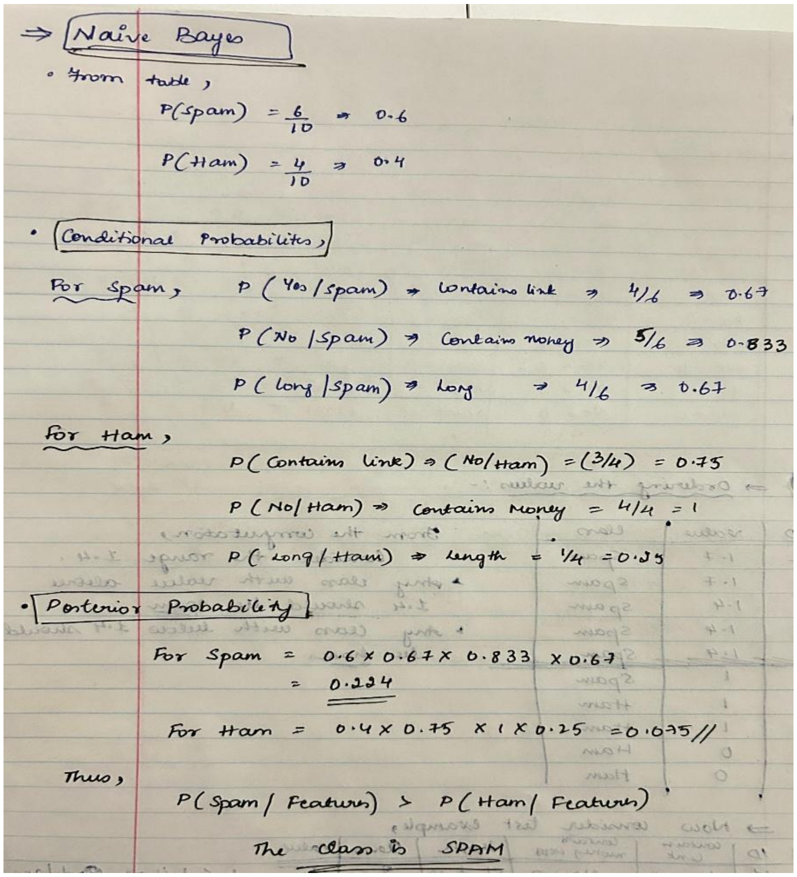
IN-CLASS ACTIVITIES/LAB (IS 733)
SUBMITTED BY: PRATHYUSHA HARISH KUMAR

Task 1: Understanding Naive Bayes and K-nearest neighbors

1a: Manually calculate prediction using the Naive Bayes Model and K nearest neighbor, K=2; Euclidean Distance for the test example for the following example:

- Use any random combination to test/report your probability

ID	Contains Link	Contains Money Words	Length	Class
1	Yes	Yes	Long	Spam
2	No	No	Short	Ham
3	Yes	No	Long	Spam
4	No	Yes	Short	Spam
5	Yes	Yes	Short	Spam
6	No	No	Long	Ham
7	Yes	No	Short	Ham
8	No	Yes	Long	Spam
9	Yes	Yes	Long	Spam
10	No	No	Short	Ham



