Message Spam/non-Spam Prediction

Sweta Swarupa

Contents

1 INTRODUCTION

The aim of this project is to classify messages into Spam vs non-Spam messages.The dataset has been taken from Kaggle. The data consists of email body and labels for text classification.

1) Message body
2) Labels

```
#importing the required libraries
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import re
import os
```

```
#importing the required sklearn libraries
import sklearn
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import TfidfTransformer, CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.pipeline import Pipeline
```

2 READING DATA

```
# Reading files
nlp_train = pd.read_csv("C:\\NLP\\SMS_train.csv", low_memory=False)
nlp_test = pd.read_csv("C:\\NLP\\SMS_test.csv", low_memory=False)
nlp_train.head()
```

```
##    S. No.                                       Message_body     Label
## 0       1                         Rofl. Its true to its name  Non-Spam
## 1       2  The guy did some bitching but I acted like i’d...  Non-Spam
## 2       3  Pity, * was in mood for that. So...any other s...  Non-Spam
## 3       4               Will ü b going to esplanade fr home?  Non-Spam
## 4       5  This is the 2nd time we have tried 2 contact u...      Spam
```

```
print(nlp_train.shape)
```

```
## (957, 3)
```

```
print(nlp_test.shape)
```

```
## (125, 3)
```

There are 957 rows and 3 columns in train dataset and 125 rows and 3 columns in test dataset.

```
#Checking for null values in train dataset
nlp_train.isnull().sum()
```

```
## S. No.          0
## Message_body    0
## Label           0
## dtype: int64
```

```
#Checking for null values in test dataset
nlp_test.isnull().sum()
```

```
## S. No.          0
## Message_body    0
## Label           0
## dtype: int64
```

There are no missing vales in train and test datasets.

```
#Lets drop Serial number column
nlp_train = nlp_train.drop(['S. No.'], axis=1)
nlp_test = nlp_test.drop(['S. No.'], axis=1)
```

Since, serial no. column is not useful in out modelling, we are going to drop that column.

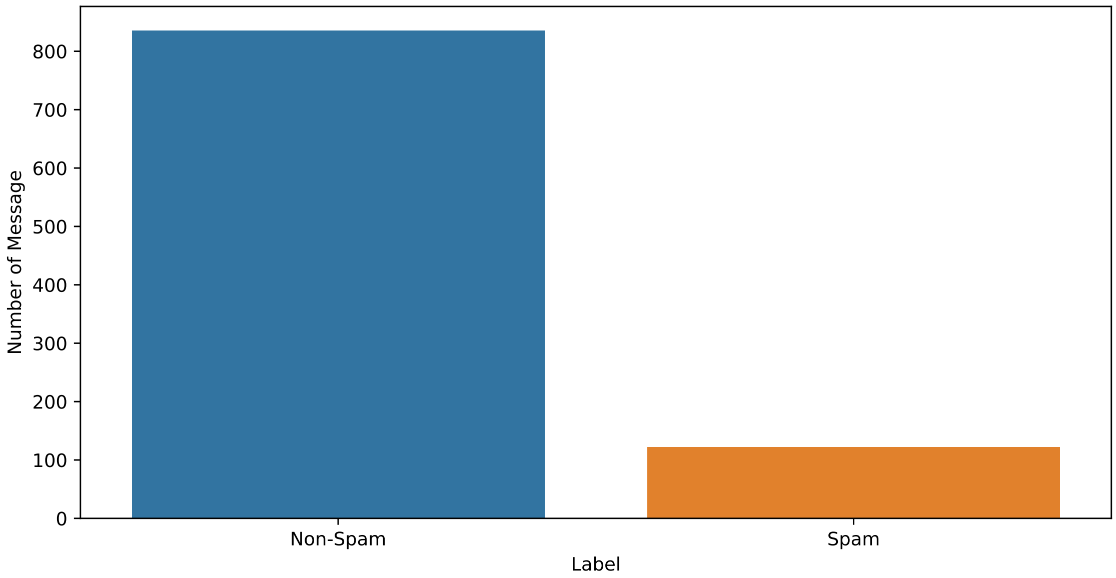
```
nlp_train['Message_body'][:9]
```

```
## 0                           Rofl. Its true to its name
## 1    The guy did some bitching but I acted like i’d...
## 2    Pity, * was in mood for that. So...any other s...
## 3                 Will ü b going to esplanade fr home?
## 4    This is the 2nd time we have tried 2 contact u...
## 5    REMINDER FROM O2: To get 2.50 pounds free call...
## 6                                         Huh y lei...
## 7    Why don’t you wait ’til at least wednesday to ...
## 8                                  Ard 6 like dat lor.
## Name: Message_body, dtype: object
```

