NavieBayes-Classifier

#lets import libraries

import pandas as pd

#lets read the dataset

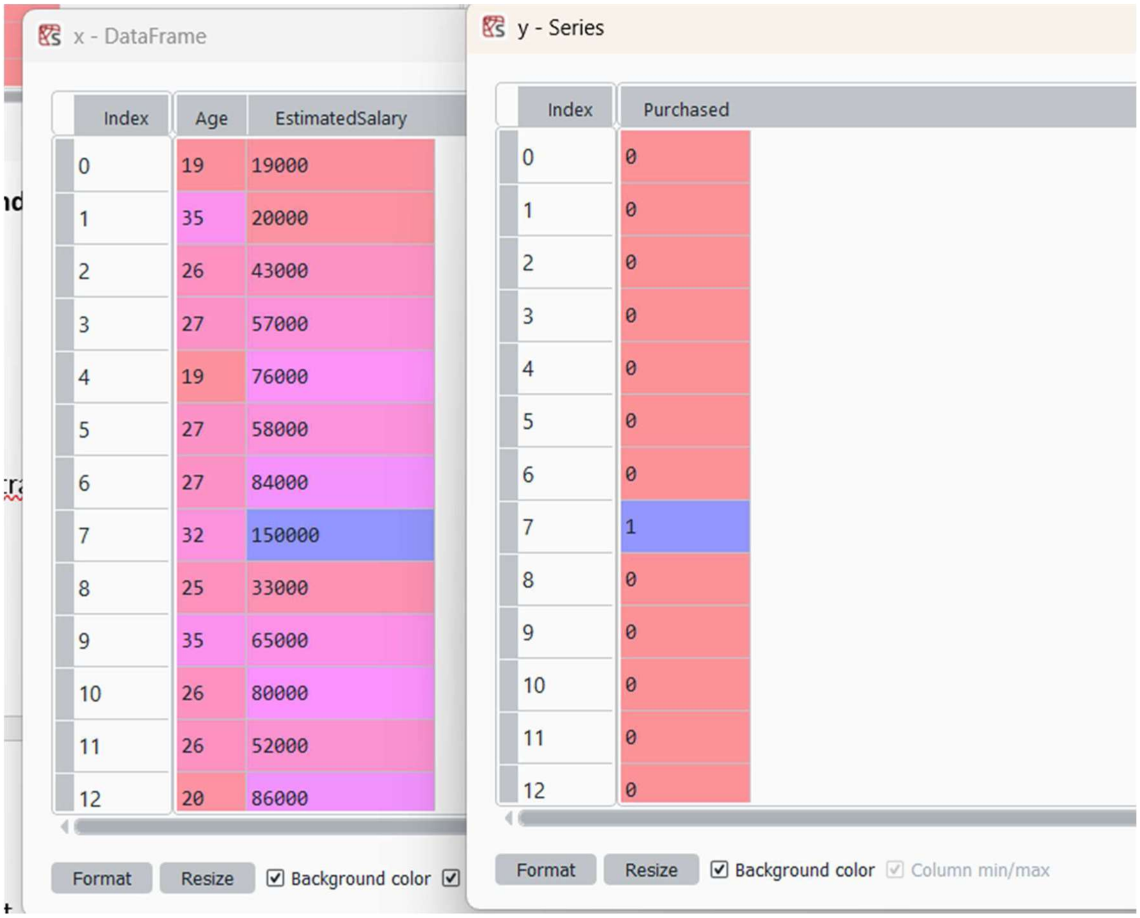
data=pd.read_csv(r"C:\Users\TharunMahendra\NIT\6.Algorithms\2.Classification\ClassificationData.csv")

Index	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0
11	15606274	Female	26	52000	0
12	15746139	Male	20	86000	0

#lets divide them into dependent & independent

x=data.iloc[:,2:4] #age,salary

y=data.iloc[:,-1] #purchased



x - DataFrame

Index	Age	EstimatedSalary
0	19	19000
1	35	20000
2	26	43000
3	27	57000
4	19	76000
5	27	58000
6	27	84000
7	32	150000
8	25	33000
9	35	65000
10	26	80000
11	26	52000
12	20	86000

Format Resize ☑ Background color ☑

y - Series

Index	Purchased
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	0
10	0
11	0
12	0

