## SMS Spam Classification

Download The Dataset

Import The Required Library

```
#importing the reqired Library

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

Read the Dataset

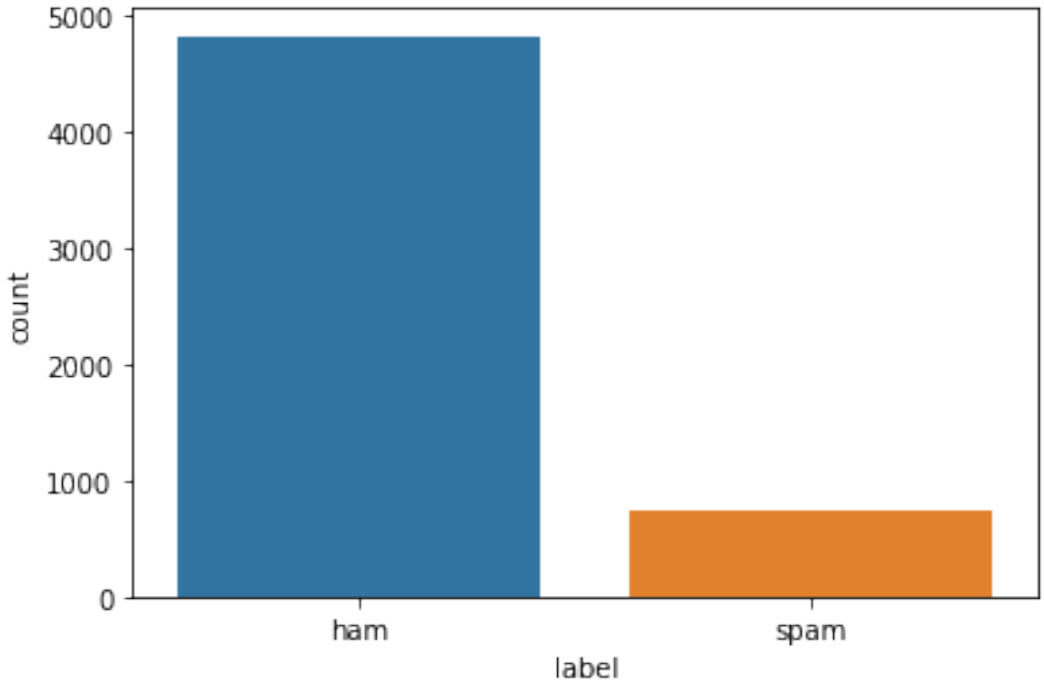
```
# Reading the data
df = pd.read_csv("/content/spam.csv",encoding='latin-1')
df.head()
```

```
     v1                                                 v2 Unnamed: 2
\
0   ham  Go until jurong point, crazy.. Available only ...         NaN

1   ham                      Ok lar... Joking wif u oni...         NaN

2  spam  Free entry in 2 a wkly comp to win FA Cup fina...         NaN

3   ham  U dun say so early hor... U c already then say...         NaN

4   ham  Nah I don't think he goes to usf, he lives aro...         NaN


  Unnamed: 3 Unnamed: 4
0        NaN        NaN
1        NaN        NaN
2        NaN        NaN
3        NaN        NaN
4        NaN        NaN
```

```
df = df.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis=1)
df = df.rename(columns={'v1':'label','v2':'Text'})
df['label_enc'] = df['label'].map({'ham':0,'spam':1})
df.head()
```

```
  label                                               Text  label_enc
0   ham  Go until jurong point, crazy.. Available only ...          0
1   ham                      Ok lar... Joking wif u oni...          0
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...          1
3   ham  U dun say so early hor... U c already then say...          0
4   ham  Nah I don't think he goes to usf, he lives aro...          0
```

```
sns.countplot(x=df['label'])
plt.show()
```




```
# Find average number of tokens in all sentences
avg_words_len=round(sum([len(i.split()) for i in
df['Text']])/len(df['Text']))
print(avg_words_len)
```

```
15
```

```
# Splitting data for Training and testing
from sklearn.model_selection import train_test_split

