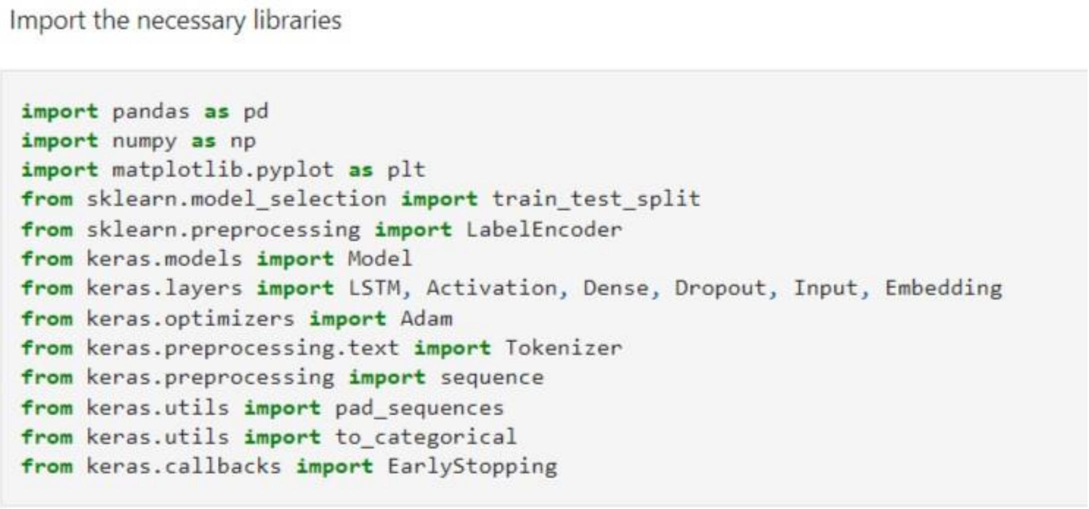
Assignment -4

SMS SPAM Classification

Assignment Date	08 October 2022
Student Name	Ashwini K
Student Roll Number	612919103008

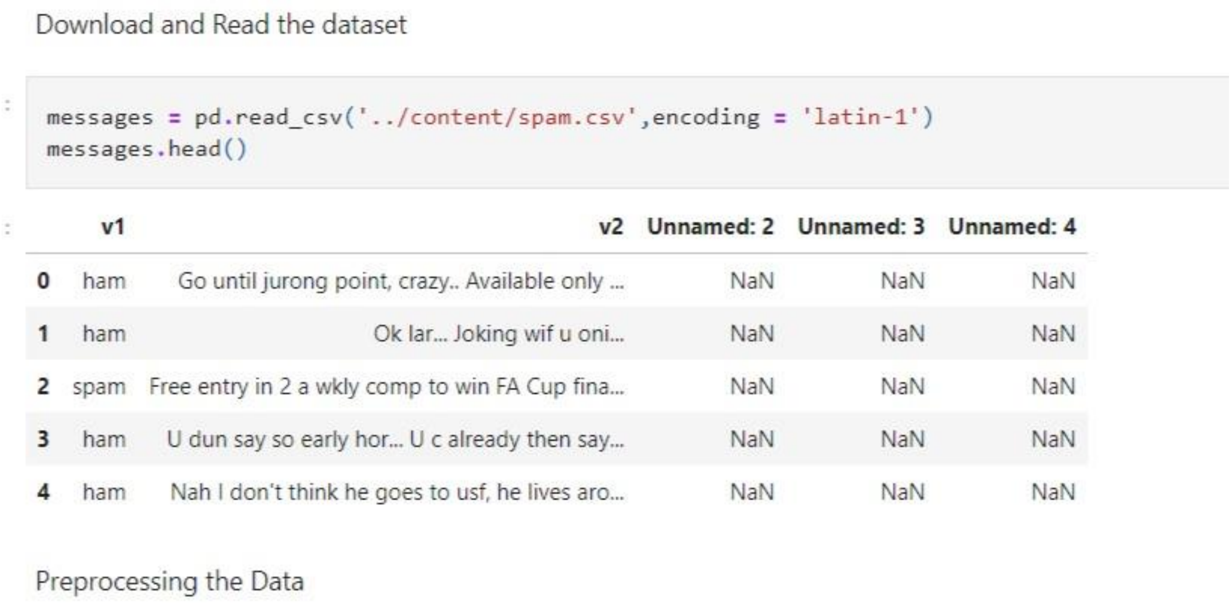
- **Download the Dataset: - Dataset**
- **Import the necessary libraries**



