**Surya Group Of Institution**

**Naan Mudhalvan**

|  |
| --- |
| **IBM - Artificial Intelligence**  ANBU A  **422221104006**  **Team 01** |

**Building a Smarter AI-Powered Spam Classifier**

**Spam Classifier with Keras**:

This project uses deep learning and AI to handle spam content. [It covers building an AI model, integrating a NoSQL database for storing inference results, and deploying the model into production](https://colab.research.google.com/github/codingforentrepreneurs/AI-as-an-API/blob/main/guides/spam-classifier/Spam_Classifier_with_Keras.ipynb" \t "https://edgeservices.bing.com/edgesvc/_blank).

**Machine Learning Email Spam Detector**:

This tutorial guides you through building an email spam detector using Python. [It demonstrates how to train a spam detector to recognize and classify emails into spam and non-spam](https://blog.logrocket.com/email-spam-detector-python-machine-learning/" \t "https://edgeservices.bing.com/edgesvc/_blank),[Spam Classifier with sklearn: This tutorial provides a step-by-step guide to building a simple spam classifier using Python](https://www.milindsoorya.com/blog/build-a-spam-classifier-in-python" \t "https://edgeservices.bing.com/edgesvc/_blank)

**Building Spam Classifier with Logistic Regression**:

This project focuses on building a logistic regression classifier using scikit-learn. [It helps predict whether an email is spam or ham](https://cloudxlab.com/assessment/playlist-intro/473/project-building-spam-classifier" \t "https://edgeservices.bing.com/edgesvc/_blank)

Indroduction:

The upsurge in the volume of unwanted emails called spam has created an intense need for the development of more dependable and robust antispam filters. Any promotional messages or advertisements that end up in our inbox can be categorised as spam as they don't provide any value and often irritates us.

Overview of the Dataset used:

We will make use of the SMS spam classification data.

The SMS Spam Collection is a set of SMS tagged messages that have been collected for SMS Spam research. It contains one set of SMS messages in English of 5,574 messages, tagged according to being ham (legitimate) or spam.

In this article, We’ll discuss:

****Data processing :****

* Import the required packages
* Loading the Dataset
* Remove the unwanted data columns
* Preprocessing and Exploring the Dataset
* Build word cloud to see which message is spam and which is not.
* Remove the stop words and punctuations
* Convert the text data into vectors

****Building a sms spam classification model :****

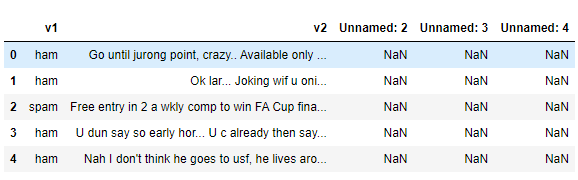
* Split the data into train and test sets
* Use Sklearn built-in classifiers to build the models
* Train the data on the model
* Make predictions on new data

Import the required packages :

%matplotlib inline  
import matplotlib.pyplot as plt  
import csv  
import sklearn  
import pickle  
from wordcloud import WordCloud  
import pandas as pd  
import numpy as np  
import nltk  
from nltk.corpus import stopwords  
from sklearn.feature\_extraction.text import CountVectorizer, TfidfTransformer  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.model\_selection import GridSearchCV,train\_test\_split,StratifiedKFold,cross\_val\_score,learning\_curve

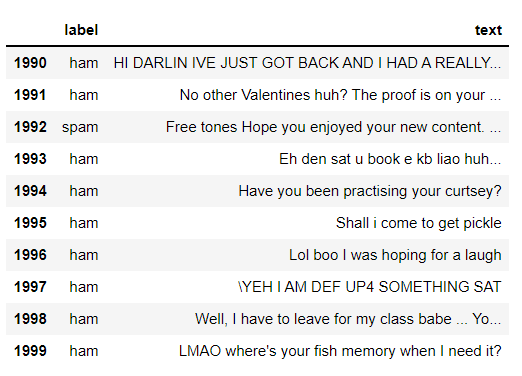
Loading the dataset :

data = pd.read\_csv('dataset/spam.csv', encoding='latin-1')  
data.head()



****Removing unwanted columns :****

data = data.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)  
data = data.rename(columns={"v2" : "text", "v1":"label"})  
data[1990:2000]



Let's move on.

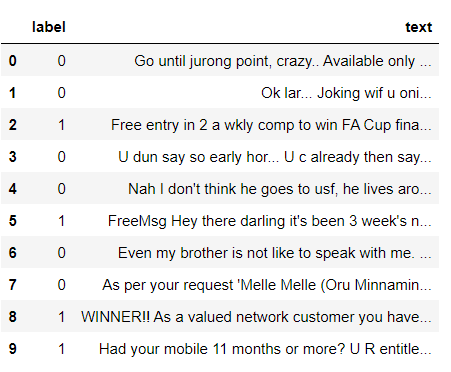
data['label'].value\_counts()  
  
# OUTPUT  
ham 4825  
spam 747  
Name: label, dtype: int64

Preprocessing and Eploring the Dataset :

# Import nltk packages and Punkt Tokenizer Models  
import nltk  
nltk.download("punkt")  
import warnings  
warnings.filterwarnings('ignore')

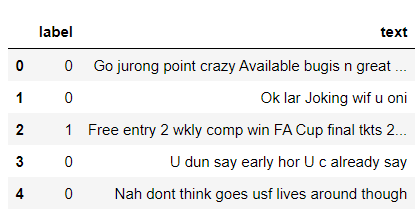
Build word cloud to see which message is spam and which is not :

ham\_words = ''  
spam\_words = ''  
  
# Creating a corpus of spam messages  
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 + ' '  
  
# Creating a corpus of ham messages  
for val in data[data['label'] == 'ham'].text:  
text = text.lower()  
tokens = nltk.word\_tokenize(text)  
for words in tokens:  
ham\_words = ham\_words + words + ' '  
  
spam\_wordcloud = WordCloud(width=500, height=300).generate(spam\_words)  
ham\_wordcloud = WordCloud(width=500, height=300).generate(ham\_words)  
  
#Spam Word cloud  
plt.figure( figsize=(10,8), facecolor='w')  
plt.imshow(spam\_wordcloud)  
plt.axis("off")  
plt.tight\_layout(pad=0)  
plt.show()  
  
#Creating Ham wordcloud  
plt.figure( figsize=(10,8), facecolor='g')  
plt.imshow(ham\_wordcloud)  
plt.axis("off")  
plt.tight\_layout(pad=0)  
plt.show()  
  
data = data.replace(['ham','spam'],[0, 1])  
data.head(10)



Removing punctuation and stopwords from the messages :

import nltk  
nltk.download('stopwords')  
  
#remove the punctuations and stopwords  
import string  
def text\_process(text):  
  
text = text.translate(str.maketrans('', '', string.punctuation))  
text = [word for word in text.split() if word.lower() not in stopwords.words('english')]  
  
return " ".join(text)  
  
data['text'] = data['text'].apply(text\_process)  
data.head()



text = pd.DataFrame(data['text'])  
label = pd.DataFrame(data['label'])

Convert the text data into vectors :

we can convert words to vectors using either Count Vectorizer or by using TF-IDF Vectorizer.

TF-IDF is better than Count Vectorizers because it not only focuses on the frequency of words present in the corpus but also provides the importance of the words. We can then remove the words that are less important for analysis, hence making the model building less complex by reducing the input dimensions.

I have included both methods for your reference.