NAME: MIHIR ASHISH VORA

SAP ID: 60004220115

DIV: C12

TOPIC: DMW Experiment 3

Aim: Implementation of Classification Algorithm

Theory:

In data mining, classification is a supervised learning technique used to categorize or assign objects into predefined classes or categories based on their characteristics. It is a fundamental task in machine learning and data analysis and is used in a wide range of applications, including text classification, image recognition, spam detection, medical diagnosis, and many more.

Here are the key components and steps involved in classification in data mining:

Dataset: Classification begins with a dataset containing labeled examples. Each example consists of a set of features (attributes) and a class label. The features represent the characteristics or attributes of the data, while the class label indicates the category or class to which the example belongs.

Training and Testing Data: The dataset is typically divided into two subsets:

Training Data: This subset is used to train the classification model. The model learns the patterns and relationships in the data, which will enable it to make predictions.

Testing Data: This subset is used to evaluate the performance of the trained model. It helps assess how well the model generalizes to new, unseen data.

Feature Selection/Extraction: Feature selection involves choosing the most relevant features for the classification task. Feature extraction can transform and reduce the dimensionality of the data to improve model performance.

Classifier Selection: Select a classification algorithm (classifier) that best fits the problem. Common classifiers include decision trees, support vector machines, logistic regression, k-nearest neighbors, and Naive Bayes, among others.

Model Training: The selected classifier is trained on the training data by learning the relationships between the features and class labels.

Model Evaluation: The trained model is tested on the testing data to assess its performance. Common evaluation metrics include accuracy, precision, recall, F1 score, and the receiver operating characteristic (ROC) curve.

Model Tuning: If the model's performance is not satisfactory, you can fine-tune hyperparameters, adjust feature selection, or try different classifiers to improve accuracy.

Model Deployment: Once you are satisfied with the model's performance, you can deploy it to make predictions on new, unlabeled data. The deployed model can be used for automated classification tasks.

Monitoring and Maintenance: Continuously monitor the model's performance in a real-world setting. Models may require periodic updates as the data distribution or requirements change.

Classification in data mining is a powerful tool for automating decision-making processes and organizing data into meaningful categories. The choice of classifier and the quality of the dataset play a significant role in the success of a classification task.

Gaussian Naive Bayes (GNB) Classification:

Gaussian Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem with the "naive" assumption that all features are conditionally independent given the class label. In GNB, each feature's distribution is assumed to be Gaussian, which is often a reasonable assumption for continuous data. GNB is particularly useful when dealing with high-dimensional data and is known for its simplicity and computational efficiency. It's widely used in various applications, including text classification and spam detection. While GNB's independence assumption doesn't always hold in practice, it can still provide surprisingly good results and serves as a baseline for more complex classification techniques.

Decision Tree Classification:

Decision Tree is a non-parametric supervised learning algorithm that recursively splits the dataset into subsets based on the most significant feature, ultimately forming a tree-like structure. Each internal node of the tree represents a decision based on a specific feature, while the leaf nodes correspond to class labels. Decision Trees are intuitive and easily interpretable, making them valuable for tasks where transparency is important. They can handle both categorical and continuous data and can be used for classification and regression tasks. However, Decision Trees can suffer from overfitting if not pruned or regularized properly, which is why techniques like Random Forests and Gradient Boosting Trees have been developed to improve their performance and robustness.

Credit Card Fraud Detection:

Cell 1:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree
Classifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import (
    accuracy_score,
    confusion_matrix,
    ConfusionMatrixDisplay,
    f1_score,
)
```

Output: Successfully Imported

Cell 2:

```
raw_mail_data=pd.read_csv("/content/card_transdata.csv")
mail_data=raw_mail_data.where(pd.notnull(raw_mail_data))
mail_data.shape
```

Output: `(13747, 8)`

Cell 3:

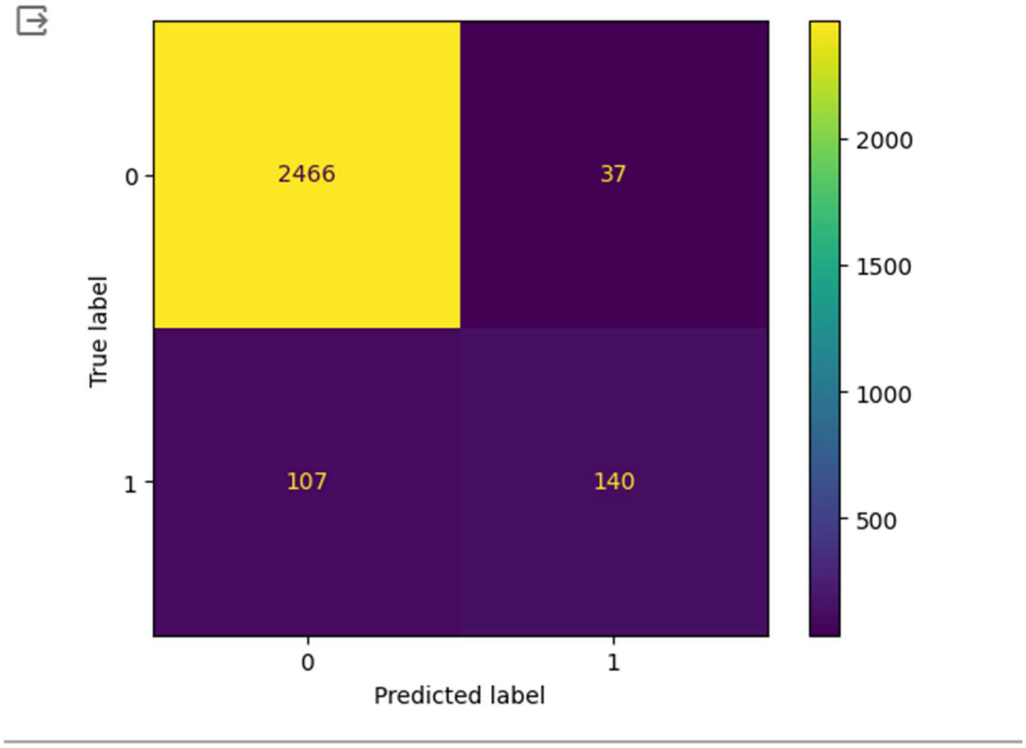
```
mail_data = mail_data.fillna(0)
X= mail_data.iloc[:, :-1].values
Y = mail_data.iloc[:, -1].values
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size =
0.2,  random_state=0)
clf = GaussianNB()
clf.fit(X_train,Y_train)
NB_predicted = clf.predict(X_test)
accuray = accuracy_score(NB_predicted, Y_test)
print("Accuracy:", accuray)
```

Output: `Accuracy: 0.9476363636363636`

Cell 4:

```
labels = [0,1]
cm = confusion_matrix(Y_test, NB_predicted, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=labels)
disp.plot();
```

Ouput:



Cell 5:

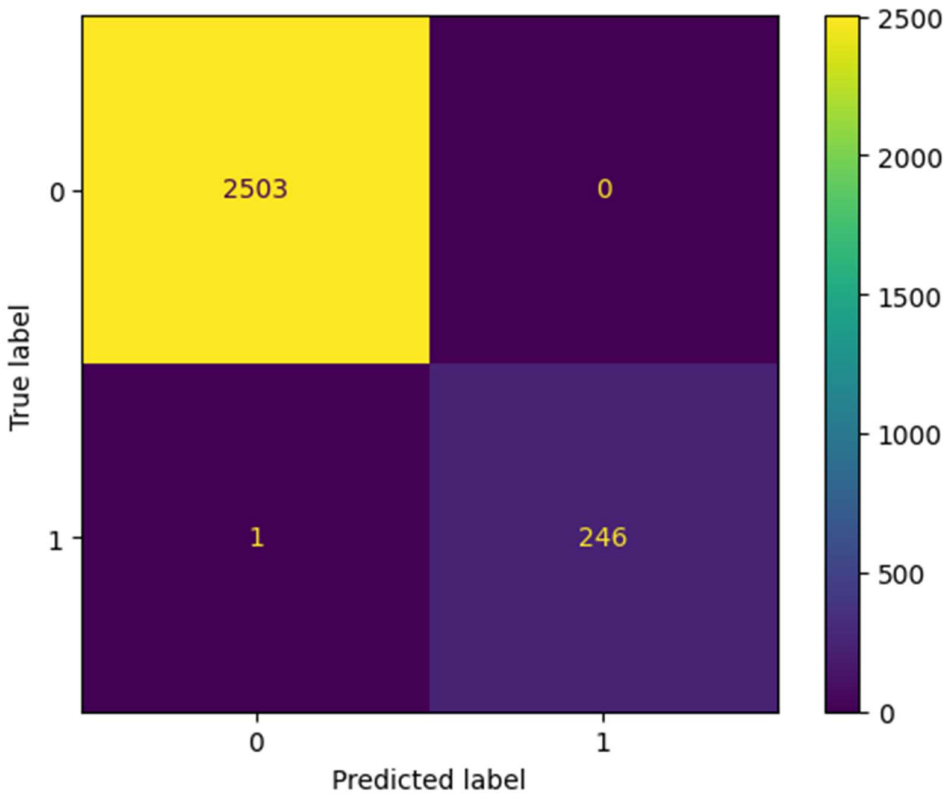
```
clf = DecisionTreeClassifier()
clf.fit(X_train,Y_train)
DT_predicted=clf.predict(X_test)
accuray = accuracy_score(DT_predicted, Y_test)
print("Accuracy:", accuray)
```

Output: Accuracy: `0.9996363636363637`

Cell 6:

```
labels = [0,1]
cm = confusion_matrix(Y_test, DT_predicted, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=labels)
disp.plot()
```

Output:



Airline Passenger Satisfaction Classification

Cell 7:

```
airline=pd.read_csv("/content/airline_passenger.csv")
airline['Gender'].replace({'Male': 0, 'Female': 1}, inplace=True)
airline['Customer Type'].replace({'Loyal Customer': 1, 'disloyal
Customer': 0}, inplace=True)
airline['Type of Travel'].replace({'Personal Travel': 0, 'Business
travel': 1}, inplace=True)
airline['Class'].replace({'Eco': 0, 'Eco Plus': 1,'Business':2},
inplace=True)
airline['satisfaction'].replace({'neutral or
dissatisfied':0,'satisfied':1}, inplace=True)
airline.head()
```

Output: Successfully Imported and Replaced Values

	Unnamed: 0	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	...	Inflight entertainment	On-board service	Leg room service	Baggage handling	Checkin service	Inflight service	Cleanliness	Departure Delay in Minutes	Arrival Delay in Minutes	satisfaction
0	0	70172	0.0	1.0	13.0	0.0	1.0	460.0	3.0	4.0	...	5.0	4.0	3.0	4.0	4.0	5.0	5.0	25.0	18.0	0.0
1	1	5047	0.0	0.0	25.0	1.0	2.0	235.0	3.0	2.0	...	1.0	1.0	5.0	3.0	1.0	4.0	1.0	1.0	6.0	0.0
2	2	110028	1.0	1.0	26.0	1.0	2.0	1142.0	2.0	2.0	...	5.0	4.0	3.0	4.0	4.0	4.0	5.0	0.0	0.0	1.0
3	3	24026	1.0	1.0	25.0	1.0	2.0	562.0	2.0	5.0	...	2.0	2.0	5.0	3.0	1.0	4.0	2.0	11.0	9.0	0.0
4	4	119299	0.0	1.0	61.0	1.0	2.0	214.0	3.0	3.0	...	3.0	3.0	4.0	4.0	3.0	3.0	3.0	0.0	0.0	1.0

5 rows × 25 columns

Cell 8:

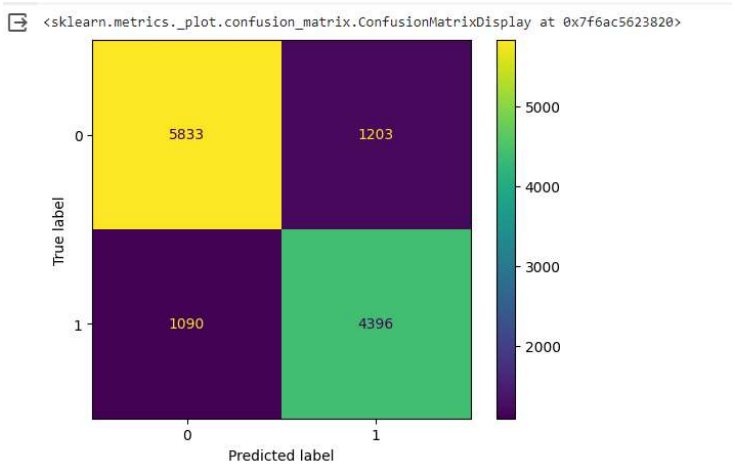
```
airline = airline.fillna(0)
X= airline.iloc[:, :-1].values
Y = airline.iloc[:, -1].values
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size =
0.2,  random_state=0)
clf = GaussianNB()
clf.fit(X_train,Y_train)
NB_predicted2 = clf.predict(X_test)
NB2 = accuracy_score(NB_predicted2, Y_test)
print("Accuracy:", NB2)
```

Output: `Accuracy: 0.8168822871745728`

Cell 9:

```
labels = [0,1]
cm = confusion_matrix(Y_test, NB_predicted2, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=labels)
disp.plot()
```

Output:



Cell 10:

```
clf = DecisionTreeClassifier()
clf.fit(X_train,Y_train)
DT_predicted2 = clf.predict(X_test)
DT2 = accuracy_score(DT_predicted2, Y_test)
print("Accuracy:", DT2)
```

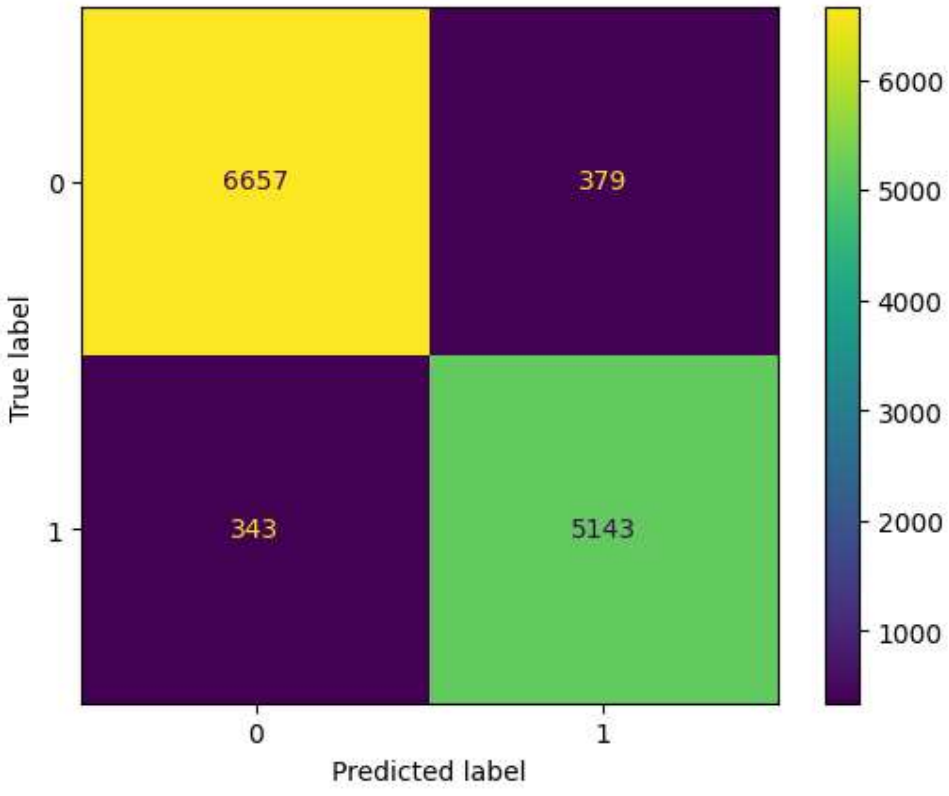
Output:

```
Accuracy: 0.9444178246286535
```

Cell 11:

```
labels = [0,1]
cm = confusion_matrix(Y_test, DT_predicted2, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=labels)
disp.plot()
```

Output:



Breast Cancer Classification

Cell 12:

