BDAA-ICP4-wiki report : Yousef Almutairi
Date: 09/20/2021

1. What you learned in the ICP:

In this ICP I learned how to build a Deep Learning classifier using Keras library for the twitter dataset. Also, to build a model with a combination of layers and visualize it using plot function.

2. ICP description what was the task you were performing:

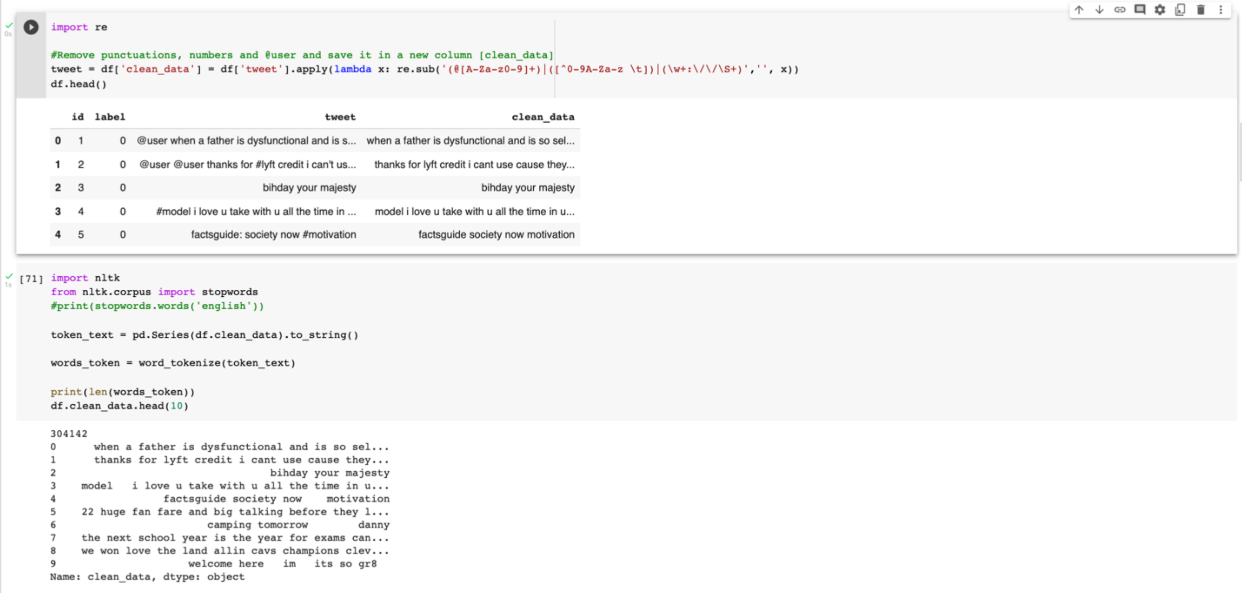
I performed the sentiment analysis task by cleaning and preprocessing the dataset as the following: removing punctuations and stop words, Tokenization.. etc. Also, I classified the text using the Convolutional Neural Network(CNN) model.

3. Challenges that you faced:

There was no big challenge in this task. However, I spent most of my time to clean and organize the data as well as

4. Screen shots that shows the successful execution of each required step of your code:

Here: I read only the tweet column and removed all punctuations, numbers and @user and save it in a new column named as "clean_data".



```
import re

#Remove punctuations, numbers and @user and save it in a new column [clean_data]
tweet = df['clean_data'] = df['tweet'].apply(lambda x: re.sub('(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)','', x))
df.head()
```

