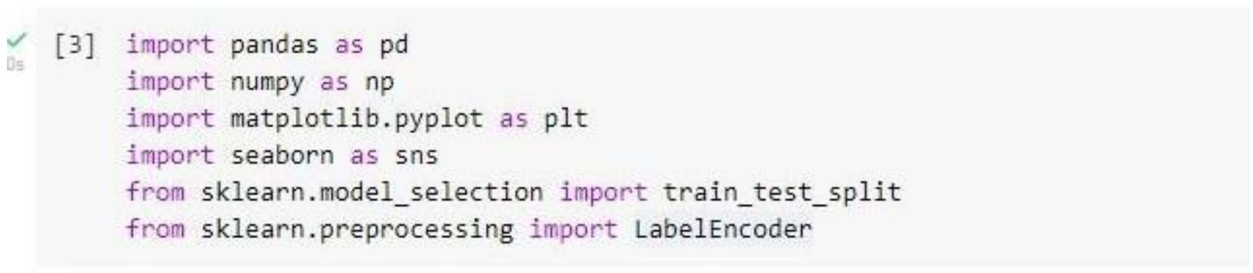
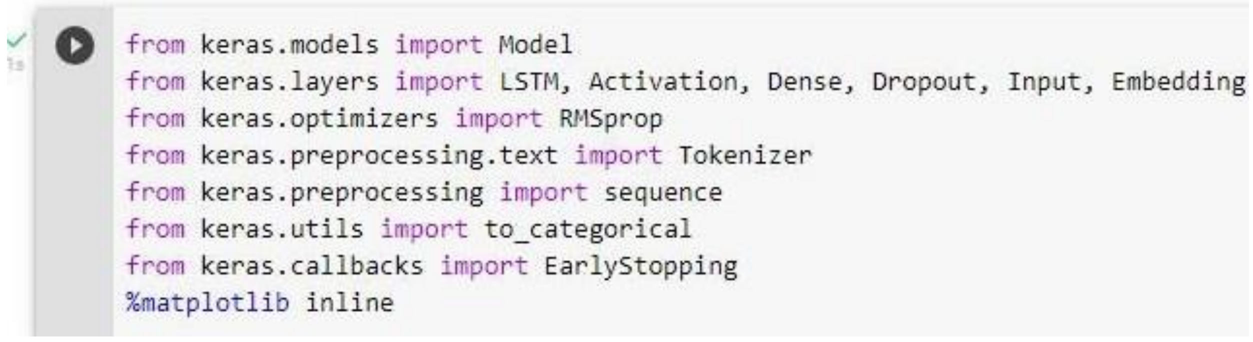
SMS SPAM CLASSIFICATION

Import the necessary libraries



```
[3] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import LabelEncoder
```

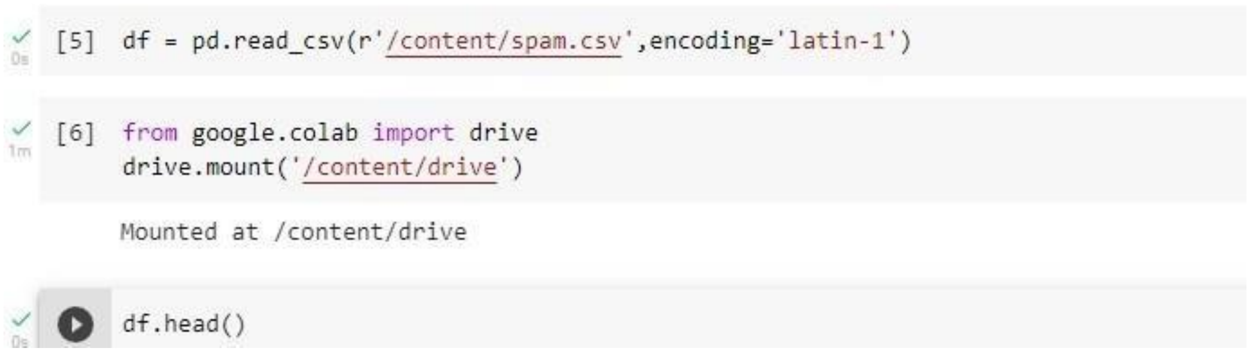


```
from keras.models import Model
from keras.layers import LSTM, Activation, Dense, Dropout, Input, Embedding
from keras.optimizers import RMSprop
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence
from keras.utils import to_categorical
from keras.callbacks import EarlyStopping
%matplotlib inline
```

