PROJECT NAME:-

RATINGS PREDICTIONS

SUBMITTED BY:

SHIVAM SHARMA

INTRODUCTION

Problem Statement :

We have a client who has a website where people write different reviews for technical products. Now they are adding a new feature to their website i.e. The reviewer will have to add stars(rating) as well with the review. The rating is out 5 stars and it only has 5 options available 1 star, 2 stars, 3 stars, 4 stars, 5 stars. Now they want to predict ratings for the reviews which were written in the past and they don’t have a rating.

So, Our goal to build an application which can predict the rating by seeing the review.

Review of Literature

The topic is about to build a Model which can predict the rating by seeing the review as the ratings are from 1 to 5 .

So after seeing the riview the model react how the rating should be assigned for that particular riview.

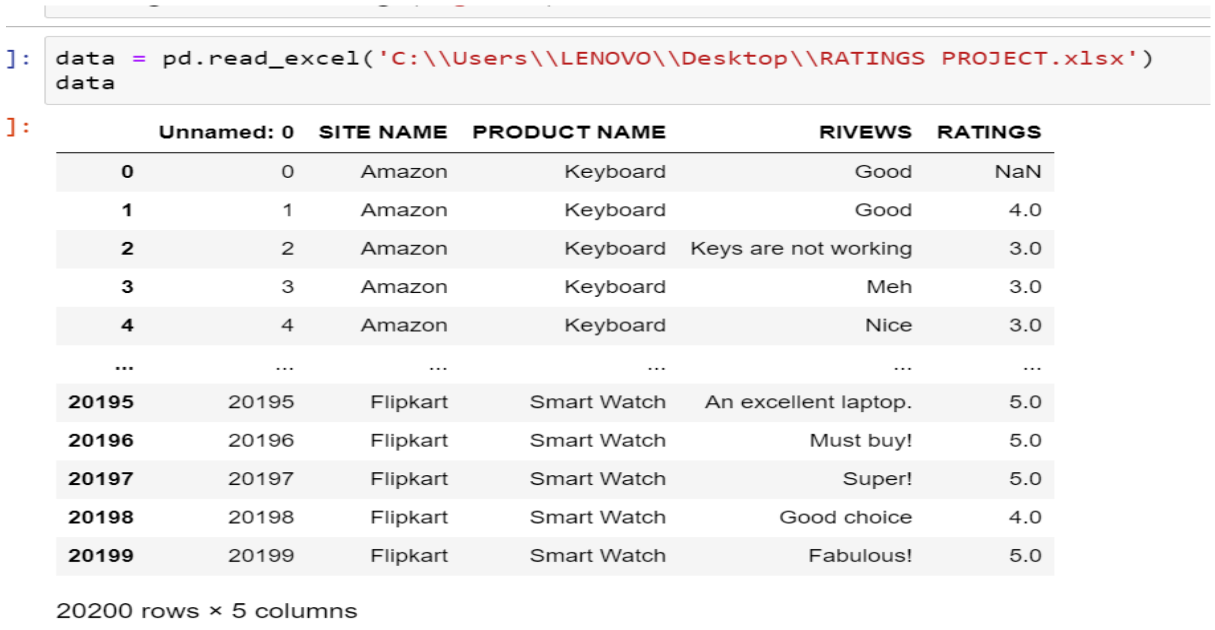
Motivation for the Problem Undertaken

As the e commerce is in very helpful bcz it makes our life easy as well as we can just simply order anything from our homes or from anywhere ,

So public is more reliable on the reviews for any product which is written on the sites . So we build a model so that is is easily predict the Ratings after seeing the reviews wriiten by any customer.

Data Sources and their format

- Amazon.com
- Flipkart.com
- Snapdeal.com



```
]: data = pd.read_excel('C:\\Users\\LENOVO\\Desktop\\RATINGS PROJECT.xlsx')
   data
```

]:

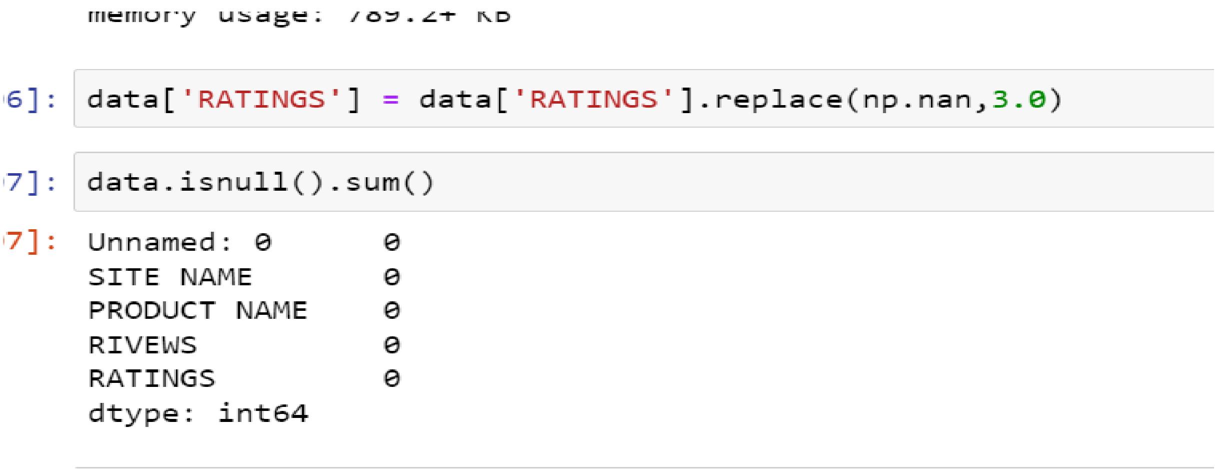
	Unnamed: 0	SITE NAME	PRODUCT NAME	RIVEWS	RATINGS
0	0	Amazon	Keyboard	Good	NaN
1	1	Amazon	Keyboard	Good	4.0
2	2	Amazon	Keyboard	Keys are not working	3.0
3	3	Amazon	Keyboard	Meh	3.0
4	4	Amazon	Keyboard	Nice	3.0
...	...	...	...	...	...
20195	20195	Flipkart	Smart Watch	An excellent laptop.	5.0
20196	20196	Flipkart	Smart Watch	Must buy!	5.0
20197	20197	Flipkart	Smart Watch	Super!	5.0
20198	20198	Flipkart	Smart Watch	Good choice	4.0
20199	20199	Flipkart	Smart Watch	Fabulous!	5.0

20200 rows × 5 columns

Data Preprocessing and EDA

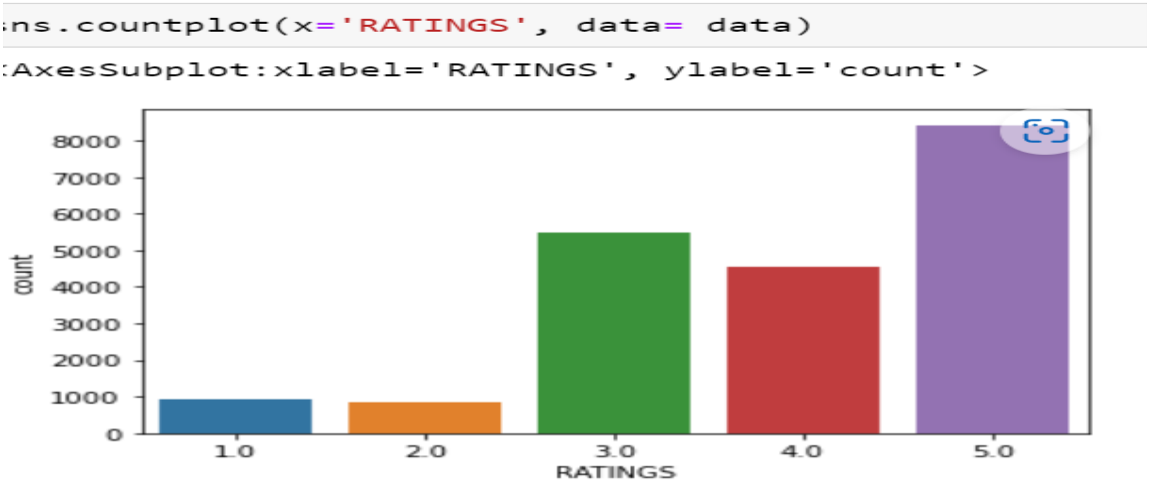
The steps followed for the cleaning and EDA of the data:-

1) First we analyse the data by simply data.info() and data.describe() methods so that we can check about the datatypes and checked themean , median and deviation.
2) After this we check for the null values and there are some some null values present in our rating data so we should fill these nans by fillna method.