Format Resize ☑ Background color ☑ Column min/max

#splitting data

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,

random_state=0)

x_test - DataFrame

Index	Age	EstimatedSalary
132	30	87000
309	38	50000
341	35	75000
196	30	79000
246	35	50000
60	27	20000
155	31	15000
261	36	144000
141	18	68000
214	47	43000
37	30	49000
134	28	55000
113	37	55000

x_train - DataFrame

Index	Age	EstimatedSalary
336	58	144000
64	59	83000
55	24	55000
106	26	35000
300	58	38000
229	42	80000
122	40	75000
373	59	130000
395	46	41000
325	41	60000
380	42	64000
253	37	146000
56	23	48000

y_test - Series

Index	Purchased
132	0
309	0
341	0
196	0
246	0
60	0
155	0
261	1
141	0
214	0
37	0
134	0
113	0

y_train - Series

Index	Purchased
336	1
64	0
55	0
106	0
300	1
229	1
122	0
373	1
395	1
325	0
380	0
253	1
56	0

#feature scaling

from sklearn.preprocessing import **StandardScaler** **#range between-> -3to3**

featurescaling=StandardScaler()

x_train=featurescaling.fit_transform(x_train)

x_test=featurescaling.transform(x_test)

x_test - NumPy object array

	0	1
0	-0.798951	0.494608
1	-0.0212649	-0.577359
2	-0.312897	0.146943
3	-0.798951	0.262831
4	-0.312897	-0.577359
5	-1.09058	-1.44652
6	-0.70174	-1.59138
7	-0.215686	2.14602
8	-1.96548	-0.0558618
9	0.853632	-0.780164
10	-0.798951	-0.606331
11	-0.993372	-0.432499
12	-0.118476	-0.432499

x_train - NumPy object array

	0	1
0	1.92295	2.14602
1	2.02016	0.378719
2	-1.38222	-0.432499
3	-1.18779	-1.01194
4	1.92295	-0.925024
5	0.367578	0.291803
6	0.173157	0.146943
7	2.02016	1.74041
8	0.756421	-0.838108
9	0.270367	-0.287638
10	0.367578	-0.17175
11	-0.118476	2.20396
12	-1.47943	-0.635303

#model building

BernoulliNB

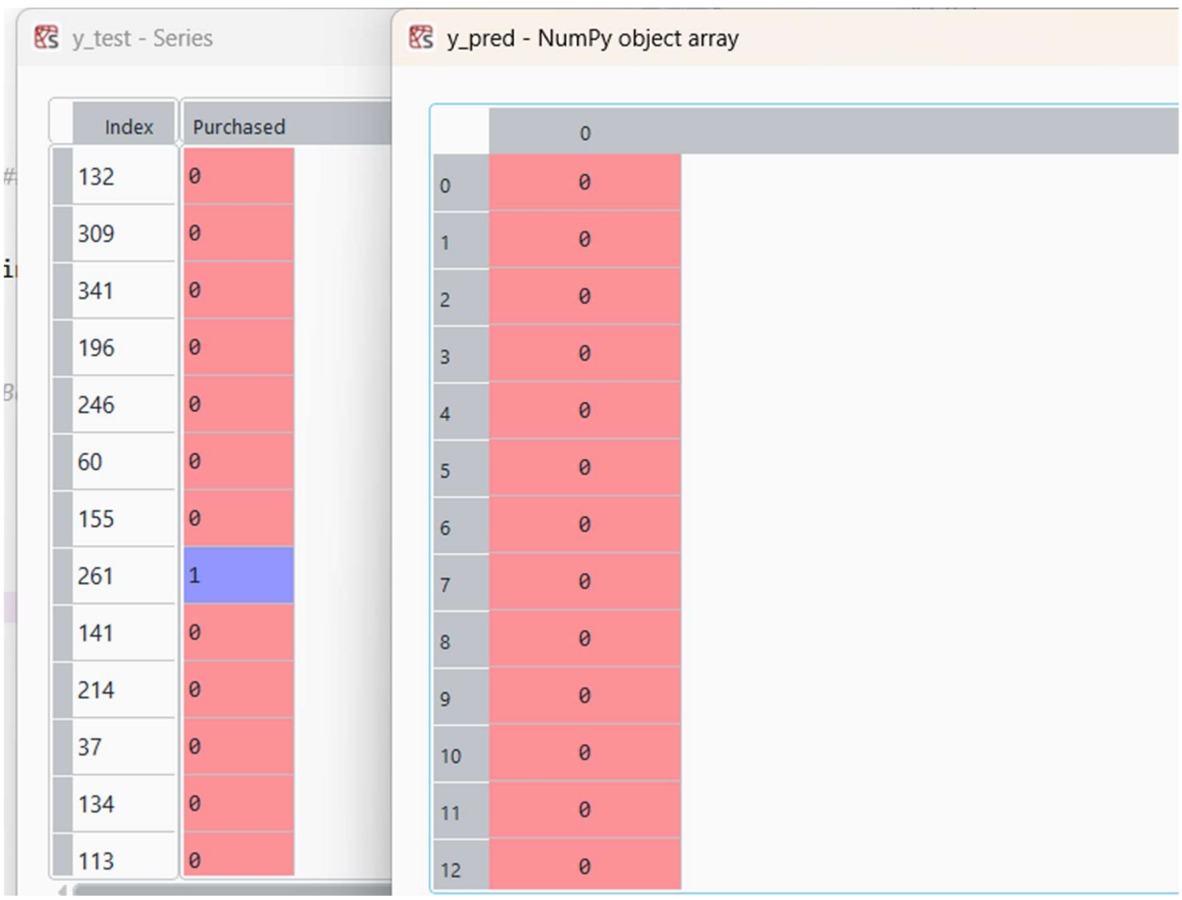
from sklearn.naive_bayes import BernoulliNB,MultinomialNB,GaussianNB