Download the Dataset

Dataset Downloaded and uploaded to drive https://www.kaggle.com/code/kredy10/simple-lstmfor-textclassification/data

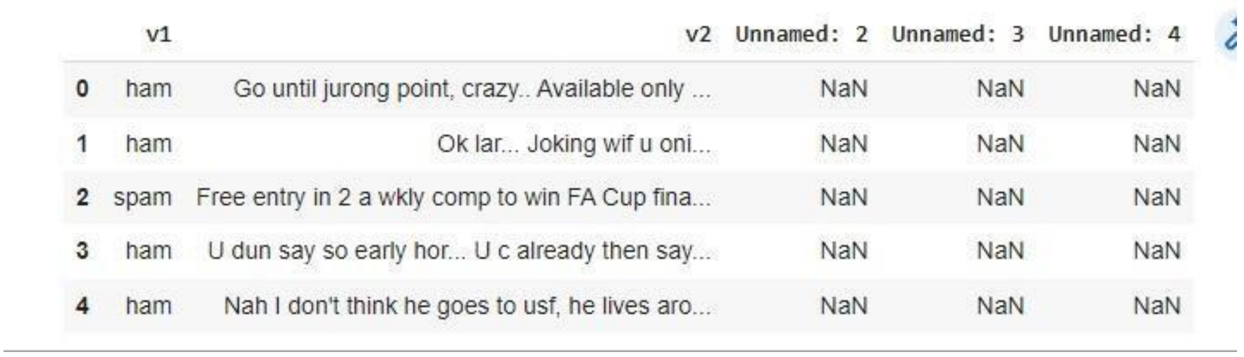
Read dataset and pre-processing



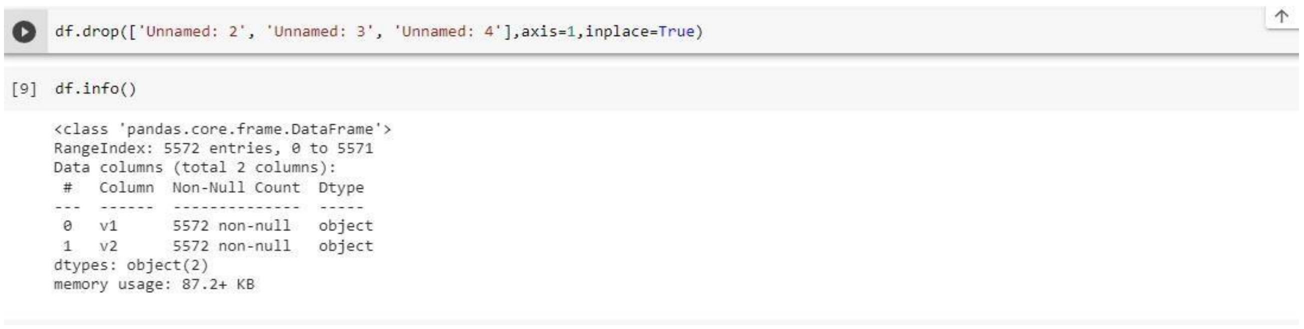
```
[5]  df = pd.read_csv(r'/content/spam.csv',encoding='latin-1')
```

```
[6]  from google.colab import drive
     drive.mount('/content/drive')

     Mounted at /content/drive
```

```
df.head()
```