⇒ Naive Bayes

- from table,

P(spam) = $\frac{6}{10}$ ⇒ 0.6

P(Ham) = $\frac{4}{10}$ ⇒ 0.4

- Conditional Probabilites,

For spam, P (Yes / spam) → Contains link ⇒ 4/6 ⇒ 0.67

P (No | spam) ⇒ Contains money ⇒ 5/6 ⇒ 0.833

P (long | spam) ⇒ long ⇒ 4/6 ⇒ 0.67

for Ham,

P(Contains link) ⇒ (No/Ham) = (3/4) = 0.75

P (No/ Ham) ⇒ Contains Money = 4/4 = 1

P (long / Ham) ⇒ length = 1/4 = 0.25

- Posterior Probability

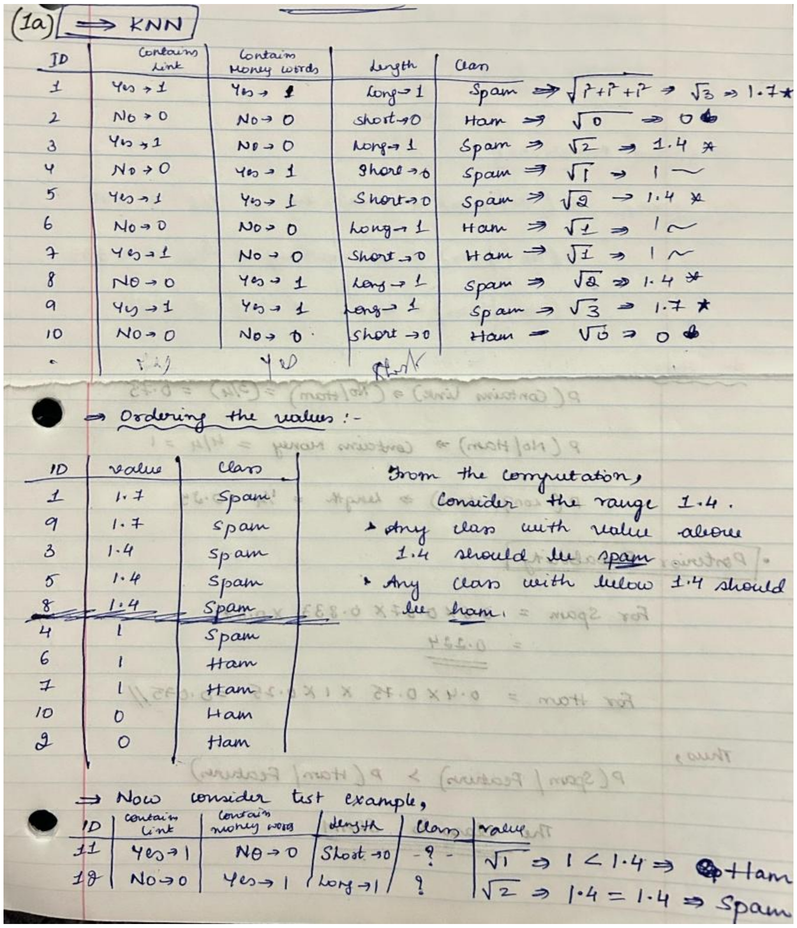
For Spam = 0.6 × 0.67 × 0.833 × 0.67
= 0.224

For Ham = 0.4 × 0.75 × 1 × 0.25 = 0.075//

Thus,

P(Spam / Features) > P(Ham / Features)

The class is SPAM



(1a) ⇒ KNN

ID	Contains Link	Contain Money words	Length	Class
1	Yes → 1	Yes → 1	Long → 1	Spam ⇒ $\sqrt{1^2+1^2+1^2}$ ⇒ $\sqrt{3}$ ⇒ 1.7★
2	No → 0	No → 0	short → 0	Ham ⇒ $\sqrt{0}$ ⇒ 0 ♣
3	Yes → 1	No → 0	Long → 1	Spam ⇒ $\sqrt{2}$ ⇒ 1.4 ✱
4	No → 0	Yes → 1	Short → 0	Spam ⇒ $\sqrt{1}$ ⇒ 1 ~
5	Yes → 1	Yes → 1	Short → 0	Spam ⇒ $\sqrt{2}$ → 1.4 ✱
6	No → 0	No → 0	Long → 1	Ham ⇒ $\sqrt{1}$ ⇒ 1 ~
7	Yes → 1	No → 0	Short → 0	Ham ⇒ $\sqrt{1}$ ⇒ 1 ~
8	No → 0	Yes → 1	Long → 1	spam ⇒ $\sqrt{2}$ ⇒ 1.4 ✱
9	Yes → 1	Yes → 1	Long → 1	Spam ⇒ $\sqrt{3}$ ⇒ 1.7 ★
10	No → 0	No → 0	short → 0	Ham ⇒ $\sqrt{0}$ ⇒ 0 ♣

