

Experiment 5

Naive Bayes Classification

Aim:

To implement the Naïve Bayes algorithm for predicting class labels and to evaluate its performance

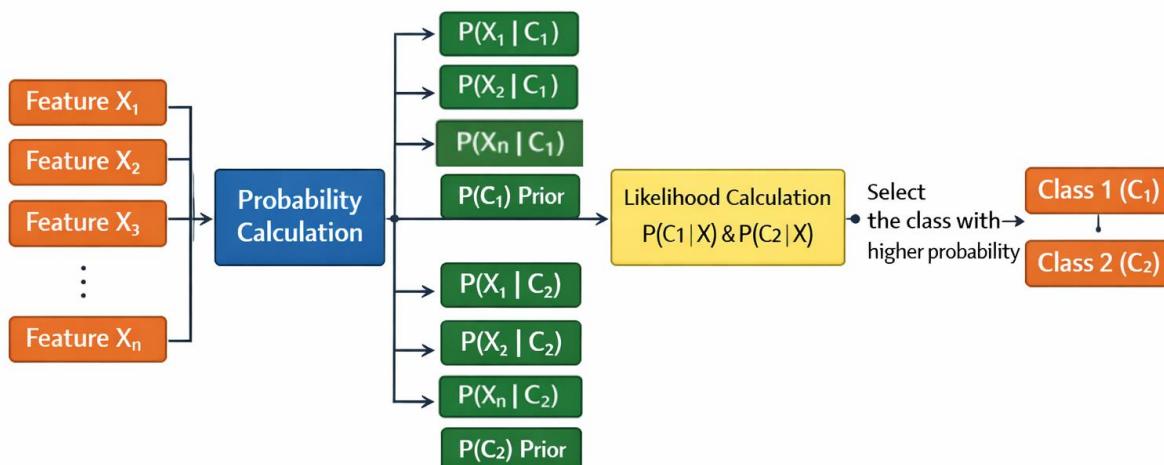
Theory:

Naive Bayes is a supervised probabilistic learning algorithm widely used for text classification tasks such as spam and ham message detection. The model is based on Bayes' theorem, which estimates the posterior probability of a class given observed features. A key assumption of Naive Bayes is that features are conditionally independent given the class label. While this assumption does not fully hold for natural language, the classifier remains effective due to the high dimensionality and sparse distribution of textual features.

Bayes' theorem is defined as:

$$P(C | X) = \frac{P(X | C) P(C)}{P(X)}$$

where $P(C | X)$ represents the probability of a message belonging to class C (spam or ham) given the feature set X , $P(X | C)$ is the likelihood of observing the features under class C , $P(X)$ is the marginal likelihood of the features, and $P(C)$ denotes the prior probability of the class. The feature-wise conditional probabilities and class priors are computed to select the class with the highest posterior probability is shown in the figure below:



In text classification, messages are modelled using the bag-of-words representation, where word order is ignored and word occurrences form the basis of feature extraction. To improve feature quality and reduce redundancy, Natural Language Processing (NLP) techniques are applied. These include text normalization, removal of stop words, and reduction of words to their base forms using stemming and lemmatization. Part-of-Speech (POS) tagging supports lemmatization by enabling accurate identification of grammatical roles such as nouns and verbs, thereby preserving semantic correctness while reducing dimensionality.

Textual data is transformed into numerical vectors using Term Frequency–Inverse Document Frequency (TF-IDF) weighting. TF-IDF enhances the importance of words that are frequent within a document but infrequent across the entire corpus, thus improving the discriminative ability of the feature space.

For classification, the Multinomial Naïve Bayes algorithm is employed, as it is well suited for word-based features and TF-IDF representations. Model effectiveness is assessed using standard metrics such as accuracy, precision, recall, confusion matrix, and classification report.

Merits of Naïve Bayes:

- Naïve Bayes is suitable when the number of features is very large, such as in text classification problems with thousands of words.
- Naïve Bayes is suitable when fast training and prediction are required because of its low computational complexity.
- Even when only a small amount of training data is available, Naïve Bayes can still give acceptable results.
- When the dataset contains unnecessary or extra features, Naïve Bayes can handle them without much loss in performance.