Import the necessary libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
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 Adam
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence
from keras.utils import pad_sequences
from keras.utils import to_categorical
from keras.callbacks import EarlyStopping
```

- **Read dataset and do pre-processing**



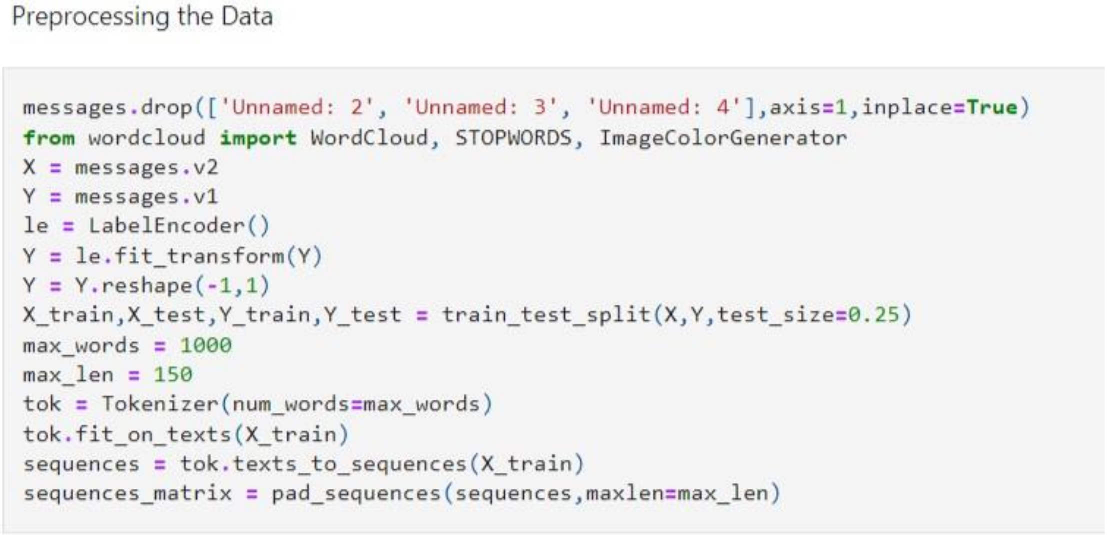
Download and Read the dataset

```
messages = pd.read_csv('../content/spam.csv',encoding = 'latin-1')
messages.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

Preprocessing the Data

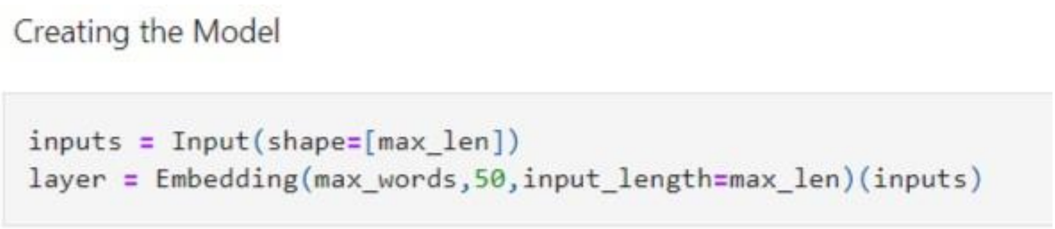
- **Pre-processing the dataset**



Preprocessing the Data

```
messages.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True)
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
X = messages.v2
Y = messages.v1
le = LabelEncoder()
Y = le.fit_transform(Y)
Y = Y.reshape(-1,1)
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.25)
max_words = 1000
max_len = 150
tok = Tokenizer(num_words=max_words)
tok.fit_on_texts(X_train)
sequences = tok.texts_to_sequences(X_train)
sequences_matrix = pad_sequences(sequences,maxlen=max_len)
```