⇒ Ordering the values :-

ID	value	Class
1	1.7	Spam
9	1.7	spam
3	1.4	spam
5	1.4	Spam
8	1.4	Spam
4	1	Spam
6	1	Ham
7	1	Ham
10	0	Ham
2	0	Ham

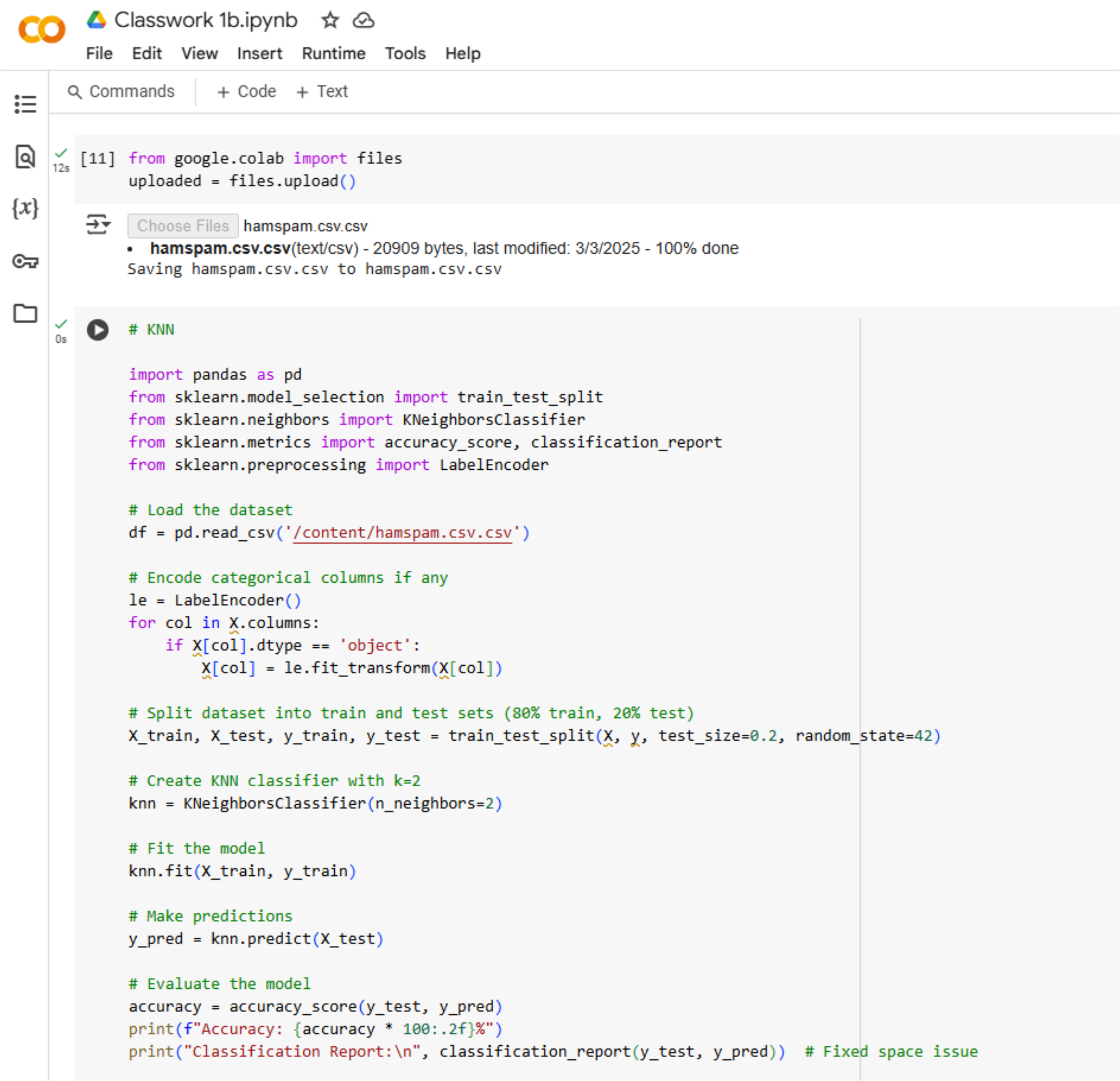
From the computation,
Consider the range 1.4.