```
bst=pd.read_csv("/content/Breast_Cancer.csv")
bst['Race'].replace({'White': 0, 'Black': 1,'Other':2}, inplace=True)
bst['Marital Status'].replace({'Single': 0, 'Married':
1,'Divorced':2,'Widowed':3,'Single ':0,'Separated':2}, inplace=True)
bst['T Stage '].replace({'T1': 1, 'T2': 2, 'T3': 3, 'T4': 4},
inplace=True)
bst['N Stage'].replace({'N1': 1, 'N2': 2, 'N3': 3}, inplace=True)
bst['6th Stage'].replace({'IIA': 0, 'IIIA':
1,'IIB':2,'IIIB':3,'IIIC':4}, inplace=True)
bst['differentiate'].replace({'Poorly differentiated':0,'Moderately
differentiated':1,'Well differentiated':2,'Undifferentiated':3},
inplace=True)
bst['Grade'].replace({'anaplastic; Grade IV':4,' anaplastic; Grade
IV':4},inplace=True)
bst['A Stage'].replace({'Regional':0,'Distant':1,}, inplace=True)
bst['Estrogen Status'].replace({'Positive':1,'Negative':0},
inplace=True)
bst['Progesterone Status'].replace({'Positive':1,'Negative':0},
inplace=True)
bst['Status'].replace({'Alive':0,'Dead':1}, inplace=True)
bst.head()
```

Output:

	Age	Race	Marital Status	T Stage	N Stage	6th Stage	differentiate	Grade	A Stage	Tumor Size	Estrogen Status	Progesterone Status	Regional Node Examined	Reginol Node Positive	Survival Months	Status
0	68	0	1	1	1	0	0	3	0	4	1	1	24	1	60	0
1	50	0	1	2	2	1	1	2	0	35	1	1	14	5	62	0
2	58	0	2	3	3	4	1	2	0	63	1	1	14	7	75	0
3	58	0	1	1	1	0	0	3	0	18	1	1	2	1	84	0
4	47	0	1	2	1	2	0	3	0	41	1	1	3	1	50	0

+ Code + Text

Cell 13:

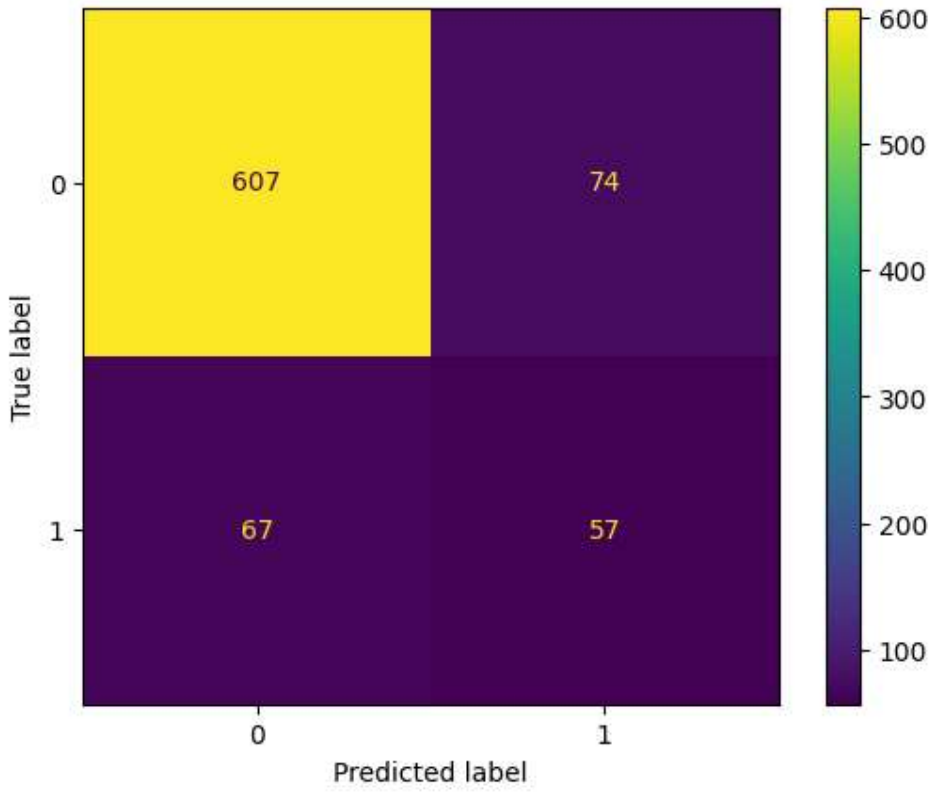
```
bst = bst.fillna(0)
X= bst.iloc[:, :-1].values
Y = bst.iloc[:, -1].values
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size =
0.2,  random_state=0)
clf = GaussianNB()
clf.fit(X_train,Y_train)
NB_predicted3 = clf.predict(X_test)
NB3 = accuracy_score(NB_predicted3, Y_test)
print("Accuracy:", NB3)
```

Output: Accuracy: 0.8248447204968944

Cell 14:

```
labels = [0,1]
cm = confusion_matrix(Y_test, NB_predicted3, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=labels)
disp.plot()
```

Output:



Cell 15:

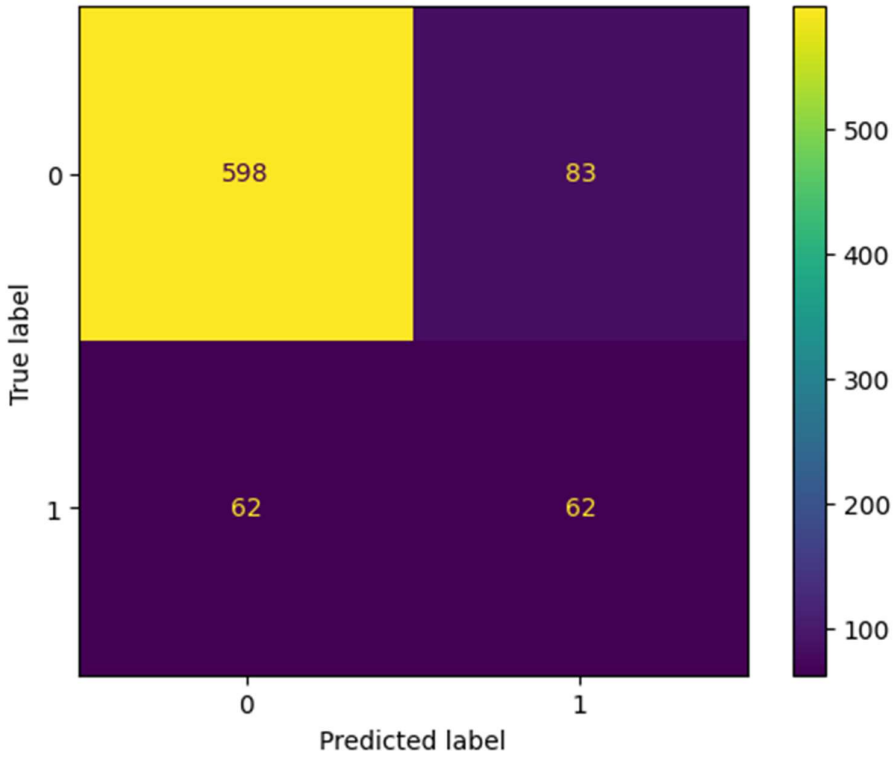
```
clf = DecisionTreeClassifier()
clf.fit(X_train,Y_train)
DT_predicted3 = clf.predict(X_test)
DT3 = accuracy_score(DT_predicted3, Y_test)
print("Accuracy:", DT3)
```

Output: `Accuracy: 0.8198757763975155`

Cell 16:

```
labels = [0,1]
cm = confusion_matrix(Y_test, DT_predicted3, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=labels)
disp.plot()
```

Output:



Mobile Price Range Classification

Cell 17:

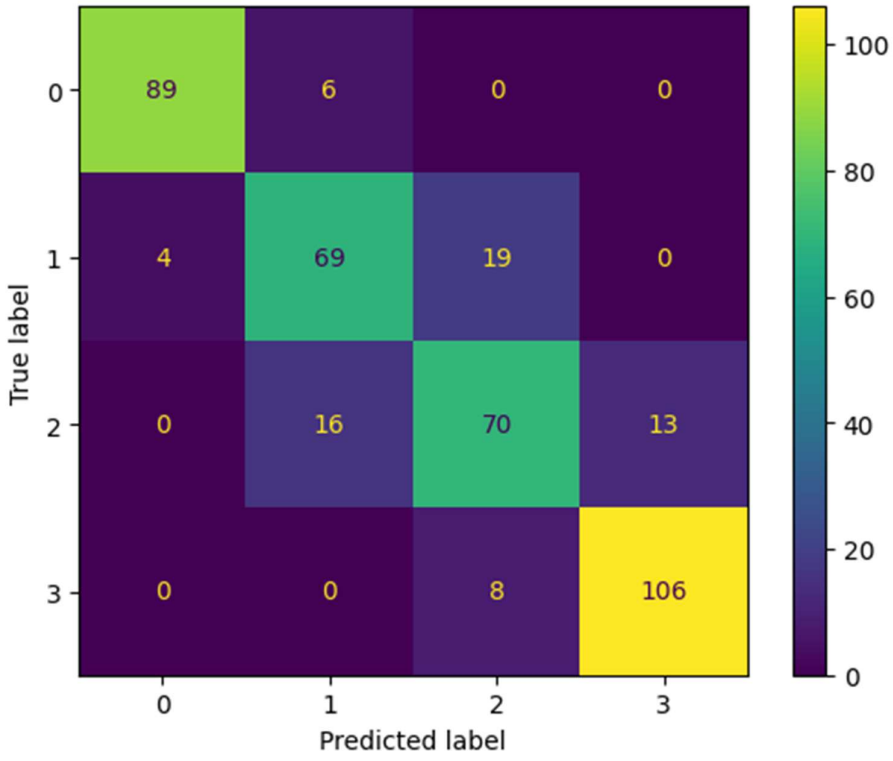
```
mobile_price=pd.read_csv("/content/mobile_price_range.csv")
X= mobile_price.iloc[:, :-1].values
Y = mobile_price.iloc[:, -1].values
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size =
0.2,  random_state=0)
clf = GaussianNB()
clf.fit(X_train,Y_train)
NB_predicted4 = clf.predict(X_test)
NB4 = accuracy_score(NB_predicted4, Y_test)
print("Accuracy:", NB4)
```

Output: Accuracy: 0.835

Cell 18:

```
labels = [0,1,2,3]
cm = confusion_matrix(Y_test, NB_predicted4, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=labels)
disp.plot()
```

Output:



Cell 19:

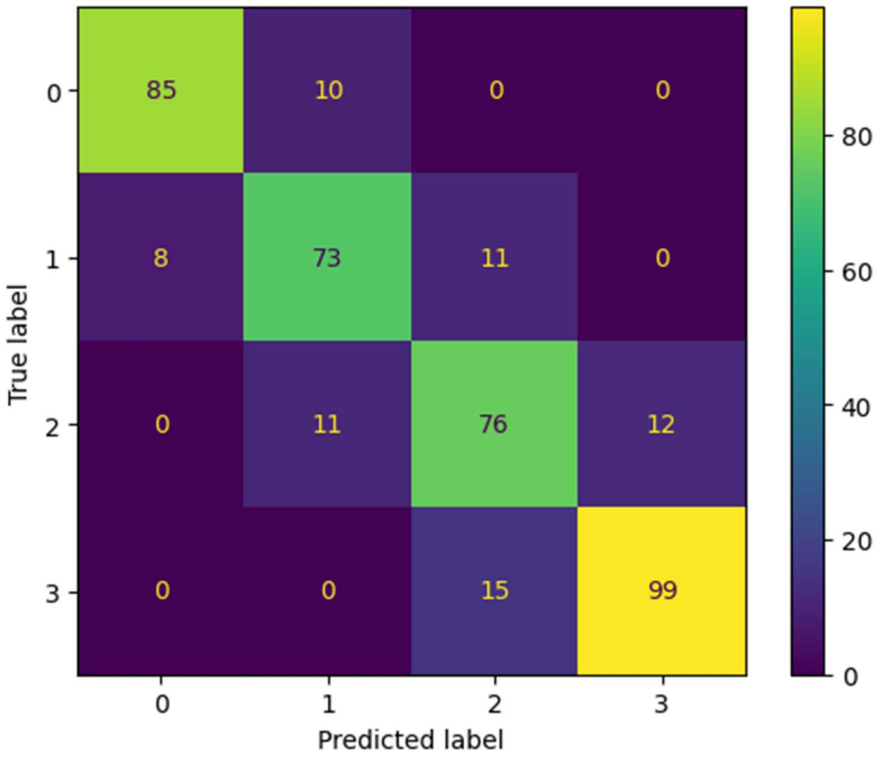
```
clf = DecisionTreeClassifier()
clf.fit(X_train,Y_train)
DT_predicted4= clf.predict(X_test)
DT4 = accuracy_score(DT_predicted4, Y_test)
print("Accuracy:", DT4)
```

Output: `Accuracy: 0.8325`

Cell 20:

```
labels = [0,1,2,3]
cm = confusion_matrix(Y_test, DT_predicted4, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=labels)
disp.plot()
```

Output:



Diabetes Prediction(Present-Absent) Classification

Cell 21:

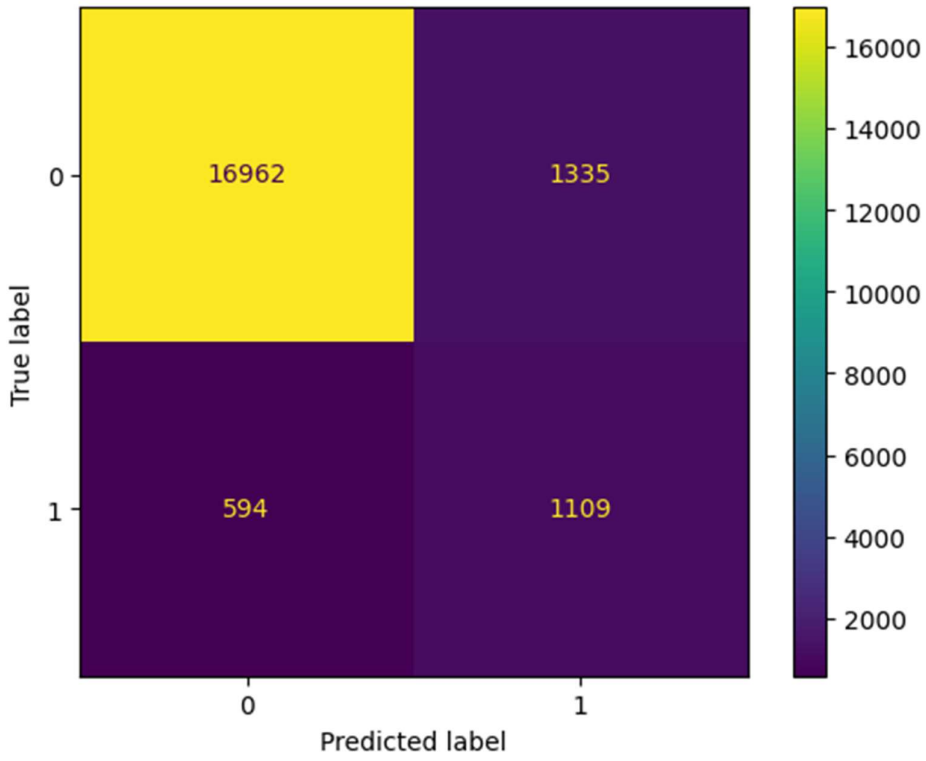
```
diabetes=pd.read_csv("/content/diabetes_prediction_dataset.csv")
diabetes['gender'].replace({'Male': 0, 'Female': 1,'Other':2},
inplace=True)
diabetes['smoking_history'].replace({'never': 0, 'No Info':
1,'current':2,'former':3,'ever':4,'not current':5}, inplace=True)
X= diabetes.iloc[:, :-1].values
Y = diabetes.iloc[:, -1].values
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size =
0.2,  random_state=0)
clf = GaussianNB()
clf.fit(X_train,Y_train)
NB_predicted5 = clf.predict(X_test)
NB5 = accuracy_score(NB_predicted5, Y_test)
print("Accuracy:", NB5)
```

Output: `Accuracy: 0.90355`

Cell 22:

```
labels = [0,1]
cm = confusion_matrix(Y_test, NB_predicted5, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=labels)
disp.plot()
```

Output:



Cell 23:

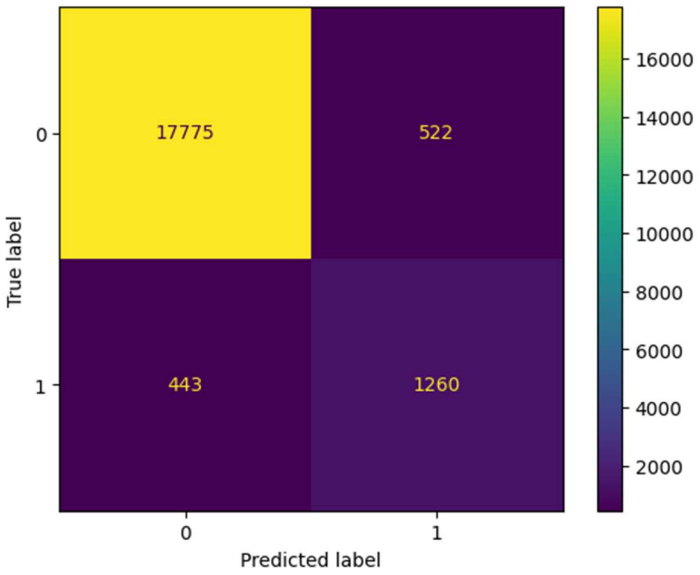
```
clf = DecisionTreeClassifier()
clf.fit(X_train,Y_train)
DT_predicted5= clf.predict(X_test)
DT5 = accuracy_score(DT_predicted5, Y_test)
print("Accuracy:", DT5)
```

Output: `Accuracy: 0.95105`

Cell 24:

```
labels = [0,1]
cm = confusion_matrix(Y_test, DT_predicted5, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=labels)
disp.plot()
```

Output:



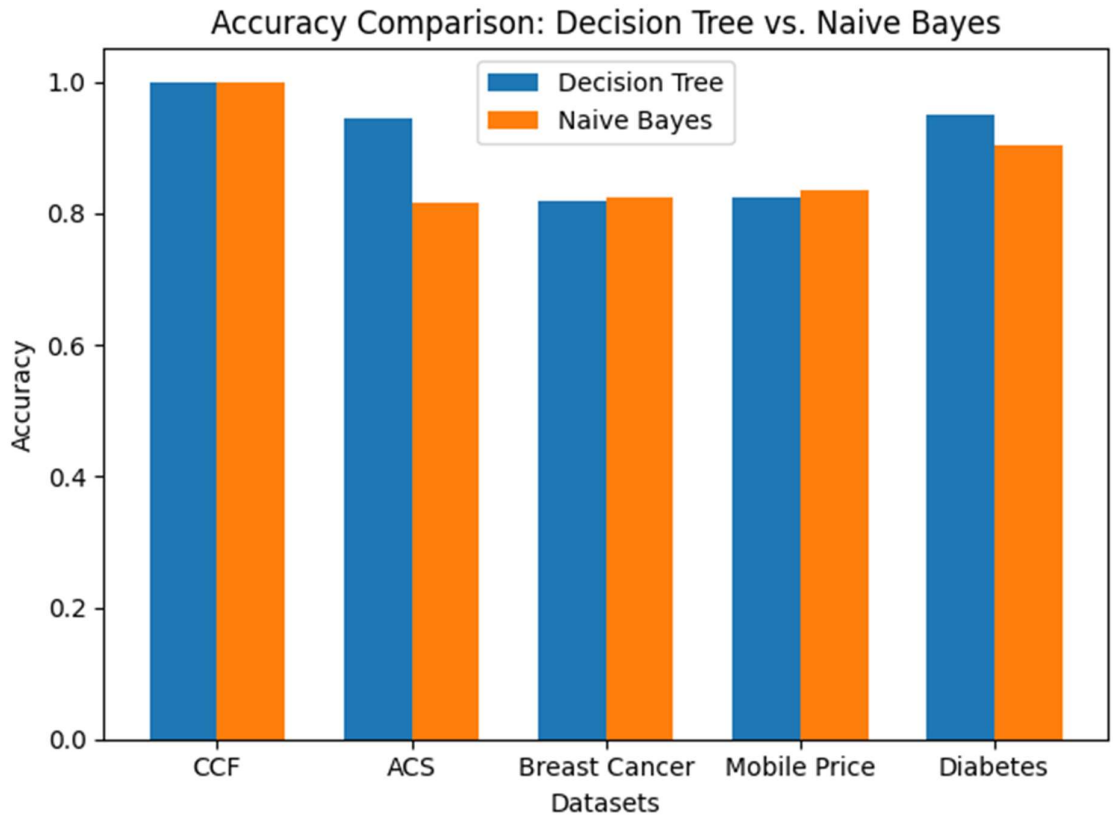
Comparison of Accuracy(Naïve Bayes vs Decision Tree)

Cell 25:

```
import matplotlib.pyplot as plt
import numpy as np
width = 0.35
datasets = ['CCF', 'ACS', 'Breast Cancer', 'Mobile Price', 'Diabetes']
x = np.arange(len(datasets))
naive_bayes_accuracy = [NB1,NB2,NB3,NB4,NB5]  # Replace with your
actual values
decision_tree_accuracy = [DT1,DT2,DT3,DT4,DT5]  # Replace with your
actual values
fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, decision_tree_accuracy, width,
label='Decision Tree')
rects2 = ax.bar(x + width/2, naive_bayes_accuracy, width, label='Naive
Bayes')
ax.set_xlabel('Datasets')
ax.set_ylabel('Accuracy')
ax.set_title('Accuracy Comparison: Decision Tree vs. Naive Bayes')
ax.set_xticks(x)
ax.set_xticklabels(datasets)
ax.legend()
```

```
plt.tight_layout()
plt.show()
```

Output:



KFOLD

Cell 26:

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
kfold = KFold(n_splits=10, shuffle=True, random_state=42)
scores = cross_val_score(clf, X, Y, cv=kfold, scoring='accuracy')
for i in scores:
  print(i)
```

Output:

```
1.0
0.9992727272727273
0.9992727272727273
1.0
0.9992727272727273
0.9992727272727273
0.9985454545454545
0.9985443959243085
0.9985443959243085
0.9978165938864629
```

ADABOOST

Cell 27:

```
from sklearn.ensemble import AdaBoostClassifier
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=42)
ada = AdaBoostClassifier(n_estimators=50,learning_rate=1)
model = ada.fit(X_train, Y_train)
Ada_pred = model.predict(X_test)
accuray = accuracy_score(Y_test, Ada_pred)
print("Accuracy",accuray)
```

Output: Accuracy 0.9996363636363637

RandomForest

Cell 28:

```
from sklearn.ensemble import RandomForestClassifier
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=42)
random_forest = RandomForestClassifier(n_estimators=100,
random_state=42)
model = random_forest.fit(X_train, Y_train)
RF_pred = model.predict(X_test)
accuray = accuracy_score(Y_test, RF_pred)
print("Accuracy",accuray)
```

Output: Accuracy 0.9996363636363637

Bagging

Cell 29:

```
from sklearn.ensemble import BaggingClassifier
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=42)
bg_clf=BaggingClassifier(n_estimators=100, random_state=42)
model=bg_clf.fit(X_train,Y_train)
BG_pred=model.predict(X_test)
accuray = accuracy_score(Y_test, BG_pred)
print("Accuracy",accuray)
```

Output: Accuracy 0.9996363636363637

GradientBoosting

Cell 30:

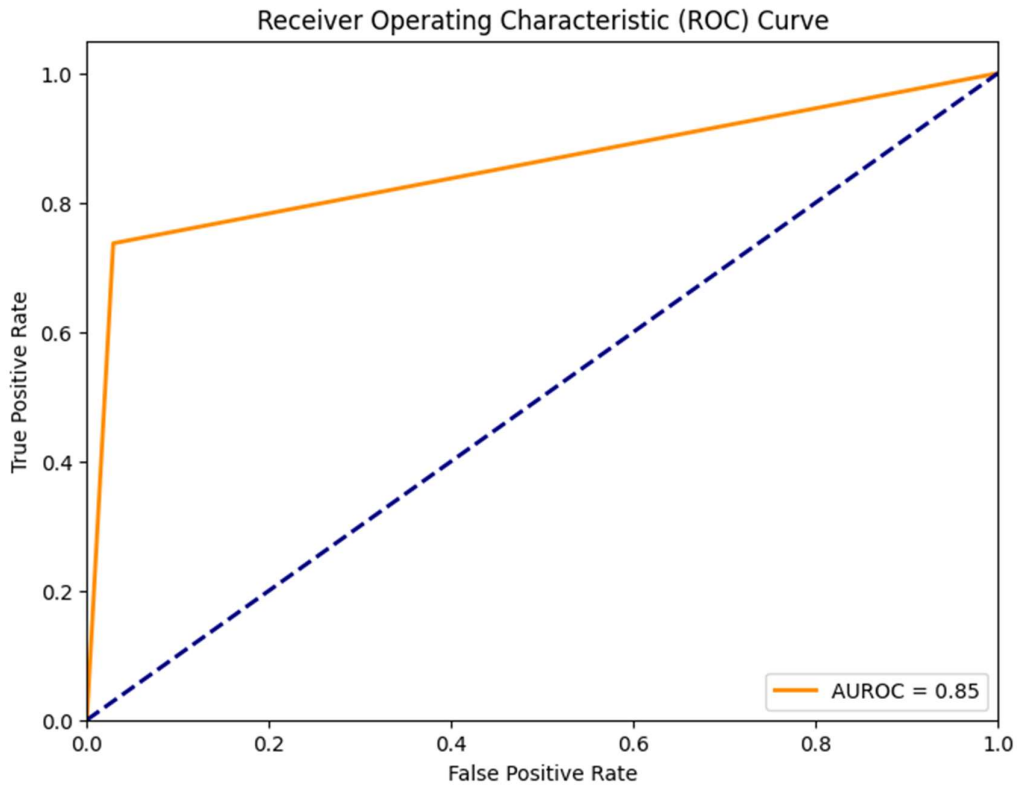
```
from sklearn.ensemble import GradientBoostingClassifier
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=42)
grad_clf=BaggingClassifier(n_estimators=100, random_state=42)
model=grad_clf.fit(X_train,Y_train)
grad_pred=model.predict(X_test)
accuray = accuracy_score(Y_test, grad_pred)
print("Accuracy",accuray)
```

Output: Accuracy 0.9996363636363637

Plotting AUROC for first dataset(Naïve Bayes):

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, roc_auc_score, auc
fpr, tpr, thresholds = roc_curve(Y_test, NB_predicted)
auroc = roc_auc_score(Y_test, NB_predicted)
print(f"AUROC: {auroc}")
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'AUROC =
{auroc:.2f}')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```

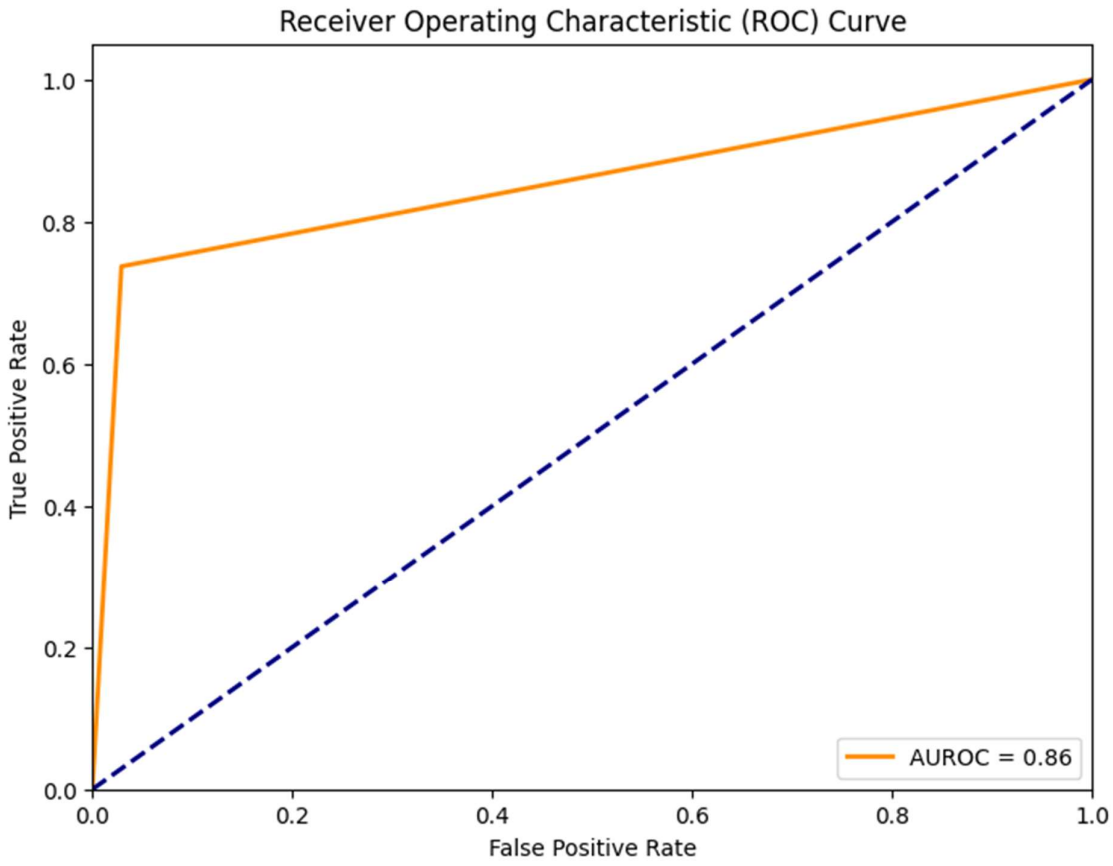
Output:



Plotting AUROC for first dataset(Decision Tree):

```
fpr, tpr, thresholds = roc_curve(Y_test, NB_predicted)
auroc = roc_auc_score(Y_test, DT_predicted)
print(f"AUROC: {auroc}")
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'AUROC =
{auroc:.2f}')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```

Output:



Conclusion: Thus, we implemented classification algorithms on five datasets and compared the accuracies of Naïve Bayes and Decision Tree Classification Algorithm. Also, implemented ensemble models and k-fold cross validation.