Lab 11.3 – Text Classification using Naive Bayes

2403A52006

MD.Mustafa

Import Required Libraries

```
import pandas as pd
import numpy as np
import string
import re
import nltk
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report, confusion
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

Load and Explore Dataset

```
data = pd.read_csv("news.csv")
data.head()
```

	Unnamed: 0	title	text	label
0	8476	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	FAKE
1	10294	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	FAKE
2	3608	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	REAL
3	10142	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	FAKE
4	875	The Battle of New York: Why This Primary Matters	It's primary day in New York and front-runners...	REAL

Next steps: Generate code with data New interactive sheet

```
print("Dataset Shape:", data.shape)
print("\nColumn Names:", data.columns)
print("\nClass Distribution:\n")
print(data['label'].value_counts())
```

```
Dataset Shape: (6335, 4)

Column Names: Index(['Unnamed: 0', 'title', 'text', 'label'], dtype='object')

Class Distribution:

label
REAL    3171
FAKE    3164
Name: count, dtype: int64
```

Text Preprocessing

```
stop_words = set(stopwords.words('english'))

def clean_text(text):
    text = str(text).lower()
    text = re.sub(r'\d+', '', text)
    text = text.translate(str.maketrans('', '', string.punctuation))
    words = text.split()
    words = [word for word in words if word not in stop_words]
    return " ".join(words)

data['clean_text'] = data['text'].apply(clean_text)

data[['text', 'clean_text']].head()
```

	text	clean_text
0	Daniel Greenfield, a Shillman Journalism Fello...	daniel greenfield shillman journalism fellow f...
1	Google Pinterest Digg Linkedin Reddit Stumbleu...	google pinterest digg linkedin reddit stumbleu...
2	U.S. Secretary of State John F. Kerry said Mon...	us secretary state john f kerry said monday st...
3	— Kaydee King (@KaydeeKing) November 9, 2016 T...	— kaydee king kaydeeking november lesson tonig...
4	It's primary day in New York and front-runners...	primary day new york frontrunners hillary clin...

Feature Extraction (TF-IDF)

```
vectorizer = TfidfVectorizer()

X = vectorizer.fit_transform(data['clean_text'])
y = data['label']
print("Feature Matrix Shape:", X.shape)
print("\nSample Feature Names:\n", vectorizer.get_feature_names_out()[:20])
```

```
Feature Matrix Shape: (6335, 79162)

Sample Feature Names:
 ['aa' 'aaa' 'aaaa' 'aaaaadd' 'aaaasetlayoutnew' 'aaas' 'aab' 'aachen'
 'aadhar' 'aadmi' 'aae' 'aahing' 'aaib' 'aaj' 'aakar' 'aakhri' 'aalia'
 'aaliya' 'aall' 'aam']
```

Train-Test Split

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

print("Training Samples:", X_train.shape[0])
print("Testing Samples:", X_test.shape[0])
```

```
Training Samples: 5068
Testing Samples: 1267
```

Train Naive Bayes Model

```
model = MultinomialNB()

model.fit(X_train, y_train)

print("Model Training Completed")
```

```
Model Training Completed
```

Model Evaluation

```
y_pred = model.predict(X_test)

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

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

print("\nConfusion Matrix:\n")
```

```
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.8342541436464088

Classification Report:

              precision    recall  f1-score   support

        FAKE       0.98      0.68      0.80       628
        REAL       0.76      0.99      0.86       639

    accuracy                           0.83      1267
   macro avg       0.87      0.83      0.83      1267
weighted avg       0.87      0.83      0.83      1267


Confusion Matrix:

[[425 203]
 [  7 632]]
```