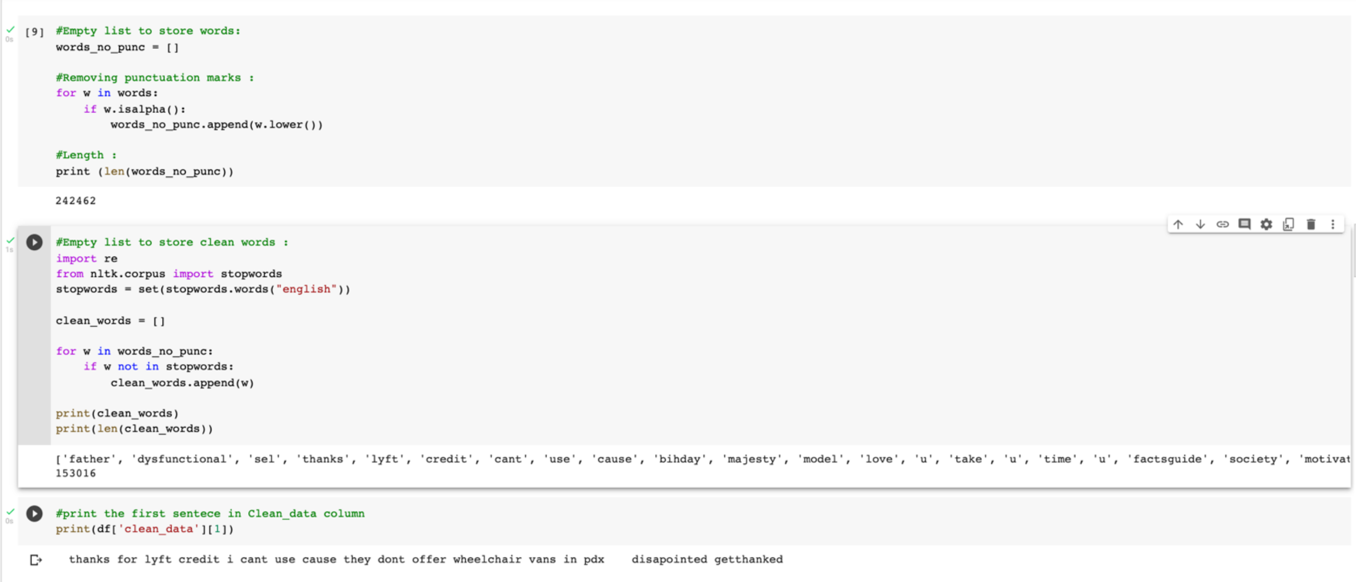
	id	label	tweet	clean_data
0	1	0	@user when a father is dysfunctional and is s...	when a father is dysfunctional and is so sel...
1	2	0	@user @user thanks for #lyft credit i can't us...	thanks for lyft credit i cant use cause they...
2	3	0	bihday your majesty	bihday your majesty
3	4	0	#model i love u take with u all the time in ...	model i love u take with u all the time in u...
4	5	0	factsguide: society now #motivation	factsguide society now motivation

```
[71] import nltk
     from nltk.corpus import stopwords
     #print(stopwords.words('english'))

     token_text = pd.Series(df.clean_data).to_string()

     words_token = word_tokenize(token_text)

     print(len(words_token))
     df.clean_data.head(10)
```

```
304142
0      when a father is dysfunctional and is so sel...
1      thanks for lyft credit i cant use cause they...
2                                  bihday your majesty
3    model   i love u take with u all the time in u...
4                  factsguide society now    motivation
5    22 huge fan fare and big talking before they l...
6                         camping tomorrow        danny
7    the next school year is the year for exams can...
8    we won love the land allin cavs champions clev...
9                      welcome here   im   its so gr8
Name: clean_data, dtype: object
```

BDAA-ICP4-wiki report : Yousef Almutairi

Date: 09/20/2021



```
[9] #Empty list to store words:
    words_no_punc = []

    #Removing punctuation marks :
    for w in words:
        if w.isalpha():
            words_no_punc.append(w.lower())

    #Length :
    print (len(words_no_punc))
```

```
242462
```