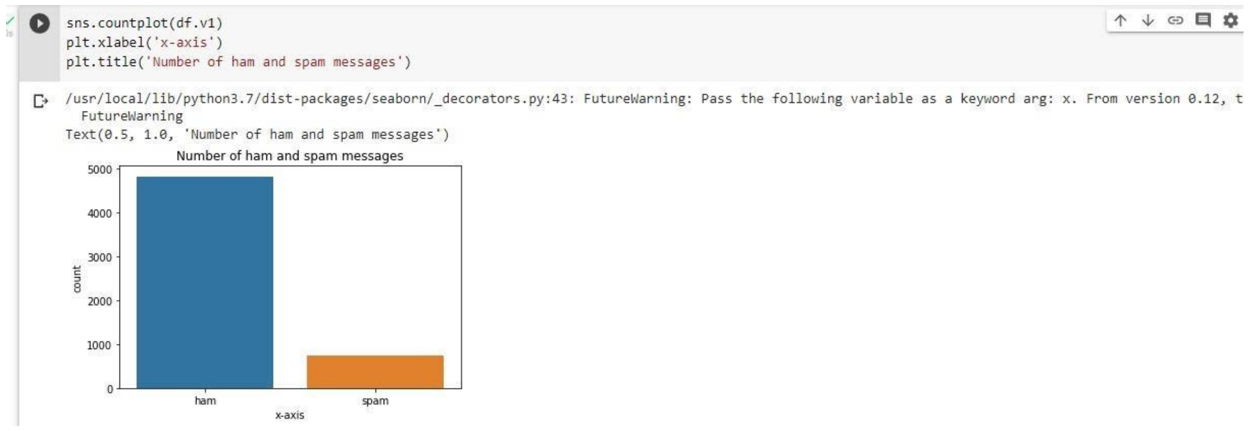


	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah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



```
df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True)
```

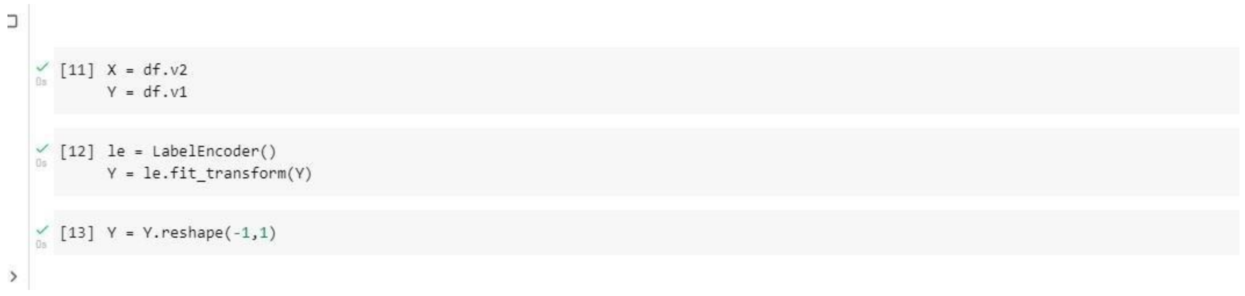
```
[9]  df.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 5572 entries, 0 to 5571
     Data columns (total 2 columns):
      #   Column  Non-Null Count  Dtype 
     ---  ------  --------------  ----- 
      0   v1      5572 non-null   object
      1   v2      5572 non-null   object
     dtypes: object(2)
     memory usage: 87.2+ KB
```



```
sns.countplot(df.v1)
plt.xlabel('x-axis')
plt.title('Number of ham and spam messages')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, t
  FutureWarning
Text(0.5, 1.0, 'Number of ham and spam messages')
```

Create input vectors and process labels

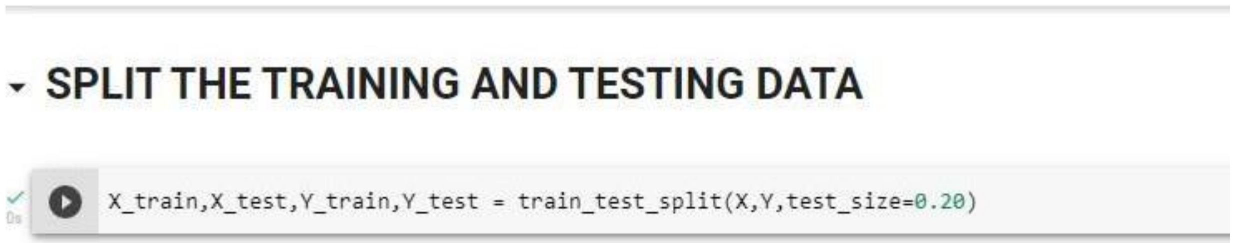


```
[11] X = df.v2
     Y = df.v1
```

```
[12] le = LabelEncoder()
     Y = le.fit_transform(Y)
```

```
[13] Y = Y.reshape(-1,1)
```