3 DATA VISUALIZATION

3.1 Visualizing Class Distribution

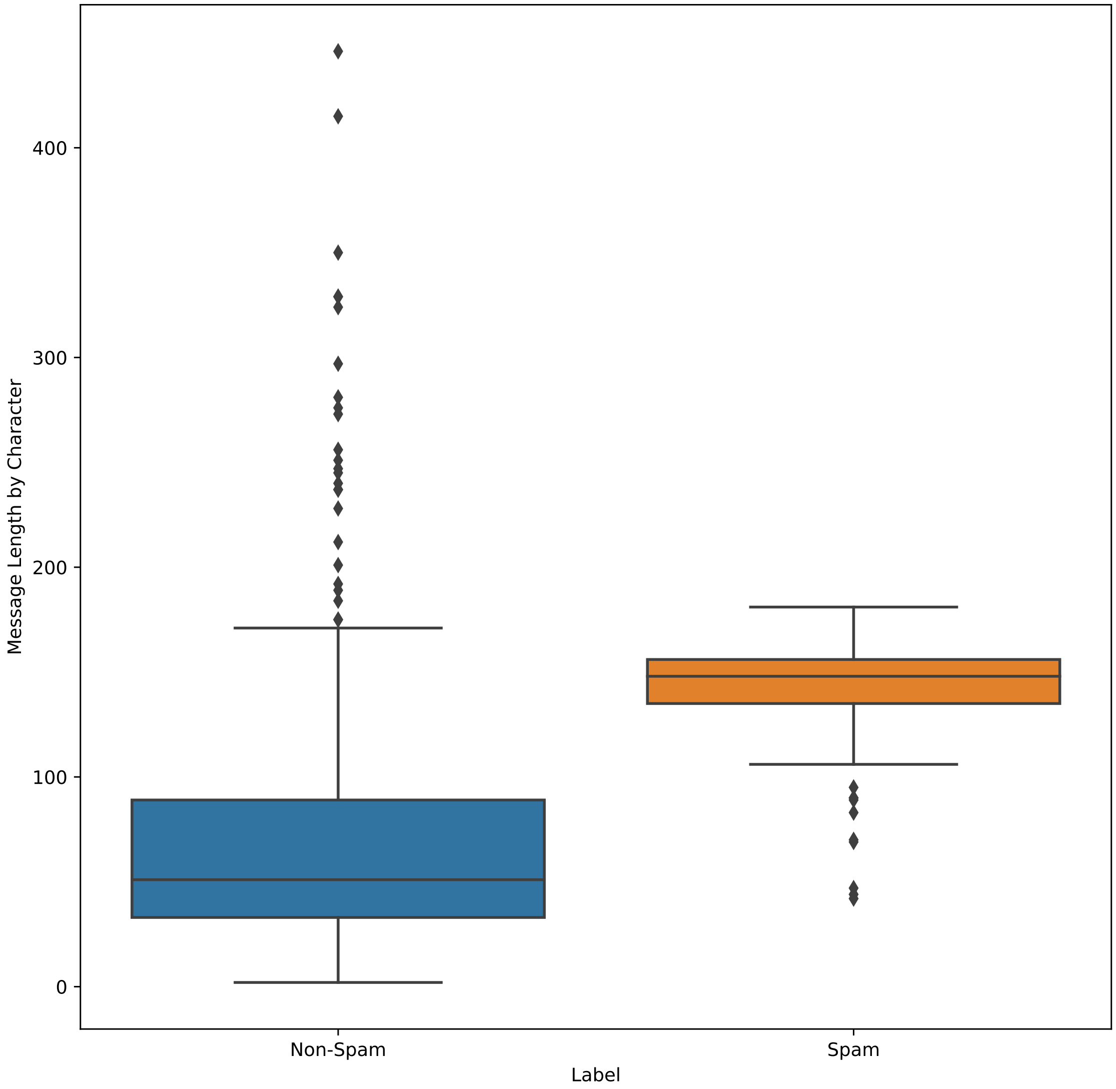
```
plt.figure(figsize=(10,5))
sns.countplot(x='Label',data = nlp_train)
plt.ylabel("Number of Message")
plt.xlabel("Label")
plt.show()
```