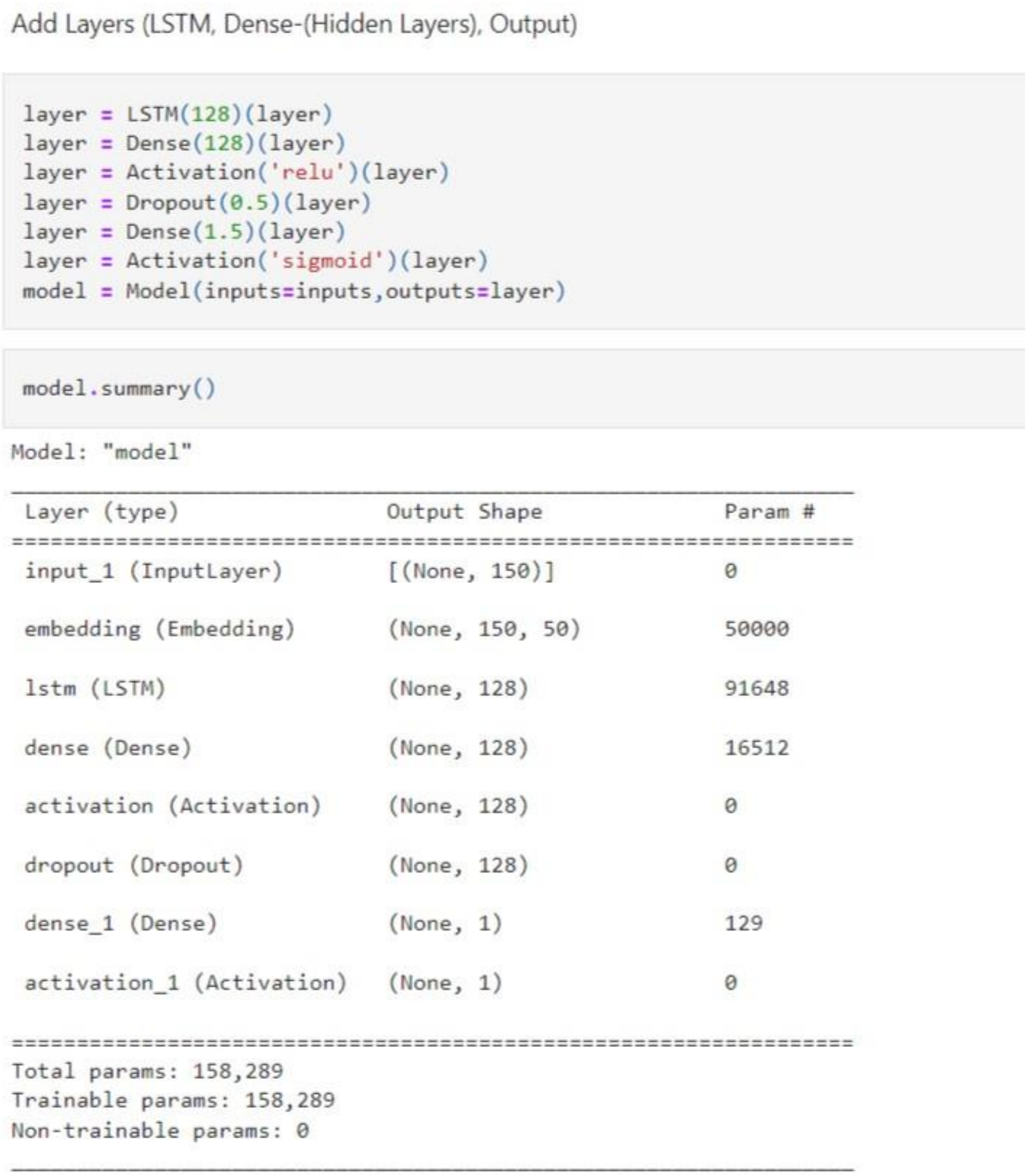
- **Create Model**



Creating the Model

```
inputs = Input(shape=[max_len])
layer = Embedding(max_words,50,input_length=max_len)(inputs)
```