- Any class with value above 1.4 should be spam
- Any class with below 1.4 should be ham.

⇒ Now consider test example,

ID	contain Link	contain money word	length	Class	value
11	Yes → 1	No → 0	Short → 0	- ? -	$\sqrt{1}$ ⇒ 1 < 1.4 ⇒ Ham
12	No → 0	Yes → 1	Long → 1	?	$\sqrt{2}$ ⇒ 1.4 = 1.4 ⇒ Spam

1b: write code (with AI assistant) to build a naive Bayes and KNN classifier. You can use the hamspam.csv to test it out.



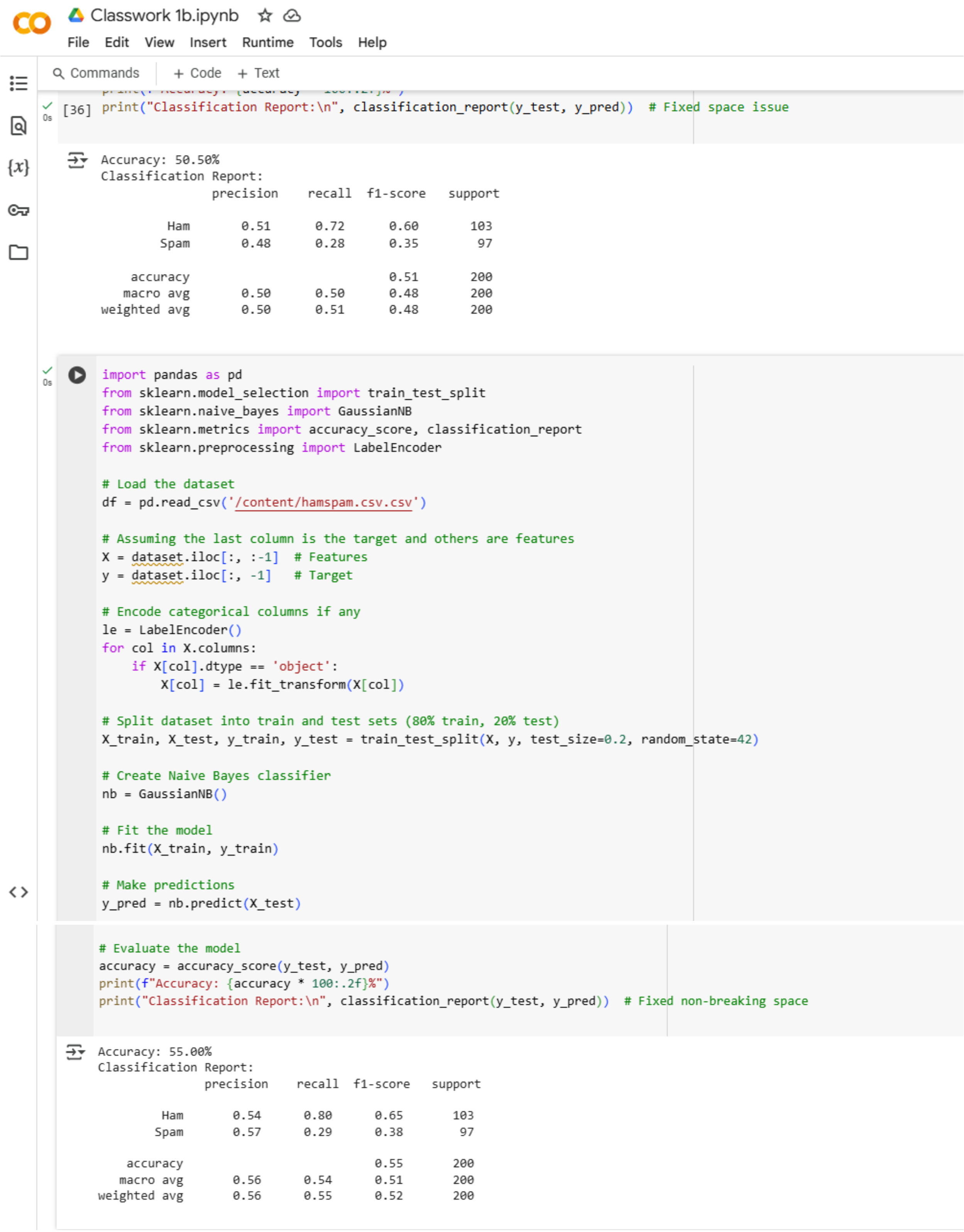
```
[11] from google.colab import files
     uploaded = files.upload()
```