model=BernoulliNB()

model.fit(x_train,y_train)

#prediction

y_pred=model.predict(x_test)



y_test - Series

Index	Purchased
132	0
309	0
341	0
196	0
246	0
60	0
155	0
261	1
141	0
214	0
37	0
134	0
113	0

y_pred - NumPy object array

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0

#confusion Matrix

from sklearn.metrics import confusion_matrix

cm=confusion_matrix(y_test, y_pred)

cm - NumPy object array

	0	1
0	55	3
1	11	11

```
from sklearn.metrics import accuracy_score
```

ac=**accuracy_score**(y_test,y_pred) **->0.825**

```
from sklearn.metrics import classification_report
cr=classification_report(y_test,y_pred)
```

Text editor - cr

	precision	recall	f1-score	support
0	0.83	0.95	0.89	58
1	0.79	0.50	0.61	22
accuracy			0.82	80
macro avg	0.81	0.72	0.75	80
weighted avg	0.82	0.82	0.81	80

bias=model.score(x_train,y_train) **-> 0.709375**

variance=model.score(x_test,y_test)**-> 0.825**

GuassianNB

model=GaussianNB()

y_test - Series

Index	Purchased
132	0
309	0
341	0
196	0
246	0
60	0
155	0
261	1
141	0
214	0
37	0
134	0
113	0

y_pred - NumPy object array

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	1
10	0
11	0
12	0

cm - NumPy object array

	0	1
0	55	3
1	4	18

ac=**accuracy_score**(y_test,y_pred) -> **0.9125**

Text editor - cr

	precision	recall	f1-score	support
0	0.93	0.95	0.94	58
1	0.86	0.82	0.84	22
accuracy			0.91	80
macro avg	0.89	0.88	0.89	80
weighted avg	0.91	0.91	0.91	80

bias=model.score(x_train,y_train) **-> 0.884375**

variance=model.score(x_test,y_test)-**> 0.9125**

MultinomialNB

 # Fetaure Scaling

from sklearn.preprocessing import Normalizer

featurescaling=Normalizer()

x_train=featurescaling.fit_transform(x_train)

