TRAVEL REVIEW RATING

END TERM PROJECT

by

PRASHANT PANDEY

Section: k20JS

Roll number: 12006445

Department of Intelligent systems

School of Computer Science Engineering

Lovely Professional University

November-2022

APPENDIX 2

Student Declaration

This is to declare that this report has been written by me. No part of the report is copied from other sources. All information included from other sources have been duly acknowledged. I aver that if any part of report is found to be copied I take the full responsibility for it.

Name of student - Prashant Pandey

Signature of student - Prashant Pandey

Roll number- 12006445

Jalandhar(Punjab)

5-11-2022

APPENDIX 3

(A typical specimen of table of contents)

TABLE OF CONTENTS

TITLE PAGE NO.

1. Background and objectives of project assigned……. 5
2. Description of Project….. 7
3. Project Work…….. 9
4. Technologies and frameworks used….. 23
5. Project analysis….. 25

APPENDIX 4

BONAFIDE CERTIFICATE

Certified that this project report "TRAVEL REVIEW RATING" is the bonafied work of "PRASHANT PANDEY" who carried out the project work under my supervision.

Name of supervisor – Dr. Dhanpratap Singh

Academic Designation – Faculty

ID of Supervisor – 25706

Department of supervisor – School of Computer Science and Engineering

Background and Objectives of Project assigned.

In this age of globalization world is now considered to be a global village. More oftenly people move from one place to another in seek of education, employment, business activities or for tourism. Now to accomplish this movement to have a successful journey people need travel and stay facilities. Back to the previous time one needs to ask his friend or call a travel agent to get details about desirable place with a limited in-hand information but with the change of time and as a boon of internet detailed information is available online on your fingertips throughout the time. Before your movement u can check for a flight or a hotel u can compare services and make your best decision. There are a lot of online travel agencies, travel sites providing u easy booking u can make your wise decision by looking over **travel review ratings.**

People prefer to check for a review before a booking In order to get the best quality. The reviews on a travel site are made by people who had already experienced the service. In order to get unbiased reviews it is necessary to have authentic reviewer.

Over the recent years there is a boom in travel sector and it was estimated that digital travel sales would cross 800 BN $. Now countries are giving priority to

tourism sector as a much potent economic contributor and also no. of travellers have grown tremendously it has created a great space for IT industry. Not only to manage the data of customers but also to help the industry to grow and expand. In this project we will be looking how machine learning is useful for travel review ratings. Online travel sites like Tripadvisor, Booking.com provide travel review ratings. These ratings are concluded over reviews made from verified customers. The supremacy of travel review ratings can be known from the fact that 77% of people don't book a trip without reading a review over Online travel sites. Travel sites like Tripadvisor,Expedia Group and others use machine learning tools to generate the ratings for Hotels, Restaurants, Beaches, Parks, Gyms or Malls and like other. So we'll be looking over the application and also how this relation(between tourism and ml) can be made better for an overall growth and development.

Description of Project

This project is based on the topic how ml is used for travel review analysis. for completing this project I had taken dataset from Kaggle where around 20000 reviews were collected from TripAdvisor and I had implemented a rating pattern of 1 to 5 stars.

The codes in this project are written in Python 3 using Jupyter Notebook.

The project aims to train a model that is capable to rate reviews on grading of 1,2,3,4,5 stars. For doing so we need to collect abundant datasets and then we would be applying algorithms in order to obtain corresponding grades. using ml we can also have the features of prediction i.e we can predict certain things like what could be the plane fairy ,expected hotel booking prices ,expenes on resorts, museums,restaurants, pools,beaches ,gyms and clubs. With this much of available details it may give tourists enough idea to plan their trip or to have a knowledge of possible expenses. This prediction is good for both customer and service provider as it gives an expected image of market so both are able to plan accordingly and thus there will be an overall betterment of

industry. To achieve this objective we have created the whole program below in which we have used jupyter notebook and tensorflow to write our code. Code has been written in python 3 and we have imported the corresponding libraries like numpy, string,spacy matplotlib.

