
SVM & Naïve Bayes – Assignment Answers

Question 1: What is a Support Vector Machine (SVM), and how does it work?

Answer:
Support Vector Machine (SVM) is a supervised machine learning algorithm primarily used for classification and regression tasks. It works by finding the optimal hyperplane that best separates data points of different classes. The closest data points to the hyperplane are called *support vectors*. SVM aims to maximize the margin — the distance between the hyperplane and the support vectors — to improve model generalization.

Question 2: Explain the difference between Hard Margin and Soft Margin SVM.

Answer:

- **Hard Margin SVM** assumes that the data is perfectly linearly separable. It does not tolerate any misclassification and tries to find a hyperplane that separates all points exactly.
- **Soft Margin SVM** allows some misclassifications using a penalty parameter C, making it suitable for real-world, noisy datasets. It balances between maximizing the margin and minimizing classification error.

Question 3: What is the Kernel Trick in SVM? Give one example of a kernel and explain its use case.

Answer:
The **Kernel Trick** allows SVM to perform in high-dimensional spaces without explicitly mapping the data. It computes the inner product in the transformed feature space using a kernel function.
Example: *Radial Basis Function (RBF) Kernel* — used when data is not linearly separable. It maps inputs into an infinite-dimensional space, useful in image classification and medical diagnostics.

Question 4: What is a Naïve Bayes Classifier, and why is it called "naïve"?

Answer:
Naïve Bayes is a probabilistic classifier based on Bayes' Theorem. It is "naïve" because it assumes all features are conditionally independent given the class label — an assumption rarely true in real data. Despite this, it performs surprisingly well in tasks like text classification and spam detection.

Question 5: Describe the Gaussian, Multinomial, and Bernoulli Naïve Bayes variants. When would you use each one?

Answer:

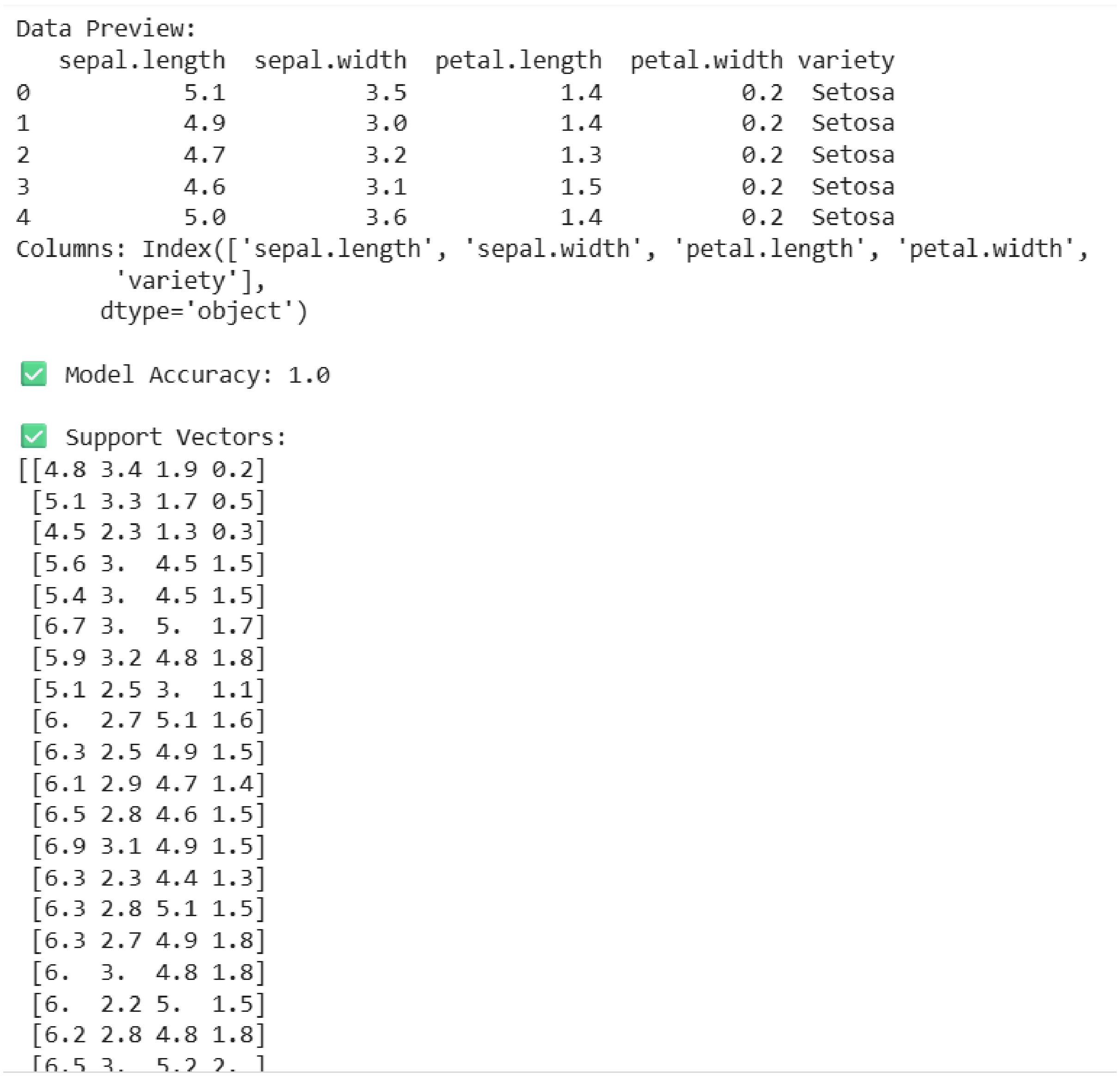
- **Gaussian Naïve Bayes:** For continuous data that follows a normal distribution (e.g., medical records).