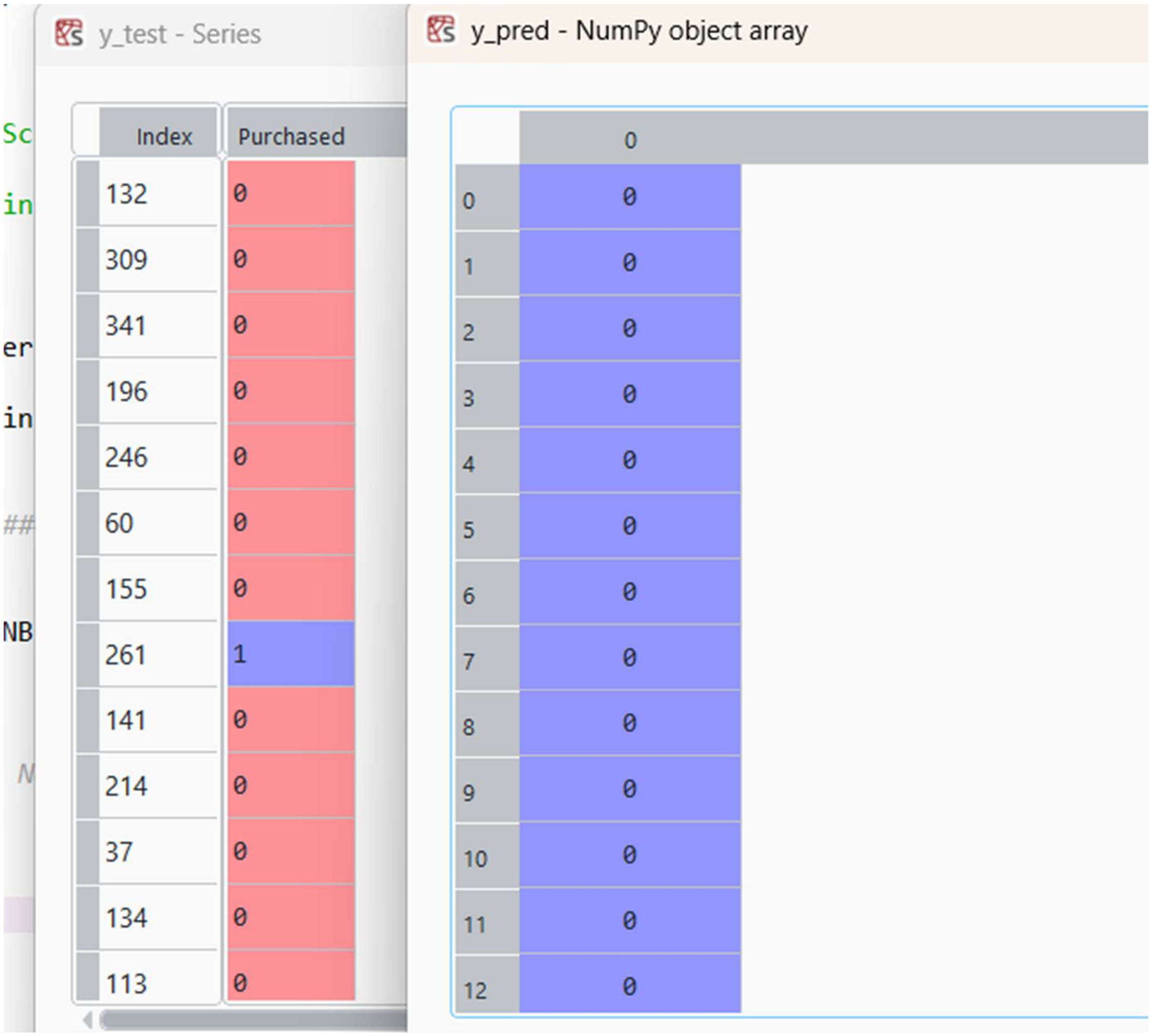
x_test=featurescaling.transform(x_test)

#we have to use Normalizer for Multionmial Navie-Bayes as Negative Values can't be passed to Multinomial where

 # Standard Scaler Ranges from -3

model=MultinomialNB()

model.fit(x_train,y_train)

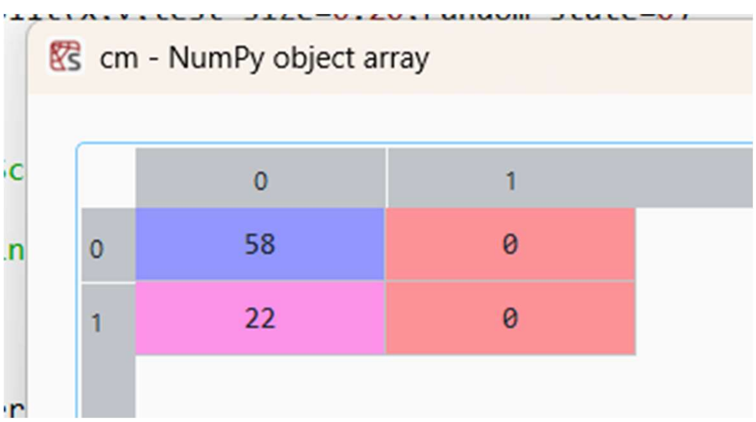


y_test - Series

Index	Purchased
132	0
309	0
341	0
196	0
246	0
60	0
155	0
261	1
141	0
214	0
37	0
134	0
113	0