Going with this approach we can create wordcloud which represents the frequency(no.of times a word occurs) also textual data contains a lot of noise like stopwords(is,an,the,because,e.t.c) punctuation,different forms of the same word (like play,plays,playing,played e.t.c where the root word is play. All of these interfere with the training process of our model and increases the vocabulary size of our model. For this purpose we will define a function to cleam our data. To effectively do preprocessing we must have texts in lowercase .

To handle negations we will use the apostrophe dictionary later we modify the apostrophe dictionary to get grammatically correct words.

We also need to use keras as ml model can’t make sense of data whereas provides a function for this very purpose which creates a vocabulary of words and assigns each word an index that is used to represent the word in its sequence notation.

Another important key is to never fit the tokenizer above on validation or testing. It must be fit only on training data.

In this project we following Lemmatization however there are other techniques also like stemming which can be implemented to get the same output.

PROJECT WORK

To begin with, we must import the relevant libraries we will be using for this task.

```
import numpy as np
import pandas as pd
import re
import spacy
from nltk.corpus import stopwords
from wordcloud import WordCloud,STOPWORDS
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
import string
```

some of the rows in the data.

```
data = pd.read_csv('input/trip-advisor-hotel-
reviews/tripadvisor_hotel_reviews.csv')
data.head()
```

	Review	Rating
0	nice hotel expensive parking got good deal sta...	4
1	ok nothing special charge diamond member hilto...	2
2	nice rooms not 4* experience hotel monaco seat...	3
3	unique, great stay, wonderful time hotel monac...	5
4	great stay great stay, went seahawk game aweso...	5

First 5 rows of the data.

Let's see a small description of the data.

```
data.describe()
```

	Rating
count	20491.000000
mean	3.952223
std	1.233030
min	1.000000
25%	3.000000
50%	4.000000
75%	5.000000
max	5.000000

As it can be seen that there are 20491 observations in the data. The mean rating is close to 4. Let's see if there are any missing data.

```
data.isnull().mean()
```

The output is as follows:

```
Review    0.0
Rating    0.0
dtype: float64
```

Nice! No data is missing. Let's see the number of unique ratings in the data.

```
data['Rating'].unique()
```

The output is as follows:

```
array([4, 2, 3, 5, 1])
```

We have only 5 unique values of the variable Rating. So we will treat this problem as a softmax classification problem. **Formally, my task would be to train a model to classify a hotel review as 1,2,3, or 5 stars.**

Let's see the distribution of each class within the table.

```
data['Rating'].value_counts(normalize=True)
```

