**ML ALGORITHMS IN PYTHON FOR HAM AND SPAM MESSAGE CLASSIFICATION**

**Program Code for Loading and Preprocessing Dataset:**

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

df = pd.read\_csv("/content/drive/MyDrive/Mail Spam-NLP/spam.csv", encoding="ISO-8859-1")

df.head(10)

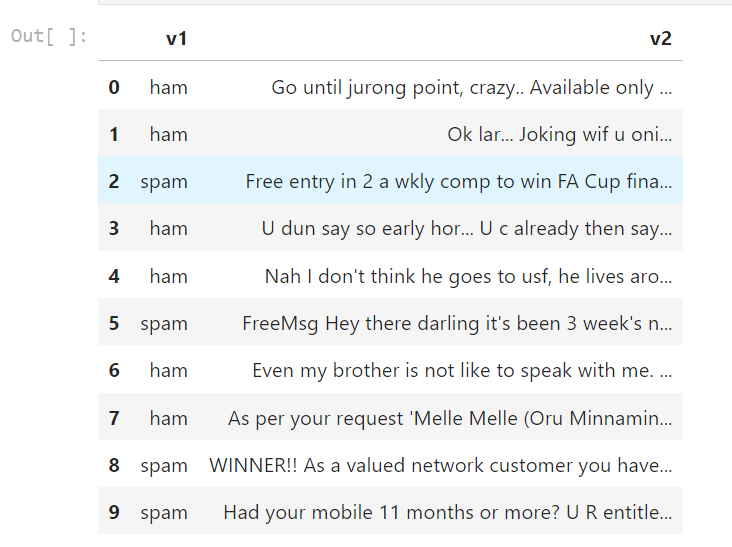


#Drop empty columns

cols = [2,3,4]

df.drop(df.columns[cols],axis=1,inplace=True)

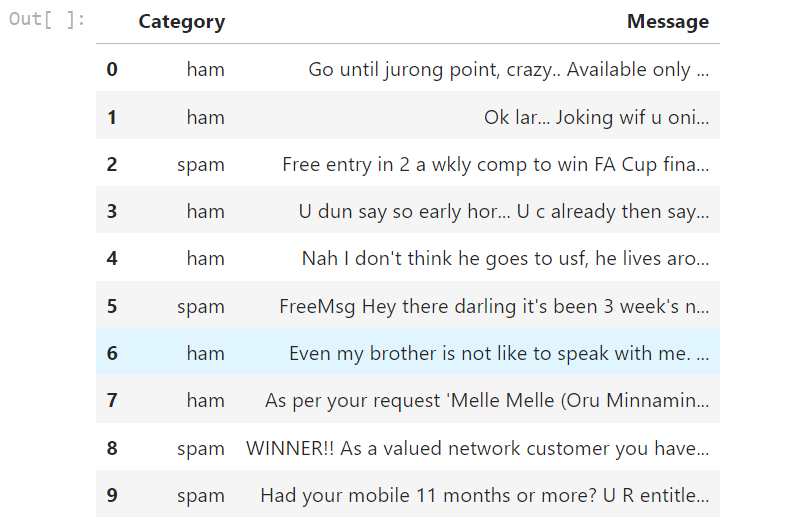
df.head(10)



#Rename columns as category and message

df.rename(columns = {'v1':'Category', 'v2':'Message'}, inplace = True)

df.head(10)



print(f'Dataset consist of {df.shape[0]} E-Mails.')

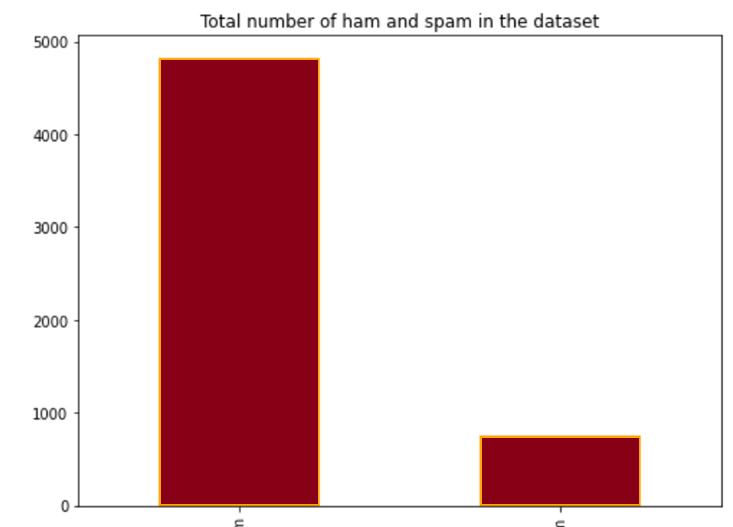
df['Category'].value\_counts()