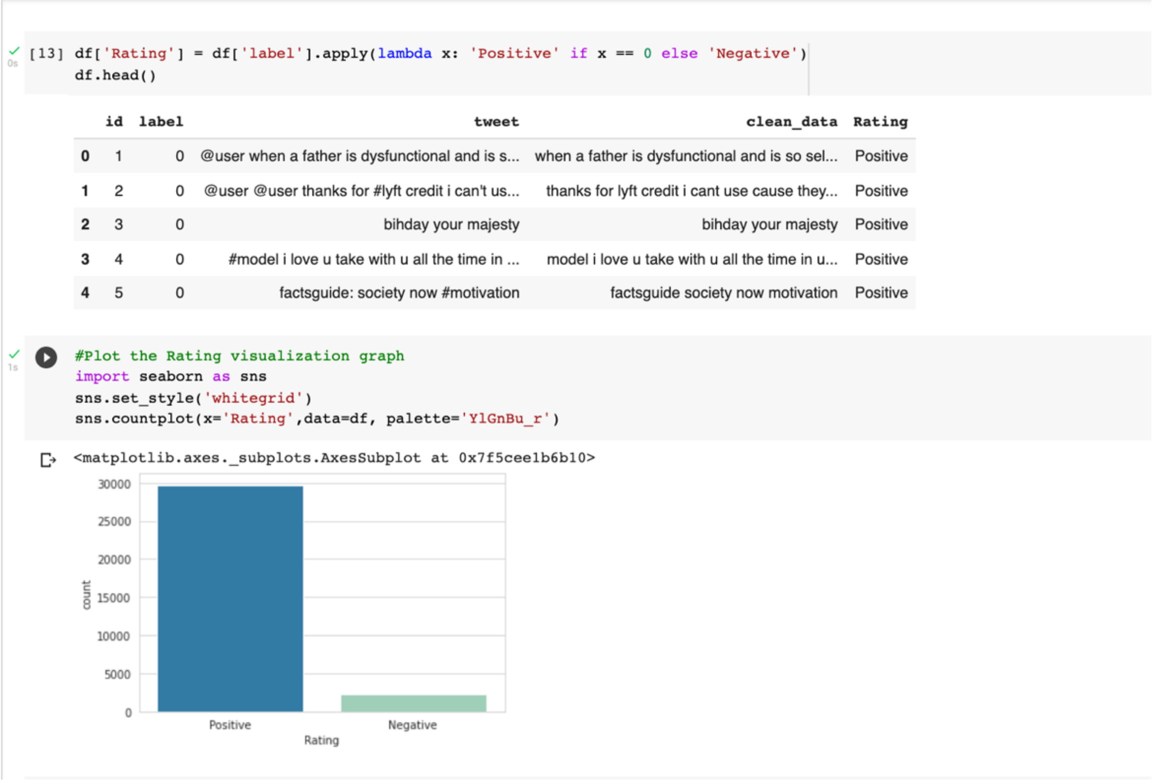
```
#Empty list to store clean words :
import re
from nltk.corpus import stopwords
stopwords = set(stopwords.words("english"))

clean_words = []

for w in words_no_punc:
    if w not in stopwords:
        clean_words.append(w)

print(clean_words)
print(len(clean_words))
```

```
['father', 'dysfunctional', 'sel', 'thanks', 'lyft', 'credit', 'cant', 'use', 'cause', 'bihday', 'majesty', 'model', 'love', 'u', 'take', 'u', 'time', 'u', 'factsguide', 'society', 'motivat
153016
```

```
#print the first sentece in Clean_data column
print(df['clean_data'][1])
```

```
thanks for lyft credit i cant use cause they dont offer wheelchair vans in pdx    disapointed getthanked
```

In this step, I created a new column named as “Rating” to show how many positive and negative text do I have in the dataset.



```
[13] df['Rating'] = df['label'].apply(lambda x: 'Positive' if x == 0 else 'Negative')
     df.head()
```

