SPAM Classifier

Importing required libraries

```
import pandas as pd
import numpy as np
import nltk
import re

nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Reading Dataset

```
df = pd.read_csv('/content/spam.csv', encoding='ISO-8859-1')
df.shape
```

```
(5572, 5)
```

Analysing Dataset

```
df
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   v1          5572 non-null   object
 1   v2          5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: object(5)
memory usage: 217.8+ KB
```

```
df.describe()
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
count	5572	5572	50	12	6
unique	2	5169	43	10	5
top	ham	Sorry, I'll call later	bt not his girlfrnd... G o o d n i g h t . . .@"	MK17 92H. 450Ppw 16"	GNT:-)"

Pre-Processing Data to create model

```
# Taking a copy of dataset

dataset = df.copy()


# Removing those columns having very less data

dataset = dataset.iloc[:,0:2]
dataset.shape
```

```
(5572, 2)
```

```
# Checking for null values
```

```
dataset.isnull().sum()
```

```
v1    0
v2    0
dtype: int64
```

```
# Seperating Independent and Dependent Columns

train_x = dataset.iloc[:,1:2]
train_y = dataset.iloc[:,0:1]
print(train_x)
print(train_y)
```

```
                                                     v2
0     Go until jurong point, crazy.. Available only ...
1                         Ok lar... Joking wif u oni...
2     Free entry in 2 a wkly comp to win FA Cup fina...
3     U dun say so early hor... U c already then say...
4     Nah I don't think he goes to usf, he lives aro...
...                                                 ...
5567  This is the 2nd time we have tried 2 contact u...
5568              Will Ì_ b going to esplanade fr home?
5569  Pity, * was in mood for that. So...any other s...
5570  The guy did some bitching but I acted like i'd...
5571                        Rofl. Its true to its name

[5572 rows x 1 columns]
        v1
0      ham
1      ham
2     spam
3      ham
4      ham
...    ...
5567  spam
5568   ham
5569   ham
5570   ham
5571   ham

[5572 rows x 1 columns]
```

Creating an Object for doing Pre-Processing

```
class SMSProcessor():

  def __init__(self,x,y):
    try:
      if len(x) == len(y):
        self.x = x
        self.y = y
        self.data = []
        self.ps = PorterStemmer()
        self.cv = CountVectorizer()
        self.re = re
```

```
            self.limit = self.x.shape[0]
        except:
          raise 'The given independent column - x  and dependent column - y   sizes ar


    def sentence_process(self,string):
        v2 = str(string)
        v2 = self.re.sub('[^a-zA-Z]',' ',v2)
        v2 = v2.lower()
        v2 = v2.split()
        v2 = [self.ps.stem(word) for word in v2 if word not in set(stopwords.words('en
        v2 = ' '.join(v2)
        return v2

    def sentence_updater(self):
        for i in range(0,self.limit):
            data = self.sentence_process(self.x.values[i])
            self.data.append(data)

    def train_process(self):
        self.x = self.cv.fit_transform(self.data).toarray()
        self.y = pd.get_dummies(self.y).drop('v1_spam', axis=1)

    def x_y_formater(self):
        self.sentence_updater()
        self.train_process()
        return self.x, self.y

    def test_process(self,string):
        string = self.sentence_process(string)
        string = self.cv.transform([string]).toarray()
        return string
```

Preprocessing Dataset

```
processor = SMSProcessor(train_x, train_y)

x_train,y_train = processor.x_y_formater()
print(x_train)
print(y_train)
```

```
     [[0 0 0 ... 0 0 0]
      [0 0 0 ... 0 0 0]
      [0 0 0 ... 0 0 0]
      ...
      [0 0 0 ... 0 0 0]
      [0 0 0 ... 0 0 0]
      [0 0 0 ... 0 0 0]]
           v1_ham
     0          1
```

```
1          1
2          0
3          1
4          1
...      ...
5567       0
5568       1
5569       1
5570       1
5571       1

[5572 rows x 1 columns]
```

Model training

Importing required libraries for model training

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

Creating Model Skeleton

```
model = Sequential()
model.add(Dense(100, activation='relu'))
model.add(Dense(150, activation='relu'))
model.add(Dense(300, activation='relu'))
model.add(Dense(500, activation='relu'))
model.add(Dense(50, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

Compiling Model to train

```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Training Model

```
model.fit(x_train,y_train,epochs=10)
```

```
Epoch 1/10
175/175 [==============================] - 4s 4ms/step - loss: 0.1615 - accur
Epoch 2/10
175/175 [==============================] - 1s 4ms/step - loss: 0.0148 - accur
Epoch 3/10
175/175 [==============================] - 1s 4ms/step - loss: 0.0040 - accur
Epoch 4/10
```

```
175/175 [==============================] - 1s 4ms/step - loss: 0.0015 - accur
Epoch 5/10
175/175 [==============================] - 1s 5ms/step - loss: 9.0278e-05 - a
Epoch 6/10
175/175 [==============================] - 1s 5ms/step - loss: 2.6569e-05 - a
Epoch 7/10
175/175 [==============================] - 1s 4ms/step - loss: 1.6063e-05 - a
Epoch 8/10
175/175 [==============================] - 1s 4ms/step - loss: 1.0974e-05 - a
Epoch 9/10
175/175 [==============================] - 1s 4ms/step - loss: 7.9298e-06 - a
Epoch 10/10
175/175 [==============================] - 1s 4ms/step - loss: 5.9530e-06 - a
<keras.callbacks.History at 0x7fad600eec10>
```

Saving Model

```
model.save('sms.h5')
```

Testing Model

```
sample_input = input('Enter the sms here : \n')
sms = processor.test_process(sample_input)
pred = model.predict(sms)
print(f'\n\nThe prodicted binary output is : {pred[0][0]}')
print(f"The SMS is {'HAM' if pred>0.5 else 'SPAM'}")
```

```
Enter the sms here :
where are you?
1/1 [==============================] - 0s 86ms/step


The prodicted binary output is : 0.9999879598617554
The SMS is HAM
```

Colab paid products - Cancel contracts here

✓ 0s completed at 11:11