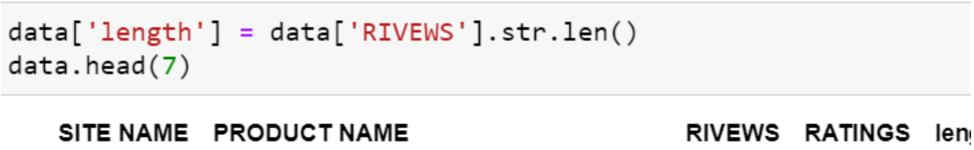
```
memory usage: 789.2+ KB

6]:  data['RATINGS'] = data['RATINGS'].replace(np.nan,3.0)

7]:  data.isnull().sum()

7]:  Unnamed: 0       0
     SITE NAME        0
     PRODUCT NAME     0
     RIVEWS           0
     RATINGS          0
     dtype: int64
```

3) Then we count the ratings by value_counts method :



```
ns.countplot(x='RATINGS', data= data)
:AxesSubplot:xlabel='RATINGS', ylabel='count'>
```

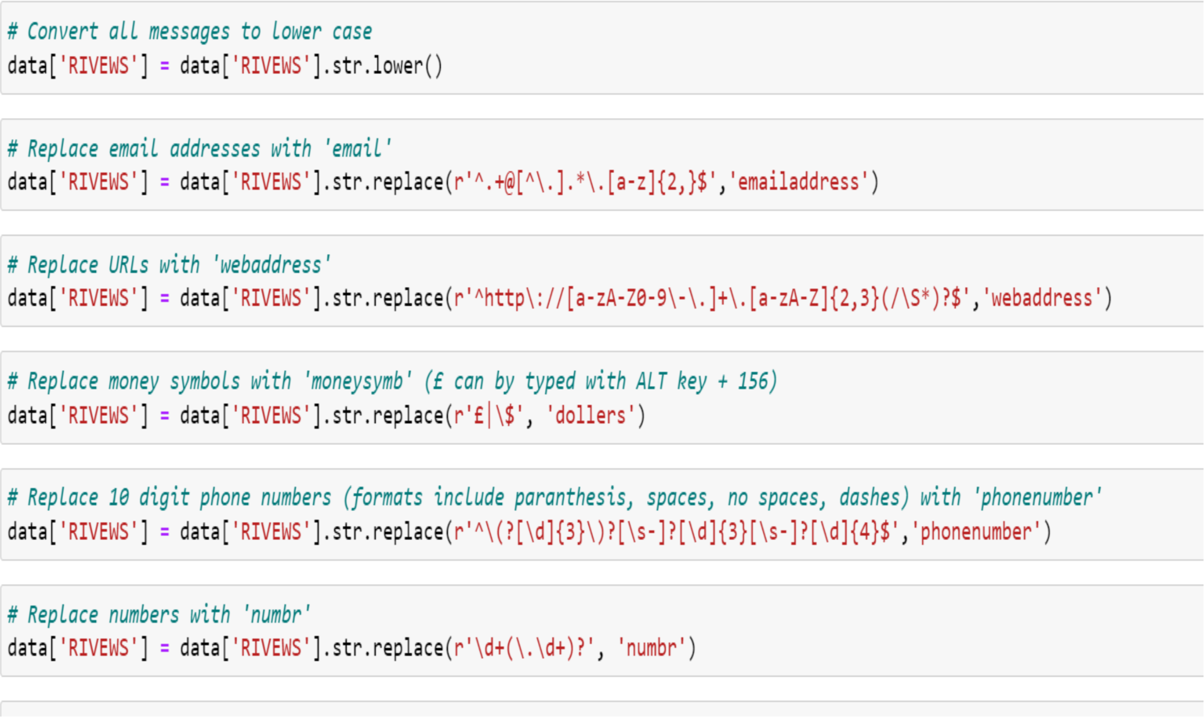
4) Then after we made a new Column of Length which contains the totallength of the Comment ..



```
data['length'] = data['RIVEWS'].str.len()
data.head(7)
```

```
     SITE NAME   PRODUCT NAME                          RIVEWS   RATINGS   len
```

5) Then after we Convert all comments in lower case, we replaced email addresses with 'email', URLs with 'webaddress’, money symbols with 'moneysymb' (£ can by typed with ALT key + 156), 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber', and numbers with 'numbr'…



```
# Convert all messages to lower case
data['RIVEWS'] = data['RIVEWS'].str.lower()
```

```
# Replace email addresses with 'email'
data['RIVEWS'] = data['RIVEWS'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$','emailaddress')
```

