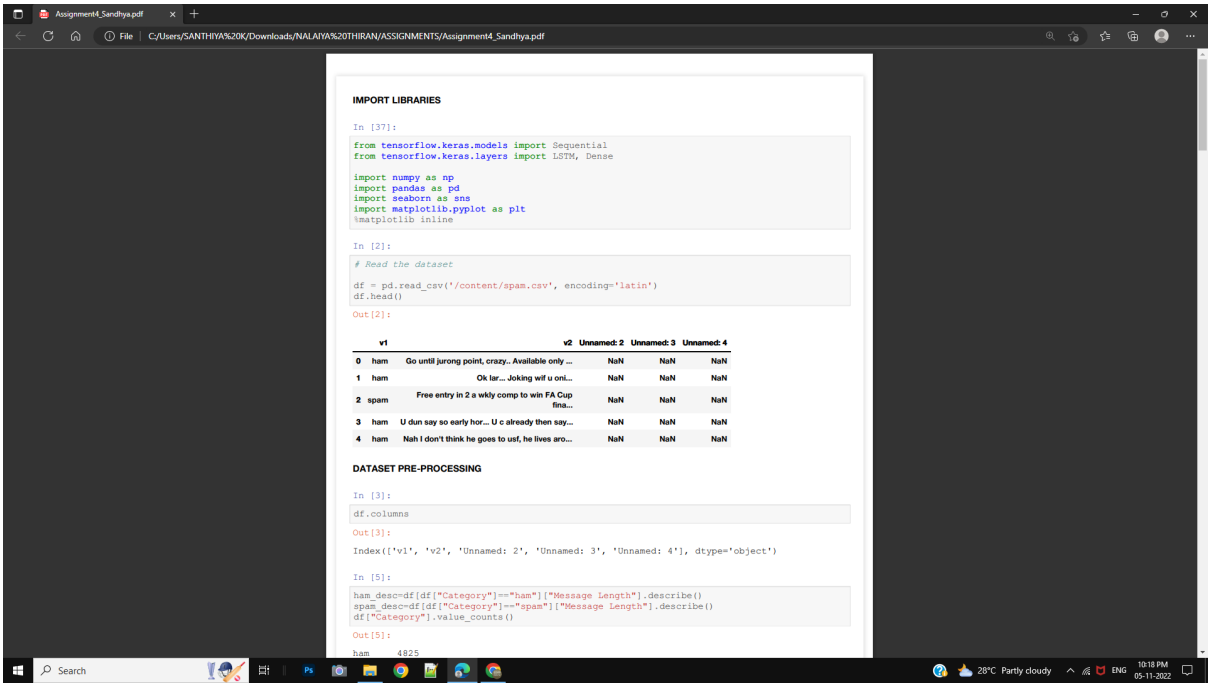
# Assignment -4

| | |
|---|---|
| Assignment Date | 5 November 2022 |
| Student Name | Sandhya K |
| Student Roll Number | 310819104073 |
| Maximum Marks | 2 Marks |

**PDF LINK:** https://drive.google.com/file/d/1pST4uVtJ6ZXmtia6QGUzQyGAX-6ykVp0/view?usp=sharing



**IMPORT LIBRARIES**

In [37]:

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

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
# Read the dataset

df = pd.read_csv('/content/spam.csv', encoding='latin')
df.head()
```

Out[2]:

| | v1 | v2 | Unnamed: 2 | Unnamed: 3 | Unnamed: 4 |
|---|---|---|---|---|---|
| 0 | ham | Go until jurong point, crazy.. Available only ... | NaN | NaN | NaN |
| 1 | ham | Ok lar... Joking wif u oni... | NaN | NaN | NaN |
| 2 | spam | Free entry in 2 a wkly comp to win FA Cup fina... | NaN | NaN | NaN |
| 3 | ham | U dun say so early hor... U c already then say... | NaN | NaN | NaN |
| 4 | ham | Nah I don't think he goes to usf, he lives aro... | NaN | NaN | NaN |

**DATASET PRE-PROCESSING**

In [3]:

```
df.columns
```

Out[3]:

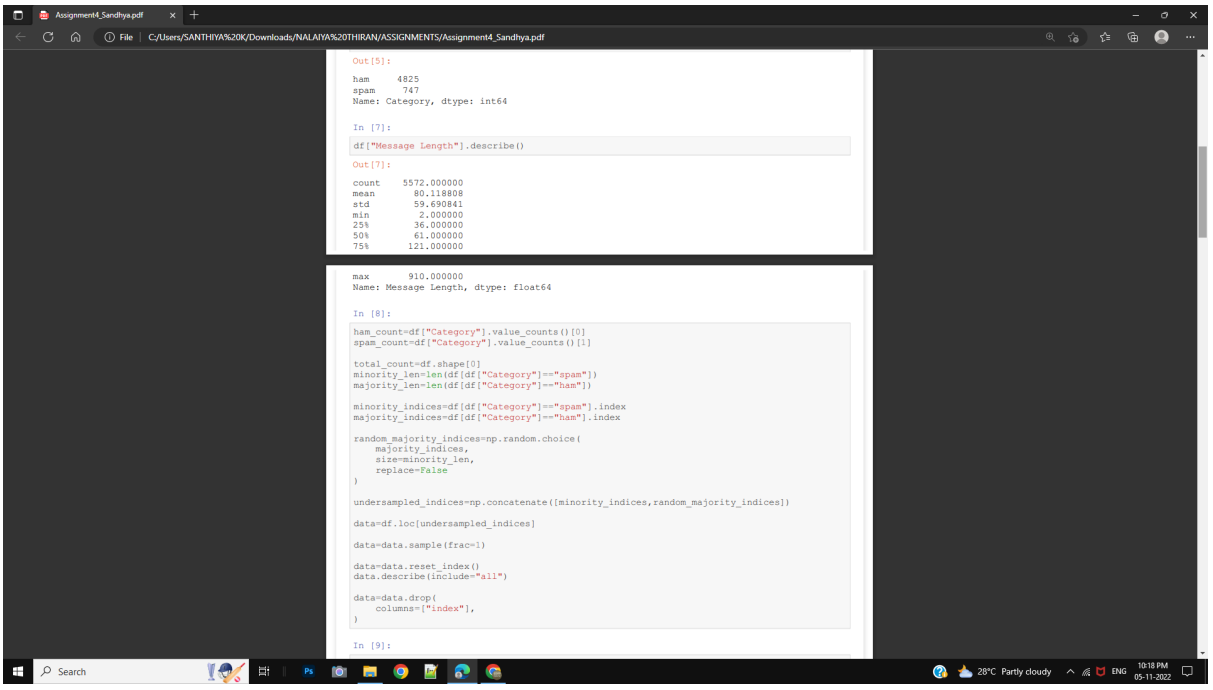
```
Index(['v1', 'v2', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], dtype='object')
```

In [5]:

```
ham_desc=df[df["Category"]=="ham"]["Message Length"].describe()
spam_desc=df[df["Category"]=="spam"]["Message Length"].describe()
df["Category"].value_counts()
```

Out[5]:

```
ham     4825
```



```
spam     747
Name: Category, dtype: int64
```

In [7]:

```
df["Message Length"].describe()
```

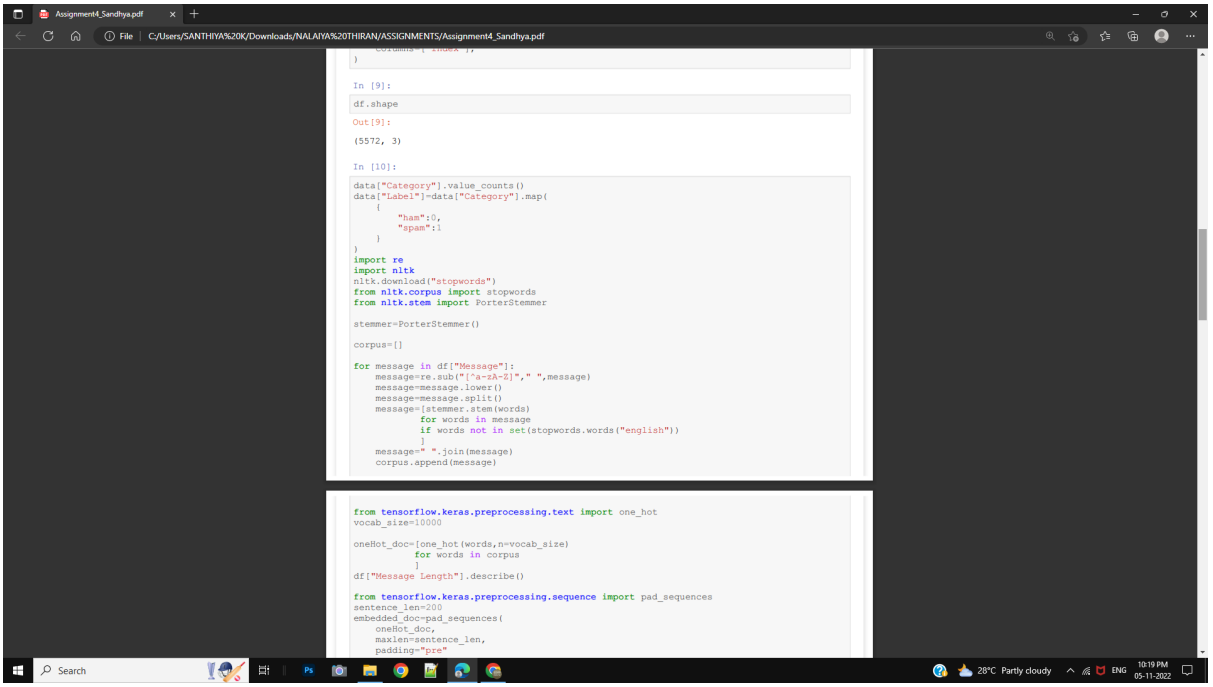
Out[7]:

```
count    5572.000000
mean       80.118808
std        59.690841
min         2.000000
25%        36.000000
50%        61.000000
75%       121.000000
max       910.000000
Name: Message Length, dtype: float64
```

In [8]:

```
ham_count=df["Category"].value_counts()[0]
spam_count=df["Category"].value_counts()[1]

total_count=df.shape[0]
minority_len=len(df[df["Category"]=="spam"])
majority_len=len(df[df["Category"]=="ham"])

minority_indices=df[df["Category"]=="spam"].index
majority_indices=df[df["Category"]=="ham"].index

random_majority_indices=np.random.choice(
    majority_indices,
    size=minority_len,
    replace=False
)

undersampled_indices=np.concatenate([minority_indices,random_majority_indices])

data=df.loc[undersampled_indices]

data=data.sample(frac=1)

data=data.reset_index()
data.describe(include="all")

data=data.drop(
    columns=["index"],
)
```

In [9]:



```
)
```

In [9]:

```
df.shape
```

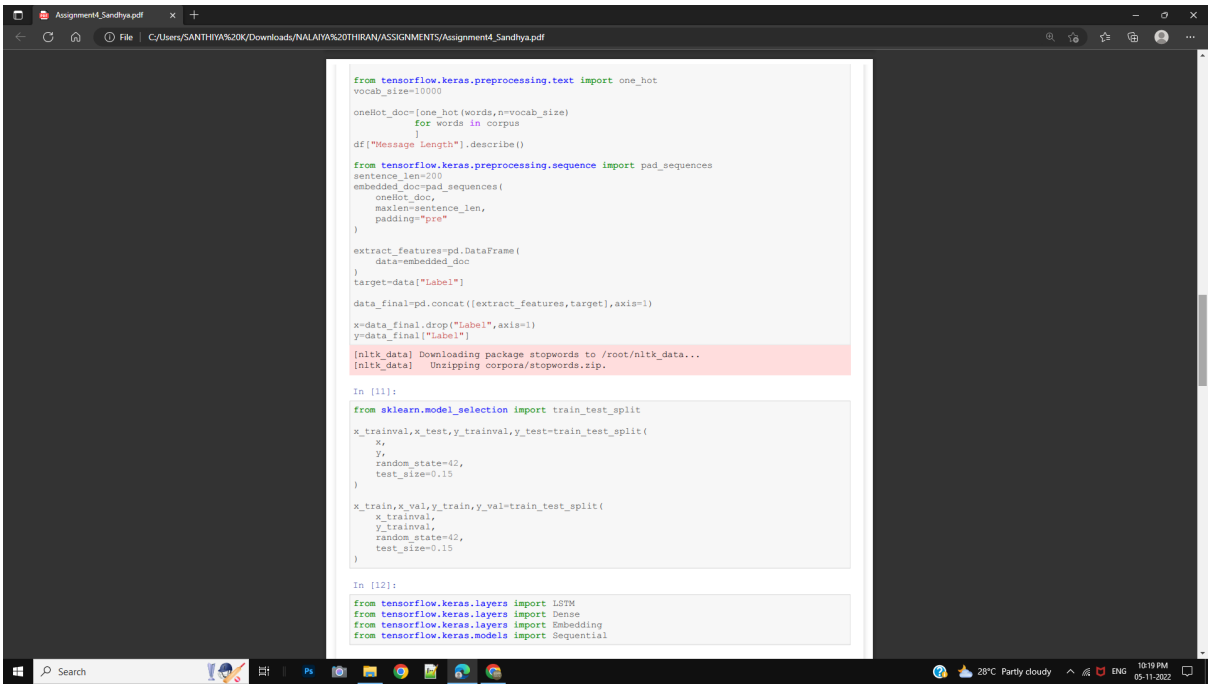
Out[9]:

```
(5572, 3)
```

In [10]:

```
data["Category"].value_counts()
data["Label"]=data["Category"].map(
    {
        "ham":0,
        "spam":1
    }
)
import re
import nltk
nltk.download("stopwords")
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

stemmer=PorterStemmer()

corpus=[]

for message in df["Message"]:
    message=re.sub("[^a-zA-Z]"," ",message)
    message=message.lower()
    message=message.split()
    message=[stemmer.stem(words)
             for words in message
             if words not in set(stopwords.words("english"))
             ]
    message=" ".join(message)
    corpus.append(message)
```

```
from tensorflow.keras.preprocessing.text import one_hot
vocab_size=10000

oneHot_doc=[one_hot(words,n=vocab_size)
            for words in corpus
            ]
df["Message Length"].describe()

from tensorflow.keras.preprocessing.sequence import pad_sequences
sentence_len=200
embedded_doc=pad_sequences(
    oneHot_doc,
    maxlen=sentence_len,
    padding="pre"
```

```
from tensorflow.keras.preprocessing.text import one_hot
vocab_size=10000

oneHot_doc=[one_hot(words,n=vocab_size)
            for words in corpus
            ]
df["Message Length"].describe()

from tensorflow.keras.preprocessing.sequence import pad_sequences
sentence_len=200
embedded_doc=pad_sequences(
    oneHot_doc,
    maxlen=sentence_len,
    padding="pre"
)

extract_features=pd.DataFrame(
    data=embedded_doc
)
target=data["Label"]

data_final=pd.concat([extract_features,target],axis=1)