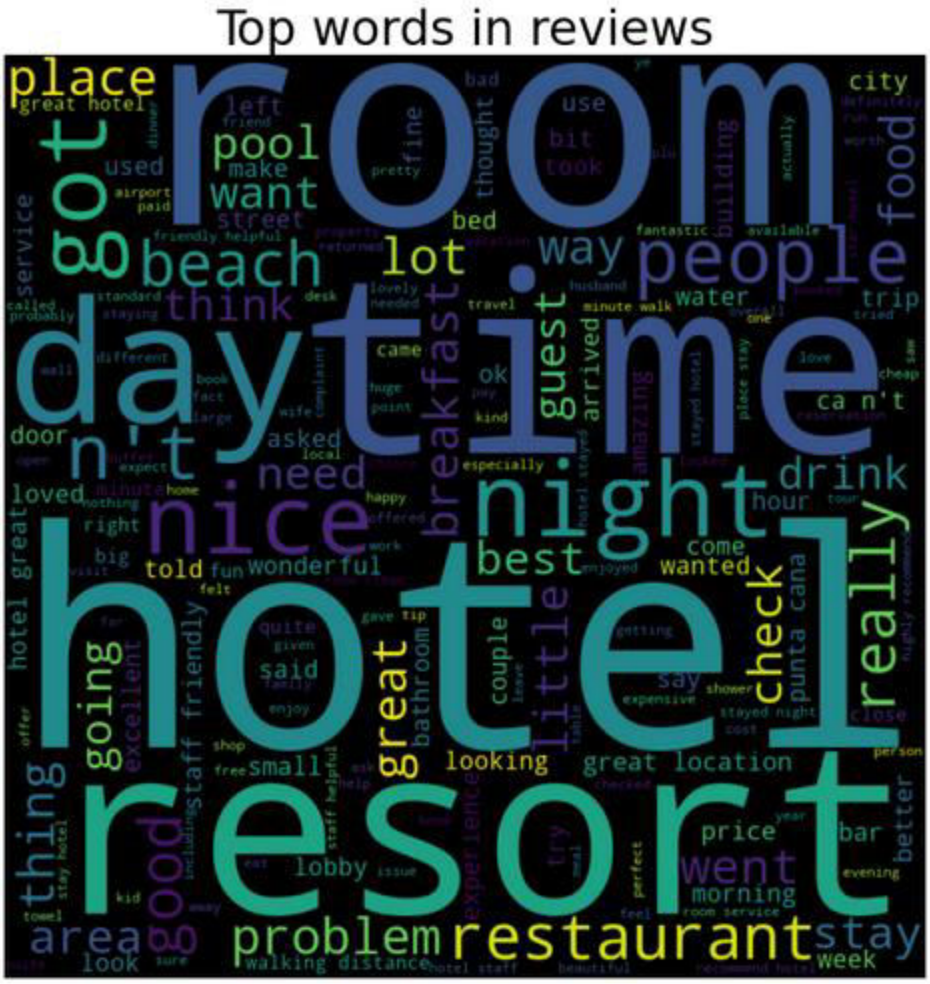
The output we get is:

```
5    0.441853
4    0.294715
3    0.106583
2    0.087502
1    0.069348
Name: Rating, dtype: float64
```

We can see that most ratings are 5-star ratings and the least are 1-star ratings. Thus we must take care to train our model on enough 1-star ratings. Thus while training the data, we must stratify our dataset. This will **ensure that the distribution of all the classes is even between the train, validation, and test set. This will help prevent unknown classes from appearing in the validation or test set.**

We will now create a word cloud to see the most commonly occurring words in our reviews.

```
def wordCloud_generator(data, title=None):
    wordcloud = WordCloud(width = 800, height = 800,
                          background_color ='black',
                          min_font_size = 10
                         ).generate(" ".join(data.values))
    plt.figure(figsize = (8, 8), facecolor = None)
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    plt.tight_layout(pad = 0)
    plt.title(title,fontsize=30)
    plt.show()wordCloud_generator(data['Review'], title="Top
words in reviews")
```



The bigger the word is, the more times it occurs

We will create copies of our data to preserve the original.

```
X = data['Review'].copy()
y = data['Rating'].copy()
```

Textual data contains a lot of noise like stopwords (is, an, the, because, etc.) punctuation, different forms of the same word (like play, plays, playing, played, etc where the root word is play. This process is called lemmatization) and negation of words (like didn’t instead of did not). All of these interfere with the training process of our model and increase the vocabulary size of our model. So we must deal with them properly.

For this purpose, we will define a function to clean our data. To effectively do our preprocessing, we must have all our text in lower case. To handle negations we will use the apostrophe dictionary which has a of list all the frequently used apostrophe words in English.

On closer inspection of the data, we can observe that the data has misspelled words (shown below). For example “not” has been spelled as “n’t” in some cases and some negations like “didn’t” have been spelled as “did n’t”. We shall address these inconsistencies in our cleaning process too.

```
"nice rooms not 4* experience hotel monaco seattle good hotel
n't 4* level.positives large bathroom mediterranean suite
comfortable bed pillowsattentive housekeeping staffnegatives ac
unit malfunctioned stay desk disorganized, missed 3 separate
wakeup calls, concierge busy hard touch, did n't provide
guidance special requests.tv hard use ipod sound dock suite non
functioning. decided book mediterranean suite 3 night weekend
stay 1st choice rest party filled, comparison w spent 45 night
larger square footage room great soaking tub whirlpool jets nice
shower.before stay hotel arrange car service price 53 tip
reasonable driver waiting arrival.checkin easy downside room
picked 2 person jacuzi tub no bath accessories salts bubble bath
did n't stay, night got 12/1a checked voucher bottle champagne
nice gesture fish waiting room, impression room huge open space
felt room big, tv far away bed chore change channel, ipod dock
broken disappointing.in morning way asked desk check thermostat
said 65f 74 2 degrees warm try cover face night bright blue
light kept, got room night no, 1st drop desk, called
maintainence came look thermostat told play settings happy
digital box wo n't work, asked wakeup 10am morning did n't
happen, called later 6pm nap wakeup forgot, 10am wakeup morning
yep forgotten.the bathroom facilities great room surprised room
sold whirlpool bath tub n't bath amenities, great relax water
jets going,  "
#sample review from the dataset
```