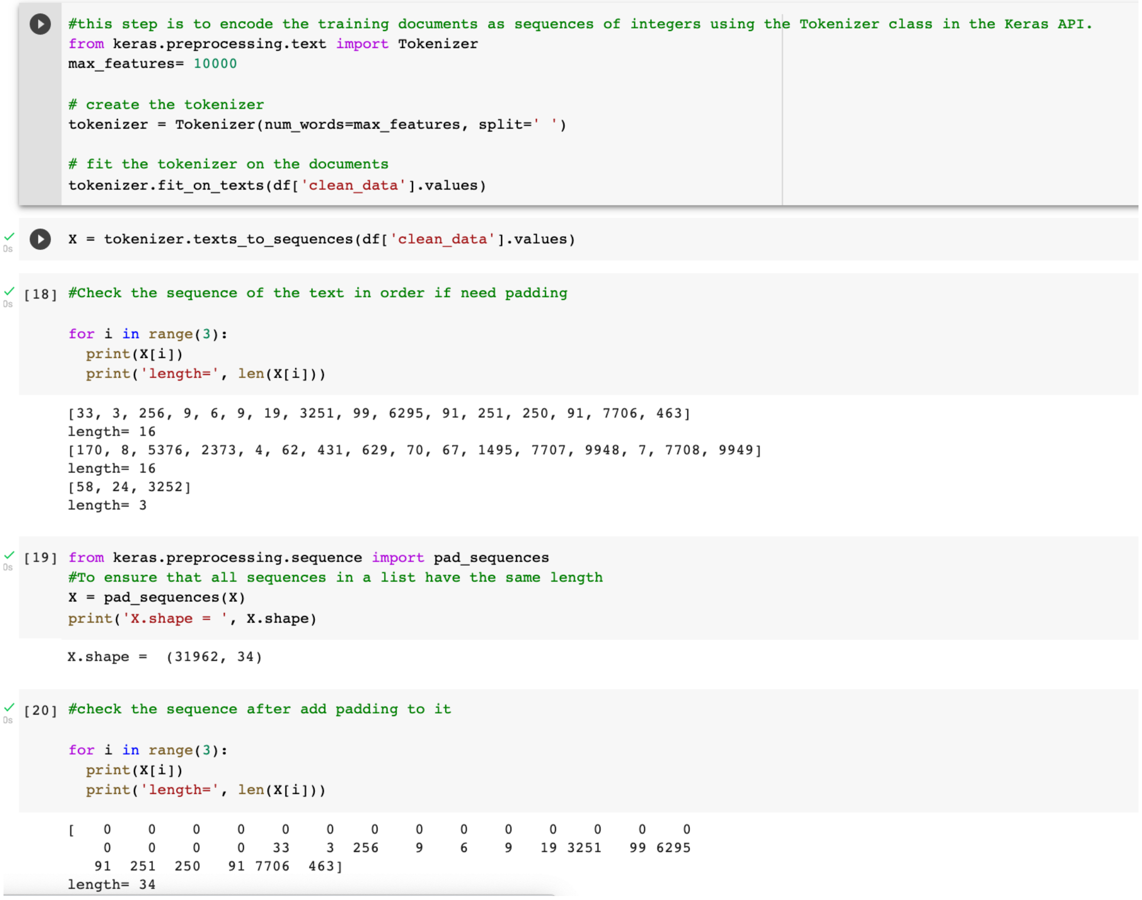
	id	label	tweet	clean_data	Rating
0	1	0	@user when a father is dysfunctional and is s...	when a father is dysfunctional and is so sel...	Positive
1	2	0	@user @user thanks for #lyft credit i can't us...	thanks for lyft credit i cant use cause they...	Positive
2	3	0	bihday your majesty	bihday your majesty	Positive
3	4	0	#model i love u take with u all the time in ...	model i love u take with u all the time in u...	Positive
4	5	0	factsguide: society now #motivation	factsguide society now motivation	Positive

```
#Plot the Rating visualization graph
import seaborn as sns
sns.set_style('whitegrid')
sns.countplot(x='Rating',data=df, palette='YlGnBu_r')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5cee1b6b10>
```

BDAA-ICP4-wiki report : Yousef Almutairi
Date: 09/20/2021

I created a tokenizer in order to encode each document as a sequence of integers and fit it to the column. Then, encode clean tweets in the training dataset using “to_sequences” function. After that I used the pad_sequences in Keras in order to ensure all the sequences have the same length.



```
#this step is to encode the training documents as sequences of integers using the Tokenizer class in the Keras API.
from keras.preprocessing.text import Tokenizer
max_features= 10000

# create the tokenizer
tokenizer = Tokenizer(num_words=max_features, split=' ')

# fit the tokenizer on the documents
tokenizer.fit_on_texts(df['clean_data'].values)
```

```
X = tokenizer.texts_to_sequences(df['clean_data'].values)
```

```
[18] #Check the sequence of the text in order if need padding

     for i in range(3):
       print(X[i])
       print('length=', len(X[i]))
```

```
[33, 3, 256, 9, 6, 9, 19, 3251, 99, 6295, 91, 251, 250, 91, 7706, 463]
length= 16
[170, 8, 5376, 2373, 4, 62, 431, 629, 70, 67, 1495, 7707, 9948, 7, 7708, 9949]
length= 16
[58, 24, 3252]
length= 3
```