y_pred - NumPy object array

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0



cm - NumPy object array

	0	1
0	58	0
1	22	0

ac=**accuracy_score**(y_test,y_pred) -> **0.725**

Text editor - cr

	precision	recall	f1-score	support
0	0.72	1.00	0.84	58
1	0.00	0.00	0.00	22
accuracy			0.72	80
macro avg	0.36	0.50	0.42	80
weighted avg	0.53	0.72	0.61	80

bias=model.score(x_train,y_train) **-> 0.621875**

variance=model.score(x_test,y_test)-**> 0.72**

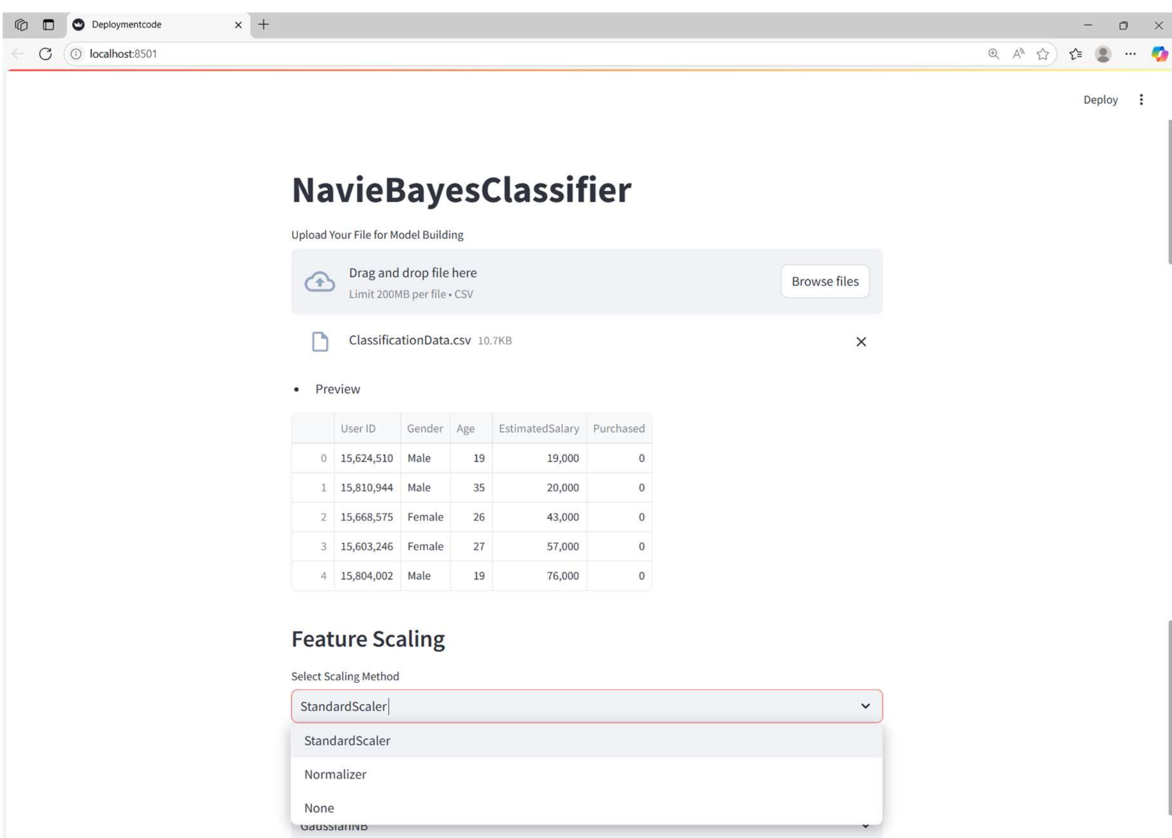
Deployment Code

```
# importing libraries

import streamlit as st

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler,Normalizer

from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB

from sklearn.metrics
import(confusion_matrix,accuracy_score,classification_report,roc_curve,ro
c_auc_score)

import seaborn as sns


st.title("NavieBayesClassifier")

# Uploading File

file=st.file_uploader('Upload Your File for Model Building',type=['csv'])

if file is not None:

   # Load

   data=pd.read_csv(file)

   st.write('- Preview')

   st.dataframe(data.head())


   # FeatureSelection

   x=data.iloc[:,2:4]
```

```
    y=data.iloc[:,-1]


    # SplittingData
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)


    # feature scaling
    st.subheader("Feature Scaling")
    scaling_method = st.selectbox("Select ScalingMethod",("StandardScaler", "Normalizer","None"))
    if scaling_method == "StandardScaler":
        featurescaling = StandardScaler()
    elif scaling_method == "Normalizer":
        featurescaling = Normalizer()
    else:
        featurescaling = None
    if featurescaling is not None:
        x_train=featurescaling.fit_transform(x_train)
        x_test=featurescaling.transform(x_test)


    # model building
    # model selection
    st.subheader("Select Model")
    model = st.selectbox("Choose a model", ("GaussianNB", "MultinomialNB", "BernoulliNB"))
    if model == "GaussianNB":
        model = GaussianNB()
    elif model == "MultinomialNB":
```

```
        model = MultinomialNB()
    else:
        model = BernoulliNB()
    model.fit(x_train,y_train)

    # prediction
    y_pred=model.predict(x_test)
    y_prob=model.predict_proba(x_test)[:,1]

    # Metrics
    st.subheader("Confusion Matrix")
    cm = confusion_matrix(y_test, y_pred)
    fig_cm, ax = plt.subplots()
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", ax=ax)
    st.pyplot(fig_cm)

    ac = accuracy_score(y_test, y_pred)
    st.write(f"**Accuracy:** {ac:.2f}")

    st.subheader("Classification Report")
    st.text(classification_report(y_test, y_pred))

    st.write(f"**Training Accuracy (Bias):** {model.score(x_train,
y_train):.2f}")
    st.write(f"**Testing Accuracy (Variance):** {model.score(x_test,
y_test):.2f}")
```

```
# ROC Curve and AUC
fpr, tpr, _ = roc_curve(y_test, y_prob)
auc_score = roc_auc_score(y_test, y_prob)
st.write(f"**AUC Score:** {auc_score:.2f}")

st.subheader("ROC Curve")
fig_roc, ax = plt.subplots()
ax.plot(fpr, tpr, color="blue", label=f"ROC curve (AUC = {auc_score:.2f})")
ax.plot([0, 1], [0, 1], color="gray", linestyle="--")
ax.set_xlabel("False Positive Rate")
ax.set_ylabel("True Positive Rate")
ax.set_title("ROC Curve")
ax.legend(loc="lower right")
st.pyplot(fig_roc)
```