- **Multinomial Naïve Bayes:** For count data like term frequency in text (e.g., spam detection).
- **Bernoulli Naïve Bayes:** For binary/boolean data (e.g., presence or absence of a word in a document).

Question 6: Load the Iris dataset, train an SVM with a linear kernel, and print accuracy & support vectors.

Answer:

Output:



```
Data Preview:
   sepal.length  sepal.width  petal.length  petal.width variety
0           5.1          3.5           1.4          0.2  Setosa
1           4.9          3.0           1.4          0.2  Setosa
2           4.7          3.2           1.3          0.2  Setosa
3           4.6          3.1           1.5          0.2  Setosa
4           5.0          3.6           1.4          0.2  Setosa
Columns: Index(['sepal.length', 'sepal.width', 'petal.length', 'petal.width',
       'variety'],
      dtype='object')

✅ Model Accuracy: 1.0

✅ Support Vectors:
[[4.8 3.4 1.9 0.2]
 [5.1 3.3 1.7 0.5]
 [4.5 2.3 1.3 0.3]
 [5.6 3.  4.5 1.5]
 [5.4 3.  4.5 1.5]
 [6.7 3.  5.  1.7]
 [5.9 3.2 4.8 1.8]
 [5.1 2.5 3.  1.1]
 [6.  2.7 5.1 1.6]
 [6.3 2.5 4.9 1.5]
 [6.1 2.9 4.7 1.4]
 [6.5 2.8 4.6 1.5]
 [6.9 3.1 4.9 1.5]
 [6.3 2.3 4.4 1.3]
 [6.3 2.8 5.1 1.5]
 [6.3 2.7 4.9 1.8]
 [6.  3.  4.8 1.8]
 [6.  2.2 5.  1.5]
 [6.2 2.8 4.8 1.8]
 [6.5 3.  5.2 2. ]
```