- **Add Layers (LSTM, Dense-(Hidden Layers), Output)**



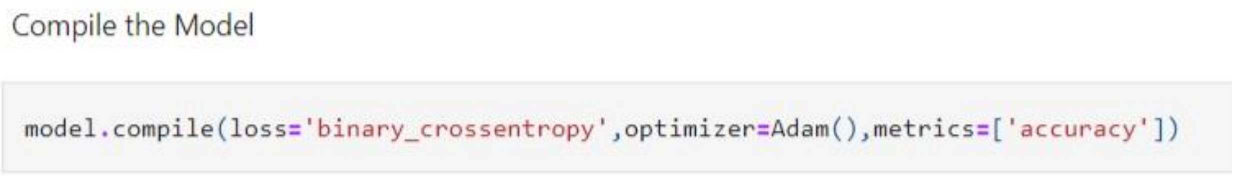
Add Layers (LSTM, Dense-(Hidden Layers), Output)

```
layer = LSTM(128)(layer)
layer = Dense(128)(layer)
layer = Activation('relu')(layer)
layer = Dropout(0.5)(layer)
layer = Dense(1.5)(layer)
layer = Activation('sigmoid')(layer)
model = Model(inputs=inputs,outputs=layer)
```

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 150)]             0

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

 lstm (LSTM)                 (None, 128)               91648

 dense (Dense)               (None, 128)               16512

 activation (Activation)     (None, 128)               0

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 1)                 129

 activation_1 (Activation)   (None, 1)                 0

=================================================================
Total params: 158,289
Trainable params: 158,289
Non-trainable params: 0
_________________________________________________________________
```

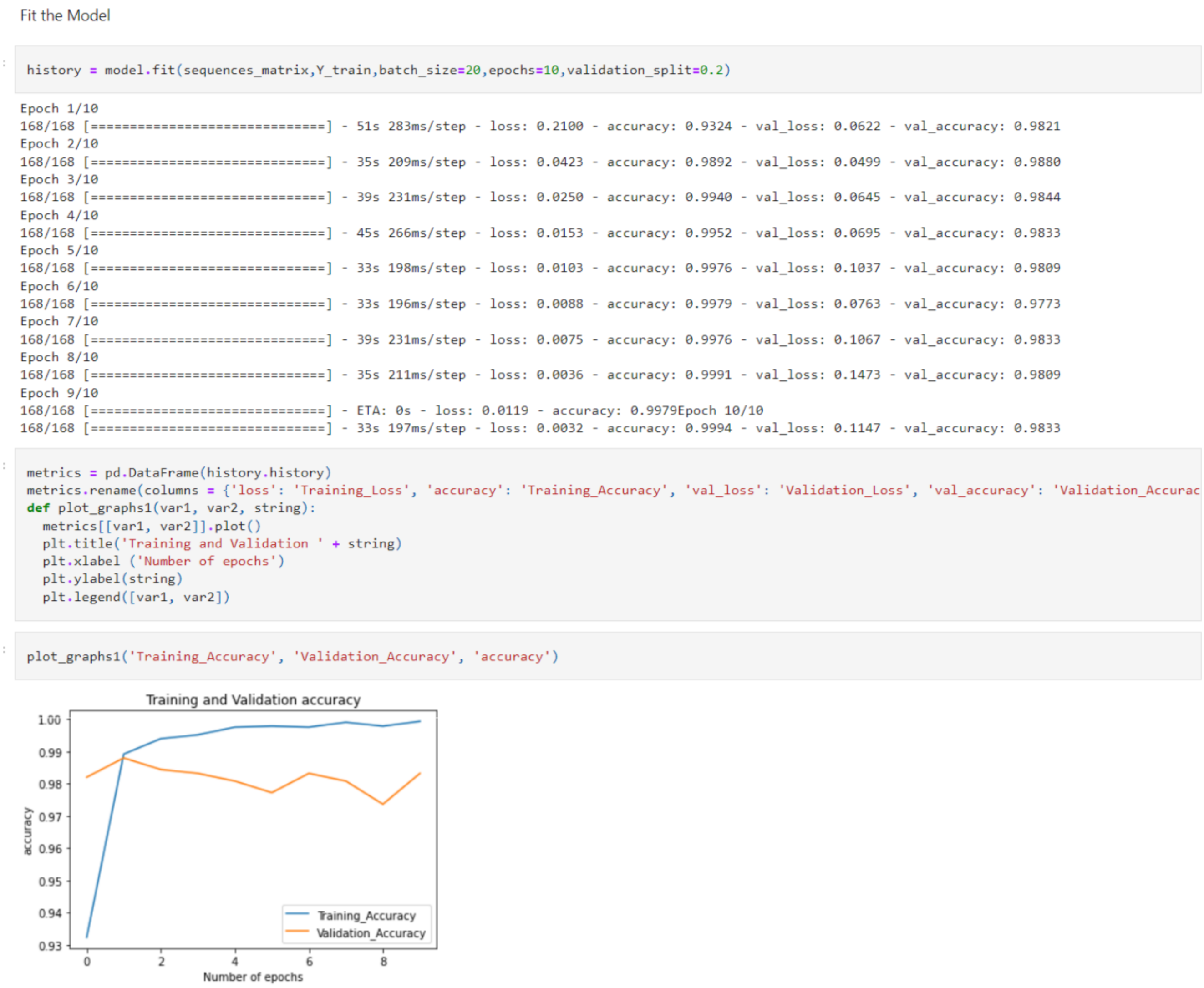
- **Compile the Model**



Compile the Model

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

- **Fit the Model**



Fit the Model

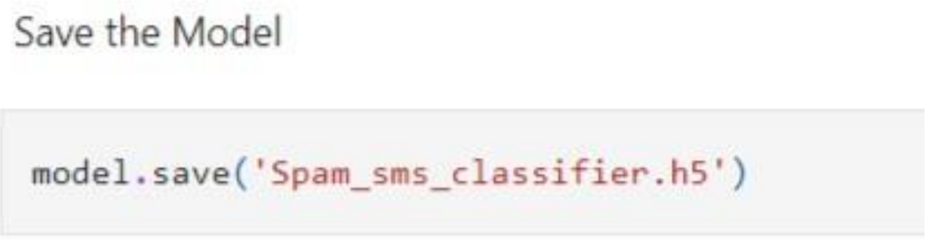
```
history = model.fit(sequences_matrix,Y_train,batch_size=20,epochs=10,validation_split=0.2)
```

```
Epoch 1/10