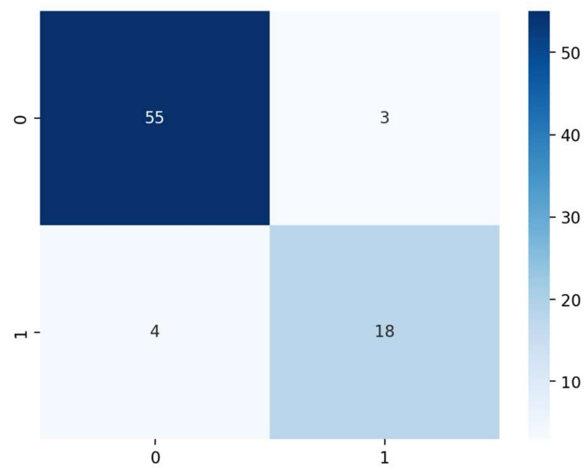
Select Model

Choose a model

GaussianNB

- GaussianNB
- MultinomialNB
- BernoulliNB

Confusion Matrix



Accuracy: 0.91

Classification Report

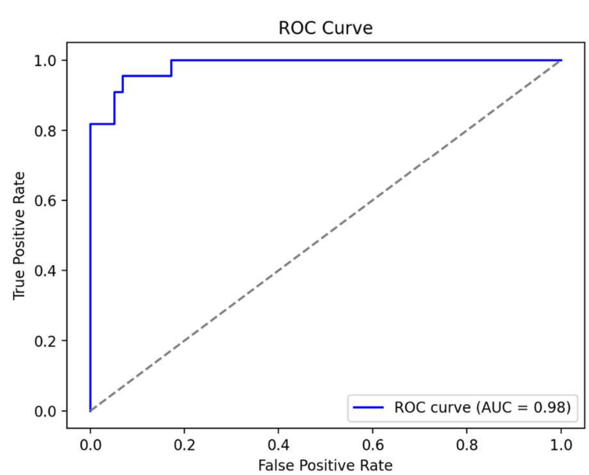
	precision	recall	f1-score	support
0	0.93	0.95	0.94	58
1	0.86	0.82	0.84	22
accuracy			0.91	80
macro avg	0.89	0.88	0.89	80
weighted avg	0.91	0.91	0.91	80