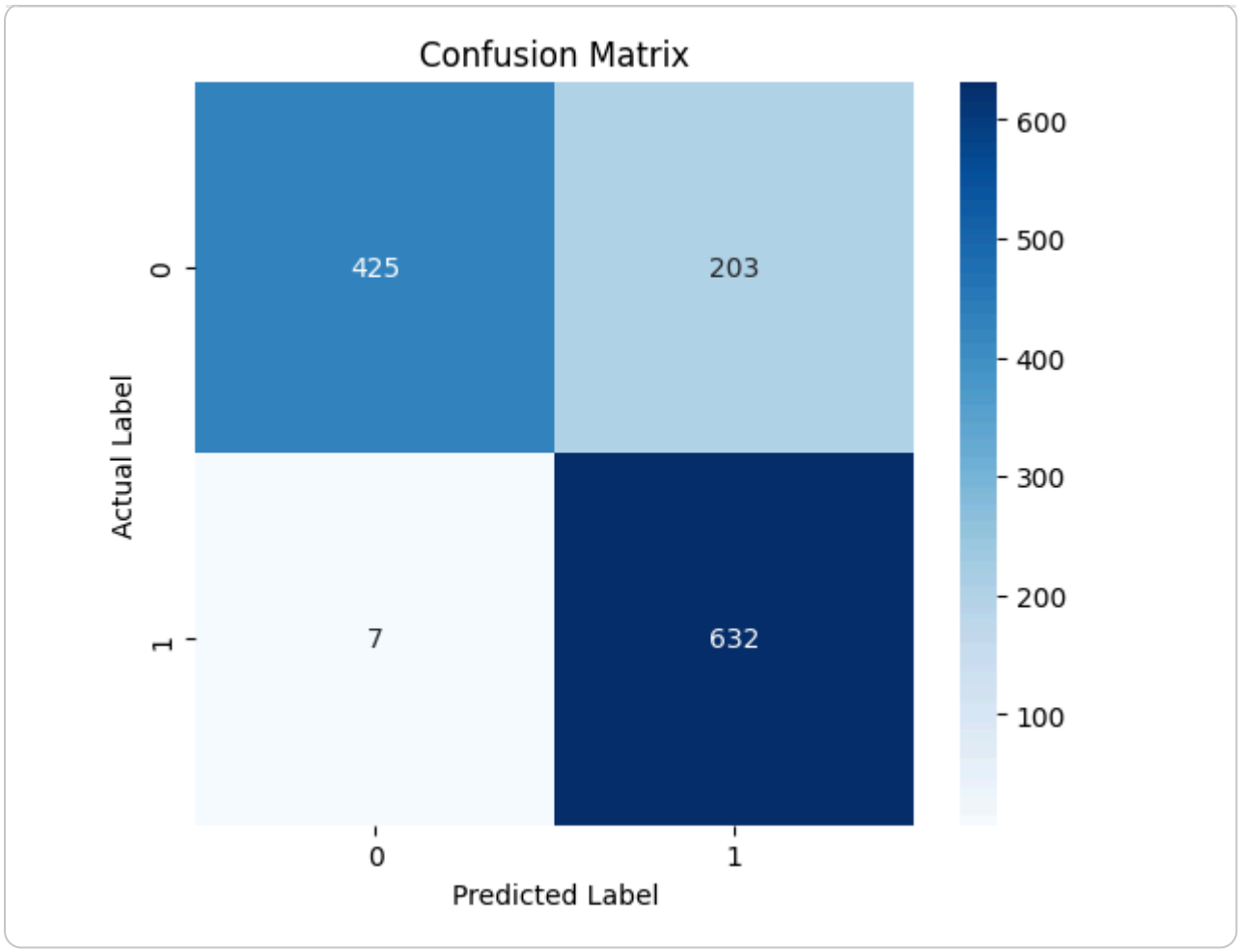
Confusion Matrix Heatmap

```
import seaborn as sns
import matplotlib.pyplot as plt

cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("Actual Label")
plt.show()
```


Confusion Matrix
425
203
7
632
0
1
Actual Label
0
1
Predicted Label
600
500
400
300
200
100