matplotl b.pyplot as plt
 mport numpy as np

graph_matr x = 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],   fi -> 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( , label=node_label)
    for j in range(len(graph_matrix[ i])):
        if graph_matrix[i][j] == 1:
            G.add_edge( , j)

plt.f igure(f igs ize=(10, 10))
pos = nx.spring_layout(G)
nx.draw_networkx(G, pos=pos, with_labels=True, labels=labels)
hubs, authorities = nx.h its(G, max_ iter=50, normal ized=True)

print("Hub Scores:")
for key, value in hubs.items():
    print(f"{labels[key]}: {value}")
print()

print("Authori ty Scores:")
for key, value in authorities.items():
    print(f"{labels[key]}: {value}")
plt.show()
```

**Graph**:




**Output**:

```
Hub Scores:
A: 0.04642540403219996
D: 0.13366037526115382
B: 0.15763599442967324
C: 0.0373891322464265
E: 0.2588144598468665
F: 0.1576359944296732
H: 0.0373891322464265
G: 0.1710495075075803

Authority Scores:
A: 0.10864044011724333
D: 0.13489685434358004
B: 0.114379740733336446
C: 0.38837280038761807
E: 0.06966521184241486
F: 0.11437974073336447
H: 0.06966521184241474
G: -0.0
PS D:\SEM-5\DMW\EXPERIMENTS>
```

**Name: Jigar Siddhpura** **SAPID:** 60004210155

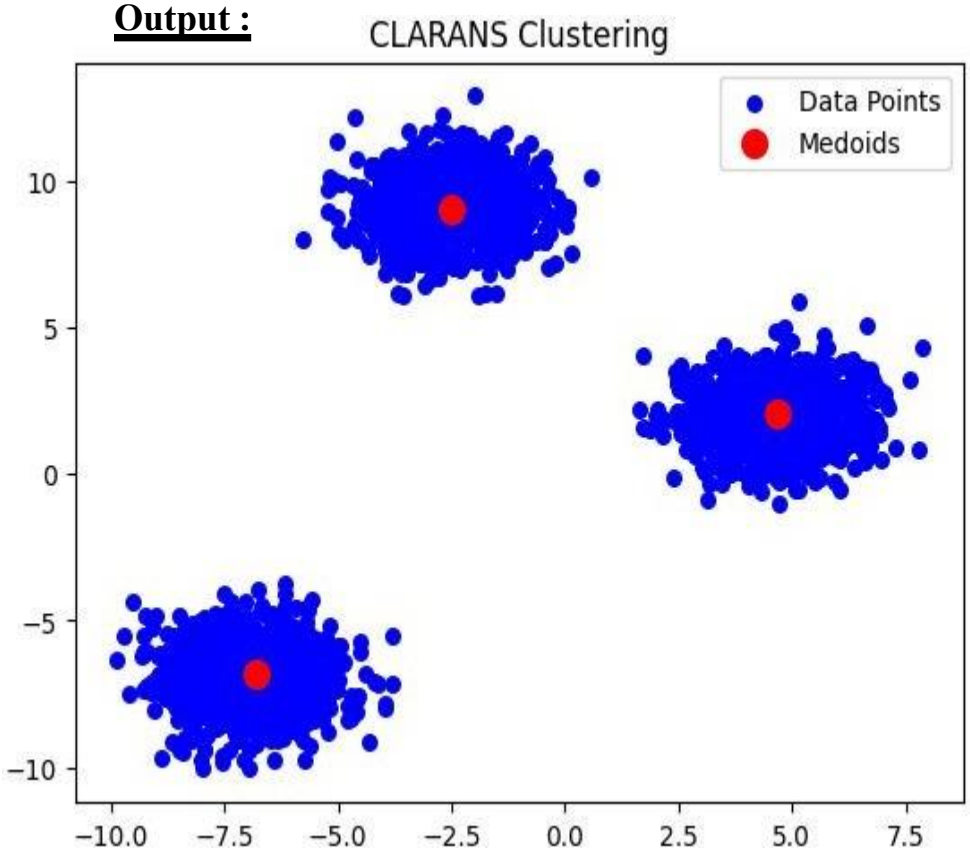
**DIV:** C/C2 **Branch:** Computer Engineering

# DMW - Experiment 10

## Code:

```
Click here to ask Blackbox to help you code faster
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 :**


CLARANS Clustering
Data Points
Medoids
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5