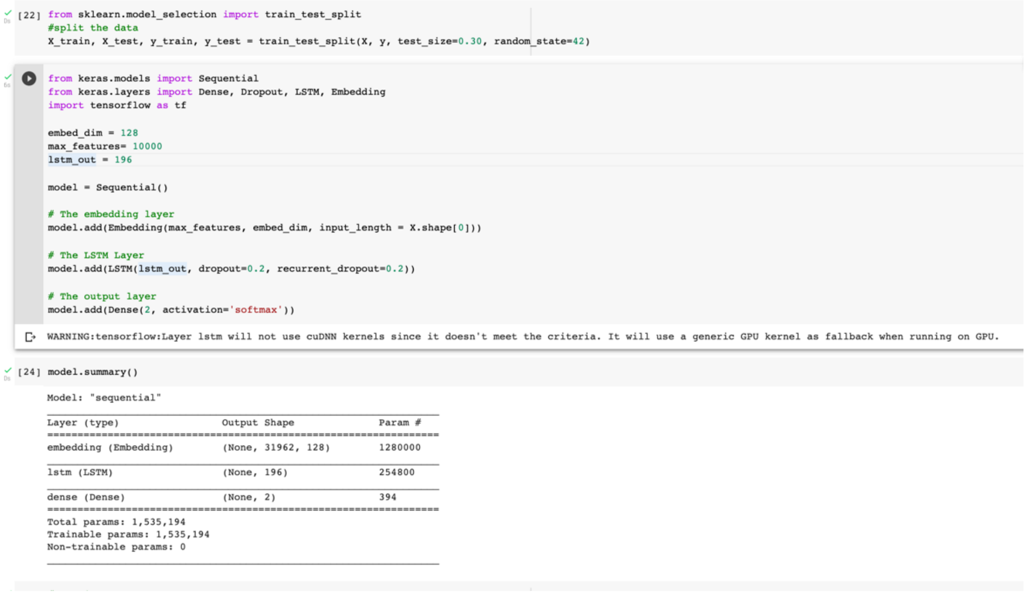
```
[19] from keras.preprocessing.sequence import pad_sequences
     #To ensure that all sequences in a list have the same length
     X = pad_sequences(X)
     print('X.shape = ', X.shape)
```

```
X.shape =  (31962, 34)
```

```
[20] #check the sequence after add padding to it

     for i in range(3):
       print(X[i])
       print('length=', len(X[i]))
```

```
[    0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0   33    3  256    9    6    9   19 3251   99 6295
    91  251  250   91 7706  463]
length= 34
```

In this step, I started splitting the data and building my Sequential model.



```
[22] from sklearn.model_selection import train_test_split
     #split the data
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)
```

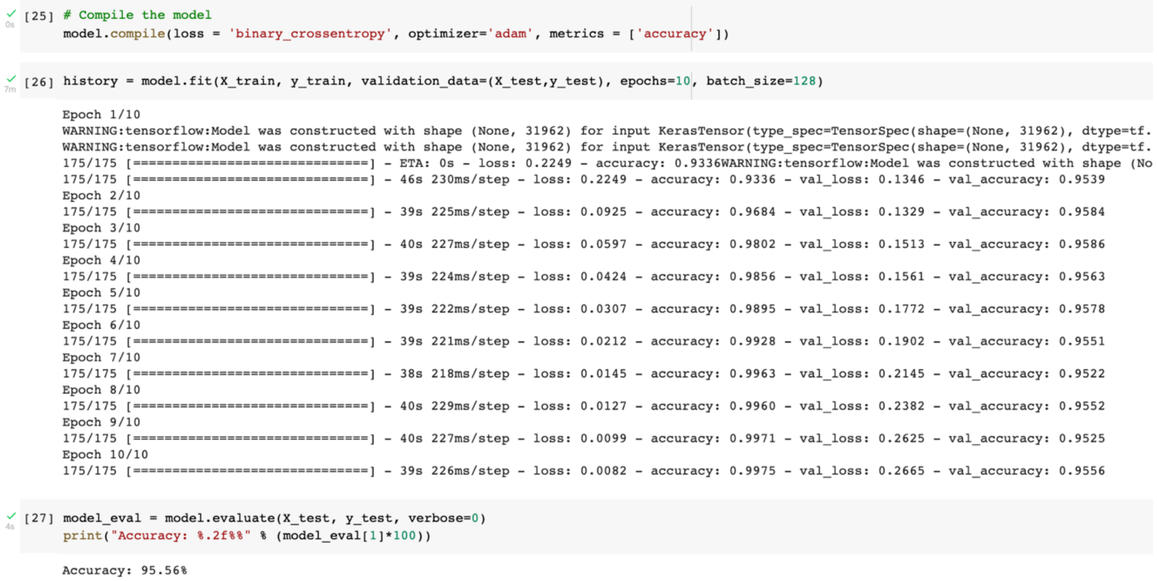
```
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM, Embedding
import tensorflow as tf

embed_dim = 128
max_features= 10000
lstm_out = 196

model = Sequential()

# The embedding layer
model.add(Embedding(max_features, embed_dim, input_length = X.shape[0]))

# The LSTM Layer
model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2))

# The output layer
model.add(Dense(2, activation='softmax'))
```

```
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
```

