

Spam Mail Classifier



Import Libraries

```
In [1]: # Handling Data
import pandas as pd
import numpy as np

# Visialization
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')

# For Text processing
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
# nltk.download('punkt')
# nltk.download('stopwords')

# ML
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

# DL
from tensorflow.keras.utils import plot_model
from tensorflow.keras.layers import Conv1D, Dense, MaxPooling1D, BatchNormalization, Flatten, Dropout
from tensorflow.keras.models import Sequential

#Accuracy Metrics
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

Reading Data

```
In [2]: df = pd.read_csv('spam_ham_dataset.csv')
# Removing Unnecessary column
df.drop('Unnamed: 0', axis=1, inplace = True)
# Changing column names
df.columns = ['label', 'text', 'class']
```

```
In [3]: df.head()
```

```
Out[3]:
```

	label	text	class
0	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	spam	Subject: photoshop , windows , office . cheap ...	1
4	ham	Subject: re : indian springs\r\nthis deal is t...	0

```
In [4]: df.shape
```

```
Out[4]: (5171, 3)
```

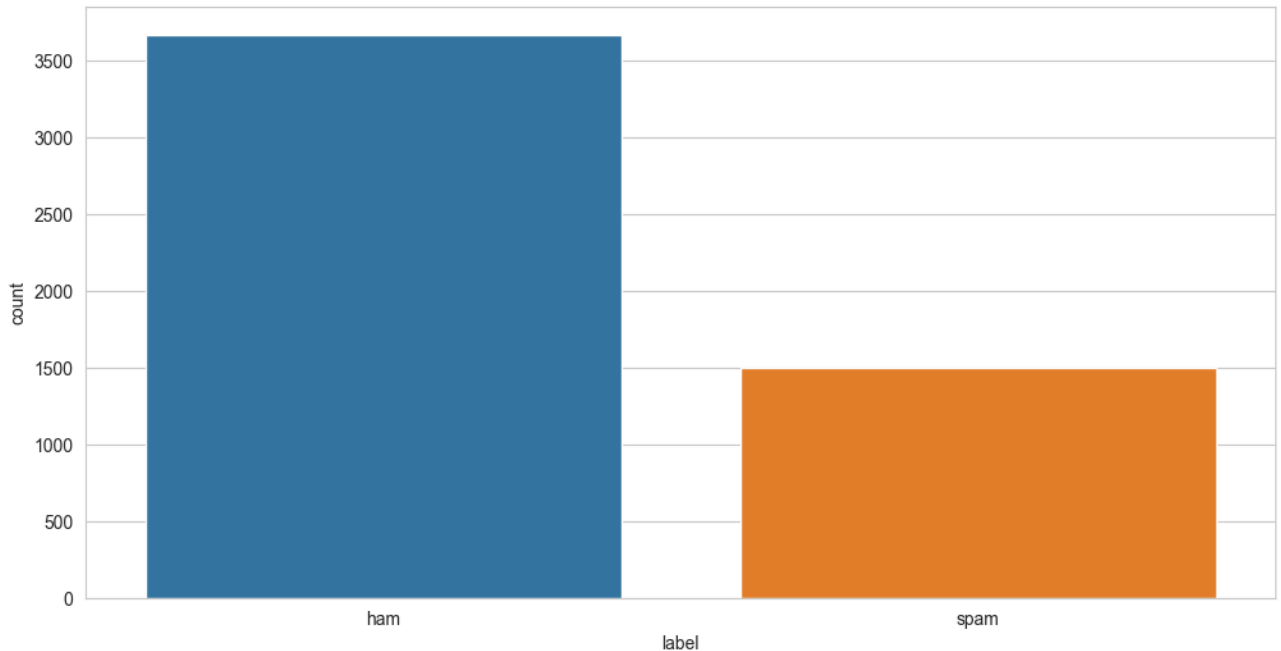
```
In [5]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5171 entries, 0 to 5170
Data columns (total 3 columns):
#   Column  Non-Null Count  Dtype
---  -
0    label    5171 non-null     object
1    text     5171 non-null     object
2    class    5171 non-null     int64
dtypes: int64(1), object(2)
memory usage: 121.3+ KB
```