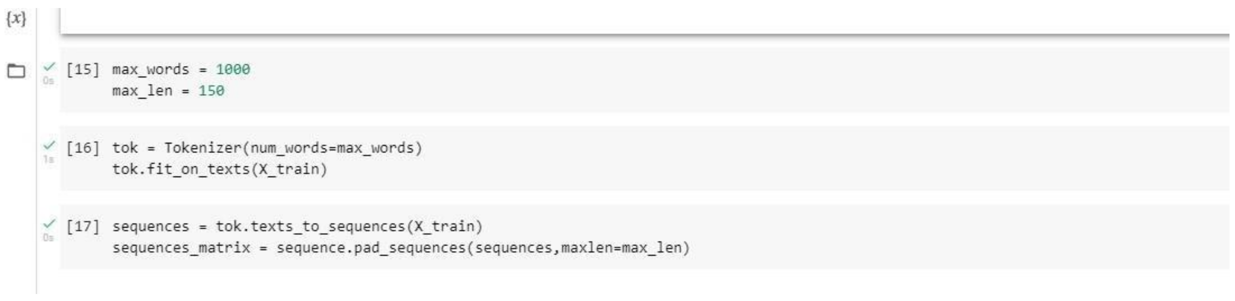
Split the training and testing data



SPLIT THE TRAINING AND TESTING DATA

```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.20)
```

Process the data

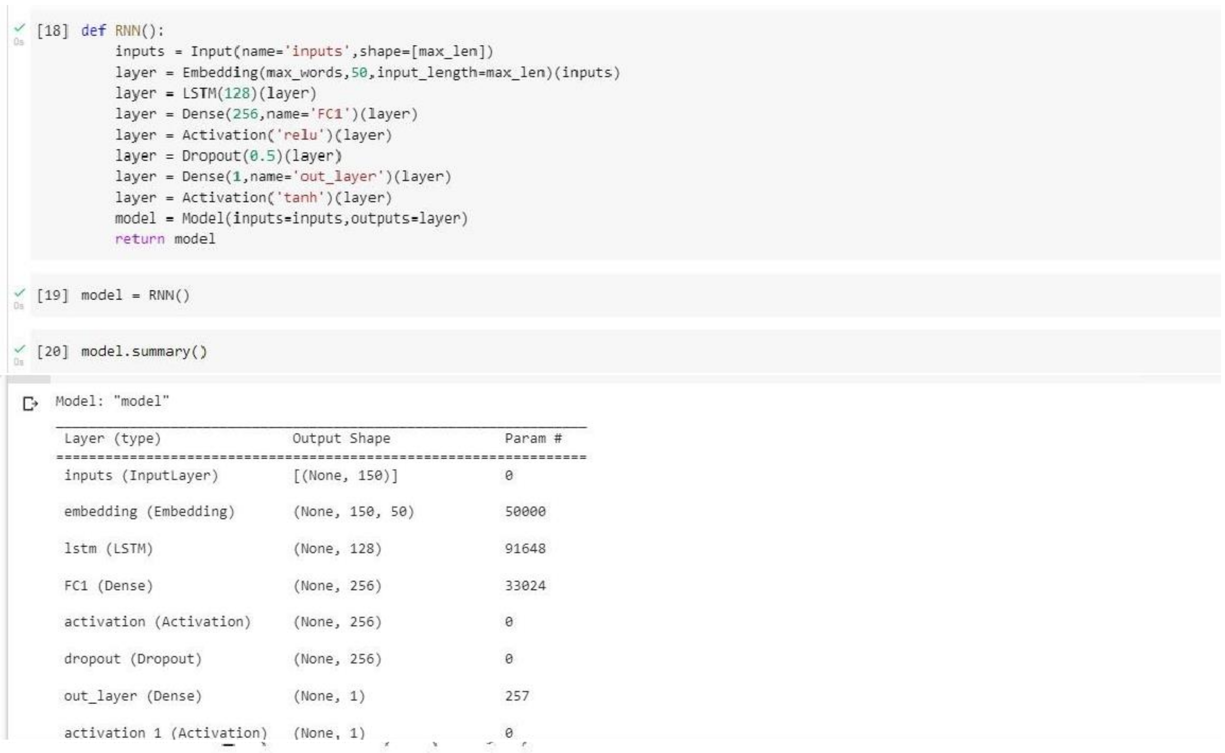


```
[15] max_words = 1000
     max_len = 150
```

```
[16] tok = Tokenizer(num_words=max_words)
     tok.fit_on_texts(X_train)
```

```
[17] sequences = tok.texts_to_sequences(X_train)
     sequences_matrix = sequence.pad_sequences(sequences,maxlen=max_len)
```