Clearly, there is an imbalance in the target values however, I am going to ignore this for now.

3.2 Visualizing Message Length by Characaters

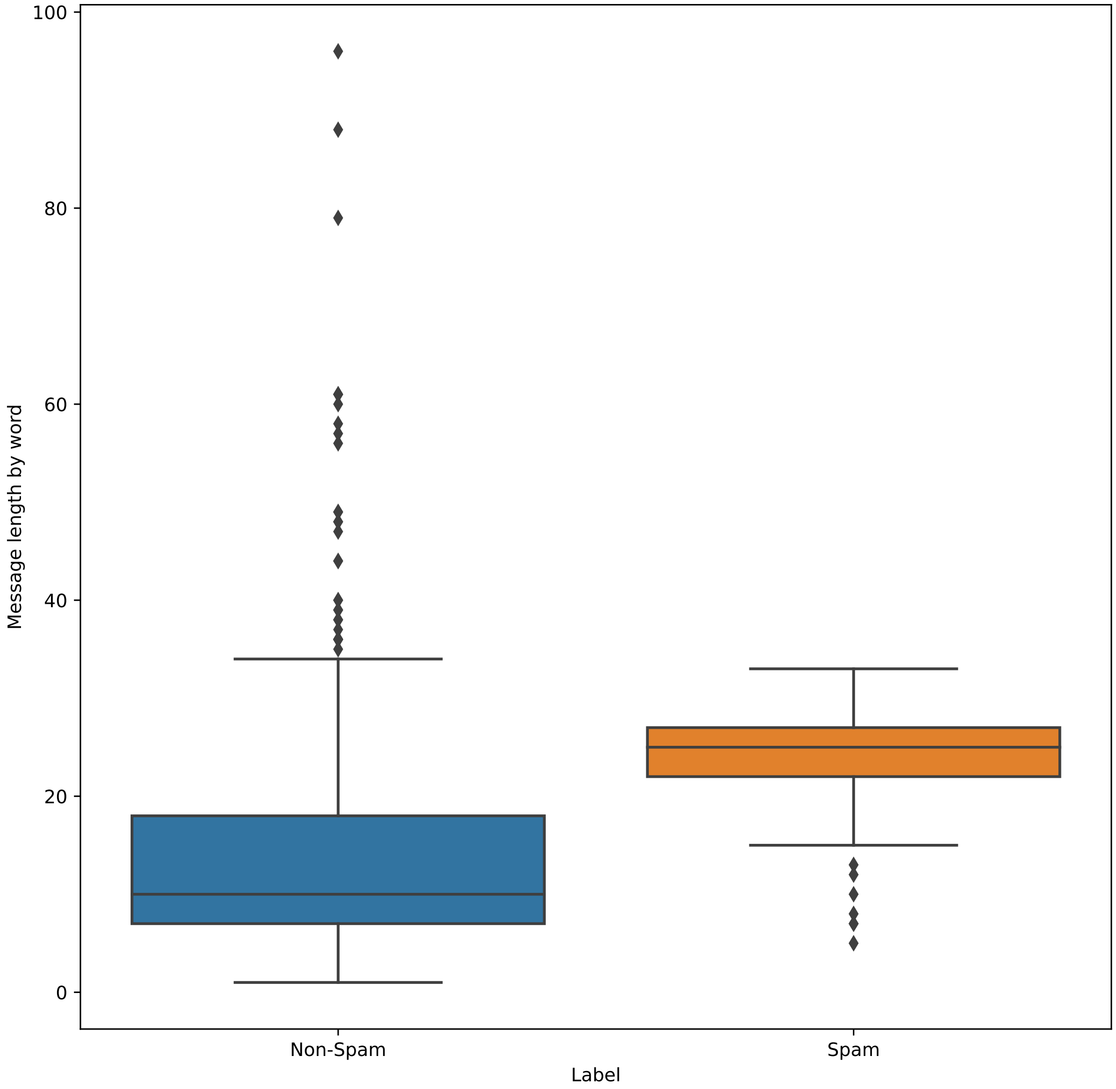
```
plt.figure(figsize=(10,10))
train_sent = nlp_train['Message_body'].str.len()
sns.boxplot(x="Label",y=train_sent,data=nlp_train)
plt.xlabel("Label")
plt.ylabel("Message Length by Character")
plt.show()
```