Training Accuracy (Bias): 0.88

Testing Accuracy (Variance): 0.91

AUC Score: 0.98

ROC Curve



Feature Scaling

Select Scaling Method

StandardScaler

Select Model

Choose a model

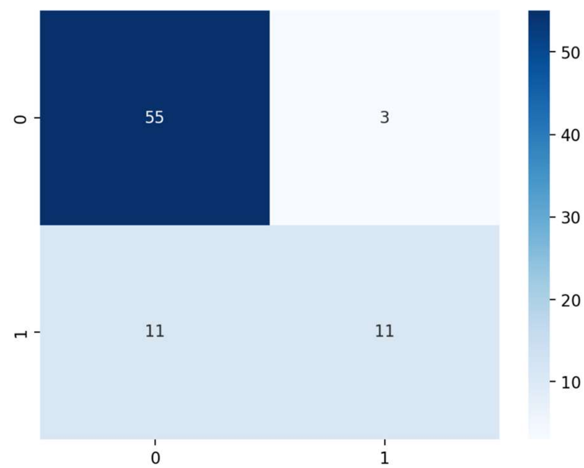
BernoulliNB

GaussianNB

MultinomialNB

BernoulliNB

Confusion Matrix



Accuracy: 0.82

Classification Report

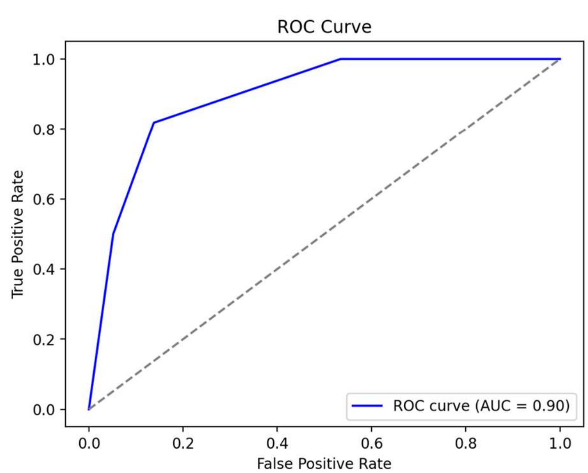
```
precision    recall  f1-score   support

           0       0.83      0.95      0.89        58
           1       0.79      0.50      0.61        22

    accuracy                           0.82        80
   macro avg       0.81      0.72      0.75        80
weighted avg       0.82      0.82      0.81        80
```

Training Accuracy (Bias): 0.71

Testing Accuracy (Variance): 0.82

AUC Score: 0.90

ROC Curve



Feature Scaling

Select Scaling Method

Normalizer

- StandardScaler
- Normalizer
- None

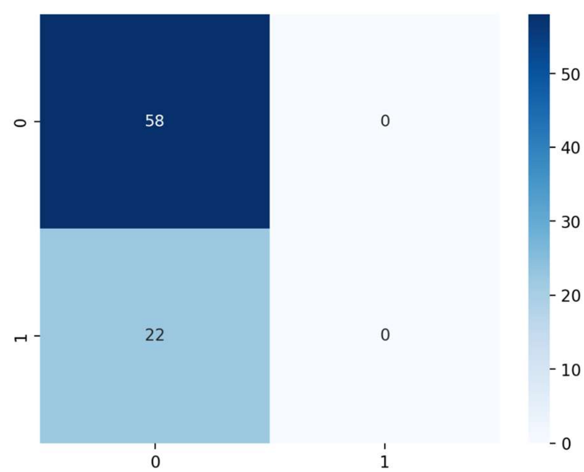
Select Model

Choose a model

MultinomialNB

- GaussianNB
- MultinomialNB
- BernoulliNB

Confusion Matrix



Accuracy: 0.72

Classification Report

```
precision    recall  f1-score   support

           0       0.72      1.00      0.84        58
           1       0.00      0.00      0.00        22

    accuracy                           0.72        80
   macro avg       0.36      0.50      0.42        80
weighted avg       0.53      0.72      0.61        80
```

Training Accuracy (Bias): 0.62

Testing Accuracy (Variance): 0.72

AUC Score: 0.57

ROC Curve