X, y = np.asanyarray(df['Text']), np.asanyarray(df['label_enc'])
new_df = pd.DataFrame({'Text': X, 'label': y})
X_train, X_test, y_train, y_test = train_test_split(
     new_df['Text'], new_df['label'], test_size=0.2, random_state=42)
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
((4457,), (4457,), (1115,), (1115,))
```

```
def word_count_plot(data):
     # finding words along with count
```

```
     word_counter = collections.Counter([word for sentence in data for
word in sentence.split()])
     most_count = word_counter.most_common(30) # 30 most common words
     # sorted data frame
     most_count = pd.DataFrame(most_count, columns=["Word",
"Count"]).sort_values(by="Count")
     most_count.plot.barh(x = "Word", y = "Count", color="green",
figsize=(10, 15))
```

Create a Model

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report,accuracy_score

tfidf_vec = TfidfVectorizer().fit(X_train)
X_train_vec,X_test_vec =
tfidf_vec.transform(X_train),tfidf_vec.transform(X_test)

baseline_model = MultinomialNB()
baseline_model.fit(X_train_vec,y_train)
```

```
MultinomialNB()
```

```
ham_words = ''
spam_words = ''

#creating an embedding layer

# load the whole embedding into memory
embeddings_index = dict()
f = open("/content/spam.csv")
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()
print('Loaded %s word vectors.' % len(embeddings_index))
```

```
---------------------------------------------------------------------
-----
UnicodeDecodeError                        Traceback (most recent call
last)
<ipython-input-8-ad0b3449a723> in <module>
      4 embeddings_index = dict()
      5 f = open("/content/spam.csv")
----> 6 for line in f:
      7     values = line.split()
      8     word = values[0]
```

```
/usr/lib/python3.7/codecs.py in decode(self, input, final)
    320         # decode input (taking the buffer into account)
    321         data = self.buffer + input
--> 322         (result, consumed) = self._buffer_decode(data,
self.errors, final)
    323         # keep undecoded input until the next call
    324         self.buffer = data[consumed:]

UnicodeDecodeError: 'utf-8' codec can't decode bytes in position 606-
607: invalid continuation byte
```

```
import pandas as pd
import numpy as np
import re
import collections
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('dark_background')
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
import warnings
warnings.simplefilter(action='ignore', category=Warning)
import keras
from keras.layers import Dense, Embedding, LSTM, Dropout
from keras.models import Sequential
from keras.preprocessing.text import Tokenizer
import pickle
```

```
for val in data[data['label'] == 'spam'].text:
    text = val.lower()
    tokens = nltk.word_tokenize(text)
    for words in tokens:
        spam_words = spam_words + words + ' '
```

```
---------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
<ipython-input-6-ed68ec7f9b51> in <module>
----> 1 for val in data[data['label'] == 'spam'].text:
      2     text = val.lower()
      3     tokens = nltk.word_tokenize(text)
      4     for words in tokens:
      5         spam_words = spam_words + words + ' '

NameError: name 'data' is not defined
```

```
from sklearn.preprocessing import LabelEncoder
```

```
lb_enc = LabelEncoder()
y = lb_enc.fit_transform(data["SpamHam"])

tokenizer = Tokenizer()
#initializing the tokenizer
tokenizer.fit_on_texts(X)
# fitting on the sms data
text_to_sequence = tokenizer.texts_to_sequences(X)
```

Fit the model

```
#fit the model
history=model.fit(sequences_matrix,Y_train,batch_size=20,epochs=15,
validation_split=0.2)
```

Save the model

```
#save the model
model.save('A4Spam_sms_classifier.h5')
```

Compile the Model

```
#compile the model

model.compile(loss='binary_crossentropy',optimizer=Adam(),metrics=['ac
curacy'])
```

Test the model

```
test_sequences = tok.texts_to_sequences(X_test)
test_sequences_matrix =
keras.utils.pad_sequences(test_sequences,maxlen=max_len)
accr = model.evaluate(test_sequences_matrix,Y_test)
print('Test set\n Loss: {:0.3f}\n Accuracy:
{:0.3f}'.format(accr[0],accr[1]))
```