

```

import pandas as pd
import numpy as np

import re
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer,
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Load dataset
pd.read_csv( " /content/news . csv" , encoding= " latin-1" )

# Display the actual column names
print ("Actual column names: " , df.columns . tolist ( ) )

label message e FAKE Daniel Greenfield, a Shillman Journalism Fello 1 FAKE Google Pinterest Digg LinkedIn Reddit Stumbleu
2 REAL U.S. Secretary of State John F . Kerry said Mon
3 FAKE ä212 Kaydee King (@KaydeeKing) November 9, 2016
4 REAL It ' s primary day in New York and front-runners
Dataset size: (6335,
2) Class distribution:
label
REAL 3171
FAKE 3164

Name : count, dtype: int64

import nltk
nltk.download( ' stopwords ' )

stop_words = 'english ' ) ) stemmer = PorterStemmer()

def preprocess_text (text) : text = text . lower()
text = text . string . punctuation))
words = text . split ( )
words = [stemmer.stem(w) for w in words if w not in stop_words]
return . join (words)

' clean_message' ] = df[ 'message' ] . apply(preprocess_text)

[nltk_data] Downloading package stopwords to /root/nltk_data. . .
[nltk_data] Unzipping corpora/ stopwords.zip.

vectorizer =
X = vectorizer.fit_transform(df[ ' clean_message' ] )
- 'ham' : 0, 'spam' : 1})

print ("Feature matrix shape: " X. shape)
print ("Sample feature names : " vectorizer. ( ) [ : 20] )

```

:47

Feature matrix shape: (6335, 3000)

Sample feature names: abandon ' 'abc ' ' abedin' ' abil ' ' abl ' ' abort ' ' abroad ' ' absenc ' ' absolut ' 'abus ' ' academ' ' accept ' ' access' ' accompani ' ' accomplish ' ' accord ' ' account' ' accur' ' accus' ' ' achiev']

' REAL' : 1})

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

```
model = MultinomialNB()
model.fit(X_train, y_train)
```

```
print ("Model parameters: " + str(model.get_params()))
```

```
Model parameters: {'alpha': 1.0, 'class_prior': None, 'fit_prior': True, 'force_alpha': True}
```

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

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

```
print ("Precision : " + str(precision_score(y_test, y_pred)))
```

```
print ("Recall: " + str(recall_score(y_test, y_pred)))
```

```
print ("F1-score: " + str(f1_score(y_test, y_pred)))
```

```
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True,
            fmt='d', cmap='Blues',
            xticklabels=['Ham', 'Spam'],
            yticklabels=['Ham', 'Spam'],
            cbar=True)
plt.xlabel("Predicted")
```

```
plt.ylabel("Actual")
```

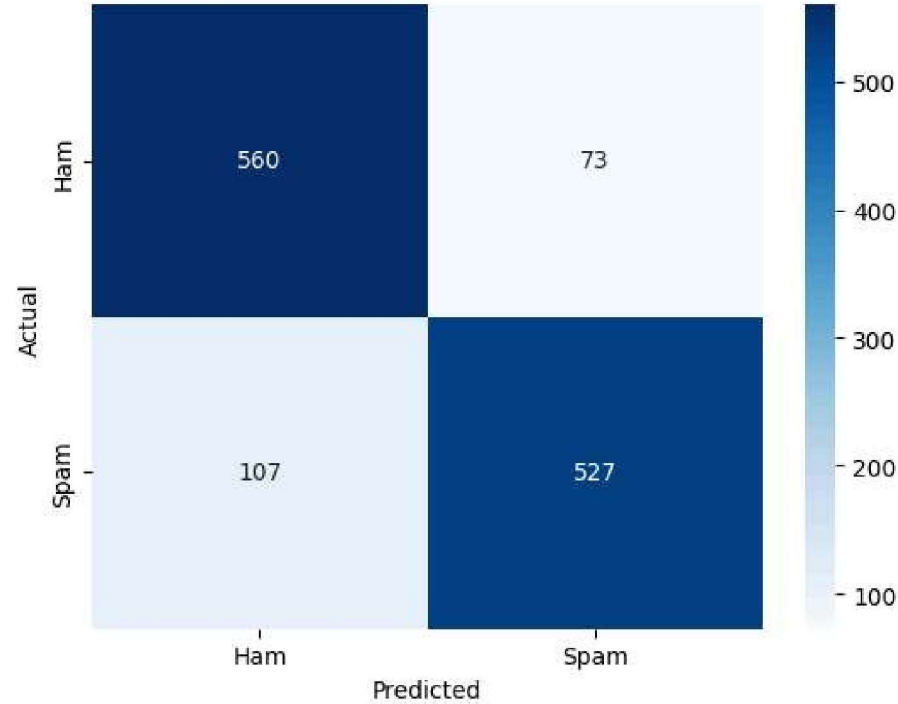
```
plt.show()
```

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

11:47

NLP (lab-11) 2403A51308 - Colab

Accuracy: 0.8579321231254933
Precision: 0.8783333333333333
Recall: 0.831230283911672
F1-score: 0.8541329011345219



Classification Report:

	precision	recall	f1-score	support
0	0.84	0.88	0.86	633
1	0.88	0.83	0.85	634
accuracy			0.86	1267
macro avg	0.86	0.86	0.86	1267
weighted avg	0.86	0.86	0.86	1267

SI P#scrollTo=7fULqkhrX5gV&printMode=true