Code:

import pandas as pd

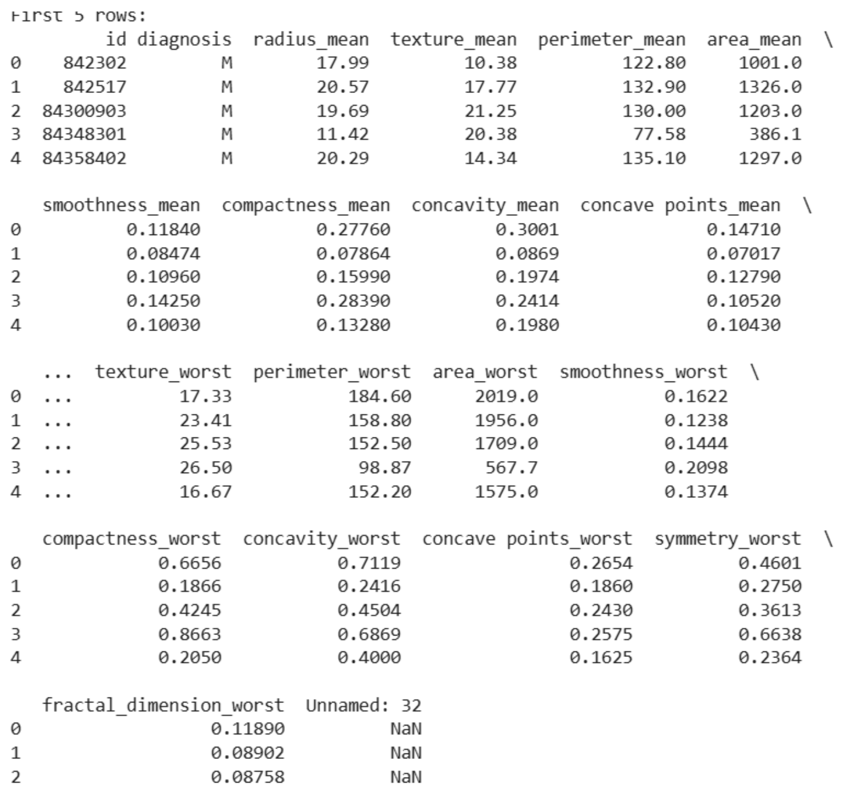
```
df = pd.read_csv("iris.csv.csv")

print(df.head())

print("Columns:", df.columns)

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.svm import SVC

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import accuracy_score

df = pd.read_csv("iris.csv.csv")

print("Data Preview:")

print(df.head())

print("Columns:", df.columns)

X = df.drop(columns=["variety"])  # Replace if your target column has a different name

y = df["variety"]

le = LabelEncoder()

y_encoded = le.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.3, random_state=42)

model = SVC(kernel='linear')

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print("\n Model Accuracy:", accuracy)

print("\n Support Vectors:")

print(model.support_vectors_)
```

Question 7: Train Gaussian Naïve Bayes on Breast Cancer dataset and print classification report.

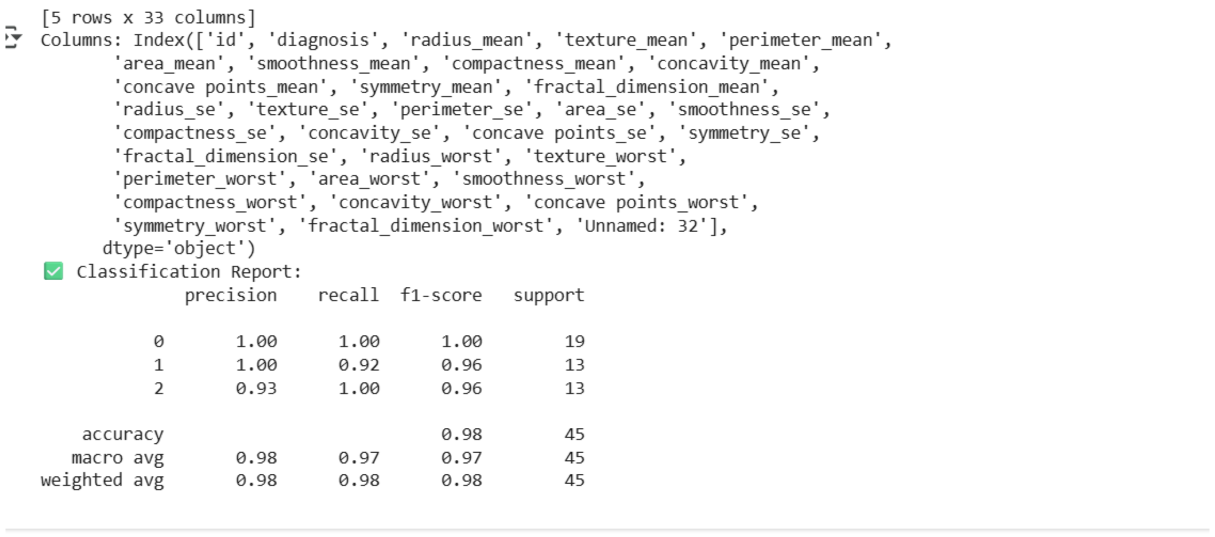
Answer:

Code:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report

from sklearn.preprocessing import LabelEncoder

df = pd.read_csv("data.csv")

print("First 5 rows:")

print(df.head())

print("Columns:", df.columns)

X = df.drop(columns=["diagnosis"])

y = df["diagnosis"]

if y.dtype == object:

   y = LabelEncoder().fit_transform(y)

model = GaussianNB()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print(" Classification Report:")

print(classification_report(y_test, y_pred))
```



```
First 5 rows:
         id diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0    842302         M        17.99         10.38          122.80     1001.0
1    842517         M        20.57         17.77          132.90     1326.0
2  84300903         M        19.69         21.25          130.00     1203.0
3  84348301         M        11.42         20.38           77.58      386.1
4  84358402         M        20.29         14.34          135.10     1297.0

   smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0          0.11840           0.27760          0.3001              0.14710
1          0.08474           0.07864          0.0869              0.07017
2          0.10960           0.15990          0.1974              0.12790
3          0.14250           0.28390          0.2414              0.10520
4          0.10030           0.13280          0.1980              0.10430

   ...  texture_worst  perimeter_worst  area_worst  smoothness_worst  \
0  ...          17.33           184.60      2019.0            0.1622
1  ...          23.41           158.80      1956.0            0.1238
2  ...          25.53           152.50      1709.0            0.1444
3  ...          26.50            98.87       567.7            0.2098
4  ...          16.67           152.20      1575.0            0.1374

   compactness_worst  concavity_worst  concave points_worst  symmetry_worst  \
0             0.6656           0.7119                0.2654          0.4601
1             0.1866           0.2416                0.1860          0.2750
2             0.4245           0.4504                0.2430          0.3613
3             0.8663           0.6869                0.2575          0.6638
4             0.2050           0.4000                0.1625          0.2364

   fractal_dimension_worst  Unnamed: 32
0                  0.11890          NaN
1                  0.08902          NaN
2                  0.08758          NaN
```



```
[5 rows x 33 columns]
Columns: Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
      dtype='object')
 ✅ Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      0.92      0.96        13
           2       0.93      1.00      0.96        13

    accuracy                           0.98        45
   macro avg       0.98      0.97      0.97        45
weighted avg       0.98      0.98      0.98        45
```

Question 8: Train SVM with GridSearchCV on Wine dataset and print best params & accuracy.

Answer:

```
import pandas as pd

from sklearn.model_selection import train_test_split, GridSearchCV

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score

from sklearn.preprocessing import LabelEncoder

df = pd.read_csv("wine.csv")  # Change if your file has a different name

print(df.head())

print("Columns:", df.columns)

X = df.drop(columns=["Wine"])  # Update column name if needed

y = df["Wine"]

if y.dtype == object:

   y = LabelEncoder().fit_transform(y)


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)


param_grid = {

   'C': [0.1, 1, 10],

   'gamma': ['scale', 0.1, 1]