plt.figure(figsize=(8,6))

df['Category'].value\_counts().plot.bar(color = ["orange","orange"])

plt.title('Total number of ham and spam in the dataset')

plt.show()



**Program Code for Feature Extraction and Classification:**

# WordCloud

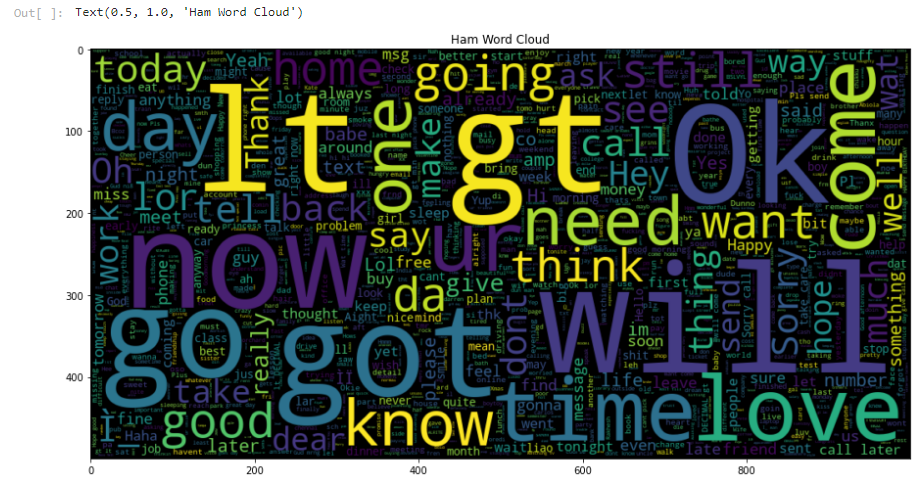
from wordcloud import WordCloud

plt.figure(figsize = (15,15))

wc = WordCloud(max\_words = 2000 , width = 1000 , height = 500).generate(" ".join(df[df.Category =="ham" ].Message))

plt.imshow(wc , interpolation = 'bilinear')

plt.title("Ham Word Cloud")



plt.figure(figsize = (15,15))

wc = WordCloud(max\_words = 2000 , width = 1000 , height = 500).generate(" ".join(df[df.Category =="spam" ].Message))

plt.imshow(wc , interpolation = 'bilinear')

plt.title("Spam Word Cloud")

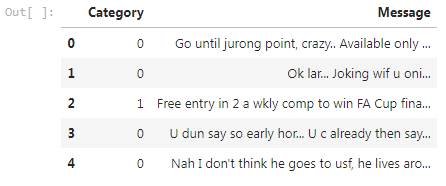


Test-Train Split

#0: Ham, 1: Spam

df['Category']=df['Category'].apply(lambda x: 1 if x=='spam' else 0)

df.head()



X=df['Message']

Y=df['Category']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,Y)

# Naive Baised Model

#Defineing Naive Baised

clf\_NaiveBaised= Pipeline([

('vectorizer', CountVectorizer()),

('nd', MultinomialNB())

])

#Fiting the algorithm

clf\_NaiveBaised.fit(X\_train,y\_train)



#Make prediction on X\_test

y\_pred\_NB=clf\_NaiveBaised.predict(X\_test)



# Random Forest Model

clf\_rf= Pipeline([

('vectorizer', CountVectorizer()),

('rf', RandomForestClassifier(n\_estimators=100))

])

clf\_rf.fit(X\_train,y\_train)



y\_pred\_RF=clf\_rf.predict(X\_test)

rf\_acc=accuracy\_score(y\_test,y\_pred\_RF)

rf\_acc