Looks like spam messages have more characters.

3.3 Visualizing Message Length by Words

```
plt.figure(figsize=(10,10))
train_sent = nlp_train['Message_body'].str.split().map(lambda x : len(x))
sns.boxplot(x="Label",y=train_sent,data=nlp_train)
plt.xlabel("Label")
plt.ylabel("Message length by word")
plt.show()
```



Similar to characters,looks like spam messages have more words compared to non-spam messages. Also, there are a lot of outliers in the non-spam messages for both message length by characters and words.

4 DATA CLEANING

4.1 Removing URLS

```
def url_clean(Message_body):
    url = re.compile(r'https?://\S+|www\.\S+')
    return url.sub(r'',Message_body)
```

4.2 Removing HTML tags

```
def html_clean(Message_body):
    html = re.compile(r'<.*?>')
    return html.sub(r'',Message_body)
```

```
def remove_emoji(Message_body):
    emoji_pattern = re.compile("["
                           u"\U0001F600-\U0001F64F"  # emoticons
                           u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                           u"\U0001F680-\U0001F6FF"  # transport & map symbols
                           u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                           u"\U00002702-\U000027B0"
                           u"\U000024C2-\U0001F251"
                           "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', Message_body)
```

```
nlp_train['Message_body']= nlp_train['Message_body'].apply(lambda x : url_clean(x))
nlp_train['Message_body']= nlp_train['Message_body'].apply(lambda x : html_clean(x))

nlp_test['Message_body']= nlp_test['Message_body'].apply(lambda x : url_clean(x))
nlp_test['Message_body']= nlp_test['Message_body'].apply(lambda x : html_clean(x))

nlp_test['Message_body']= nlp_test['Message_body'].apply(lambda x : remove_emoji(x))
nlp_test['Message_body']= nlp_test['Message_body'].apply(lambda x : remove_emoji(x))
```

5 DATA PREPARATION

5.1 Extracting the messages and labels for modelling

```
X_train = nlp_train["Message_body"]
y_train = nlp_train["Label"]
X_test = nlp_test['Message_body']
y_test = nlp_test["Label"]
```

5.2 Bag of words vectorization

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

vectorizer = TfidfVectorizer()
train_x_vectors = vectorizer.fit_transform(X_train)

test_x_vectors = vectorizer.transform(X_test)

print(X_train[0])
```

```
## Rofl. Its true to its name
```

```
print(train_x_vectors[0].toarray())
```

```
## [[0. 0. 0. ... 0. 0. 0.]]
```

6 CLASSIFICATION

6.1 Linear SMV

```
from sklearn import svm

clf_svm = svm.SVC(kernel='linear')

clf_svm.fit(train_x_vectors, y_train)
```

```
## SVC(kernel='linear')
```

```
X_test[0]
```

```
## "UpgrdCentre Orange customer, you may now claim your FREE CAMERA PHONE upgrade for your loyalty. Call n
```