x=data_final.drop("Label",axis=1)
y=data_final["Label"]
```

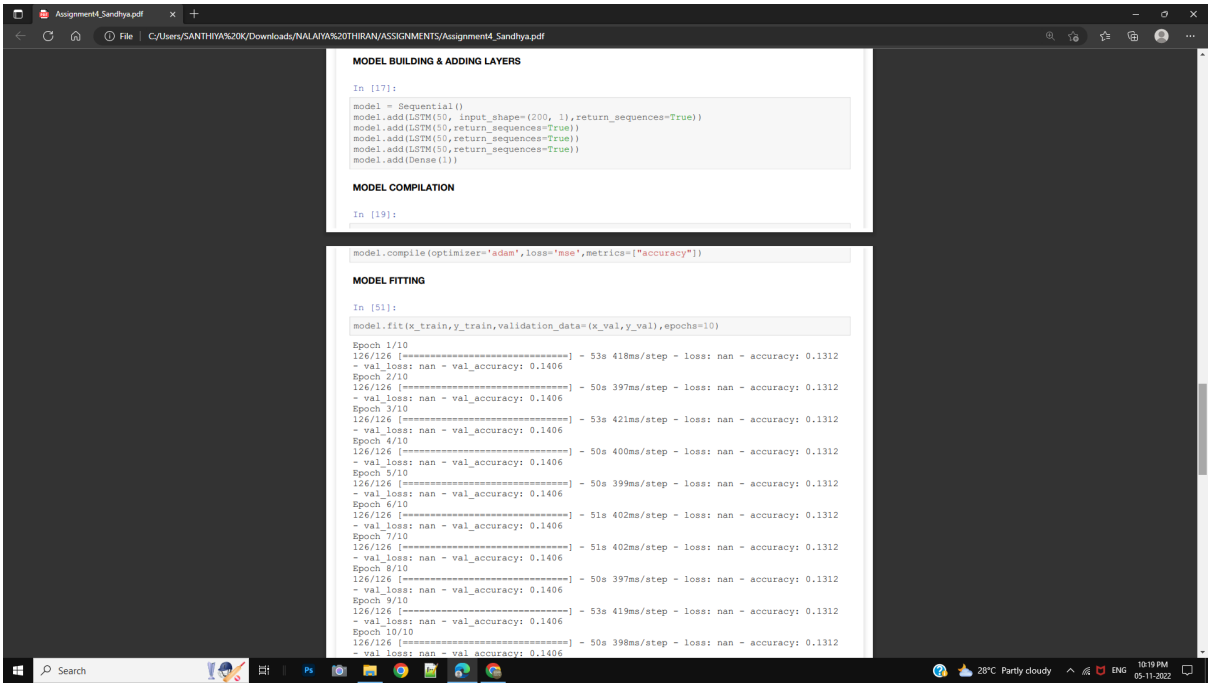
```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

In [11]:

```
from sklearn.model_selection import train_test_split

x_trainval,x_test,y_trainval,y_test=train_test_split(
    x,
    y,
    random_state=42,
    test_size=0.15
)

x_train,x_val,y_train,y_val=train_test_split(
    x_trainval,
    y_trainval,
    random_state=42,
    test_size=0.15
)
```

In [12]:

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



Assignment4_Sandhya.pdf

C:/Users/SANTHIYA%20K/Downloads/NALAIYA%20THIRAN/ASSIGNMENTS/Assignment4_Sandhya.pdf

**MODEL BUILDING & ADDING LAYERS**

In [17]:

```
model = Sequential()
model.add(LSTM(50, input_shape=(200, 1),return_sequences=True))
model.add(LSTM(50,return_sequences=True))
model.add(LSTM(50,return_sequences=True))
model.add(LSTM(50,return_sequences=True))
model.add(Dense(1))
```

**MODEL COMPILATION**

In [19]:

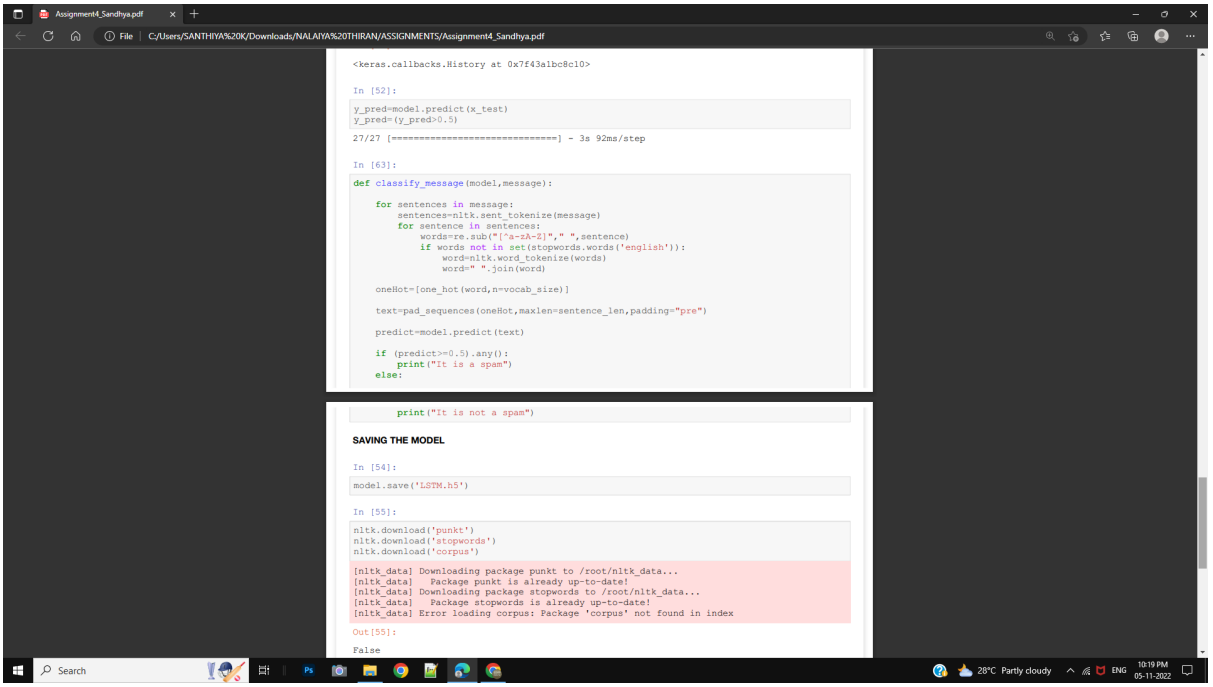
```
model.compile(optimizer='adam',loss='mse',metrics=["accuracy"])
```

**MODEL FITTING**

In [51]:

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

```
Epoch 1/10
126/126 [==============================] - 53s 418ms/step - loss: nan - accuracy: 0.1312
- val_loss: nan - val_accuracy: 0.1406
Epoch 2/10
126/126 [==============================] - 50s 397ms/step - loss: nan - accuracy: 0.1312
- val_loss: nan - val_accuracy: 0.1406
Epoch 3/10
126/126 [==============================] - 53s 421ms/step - loss: nan - accuracy: 0.1312
- val_loss: nan - val_accuracy: 0.1406
Epoch 4/10
126/126 [==============================] - 50s 400ms/step - loss: nan - accuracy: 0.1312
- val_loss: nan - val_accuracy: 0.1406
Epoch 5/10
126/126 [==============================] - 50s 399ms/step - loss: nan - accuracy: 0.1312
- val_loss: nan - val_accuracy: 0.1406
Epoch 6/10
126/126 [==============================] - 51s 402ms/step - loss: nan - accuracy: 0.1312
- val_loss: nan - val_accuracy: 0.1406
Epoch 7/10
126/126 [==============================] - 51s 402ms/step - loss: nan - accuracy: 0.1312
- val_loss: nan - val_accuracy: 0.1406
Epoch 8/10
126/126 [==============================] - 50s 397ms/step - loss: nan - accuracy: 0.1312
- val_loss: nan - val_accuracy: 0.1406
Epoch 9/10
126/126 [==============================] - 53s 419ms/step - loss: nan - accuracy: 0.1312
- val_loss: nan - val_accuracy: 0.1406
Epoch 10/10
126/126 [==============================] - 50s 398ms/step - loss: nan - accuracy: 0.1312
- val_loss: nan - val_accuracy: 0.1406
```



```
<keras.callbacks.History at 0x7f43a1bc8c10>
```

In [52]:

```
y_pred=model.predict(x_test)
y_pred=(y_pred>0.5)
```

```
27/27 [==============================] - 3s 92ms/step
```

In [63]:

```
def classify_message(model,message):

    for sentences in message:
        sentences=nltk.sent_tokenize(message)
        for sentence in sentences:
            words=re.sub("[^a-zA-Z]"," ",sentence)
            if words not in set(stopwords.words('english')):
                word=nltk.word_tokenize(words)
                word=" ".join(word)

    oneHot=[one_hot(word,n=vocab_size)]

    text=pad_sequences(oneHot,maxlen=sentence_len,padding="pre")

    predict=model.predict(text)

    if (predict>=0.5).any():
        print("It is a spam")
    else:
        print("It is not a spam")
```

