Name- VAIBHAV PATEL

Scholar No: 211112262

In [ ]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.naive_bayes import GaussianNB

data=pd.read_csv('iris.csv')
x=data[["sepal.length","sepal.width","petal.length","petal.width"]]
y=data['variety']

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_s

log_reg=GaussianNB()
log_reg.fit(x_train,y_train)
y_pred=log_reg.predict(x_test)
print(confusion_matrix(y_test,y_pred))
from sklearn.metrics import accuracy_score, precision_score, recall_score

# Assuming y_true and y_pred are the true labels and predicted labels


# Calculate performance metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred,average="micro")
recall = recall_score(y_test, y_pred,average="micro")
f1 = f1_score(y_test, y_pred,average="micro")
gmean = np.sqrt(recall*(1-precision))
tpr = recall
fpr = 1 - recall







print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("Gmean:", gmean)
print("TPR:", tpr)
print("FAR:", fpr)
```

```
[[16  0  0]
 [ 0 18  0]
 [ 0  0 11]]
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
Gmean: 0.0
TPR: 1.0
FAR: 0.0
```

In [ ]:

```
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.naive_bayes import GaussianNB

data=pd.read_csv('PlayTennis.csv')
le = preprocessing.LabelEncoder()
data_train_df = pd.DataFrame(data)
data_train_df_encoded = data_train_df.apply(le.fit_transform)
x=data_train_df_encoded[["Outlook","Temperature","Humidity","Wind"]]
y=data_train_df_encoded["Play Tennis"]

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_s

log_reg=GaussianNB()
log_reg.fit(x_train,y_train)
y_pred=log_reg.predict(x_test)
print(confusion_matrix(y_test,y_pred))
from sklearn.metrics import accuracy_score, precision_score, recall_score

# Assuming y_true and y_pred are the true labels and predicted labels


# Calculate performance metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
gmean = np.sqrt(recall*(1-precision))
tpr = recall
fpr = 1 - recall









print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("Gmean:", gmean)
print("TPR:", tpr)
print("FAR:", fpr)
```

```
[[0 0]
 [3 2]]
Accuracy: 0.4
Precision: 1.0
Recall: 0.4
F1 Score: 0.5714285714285715
Gmean: 0.0
TPR: 0.4
FAR: 0.6
```

In [ ]:

```
# Importing necessary libraries
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report

# Loading the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,

# Initializing Gaussian Naive Bayes classifier
classifier = GaussianNB()

# Training the classifier
classifier.fit(X_train, y_train)

# Predicting the classes for test set
y_pred = classifier.predict(X_test)

# Calculating accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Generating classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=iris.target_name
```

```
Accuracy: 1.0

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   virginica       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

In [ ]:

```
import numpy as np
from sklearn.metrics import confusion_matrix, precision_score, recall_sco
```

```python
def performance_metrics(y_true, y_pred, y_prob=None):
    # 1. Confusion Matrix
    cm = confusion_matrix(y_true, y_pred)

    # 2. Precision
    precision = precision_score(y_true, y_pred,average="micro")

    # 3. Recall
    recall = recall_score(y_true, y_pred,average="micro")

    # 4. Gmean
    gmean = np.sqrt(recall * (1 - precision))

    # 5. True Positive Rate
    tpr = recall

    # 6. Area under Curve (ROC AUC Score)
    auc = roc_auc_score(y_true, y_prob) if y_prob is not None else None

    # 7. False Alarm Rate (Fallout)
    fpr = 1 - recall

    # 8. F1 Score (F-Measure)
    f1 = f1_score(y_true, y_pred,average="micro")

    # 9. Overall Accuracy
    accuracy = accuracy_score(y_true, y_pred)

    return {
        "Confusion Matrix": cm,
        "Precision": precision,
        "Recall": recall,
        "Gmean": gmean,
        "True Positive Rate": tpr,
        "Area under Curve": auc,
        "False Alarm Rate": fpr,
        "F1 Score": f1,
        "Overall Accuracy": accuracy
    }
```

In [ ]:

```python
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics

df = pd.read_csv("IMDB Dataset.csv")

vectorizer = CountVectorizer(stop_words='english', max_features=5000)
X = vectorizer.fit_transform(df['review'].values.astype('U')).toarray()
y = df['sentiment'].map({'positive': 1, 'negative': 0})

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,

clf = MultinomialNB()
```

```python
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

metrics = performance_metrics(y_test, y_pred)
for metric, value in metrics.items():
    print(f"{metric}: {value}")
```

```
Confusion Matrix: [[6295 1116]
 [1218 6371]]
Precision: 0.8444
Recall: 0.8444
Gmean: 0.36247570953099734
True Positive Rate: 0.8444
Area under Curve: None
False Alarm Rate: 0.15559999999999996
F1 Score: 0.8444
Overall Accuracy: 0.8444
```

In [ ]:

```python
import numpy as np

# Sigmoid function
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

# Predicted probabilities using a logistic model
def predict_proba(w0, w1, x1):
    z = w0 + w1 * x1
    return sigmoid(z)

# Binary cross-entropy loss or logistic loss
def logistic_loss(y_true, y_pred):
    return -np.mean(y_true * np.log(y_pred + 1e-15) + (1 - y_true) * np.l

# Least squares error function
def least_squares_error(y_true, y_pred):
    return np.mean((y_true - y_pred) ** 2)

# Example usage:

# True labels
y_true = np.array([0, 1, 1, 0])

# Features
x1 = np.array([0.5, 2.3, -1.5, 0.7])

# Model parameters
w0, w1 = 0.1, 1.0

# Predicted probabilities
y_pred = predict_proba(w0, w1, x1)

# Logistic loss
print("Logistic Loss:", logistic_loss(y_true, y_pred))

# Least squares error
```

```python
print("Least Squares Error:", least_squares_error(y_true, y_pred))
```

```
Logistic Loss: 0.9789605446265129
Least Squares Error: 0.3858383847203733
```