168/168 [==============================] - 51s 283ms/step - loss: 0.2100 - accuracy: 0.9324 - val_loss: 0.0622 - val_accuracy: 0.9821
Epoch 2/10
168/168 [==============================] - 35s 209ms/step - loss: 0.0423 - accuracy: 0.9892 - val_loss: 0.0499 - val_accuracy: 0.9880
Epoch 3/10
168/168 [==============================] - 39s 231ms/step - loss: 0.0250 - accuracy: 0.9940 - val_loss: 0.0645 - val_accuracy: 0.9844
Epoch 4/10
168/168 [==============================] - 45s 266ms/step - loss: 0.0153 - accuracy: 0.9952 - val_loss: 0.0695 - val_accuracy: 0.9833
Epoch 5/10
168/168 [==============================] - 33s 198ms/step - loss: 0.0103 - accuracy: 0.9976 - val_loss: 0.1037 - val_accuracy: 0.9809
Epoch 6/10
168/168 [==============================] - 33s 196ms/step - loss: 0.0088 - accuracy: 0.9979 - val_loss: 0.0763 - val_accuracy: 0.9773
Epoch 7/10
168/168 [==============================] - 39s 231ms/step - loss: 0.0075 - accuracy: 0.9976 - val_loss: 0.1067 - val_accuracy: 0.9833
Epoch 8/10
168/168 [==============================] - 35s 211ms/step - loss: 0.0036 - accuracy: 0.9991 - val_loss: 0.1473 - val_accuracy: 0.9809
Epoch 9/10
168/168 [==============================] - ETA: 0s - loss: 0.0119 - accuracy: 0.9979Epoch 10/10
168/168 [==============================] - 33s 197ms/step - loss: 0.0032 - accuracy: 0.9994 - val_loss: 0.1147 - val_accuracy: 0.9833
```