Choose Files hamspam.csv.csv

- **hamspam.csv.csv**(text/csv) - 20909 bytes, last modified: 3/3/2025 - 100% done

```
Saving hamspam.csv.csv to hamspam.csv.csv
```

```
# KNN

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import LabelEncoder

# Load the dataset
df = pd.read_csv('/content/hamspam.csv.csv')

# Encode categorical columns if any
le = LabelEncoder()
for col in X.columns:
    if X[col].dtype == 'object':
        X[col] = le.fit_transform(X[col])

# Split dataset into train and test sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create KNN classifier with k=2
knn = KNeighborsClassifier(n_neighbors=2)

# Fit the model
knn.fit(X_train, y_train)

# Make predictions
y_pred = knn.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
print("Classification Report:\n", classification_report(y_test, y_pred))  # Fixed space issue
```



```
print(f"Accuracy: {accuracy * 100:.2f}%")
[36] print("Classification Report:\n", classification_report(y_test, y_pred))  # Fixed space issue
```

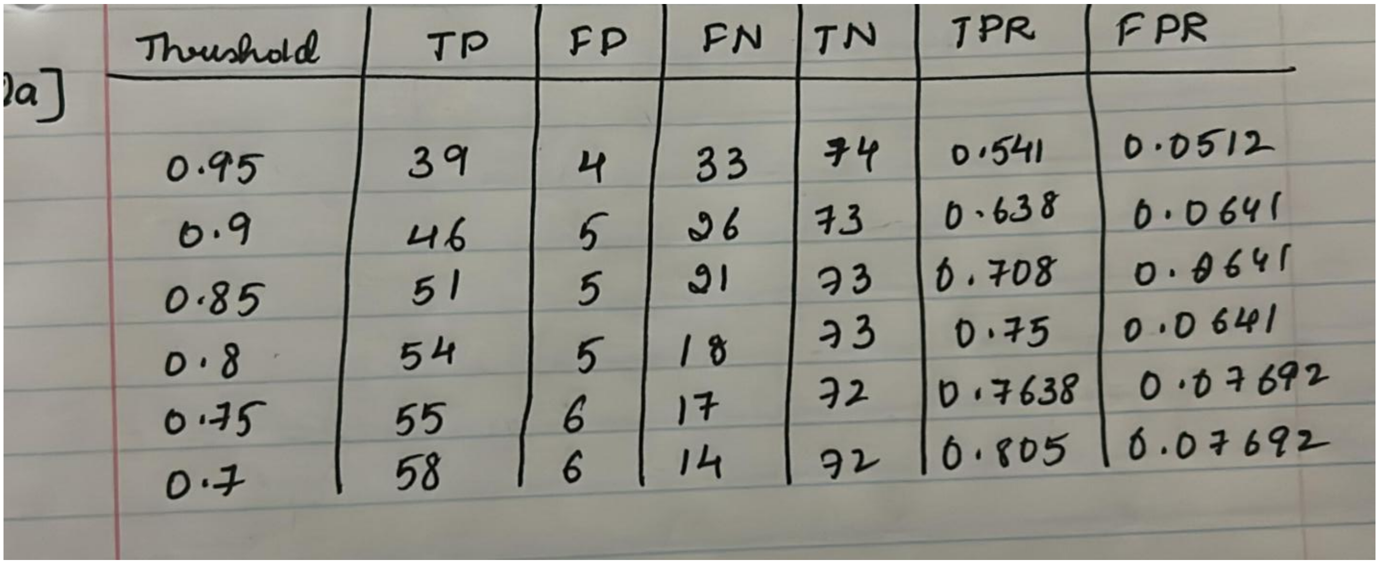
```
Accuracy: 50.50%
Classification Report:
               precision    recall  f1-score   support

         Ham       0.51      0.72      0.60       103
        Spam       0.48      0.28      0.35        97

    accuracy                           0.51       200
   macro avg       0.50      0.50      0.48       200
weighted avg       0.50      0.51      0.48       200
```