```
In [6]: # No NaN in the data
df.isna().sum()
```

```
Out[6]: label    0
text          0
class         0
dtype: int64
```

```
In [7]: # Barplot describes the count of the class labels
plt.figure(figsize = (12, 6))
sns.countplot(data = df, x = 'label');
```



Viewing samples of the data

```
In [8]: # Let's see few examples of the data

for i in df.iterrows():
    print("Class Label: {}\nMail: \n{}\n\n".format(i[1][0], i[1][1]))
    if i[0] == 6: break

( see attached file : hplnol 09 . xls )
- hplnol 09 . xls
```

```
Class Label: ham
Mail:
Subject: neon retreat
ho ho ho , we ' re around to that most wonderful time of the year - - - neon leaders retreat time !
i know that this time of year is extremely hectic , and that it ' s tough to think about anything past the ho
lidays , but life does go on past the week of december 25 through january 1 , and that ' s what i ' d like yo
u to think about for a minute .
on the calender that i handed out at the beginning of the fall semester , the retreat was scheduled for the w
eekend of january 5 - 6 . but because of a youth ministers conference that brad and dustin are connected with
that week , we ' re going to change the date to the following weekend , january 12 - 13 . now comes the part
you need to think about .
i think we all agree that it ' s important for us to get together and have some time to recharge our batterie
s before we get to far into the spring semester , but it can be a lot of trouble and difficult for us to get
away without kids , etc . so , brad came up with a potential alternative for how we can get together on that
weekend , and then you can let me know which you prefer .
the first option would be to have a retreat similar to what we ' ve done the past several years . this year w
```

Remove stopwords from the data

```
In [9]: %%time
stop_words = set(stopwords.words('english'))

df['text'] = df['text'].apply(lambda x: ' '.join([ word for word in word_tokenize(x) if not word in stop_words])

CPU times: total: 6.3 s
Wall time: 6.72 s
```

```
In [10]: df.sample(10)
```

```
Out[10]:
```

	label	text	class
4343	ham	Subject : hpl nom november 28 , 2000 (see att...	0
373	spam	Subject : @ \$ 89 . 00 w / speakers hp ...	1
2111	spam	Subject : : wed , 21 jan 2004 08 : 15 : 42 - 0...	1
1746	ham	Subject : nom 3 / 14 3 / 15 well , goofed emai...	0
2304	ham	Subject : fw : fw : march 2001 invoice mary - ...	0
1994	ham	Subject : look -- 'important !!!!! ok , ...	0
3628	ham	Subject : meter 986296 scherlyn , per conversa...	0
3459	spam	Subject : hello mail transaction failed . part...	1
3525	spam	Subject : fix penis growth limited time offer ...	1
1633	ham	Subject : phillips petroleum wanted update phi...	0

```
In [11]: X = df.loc[:, 'text']
y = df.loc[:, 'class']

print(f"Shape of X: {X.shape}\nshape of y: {y.shape}")

Shape of X: (5171,)
shape of y: (5171,)
```

Split data into train and test in 80:20

```
In [12]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=11)
```

```
In [13]: print(f"Train Data Shape: {X_train.shape}\nTest Data Shape: {X_test.shape}")

Train Data Shape: (4136,)
Test Data Shape: (1035,)
```

Preprocess text to build the ML model

```
In [14]: cVect = CountVectorizer()
cVect.fit(X_train)
```

```
Out[14]: CountVectorizer()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.
```

Let's see the vocabulary that has extracted by the count vectorizer

```
In [15]: print('NO. of Tokens: ', len(cVect.vocabulary_.keys()))

NO. of Tokens: 44255
```

```
In [16]: # document term vector (dtv)
dtv = cVect.transform(X_train)
```

```
In [17]: type(dtv)
```

```
Out[17]: scipy.sparse._csr.csr_matrix
```

```
In [18]: dtv = dtv.toarray()
```

```
In [19]: print(f"Number of Observations: {dtv.shape[0]}\nTokens/Features: {dtv.shape[1]}")

Number of Observations: 4136
Tokens/Features: 44255
```

```
In [20]: # Let's see an sample that has been preprocessed
dtv[1]
```

```
Out[20]: array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

Logistic Regression.

Logistic Regression could help use predict whether the student passed or failed. Logistic regression predictions are discrete (only specific values or categories are allowed). We can also view probability scores underlying the model's classifications.