```
metrics = pd.DataFrame(history.history)
metrics.rename(columns = {'loss': 'Training_Loss', 'accuracy': 'Training_Accuracy', 'val_loss': 'Validation_Loss', 'val_accuracy': 'Validation_Accurac
def plot_graphs1(var1, var2, string):
  metrics[[var1, var2]].plot()
  plt.title('Training and Validation ' + string)
  plt.xlabel ('Number of epochs')
  plt.ylabel(string)
  plt.legend([var1, var2])
```

```
plot_graphs1('Training_Accuracy', 'Validation_Accuracy', 'accuracy')
```

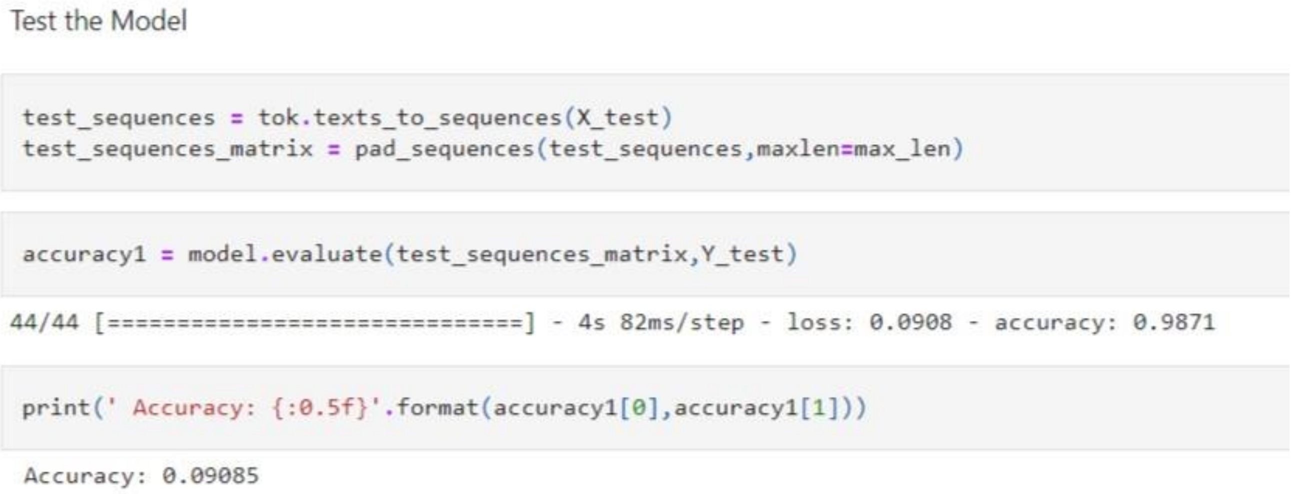
- **Save The Model**



Save the Model

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

- **Test The Model**



Test the Model

```
test_sequences = tok.texts_to_sequences(X_test)
test_sequences_matrix = pad_sequences(test_sequences,maxlen=max_len)
```

```
accuracy1 = model.evaluate(test_sequences_matrix,Y_test)
```

```
44/44 [==============================] - 4s 82ms/step - loss: 0.0908 - accuracy: 0.9871
```

```
print(' Accuracy: {:0.5f}'.format(accuracy1[0],accuracy1[1]))
```

```
 Accuracy: 0.09085
```