Thus, the apostrophe dictionary we got earlier, will be modified to account for this as follows:

```
apposV2 = {
"are not" : "are not",
"ca" : "can",
"could n't" : "could not",
"did n't" : "did not",
"does n't" : "does not",
"do n't" : "do not",
"had n't" : "had not",
"has n't" : "has not",
"have n't" : "have not",
"he'd" : "he would",
"he'll" : "he will",
"he's" : "he is",
"i'd" : "I would",
"i'd" : "I had",
"i'll" : "I will",
"i'm" : "I am",
"is n't" : "is not",
"it's" : "it is",
"it'll":"it will",
"i've" : "I have",
"let's" : "let us",
"might n't" : "might not",
"must n't" : "must not",
"sha" : "shall",
"she'd" : "she would",
"she'll" : "she will",
"she's" : "she is",
"should n't" : "should not",
"that's" : "that is",
"there's" : "there is",
"they'd" : "they would",
"they'll" : "they will",
"they're" : "they are",
"they've" : "they have",
"we'd" : "we would",
"we're" : "we are",
"were n't" : "were not",
"we've" : "we have",
"what'll" : "what will",
"what're" : "what are",
"what's" : "what is",
"what've" : "what have",
"where's" : "where is",
"who'd" : "who would",
"who'll" : "who will",
"who're" : "who are",
"who's" : "who is",
"who've" : "who have",
"wo" : "will",
"would n't" : "would not",
```

```
"you'd" : "you would",
"you'll" : "you will",
"you're" : "you are",
"you've" : "you have",
"'re": " are",
"was n't": "was not",
"we'll":"we will",
"did n't": "did not"
}appos = {
"aren't" : "are not",
"can't" : "cannot",
"couldn't" : "could not",
"didn't" : "did not",
"doesn't" : "does not",
"don't" : "do not",
"hadn't" : "had not",
"hasn't" : "has not",
"haven't" : "have not",
"he'd" : "he would",
"he'll" : "he will",
"he's" : "he is",
"i'd" : "I would",
"i'd" : "I had",
"i'll" : "I will",
"i'm" : "I am",
"isn't" : "is not",
"it's" : "it is",
"it'll":"it will",
"i've" : "I have",
"let's" : "let us",
"mightn't" : "might not",
"mustn't" : "must not",
"shan't" : "shall not",
"she'd" : "she would",
"she'll" : "she will",
"she's" : "she is",
"shouldn't" : "should not",
"that's" : "that is",
"there's" : "there is",
"they'd" : "they would",
"they'll" : "they will",
"they're" : "they are",
"they've" : "they have",
"we'd" : "we would",
"we're" : "we are",
"weren't" : "were not",
"we've" : "we have",
"what'll" : "what will",
"what're" : "what are",
"what's" : "what is",
"what've" : "what have",
```

```
"where's" : "where is",
"who'd" : "who would",
"who'll" : "who will",
"who're" : "who are",
"who's" : "who is",
"who've" : "who have",
"won't" : "will not",
"wouldn't" : "would not",
"you'd" : "you would",
"you'll" : "you will",
"you're" : "you are",
"you've" : "you have",
"'re": " are",
"wasn't": "was not",
"we'll":" will",
"didn't": "did not"
}
```