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

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 31962, 128)        1280000
_________________________________________________________________
lstm (LSTM)                  (None, 196)               254800
_________________________________________________________________
dense (Dense)                (None, 2)                 394
=================================================================
Total params: 1,535,194
Trainable params: 1,535,194
Non-trainable params: 0
_________________________________________________________________
```

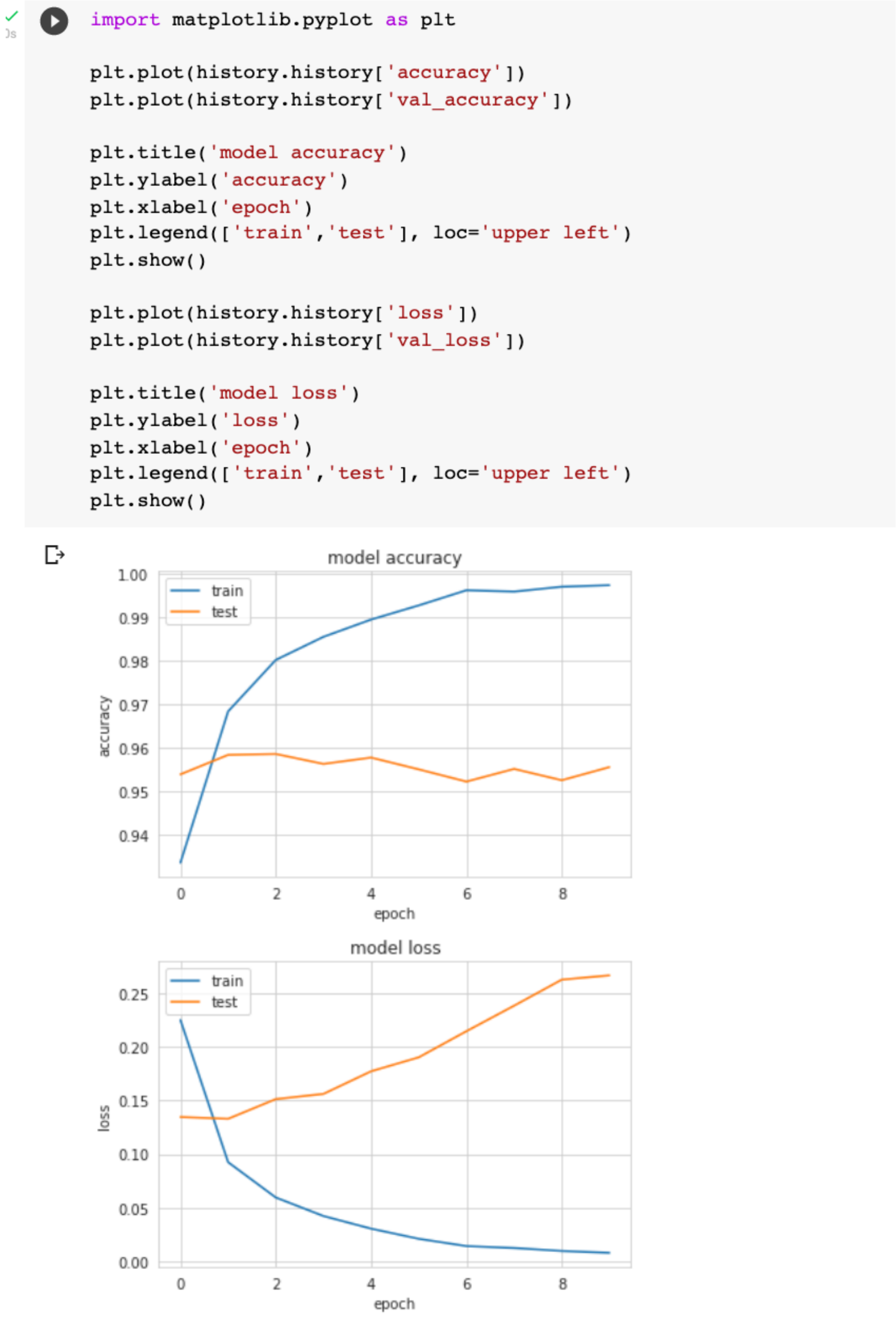
Compile, fit the model and print out the Accuracy score.



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

[26] history = model.fit(X_train, y_train, validation_data=(X_test,y_test), epochs=10, batch_size=128)

     Epoch 1/10
     WARNING:tensorflow:Model was constructed with shape (None, 31962) for input KerasTensor(type_spec=TensorSpec(shape=(None, 31962), dtype=tf.
     WARNING:tensorflow:Model was constructed with shape (None, 31962) for input KerasTensor(type_spec=TensorSpec(shape=(None, 31962), dtype=tf.
     175/175 [==============================] - ETA: 0s - loss: 0.2249 - accuracy: 0.9336WARNING:tensorflow:Model was constructed with shape (No
     175/175 [==============================] - 46s 230ms/step - loss: 0.2249 - accuracy: 0.9336 - val_loss: 0.1346 - val_accuracy: 0.9539
     Epoch 2/10
     175/175 [==============================] - 39s 225ms/step - loss: 0.0925 - accuracy: 0.9684 - val_loss: 0.1329 - val_accuracy: 0.9584
     Epoch 3/10
     175/175 [==============================] - 40s 227ms/step - loss: 0.0597 - accuracy: 0.9802 - val_loss: 0.1513 - val_accuracy: 0.9586