}
```

grid = GridSearchCV(SVC(), param_grid, cv=3)

grid.fit(X_train, y_train)

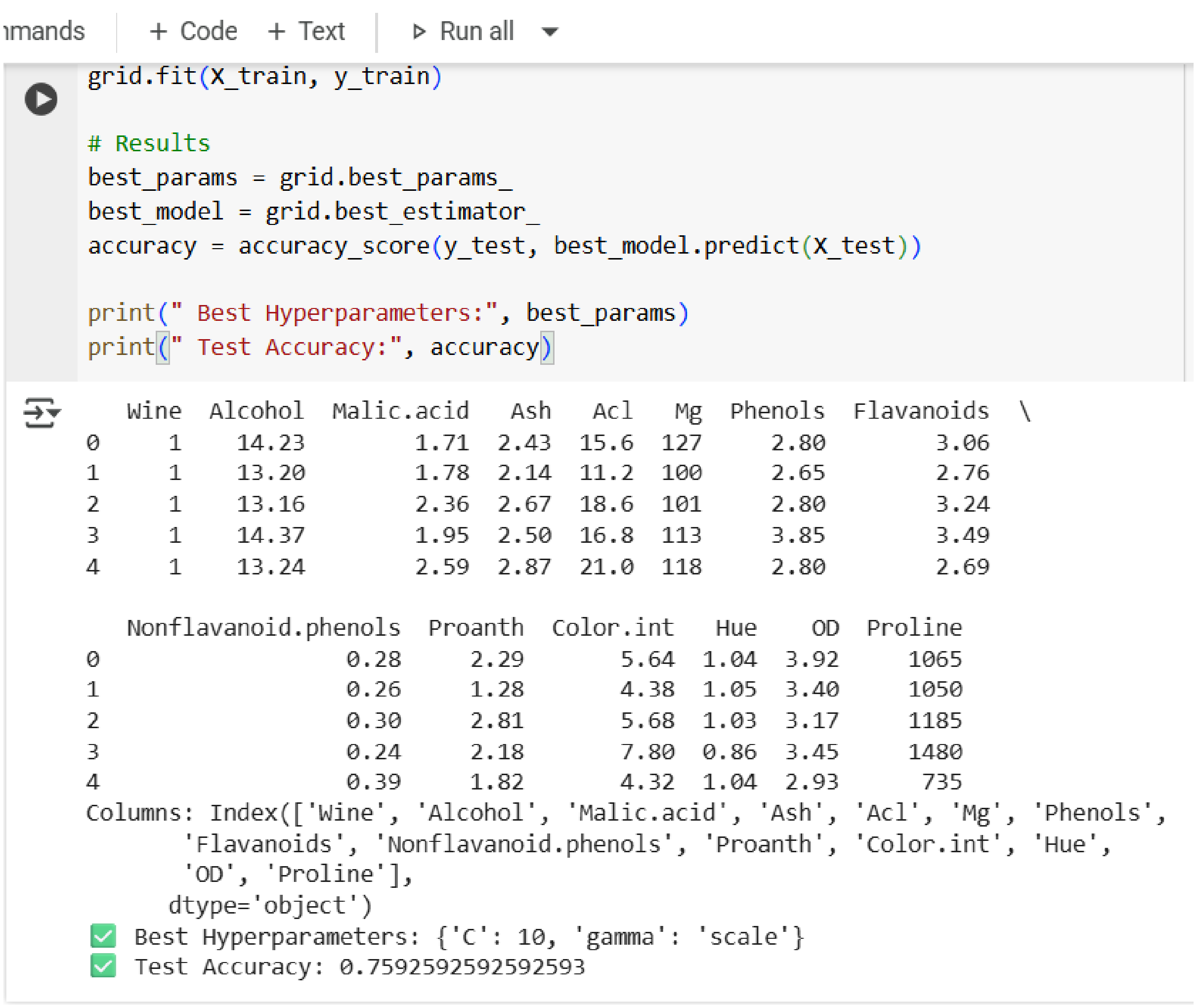
best_params = grid.best_params_

best_model = grid.best_estimator_

accuracy = accuracy_score(y_test, best_model.predict(X_test))

print(" Best Hyperparameters:", best_params)

print(" Test Accuracy:", accuracy)



```
grid.fit(X_train, y_train)