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import LabelEncoder

# Load the dataset
df = pd.read_csv('/content/hamspam.csv.csv')

# Assuming the last column is the target and others are features
X = dataset.iloc[:, :-1]  # Features
y = dataset.iloc[:, -1]   # Target

# Encode categorical columns if any
le = LabelEncoder()
for col in X.columns:
    if X[col].dtype == 'object':
        X[col] = le.fit_transform(X[col])

# Split dataset into train and test sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create Naive Bayes classifier
nb = GaussianNB()

# Fit the model
nb.fit(X_train, y_train)

# Make predictions
y_pred = nb.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
print("Classification Report:\n", classification_report(y_test, y_pred))  # Fixed non-breaking space
```

```
Accuracy: 55.00%
Classification Report:
               precision    recall  f1-score   support

         Ham       0.54      0.80      0.65       103
        Spam       0.57      0.29      0.38        97

    accuracy                           0.55       200
   macro avg       0.56      0.54      0.51       200
weighted avg       0.56      0.55      0.52       200
```

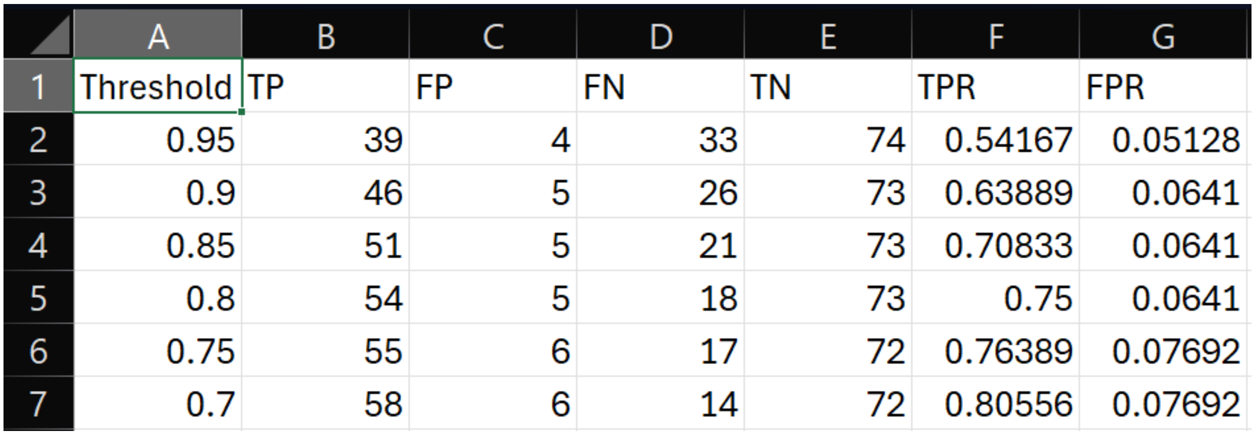
Task2: Understanding ROC and AUC

2a: Create a ROC (with AI assistant/Excel) **(Refer to** roc_data.csv**)**
Step1: Given the threshold (0.95,0.90,0.85,0.80,0.75,0.70), derive True Positive and False Positive
Step2: Calculate the True Positive Rate (TPR) and False Positive Rate (FPR), enter the values into the sheet
Step3: plot the set points (FRP, TPR) on the ROC diagram

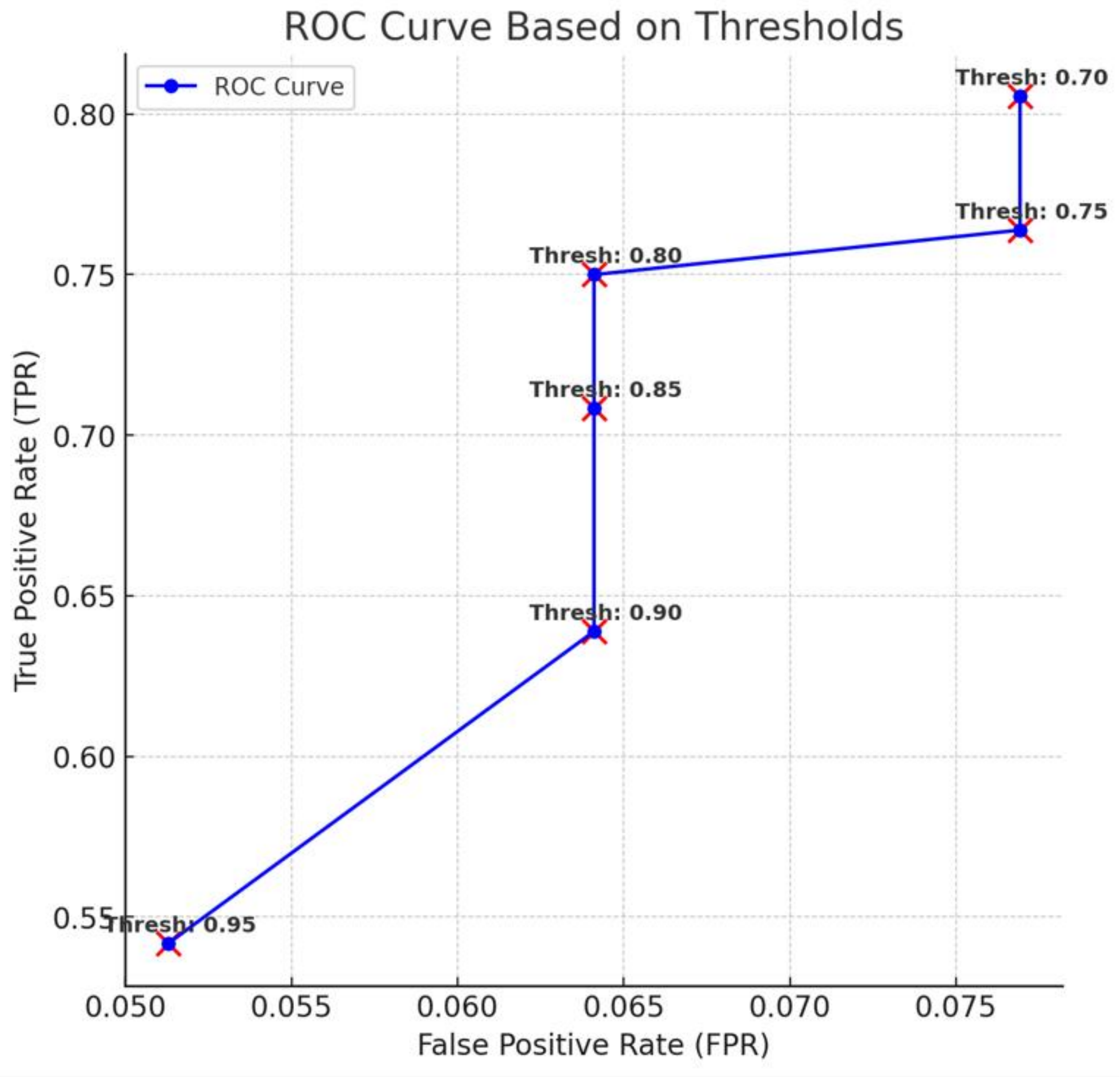


a]

Threshold	TP	FP	FN	TN	TPR	FPR
0.95	39	4	33	74	0.541	0.0512
0.9	46	5	26	73	0.638	0.0641
0.85	51	5	21	73	0.708	0.0641
0.8	54	5	18	73	0.75	0.0641
0.75	55	6	17	72	0.7638	0.07692
0.7	58	6	14	72	0.805	0.07692



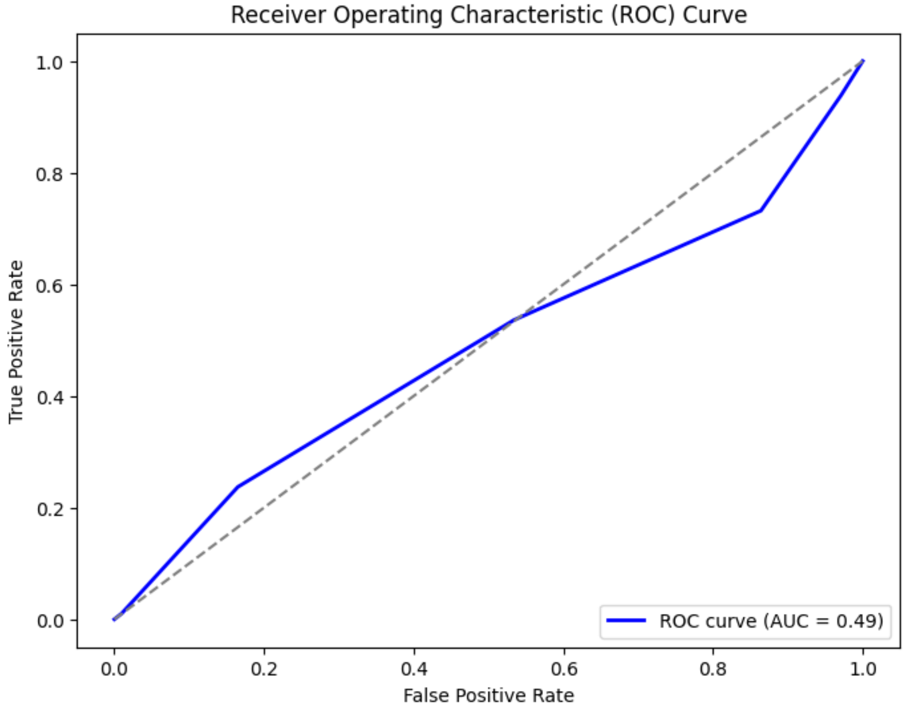
	A	B	C	D	E	F	G
1	Threshold	TP	FP	FN	TN	TPR	FPR
2	0.95	39	4	33	74	0.54167	0.05128
3	0.9	46	5	26	73	0.63889	0.0641
4	0.85	51	5	21	73	0.70833	0.0641
5	0.8	54	5	18	73	0.75	0.0641
6	0.75	55	6	17	72	0.76389	0.07692
7	0.7	58	6	14	72	0.80556	0.07692



2b. Write code (with AI assistant) to fit the model using your favorite classifier (NB, KNN, or Decision tree); using the hamspam.csv**,** ask to output an ROC curve and AUC score. (Hint: if you fit a decision tree, you might want to reduce max_depth)

```python
#2b

#KNN ROC
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import LabelEncoder

# Assuming the last column is the target and others are features
X = dataset.iloc[:, :-1]  # Features
y = dataset.iloc[:, -1]   # Target

# Encode categorical columns if any
le = LabelEncoder()
for col in X.columns:
    if X[col].dtype == 'object':
        X[col] = le.fit_transform(X[col])

# Encode target variable if it's categorical
y = le.fit_transform(y) if y.dtype == 'object' else y

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train KNN classifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# Predict probabilities
if hasattr(knn, "predict_proba"):
    y_scores = knn.predict_proba(X_test)[:, 1]
else:
    y_scores = knn.predict(X_test)  # Fallback if predict_proba is not available

# Compute ROC curve
fpr, tpr, _ = roc_curve(y_test, y_scores)
roc_auc = auc(fpr, tpr)
```

```python
# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()

# Print AUC Score
print(f'AUC Score: {roc_auc:.2f}')
```



```
AUC Score: 0.49
```