     Epoch 4/10
     175/175 [==============================] - 39s 224ms/step - loss: 0.0424 - accuracy: 0.9856 - val_loss: 0.1561 - val_accuracy: 0.9563
     Epoch 5/10
     175/175 [==============================] - 39s 222ms/step - loss: 0.0307 - accuracy: 0.9895 - val_loss: 0.1772 - val_accuracy: 0.9578
     Epoch 6/10
     175/175 [==============================] - 39s 221ms/step - loss: 0.0212 - accuracy: 0.9928 - val_loss: 0.1902 - val_accuracy: 0.9551
     Epoch 7/10
     175/175 [==============================] - 38s 218ms/step - loss: 0.0145 - accuracy: 0.9963 - val_loss: 0.2145 - val_accuracy: 0.9522
     Epoch 8/10
     175/175 [==============================] - 40s 229ms/step - loss: 0.0127 - accuracy: 0.9960 - val_loss: 0.2382 - val_accuracy: 0.9552
     Epoch 9/10
     175/175 [==============================] - 40s 227ms/step - loss: 0.0099 - accuracy: 0.9971 - val_loss: 0.2625 - val_accuracy: 0.9525
     Epoch 10/10
     175/175 [==============================] - 39s 226ms/step - loss: 0.0082 - accuracy: 0.9975 - val_loss: 0.2665 - val_accuracy: 0.9556

[27] model_eval = model.evaluate(X_test, y_test, verbose=0)
     print("Accuracy: %.2f%%" % (model_eval[1]*100))

     Accuracy: 95.56%
```

In the following screenshot, I used the “matplotlib.pyplot” library to visualize the accuracy and loss of the model.

BDAA-ICP4-wiki report : Yousef Almutairi
Date: 09/20/2021



```
import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])

plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])

plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()
```

5. Video link:

https://drive.google.com/file/d/1eygpPoyE1BKcmcDRODNHsHEwwMhaSrvO/view?usp=sharing