Create models and add layers

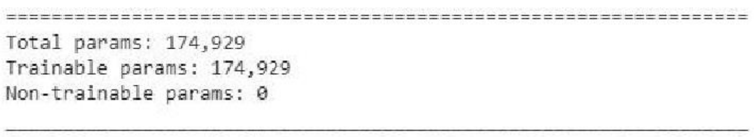


```
[18] def RNN():
         inputs = Input(name='inputs',shape=[max_len])
         layer = Embedding(max_words,50,input_length=max_len)(inputs)
         layer = LSTM(128)(layer)
         layer = Dense(256,name='FC1')(layer)
         layer = Activation('relu')(layer)
         layer = Dropout(0.5)(layer)
         layer = Dense(1,name='out_layer')(layer)
         layer = Activation('tanh')(layer)
         model = Model(inputs=inputs,outputs=layer)
         return model
```

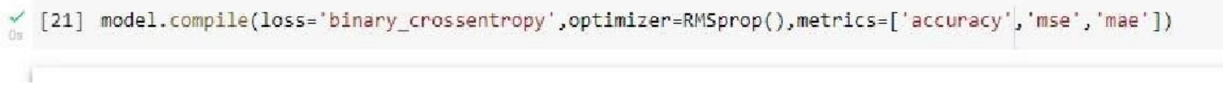
```
[19] model = RNN()
```

```
[20] model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inputs (InputLayer)         [(None, 150)]             0

 embedding (Embedding)       (None, 150, 50)           50000

 lstm (LSTM)                 (None, 128)               91648

 FC1 (Dense)                 (None, 256)               33024

 activation (Activation)     (None, 256)               0

 dropout (Dropout)           (None, 256)               0

 out_layer (Dense)           (None, 1)                 257

 activation_1 (Activation)   (None, 1)                 0
```

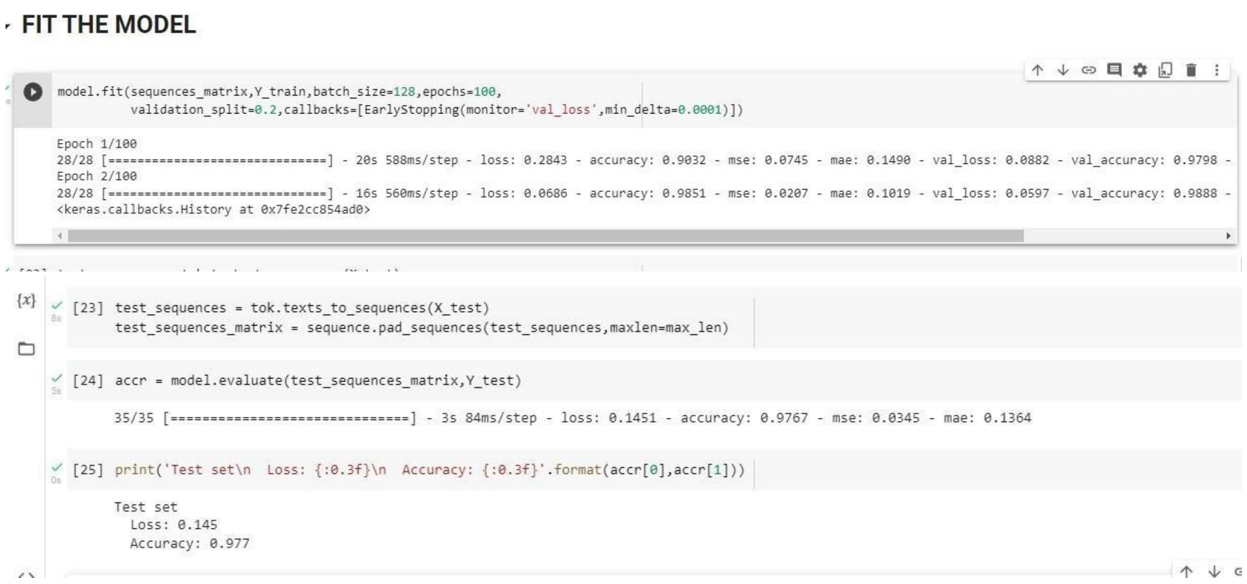


```
=================================================================
Total params: 174,929
Trainable params: 174,929
Non-trainable params: 0
_________________________________________________________________
```



```
[21] model.compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=['accuracy','mse','mae'])
```