Hyperparameter Tuning

```
In [21]: lr = LogisticRegression(verbose=1)

grid={ "C": [float(i) for i in range(1, 3)], "penalty": "l2", "solver": [ 'lbfgs', 'liblinear' ] }
logreg_cv = GridSearchCV(lr, grid, cv=4)
logreg_cv.fit(dtv, y_train)

print("Tuned Hpyerparameters :", logreg_cv.best_params_)
print("accuracy :", logreg_cv.best_score_)

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 1 out of 1 | elapsed: 6.3s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 1 out of 1 | elapsed: 5.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 1 out of 1 | elapsed: 5.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 1 out of 1 | elapsed: 6.1s finished

[LibLinear][LibLinear][LibLinear][LibLinear]

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 1 out of 1 | elapsed: 7.8s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 1 out of 1 | elapsed: 6.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 1 out of 1 | elapsed: 5.9s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 1 out of 1 | elapsed: 6.0s finished

[LibLinear][LibLinear][LibLinear][LibLinear][LibLinear]Tuned Hpyerparameters : {'C': 1.0, 'penalty': 'l2', 'solver': 'liblinear'}
accuracy : 0.9765473887814313
```

```
In [22]: %%time
lr = LogisticRegression(solver='liblinear', penalty='l2', C=1.0)
lr.fit(dtv, y_train)
```

```
CPU times: total: 1.28 s
Wall time: 1.71 s
```

```
Out[22]: LogisticRegression(solver='liblinear')
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

Evaluate on the Test data

```
In [23]: # Preprocess the test data
test_dtv = cvect.transform(X_test)
test_dtv = test_dtv.toarray()
print(f"Number of Observations: {test_dtv.shape[0]}\nTokens/Features: {test_dtv.shape[1]}")
```

```
Number of Observations: 1035
Tokens/Features: 44255
```

```
In [24]: %%time  
pred = lr.predict(test_dtv)
```

CPU times: total: 156 ms
Wall time: 169 ms

```
In [25]: print('Accuracy: ', accuracy_score(y_test, pred) * 100)
```

Accuracy: 98.35748792270532

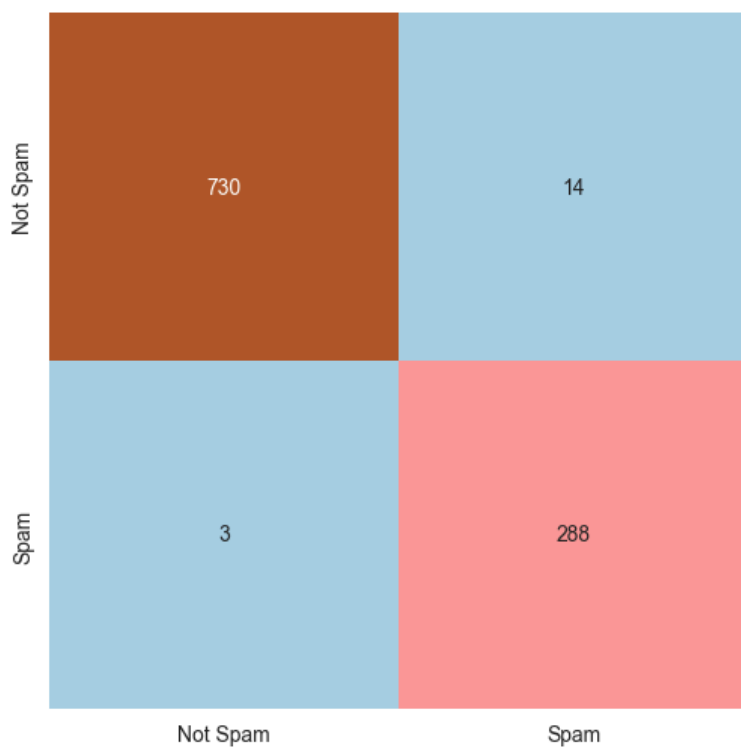
Classification Report of the classifier

```
In [26]: # 0 - Not Spam / Ham  
# 1 - Spam  
print(classification_report(y_test, pred))
```

	precision	recall	f1-score	support
0	1.00	0.98	0.99	744
1	0.95	0.99	0.97	291
accuracy			0.98	1035
macro avg	0.97	0.99	0.98	1035
weighted avg	0.98	0.98	0.98	1035

Confusion Matrix