We will now set up our cleaning function.

```
nlp = spacy.load('en',disable=['parser','ner'])
stop = stopwords.words('english')
def cleanData(reviews):
    all_=[]
    for review in reviews:
        lower_case = review.lower() #lower case the text
        lower_case = lower_case.replace(" n't"," not") #correct
n't as not
        lower_case = lower_case.replace("."," . ")
        lower_case = ' '.join(word.strip(string.punctuation) for
word in lower_case.split()) #remove punctuation
        words = lower_case.split() #split into words
        words = [word for word in words if word.isalpha()]
#remove numbers
        split = [apposV2[word] if word in apposV2 else word for
word in words] #correct using apposV2 as mentioned above
        split = [appos[
ord] if word in appos else word for word in split] #correct
using appos as mentioned above
        split = [word for word in split if word not in stop]
#remove stop words
        reformed = " ".join(split) #join words back to the text
        doc = nlp(reformed)
        reformed = " ".join([token.lemma_ for token in doc])
#lemmatiztion
        all_.append(reformed)
    df_cleaned = pd.DataFrame()
    df_cleaned['clean_reviews'] = all_
    return df_cleaned['clean_reviews']X_cleaned = cleanData(X)
X_cleaned.head()
```

We get the following output:

```
0    nice hotel expensive parking get good deal sta...
1    ok nothing special charge diamond member hilto...
2    nice room experience hotel monaco seattle good...
3    unique great stay wonderful time hotel monaco ...
4    great stay great stay go seahawk game awesome ...
Name: clean_reviews, dtype: object
```

We will also encode our target variable Rating, into a one hot vector.

```
encoding = {1: 0,
            2: 1,
            3: 2,
            4: 3,
            5: 4
           }labels = ['1', '2', '3', '4', '5']

y = data['Rating'].copy()
y.replace(encoding, inplace=True)
y = to_categorical(y,5)
```

It's time to split our data into train and test sets. We will use 80% of the data for training, 10% for validation, and 10% for testing.

```
X_train, X_test, y_train, y_test = train_test_split(X_cleaned,
y, stratify=y, random_state=42,test_size=0.1)
#validation split will done when fitting the model
```

ML models can't make sense of text data. To feed them text data, we convert our text into sequences that be fed to the model. Keras provides a function for this very purpose. It creates a vocabulary of words and assigns each word an index that is used to represent the word in its sequence notation. Each sentence may not be the same length as the others. We pad them using Keras as our model expects each sentence to be of the same length.

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import
pad_sequencestokenizer = Tokenizer()
tokenizer.fit_on_texts(X_train)X_train =
tokenizer.texts_to_sequences(X_train)max_length = max([len(x)
for x in X_train])
vocab_size = len(tokenizer.word_index)+1 #add 1 to account for
```

```
unknown wordprint("Vocabulary size: {}".format(vocab_size))
print("Max length of sentence: {}".format(max_length))X_train =
pad_sequences(X_train, max_length ,padding='post')
```

We get the following output:

```
Vocabulary size: 41115
Max length of sentence: 1800
```

An important note here is to never fit the Tokenizer above on the validation or testing data. It must be fit ONLY on the training data. As a general note, any kind of fitting must be done only on the training data.

It's time to start modeling. Let's create our model. We will create a sequential model.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM,Dense,Dropout
from tensorflow.keras.layers import
Bidirectional,Embedding,Flatten
from tensorflow.keras.callbacks import
EarlyStopping,ModelCheckpointembedding_vector_length=32
num_classes = 5
model =
Sequential()model.add(Embedding(vocab_size,embedding_vector_leng
th,input_length=X_train.shape[1]))
model.add(Bidirectional(LSTM(250,return_sequences=True)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(128,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(64,activation='relu'))
model.add(Dense(32,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(16,activation='relu'))
model.add(Dense(num_classes,activation='softmax'))model.compile(
loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])callbacks =
[EarlyStopping(monitor='val_loss', patience=5),
             ModelCheckpoint('../model/model.h5',
save_best_only=True,
                             save_weights_only=False)]
model.summary()
```