**SAVING THE MODEL**

In [54]:

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

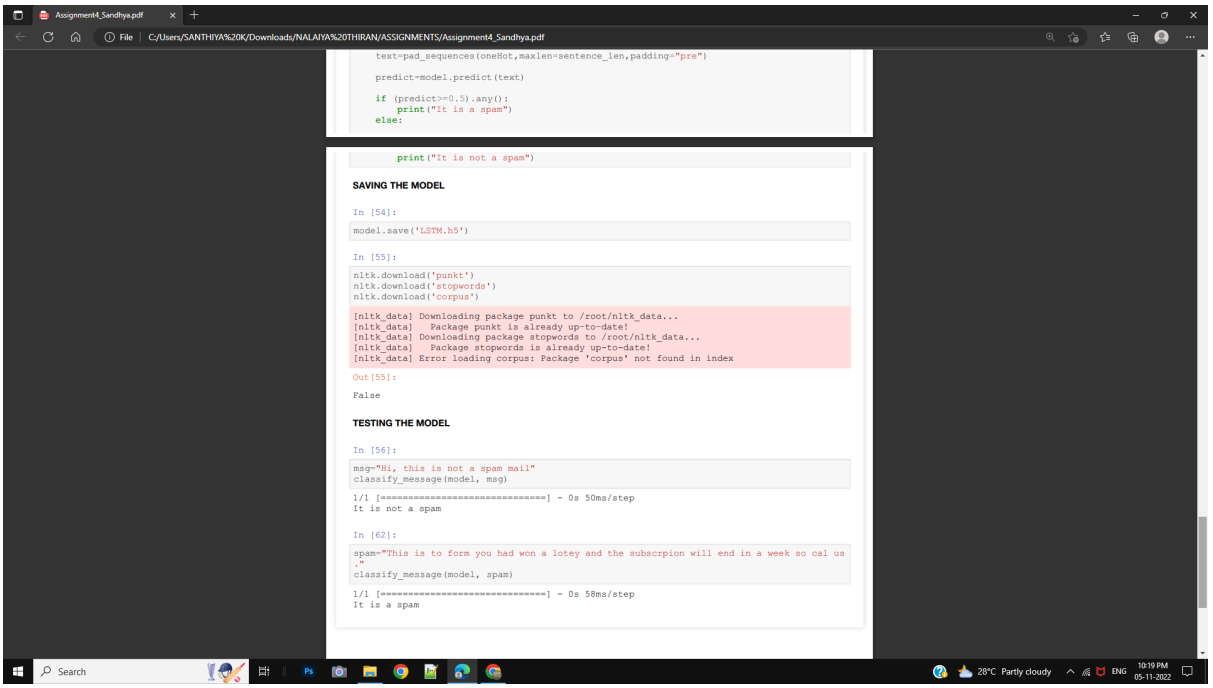
In [55]:

```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('corpus')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Error loading corpus: Package 'corpus' not found in index
```

Out[55]:

```
False
```




```
    text=pad_sequences(oneHot,maxlen=sentence_len,padding="pre")

    predict=model.predict(text)

    if (predict>=0.5).any():
        print("It is a spam")
    else:
        print("It is not a spam")
```

**SAVING THE MODEL**

In [54]:

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

In [55]:

```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('corpus')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Error loading corpus: Package 'corpus' not found in index
```

Out[55]:

```
False
```


**TESTING THE MODEL**

In [56]:

```
msg="Hi, this is not a spam mail"
classify_message(model, msg)
```

```
1/1 [==============================] - 0s 50ms/step
It is not a spam
```

In [62]:

```
spam="This is to form you had won a lotey and the subscrpion will end in a week so cal us
."
classify_message(model, spam)
```

```
1/1 [==============================] - 0s 58ms/step
It is a spam
```