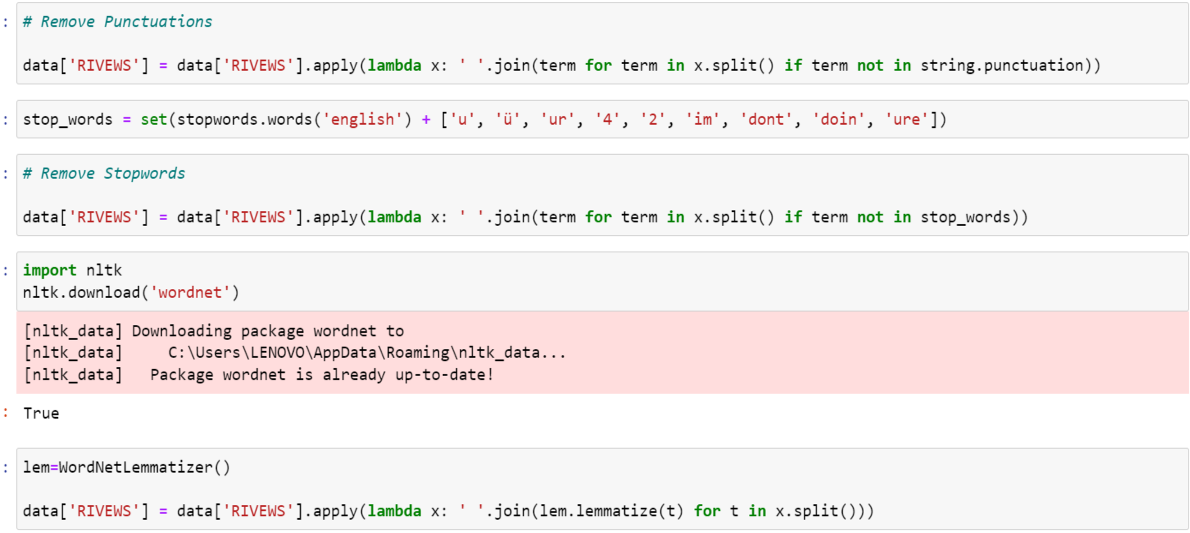
```
# Replace URLs with 'webaddress'
data['RIVEWS'] = data['RIVEWS'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$','webaddress')
```

```
# Replace money symbols with 'moneysymb' (£ can by typed with ALT key + 156)
data['RIVEWS'] = data['RIVEWS'].str.replace(r'£|\$', 'dollers')
```

```
# Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber'
data['RIVEWS'] = data['RIVEWS'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$','phonenumber')
```

```
# Replace numbers with 'numbr'
data['RIVEWS'] = data['RIVEWS'].str.replace(r'\d+(\.\d+)?', 'numbr')
```

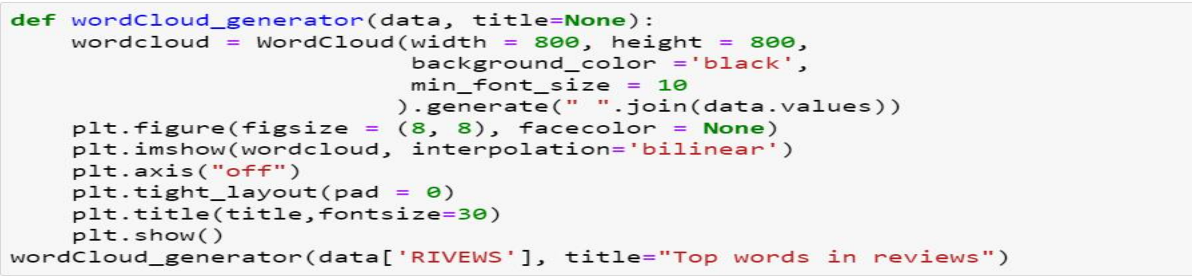
6) Then after we apply lambda function to remove the Stopwords and punctuations from the comment and then we used WordNetLemmatizer so that we can make Lemmas and words can bereduced into their meaningful roots..



```
: # Remove Punctuations

  data['RIVEWS'] = data['RIVEWS'].apply(lambda x: ' '.join(term for term in x.split() if term not in string.punctuation))

: stop_words = set(stopwords.words('english') + ['u', 'ü', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure'])

: # Remove Stopwords

  data['RIVEWS'] = data['RIVEWS'].apply(lambda x: ' '.join(term for term in x.split() if term not in stop_words))

: import nltk
  nltk.download('wordnet')

  [nltk_data] Downloading package wordnet to
  [nltk_data]     C:\Users\LENOVO\AppData\Roaming\nltk_data...
  [nltk_data]   Package wordnet is already up-to-date!

: True

: lem=WordNetLemmatizer()

  data['RIVEWS'] = data['RIVEWS'].apply(lambda x: ' '.join(lem.lemmatize(t) for t in x.split()))
```