We add some dropout layers and callbacks to prevent overfitting. We get the following output.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 1800, 32)          1315680
_________________________________________________________________
bidirectional_1 (Bidirection (None, 1800, 500)         566000
_________________________________________________________________
dropout_3 (Dropout)          (None, 1800, 500)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 900000)            0
_________________________________________________________________
dense_5 (Dense)              (None, 128)               115200128
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_6 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_7 (Dense)              (None, 32)                2080
_________________________________________________________________
dropout_5 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_8 (Dense)              (None, 16)                528
_________________________________________________________________
dense_9 (Dense)              (None, 5)                 85
=================================================================
Total params: 117,092,757
Trainable params: 117,092,757
Non-trainable params: 0
_________________________________________________________________
```

Let’s fit our model and start the training process.

```
history = model.fit(X_train, y_train, validation_split=0.11,
                    epochs=15, batch_size=32, verbose=1,
                    callbacks=callbacks)
```

Our training yields the following results.

```
Epoch 1/15
513/513 [==============================] - 195s 368ms/step -
loss: 1.6616 - accuracy: 0.4265 - val_loss: 0.9994 -
val_accuracy: 0.5002
Epoch 2/15
513/513 [==============================] - 189s 369ms/step -
loss: 0.9146 - accuracy: 0.5659 - val_loss: 0.9282 -
val_accuracy: 0.5619
Epoch 3/15
513/513 [==============================] - 188s 367ms/step -
loss: 0.7914 - accuracy: 0.6282 - val_loss: 0.9804 -
val_accuracy: 0.5722
Epoch 4/15
513/513 [==============================] - 189s 368ms/step -
loss: 0.6786 - accuracy: 0.7044 - val_loss: 0.9794 -
val_accuracy: 0.6077
Epoch 5/15
513/513 [==============================] - 189s 368ms/step -
loss: 0.5620 - accuracy: 0.7673 - val_loss: 1.0368 -
val_accuracy: 0.5973
Epoch 6/15
513/513 [==============================] - 188s 367ms/step -
loss: 0.4566 - accuracy: 0.8180 - val_loss: 1.2449 -
val_accuracy: 0.5875
Epoch 7/15
513/513 [==============================] - 189s 369ms/step -
loss: 0.3666 - accuracy: 0.8607 - val_loss: 1.3929 -
val_accuracy: 0.5954
```

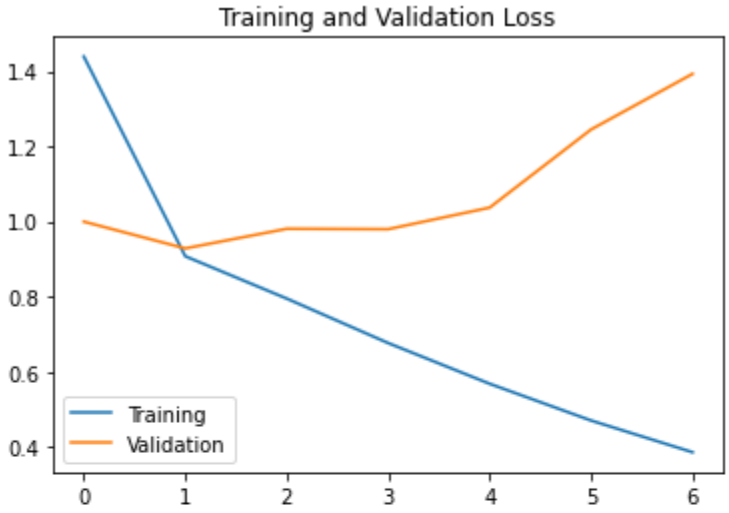
We can notice that **the model stopped with 7 epochs although it was set for 15 epochs. This is because of our callback. It stopped the training process as soon as it observed that there was no improvement in the validation accuracy after 5 epochs.** Additionally, it also saved the weights of the model which best prevent overfitting.

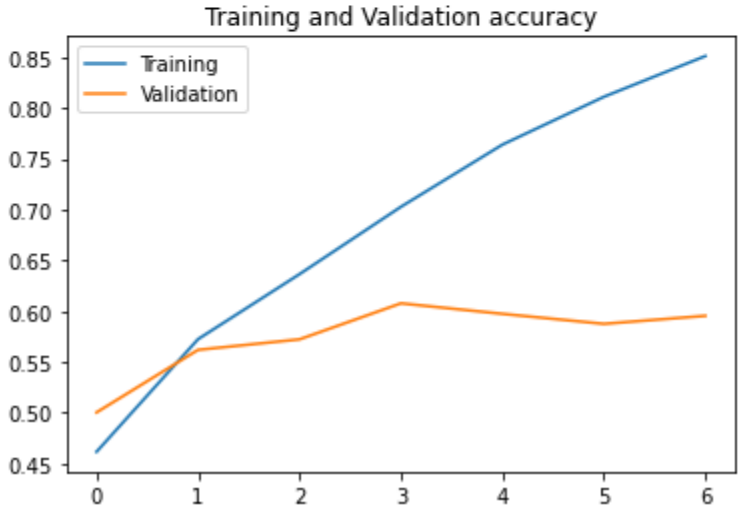
We reached an accuracy of 60% on the validation set and 70% on our train set (4th epoch). We can reduce the overfitting further by increasing the Dropout layers. Let's plot our results.