Demerits of Naïve Bayes:

- When features are highly correlated or dependent, Naïve Bayes should not be used because the independence assumption is violated.
- When the problem is complex and non-linear, Naïve Bayes is not suitable since it cannot model complex patterns.

Overall, the Naïve Bayes classifier, combined with NLP-based feature normalization and TF-IDF representation, provides an efficient and robust solution for spam and ham message classification.

Code and Results Section:

Block 1: Importing Required Libraries

Input

```
import pandas as pd  
  
import numpy as np  
  
import re  
  
import matplotlib.pyplot as plt  
  
import nltk  
  
from nltk.stem import PorterStemmer  
  
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS  
  
from sklearn.model_selection import train_test_split  
  
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
from sklearn.naive_bayes import MultinomialNB  
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix,  
precision_recall_curve  
print("Libraries Imported")
```

Output

Libraries Imported

Block 2: Reading the Dataset

Input

```
data = pd.read_csv('/content/Spam_Detection.csv', encoding='latin1', header=0, usecols=[0, 1],  
engine="python")  
print("Dataset reading complete")
```

Output

Dataset reading complete

Block 3: Dataset Preview (First 5 rows)

Input

```
data.head()
```

Output

	Category	Message
0	spam	Get your garden ready for summer with a free selection of summer bulbs and seeds worth £33.50, available only with The Scotsman this Saturday. To stop messages, visit notxt.co.uk.
1	spam	This is your chance to be on a reality fantasy show. Call now at 08707509020. The cost is 20p per minute. NTT Ltd, PO Box 1327, Croydon CR9 5WB. 0870 is a national-rate call.
2	ham	Love is not a decision; it is a feeling. If we could choose whom to love, life would be much simpler, but it would also be less magical.
3	ham	Good friends care for each other, close friends understand each other, and true friends remain forever—beyond words and beyond time. Good night.
4	spam	Get your garden ready for summer with a free selection of summer bulbs and seeds worth £33.50, available only with The Scotsman this Saturday. To stop messages, visit notxt.co.uk.

Block 4: Dataset Preview (Last 5 rows)

Input

```
data.tail()
```

Output

	Category	Message
1737	spam	Do you want 750 anytime any network mins 150 text and a NEW VIDEO phone for only five pounds per week call 08002888812 or reply for delivery tomorrow
1738	ham	Thanx...
1739	spam	88800 and 89034 are premium phone services call 08718711108
1740	spam	Today's Offer! Claim ur £150 worth of discount vouchers! Text YES to 85023 now! SavaMob, member offers mobile! T Cs 08717898035. £3.00 Sub. 16 . Unsub reply X
1741	ham	am up to my eyes in philosophy

Block 5: Dataset Information

Input

```
data.info()
```

Output

Range Index: 1742 entries, 0 to 1741

Data columns (total 2 columns):

	Column	Non-Null Count	Dtype
0	Category	1742 non-null	object
1	Message	1742 non-null	object

dtypes: object(2)

memory usage: 27.3+ KB

Block 6: Missing Values

Input

```
data.isnull().sum()
```

Output

Category	0
Message	0

dtype: int64

Block 7: Size of the dataset

Input

```
data.shape
```

Output

```
(1742,2)
```

Block 8: Value count of each category**Input**

```
data['Category'].value_counts()
```