```
clf_svm.predict(test_x_vectors[0])
```

```
## array(['Spam'], dtype=object)
```

6.2 Decision Tree

```
from sklearn.tree import DecisionTreeClassifier

clf_dec = DecisionTreeClassifier()
clf_dec.fit(train_x_vectors, y_train)
```

```
## DecisionTreeClassifier()
```

```
clf_dec.predict(test_x_vectors[0])
```

```
## array(['Spam'], dtype=object)
```

6.3 Naive Bayes

```
from sklearn.naive_bayes import GaussianNB

clf_gnb = DecisionTreeClassifier()
clf_gnb.fit(train_x_vectors, y_train)
```

```
## DecisionTreeClassifier()
```

```
clf_gnb.predict(test_x_vectors[0])
```

```
## array([’Spam’], dtype=object)
```

6.4 Logistic Regression

```
from sklearn.linear_model import LogisticRegression

clf_log = LogisticRegression()
clf_log.fit(train_x_vectors, y_train)
```

```
## LogisticRegression()
```

```
clf_log.predict(test_x_vectors[0])
```

```
## array([’Spam’], dtype=object)
```

7 EVALUATION

7.1 Mean Accuracy

```
print(clf_svm.score(test_x_vectors, y_test))
```

```
## 0.912
```

```
print(clf_dec.score(test_x_vectors, y_test))
```

```
## 0.864
```

```
print(clf_gnb.score(test_x_vectors, y_test))
```

```
## 0.88
```

```
print(clf_log.score(test_x_vectors, y_test))
```

```
## 0.64
```

7.2 F1 Scores

```
from sklearn.metrics import f1_score

f1_score(y_test, clf_svm.predict(test_x_vectors), average=None)
```

```
## array([0.89908257, 0.92198582])
```

```
f1_score(y_test, clf_dec.predict(test_x_vectors), average=None)
```

```
## array([0.85217391, 0.87407407])
```

```
f1_score(y_test, clf_gnb.predict(test_x_vectors), average=None)
```

```
## array([0.86486486, 0.89208633])
```

```
f1_score(y_test, clf_log.predict(test_x_vectors), average=None)
```

```
## array([0.68531469, 0.57943925])
```

7.3 Testing on a new set

```
test_set = ['very fun', "bad book do not buy", 'pls reply 2 this text with your valid name', 'for your inc
new_test = vectorizer.transform(test_set)

clf_svm.predict(new_test)
```

```
## array(['Non-Spam', 'Non-Spam', 'Spam', 'Spam'], dtype=object)
```

As you can see above that I entered 4 random messages to test the model and the model is predicting the message type correctly. Overall, I am satisfied with the model.

8 TUNING OUR MODEL (with Grid Search)

```
from sklearn.model_selection import GridSearchCV

parameters = {'kernel': ('linear', 'rbf'), 'C': (1,4,8,16,32)}

svc = svm.SVC()
clf = GridSearchCV(svc, parameters, cv=5)
clf.fit(train_x_vectors, y_train)
```

```
## GridSearchCV(cv=5, estimator=SVC(),
##              param_grid={'C': (1, 4, 8, 16, 32), 'kernel': ('linear', 'rbf')})
```

```
print(clf.score(test_x_vectors, y_test))
```

```
## 0.92
```

9 CONCLUSION

I built 4 different models (linear SMV, Decision Tree, Naive Bayes, Logistic Regression) for predicting spam vs non-spam messages. Mean accuracy and f1 score for linear SMV is the best out of the 4 models. I also tuned the model with grid search to improve the score further.