# Results
best_params = grid.best_params_
best_model = grid.best_estimator_
accuracy = accuracy_score(y_test, best_model.predict(X_test))

print(" Best Hyperparameters:", best_params)
print(" Test Accuracy:", accuracy)
```

```
   Wine  Alcohol  Malic.acid   Ash   Acl   Mg  Phenols  Flavanoids  \
0     1    14.23        1.71  2.43  15.6  127     2.80        3.06
1     1    13.20        1.78  2.14  11.2  100     2.65        2.76
2     1    13.16        2.36  2.67  18.6  101     2.80        3.24
3     1    14.37        1.95  2.50  16.8  113     3.85        3.49
4     1    13.24        2.59  2.87  21.0  118     2.80        2.69

   Nonflavanoid.phenols  Proanth  Color.int   Hue    OD  Proline
0                  0.28     2.29       5.64  1.04  3.92     1065
1                  0.26     1.28       4.38  1.05  3.40     1050
2                  0.30     2.81       5.68  1.03  3.17     1185
3                  0.24     2.18       7.80  0.86  3.45     1480
4                  0.39     1.82       4.32  1.04  2.93      735
Columns: Index(['Wine', 'Alcohol', 'Malic.acid', 'Ash', 'Acl', 'Mg', 'Phenols',
       'Flavanoids', 'Nonflavanoid.phenols', 'Proanth', 'Color.int', 'Hue',
       'OD', 'Proline'],
      dtype='object')
✅ Best Hyperparameters: {'C': 10, 'gamma': 'scale'}
✅ Test Accuracy: 0.7592592592592593
```

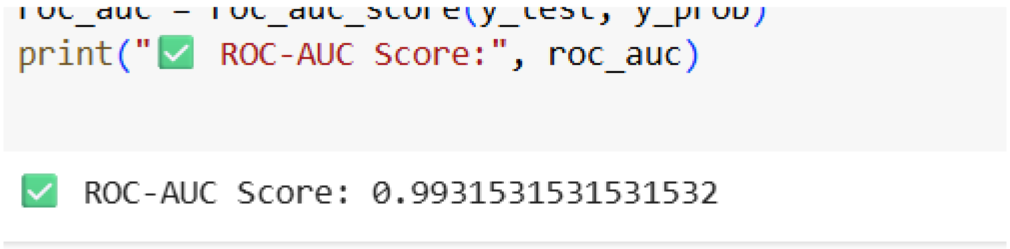
Question 9: Train Naïve Bayes on fetch_20newsgroups and print ROC-AUC score.

Answer:

Code:

```
from sklearn.datasets import fetch_20newsgroups

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.naive_bayes import MultinomialNB

from sklearn.model_selection import train_test_split

from sklearn.metrics import roc_auc_score

data = fetch_20newsgroups(subset='train', categories=['sci.space', 'rec.sport.hockey'])

X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, random_state=42)

vectorizer = TfidfVectorizer()

X_train_vec = vectorizer.fit_transform(X_train)

X_test_vec = vectorizer.transform(X_test)

model = MultinomialNB()

model.fit(X_train_vec, y_train)

y_prob = model.predict_proba(X_test_vec)[:, 1]

print("ROC-AUC Score:", roc_auc_score(y_test, y_prob))
```

output:



```
roc_auc = roc_auc_score(y_test, y_prob)
print("✅ ROC-AUC Score:", roc_auc)
```

```
✅ ROC-AUC Score: 0.9931531531531532
```

Question 10: Spam Classification Strategy – Email Dataset

Answer:

1. Preprocessing:

- Clean and tokenize text.
- Handle missing data with imputation or dropping.
- Convert text using TfidfVectorizer.

2. Model Choice:
Use **Naïve Bayes** for speed and scalability on text data. SVM is powerful but slower.

3. Handle Class Imbalance:

- Use SMOTE (oversampling minority).

- Assign class weights in the model.

4. Evaluation Metrics:

- Precision, Recall, F1-score
- ROC-AUC Score for overall performance

5. Business Impact:

- Automatically detects spam with high accuracy.
- Reduces security risks and enhances user trust.
- Saves employee time and increases productivity.