Output

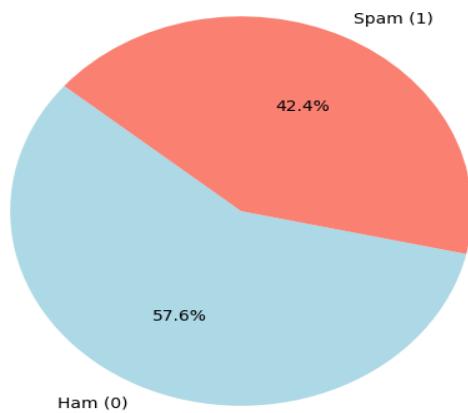
Category	Count
Ham	1003
Spam	739

Block 9: Visualizes the distribution of spam and ham messages in the dataset**Input**

```
category_counts = data['Category'].value_counts()  
  
plt.figure(figsize=(6, 6))  
  
plt.pie(category_counts, labels=['Ham (0)', 'Spam (1)'], autopct='%1.1f%%', startangle=140,  
colors=['lightblue', 'salmon'])  
  
plt.title('Distribution of Ham and Spam in Category')  
  
plt.show()
```

Output

Distribution of Ham and Spam in Category



Block 10: Text preprocessing

Input

```
stemmer = PorterStemmer()  
stop_words = set(ENGLISH_STOP_WORDS)  
print("Initialized Porter Stemmer and English stopword list.")
```

Output

Initialized Porter Stemmer and English stopword list.

Block 11: Text cleaning and Normalization

Input

```
def preprocess_text(s):  
    if pd.isnull(s):  
        return ""  
    s = s.lower()  
    s = re.sub(r'http\S+|www\.\S+', '', s)  
    s = re.sub(r'[^a-zA-Z\s]', ' ', s)  
    s = re.sub(r'\s+', ' ', s).strip()  
    return s  
  
print("Text Cleaned")
```

Output

Text Cleaned

Block 12: Splits the dataset

Input

```
data['label'] = (data['Category'].str.lower().str.strip() == 'spam').astype(int)

X_train, X_test, y_train, y_test = train_test_split( data['Message'], data['label'],
                                                    test_size=0.25, random_state=42, stratify=data['label'] )

print("Labels encoded and data split into training and testing sets.")
```

Output

Labels encoded and data split into training and testing sets.

Block 13: Converts text messages into TF-IDF feature vectors

Input

```
vectorizer = TfidfVectorizer(ngram_range=(1,3), min_df=2, max_features=50000)

X_train_vec = vectorizer.fit_transform(X_train)

X_test_vec = vectorizer.transform(X_test)

print("Text data vectorized using TF-IDF.")
```

Output

Text data vectorized using TF-IDF.

Block 14: Multinomial Naive Bayes classifier model trains

Input

```
model=MultinomialNB()

model.fit(X_train_vec,y_train)
```

Output

MultinomialNB

MultinomialNB()

Block 15: Model Evaluation

Input

```
y_pred = model.predict(X_test_vec)

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

print("Accuracy:", accuracy_score(y_test, y_pred))

print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

Output

Accuracy: 0.9288990825688074

Classification Report:

	precision	recall	f1-score	support
0(ham)	0.90	0.99	0.94	251
1(spam)	0.99	0.84	0.91	185
accuracy			0.93	436
macro avg	0.94	0.92	0.93	436
weighted avg	0.93	0.93	0.93	436

Block 16: Displays confusion matrix**Input**

```
cm = confusion_matrix(y_test, y_pred)

plt.matshow(cm, cmap='Blues')

for (i, j), val in np.ndenumerate(cm):

    plt.text(j, i, val, ha='center', va='center',
             color='white' if val > cm.max()/2 else 'black', fontsize=12)

plt.xticks([0,1], ['Pred Ham','Pred Spam'])

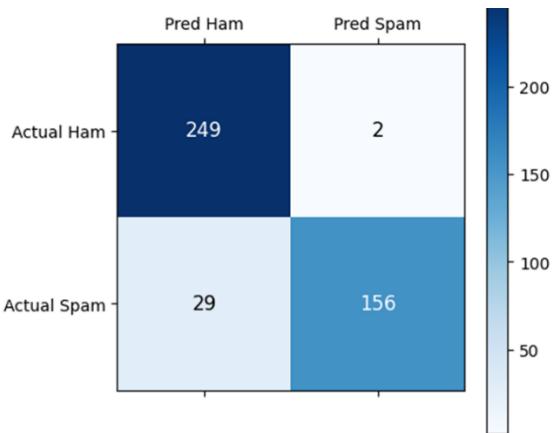
plt.yticks([0,1], ['Actual Ham','Actual Spam'])

plt.title("Confusion Matrix", fontsize=14, fontweight='bold')

plt.colorbar()

plt.show()
```

Output



Block 17: Displays the Precision-Recall curve for the model

Input