```
In [27]: cmat = confusion_matrix(y_test, pred)  
plt.figure(figsize = (6, 6))  
sns.heatmap(cmat, annot = True, cmap = 'Paired', cbar = False, fmt="d", xticklabels=['Not Spam', 'Spam'], ytickl
```

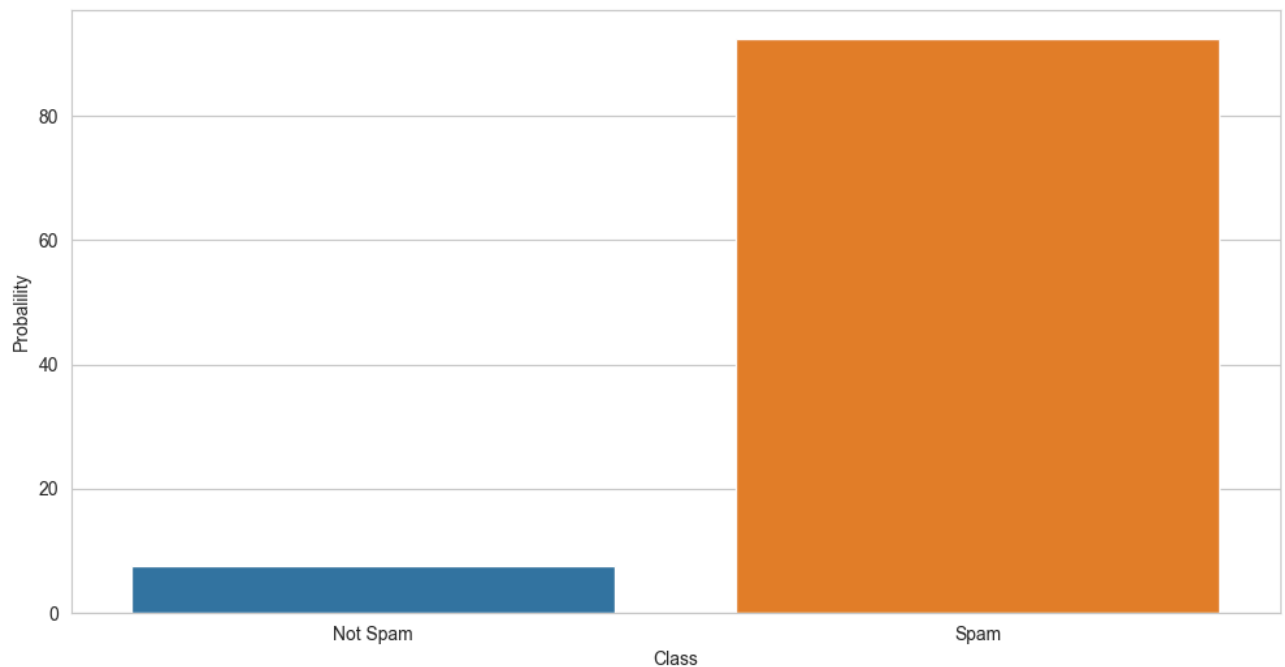


Predict Class label for the unseen data i.e., Spam or Not Spam

```
In [28]: # 'You won 1000$ prize money in Lottery. Click here to avail'
def predict_class(lr):
    text = input('Enter Text(Subject of the mail): ')
    text = [ ' '.join([ word for word in word_tokenize(text) if not word in stop_words])]
    t_dtv = cVect.transform(text).toarray()
    print('Predicted Class:', end = ' ')
    print('Spam' if lr.predict(t_dtv)[0] else 'Not Spam')
    prob = lr.predict_proba(t_dtv)*100
    print(f'Not Spam: {prob[0][0]}%\nSpam: {prob[0][1]}%')
    plt.figure(figsize=(12, 6))
    sns.barplot(x = ['Not Spam', 'Spam'] , y = [prob[0][0], prob[0][1]])
    plt.xlabel('Class')
    plt.ylabel('Probalility')
    plt.show()
```

```
In [29]: predict_class(lr)
```

Enter Text(Subject of the mail): You won 1000\$ prize money in lottery. Click here to avail
Predicted Class: Spam
Not Spam: 7.454886754592693%
Spam: 92.54511324540731%



The Above Mail Text is Exactly predict the Spam mail, Hence our model is predict the Correct with 92.42% Accuracy