FIT THE MODEL

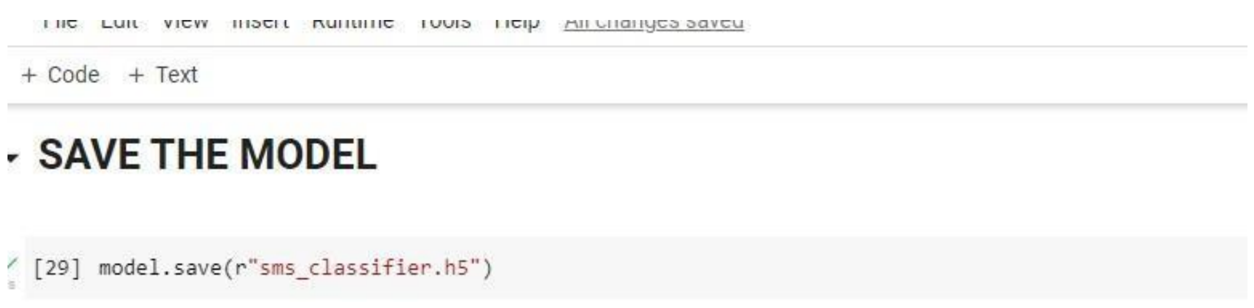


FIT THE MODEL

```
model.fit(sequences_matrix,Y_train,batch_size=128,epochs=100,
          validation_split=0.2,callbacks=[EarlyStopping(monitor='val_loss',min_delta=0.0001)])

Epoch 1/100
28/28 [==============================] - 20s 588ms/step - loss: 0.2843 - accuracy: 0.9032 - mse: 0.0745 - mae: 0.1490 - val_loss: 0.0882 - val_accuracy: 0.9798 -
Epoch 2/100
28/28 [==============================] - 16s 560ms/step - loss: 0.0686 - accuracy: 0.9851 - mse: 0.0207 - mae: 0.1019 - val_loss: 0.0597 - val_accuracy: 0.9888 -
<keras.callbacks.History at 0x7fe2cc854ad0>
```

```
[23] test_sequences = tok.texts_to_sequences(X_test)
     test_sequences_matrix = sequence.pad_sequences(test_sequences,maxlen=max_len)

[24] accr = model.evaluate(test_sequences_matrix,Y_test)

     35/35 [==============================] - 3s 84ms/step - loss: 0.1451 - accuracy: 0.9767 - mse: 0.0345 - mae: 0.1364

[25] print('Test set\n  Loss: {:0.3f}\n  Accuracy: {:0.3f}'.format(accr[0],accr[1]))

     Test set
       Loss: 0.145
       Accuracy: 0.977
```

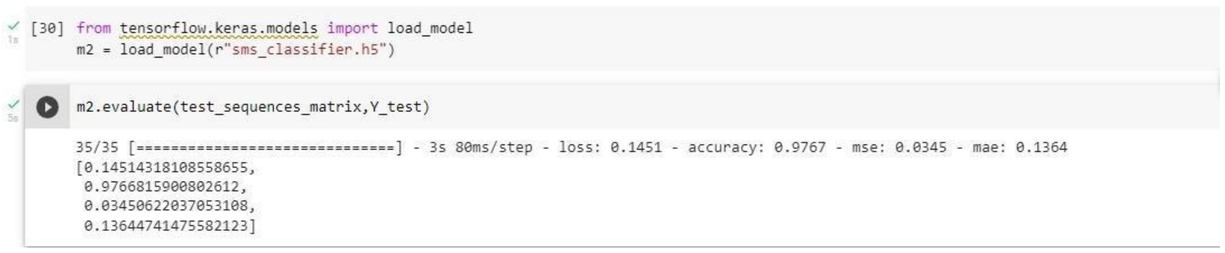
SAVE THE MODEL



SAVE THE MODEL

```
[29] model.save(r"sms_classifier.h5")
```

TEST THE MODEL



```
[30] from tensorflow.keras.models import load_model
     m2 = load_model(r"sms_classifier.h5")

m2.evaluate(test_sequences_matrix,Y_test)

35/35 [==============================] - 3s 80ms/step - loss: 0.1451 - accuracy: 0.9767 - mse: 0.0345 - mae: 0.1364
[0.14514318108558655,
 0.9766815900802612,
 0.03450622037053108,
 0.13644741475582123]
```