```
precision, recall, _ = precision_recall_curve(y_test, y_prob)

plt.figure(figsize=(6,4))

plt.plot(recall, precision)

plt.title('Precision-Recall Curve')

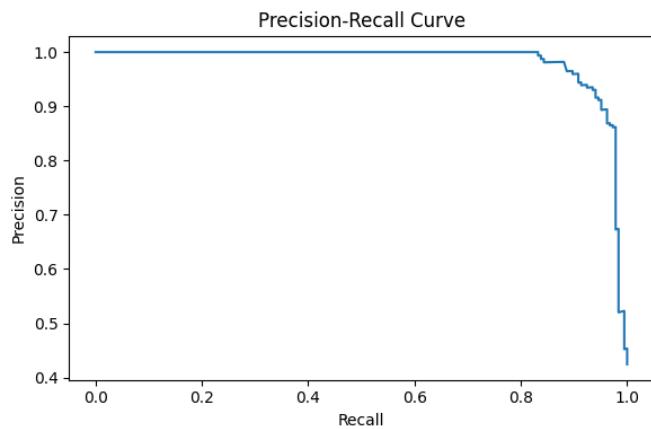
plt.xlabel('Recall')

plt.ylabel('Precision')

plt.tight_layout()

plt.show()
```

Output



Block 18: Highlights words using color intensity

Input

```

from IPython.display import HTML, display

def color_text(words, contributions): max_val = max(abs(c) for c in contributions) or 1

html = """

    for w, c in zip(words, contributions): intensity = int(255 * (abs(c) / max_val))

        if c > 0: color = f"rgb(255,{255-intensity},{255-intensity})"

        else: color = f"rgb({255-intensity},255,{255-intensity})"

    html += f"<span style='background-color:{color}; padding:3px; margin:2px; border-radius:4px;'>{w}</span>"

return html

print("Text contribution visualization function defined.")

```

Output

Text contribution visualization function defined.

Block 19: Computes precision and recall of the model on test data

Input

```

def show_precision_recall(model, X_test_vec, y_test):

    preds = model.predict(X_test_vec)

    precision = precision_score(y_test, preds)

    recall = recall_score(y_test, preds)

print("Precision and recall calculation function ready.")

```

Output

Precision and recall calculation function ready.

Block 20: Function for 'n' random test messages

Input

```

def analyze_multiple(model, vectorizer, X_test, y_test, n=5):

    show_precision_recall(model, X_test_vec, y_test)

    for i in range(n):

        analyze_random_message(model, vectorizer, X_test, y_test)

print("Multiple message analysis function defined.")

```

Output

Multiple message analysis function defined.

Block 21: Evaluation on test messages

Input

```
analyze_multiple(model, vectorizer, X_test, y_test, n=5)  
print(" MODEL PERFORMANCE ON TEST DATA");  
print("Precision:")  
print("Recall:")  
print(" Choose one random test sample.")
```

Output

MODEL PERFORMANCE ON TEST DATA

Precision: 0.9873

Recall: 0.8432

Choose one random test sample.

Category	Message
ham	It's getting me down just waiting around.
spam	Text and meet someone today. Reply with your name and age.
ham	I.ll give her once i have it. Plus she said grinule greet you whenever we speak
spam	Congratulations U can claim 2 VIP row A Tickets 2 C Blu in concert in November or Blu gift guaranteed Call 09061104276 to claim TS&Cs www.smsco.net cost£3.75max
ham	thinking about you.

Message Analysis

it's getting me down just waiting around.

Actual: ham

Predicted: ham

Spam Probability: 0.0408