



Data Analysis 2

Task1

Dataset:

Spambase

Student Name	ID
Mawaddah Alharthi	443004953
Bayan Alqarni	443002589

Naive Bayes Classifier

1. Data Processing Steps

Loading the Data:

- The dataset was loaded using pandas.

```
import pandas as pd  
data = pd.read_csv('/content/spambase.data', header=None)
```

Splitting the Data:

- The dataset was split into features (X) and target (y). In this case, the target variable is assumed to be in the last column.

```
X = data.drop(columns=[57]) # Features  
y = data[57] # Target variable
```

Scaling Features:

- Used MinMaxScaler to scale the features between 0 and 1 for better performance of the Naive Bayes model.

```
from sklearn.preprocessing import MinMaxScaler  
scaler = MinMaxScaler()  
X_scaled = scaler.fit_transform(X)
```

2. Model Choice

Selected Model:

- I used the Multinomial Naive Bayes model for the classification task.

```
from sklearn.naive_bayes import MultinomialNB  
nb_classifier = MultinomialNB()
```

Reason for Choice:

- This model is particularly effective for classification problems, especially with text data or count features, which fits well for spam detection.

3. Performance Evaluation

Splitting the Dataset:

- The data was split into training and testing sets using `train_test_split()`.

```
from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

Training the Model:

- The model was trained on the training data.

```
nb_classifier.fit(X_train, y_train)
```

Making Predictions:

- Predictions were made on the test data.

```
y_pred = nb_classifier.predict(X_test)
```

Evaluating Performance:

- We calculated the accuracy and generated a classification report to evaluate the model's performance.

```
from sklearn.metrics import accuracy_score, classification_report  
accuracy = accuracy_score(y_test, y_pred)  
classification_report_output = classification_report(y_test, y_pred)  
print(f"Accuracy: {accuracy * 100:.2f}%")  
print("Classification Report:")  
print(classification_report_output)
```

4. Confusion Matrix Visualization

- A confusion matrix was generated to visualize the model's performance.

```
from sklearn.metrics import confusion_matrix  
import matplotlib.pyplot as plt  
import seaborn as sns  
conf_matrix = confusion_matrix(y_test, y_pred)  
plt.figure(figsize=(8, 6))  
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False, xticklabels=['Not Spam', 'Spam'], yticklabels=['Not Spam', 'Spam'])  
plt.title(f"Confusion Matrix (Accuracy: {accuracy * 100:.2f}%")  
plt.xlabel('Predicted Labels')  
plt.ylabel('True Labels')  
plt.show()
```

5. Conclusion and Future Work

- The model demonstrated a solid performance with an accuracy of approximately 87.19%.

To improve the results, I could:

- Experiment with different feature scaling techniques.
- Tune the model's hyperparameters.
- Gather more data or explore advanced models like Support Vector Machines (SVM).