```
import matplotlib.pyplot as plt

plt.plot(history.history['loss'], label='Training')
plt.plot(history.history['val_loss'], label='Validation')
plt.legend()
plt.title('Training and Validation Loss')
plt.figure()

plt.plot(history.history['accuracy'],label='Training')
plt.plot(history.history['val_accuracy'],label='Validation')
plt.legend()
plt.title('Training and Validation accuracy')
```





Plots

Let’s do some predictions on our test set.

```
X_test_token = tokenizer.texts_to_sequences(X_test)
X_test_token = pad_sequences(X_test_token, max_length
,padding='post')
pred = model.predict(X_test_token)
pred = to_categorical(pred,5)
```

We can get the accuracy score and the classification report as follows:

```
from sklearn.metrics import classification_report,accuracy_score
print('Test Accuracy: {}'.format(accuracy_score(pred, y_test)))
print(classification_report(y_test, pred, target_names=labels))
```

The output is follows:

```
Test Accuracy: 0.5936585365853658             precision    recall
f1-score   support

           1       0.61      0.64      0.63       142
           2       0.40      0.31      0.35       179
           3       0.45      0.32      0.38       219
           4       0.47      0.57      0.52       604
           5       0.75      0.72      0.73       906

   micro avg       0.59      0.59      0.59      2050
   macro avg       0.54      0.51      0.52      2050
weighted avg       0.59      0.59      0.59      2050
 samples avg       0.59      0.59      0.59      2050
```

From our classification report, we can observe that our model performs reasonably well on above graded ratings of 1,2,3,4 and 5-star ratings

Technologies and frameworks used

for completing this project I have used

Language - Python 3

Platform - Jupyter Notebook, TensorFlow.

Libraries - numpy

pandas

re

spacy

string

matplotlib

API - Keras

PROJECT ANALYSIS

Completing this project I learned how machine learning is applicable in the context of travel review ratings. Working on this project gave me an idea about Strengths, Weaknesses, Opportunities and Threats (SWOT) associated with this aim. In this module we will discuss about these points.

STRENGTH :- The project has major strength as it provides an automated system for generating ratings regarding travel reviews. Implementing machine learning algorithm gives enterprises a view of trends in customer behavior and business operational patterns.

LIMITATION :- The technology works good only when properly implemented so we need to consider the following things at utmost priority if we want to get the best results.

- **Gather quality data**.
- **Deterministic Problem.**
- **Misapplication.**
- **Time and Resources.**

OPPORTUNITIES:- ML has boosted the sector of tour and travels. when properly trained ml algorithms are capable to predict the upcoming

travel sentiments,estimated tour and travel prices including the plane fairy,hotel charges and about other essentials so it helps clients in making their decisions and service providers to give their best.

THREATS :- The biggest hurdle in implementing machine learning is its vast demand of data . Not only abundant data but quality data. Data needs not to be corrupted it should be very precise and accurate in order to get the best result.

We need good and experienced professionals to carry out the above task.