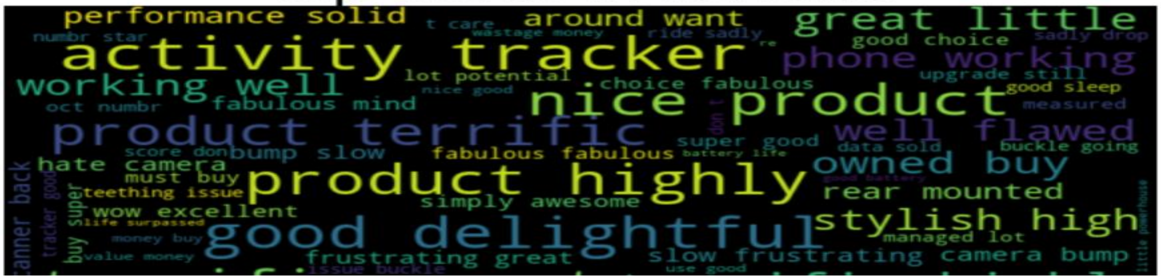
7) Then after lemmatizing the comment we made a new column(name as Clear_lenght) so that we can check the original nad clear length separately by doing these cleanings(like stopwords, punctuations andmany more as written above)..

8) Then we used wordcloud for some words that are Loud.

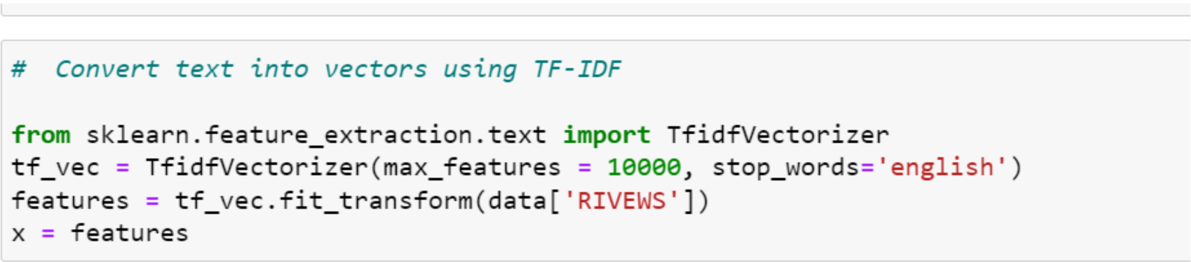


```
def wordCloud_generator(data, title=None):
    wordcloud = WordCloud(width = 800, height = 800,
                          background_color ='black',
                          min_font_size = 10
                         ).generate(" ".join(data.values))
    plt.figure(figsize = (8, 8), facecolor =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

Top words in reviews



9) Then after this we are ready to convert the comments into the Vector form as machine can not understand object/strings with thehelp of TF-IDF Vectorizer…



```
#  Convert text into vectors using TF-IDF

from sklearn.feature_extraction.text import TfidfVectorizer
tf_vec = TfidfVectorizer(max_features = 10000, stop_words='english')
features = tf_vec.fit_transform(data['RIVEWS'])
x = features
```

10) Now we are ready for the model building..

Algorithm used in this project:-

For building machine learning models there are several modelspresent inside the Sklearn module.

Sklearn provides two types of models i.e. regression and

classification. Our dataset's target variable is to predict whether fraud is reported or not. So for this kind of problem we use classification models.

But before the model fitting we have to seprate the predictor and target variable, then we pass this variable to the train_test_split method to create the training set and testing setfor the model training and prediction.

We can build as many models as we want to compare the accuracy given by these models and to select the best model among them.

I have selected 5 models:

1. MultinomialNB
2. Logistic Regression
3. Decision Tree Classifier
4. Random forest Classifier
5. Gradient boosting Classifier

BEST Algo. From these And why?

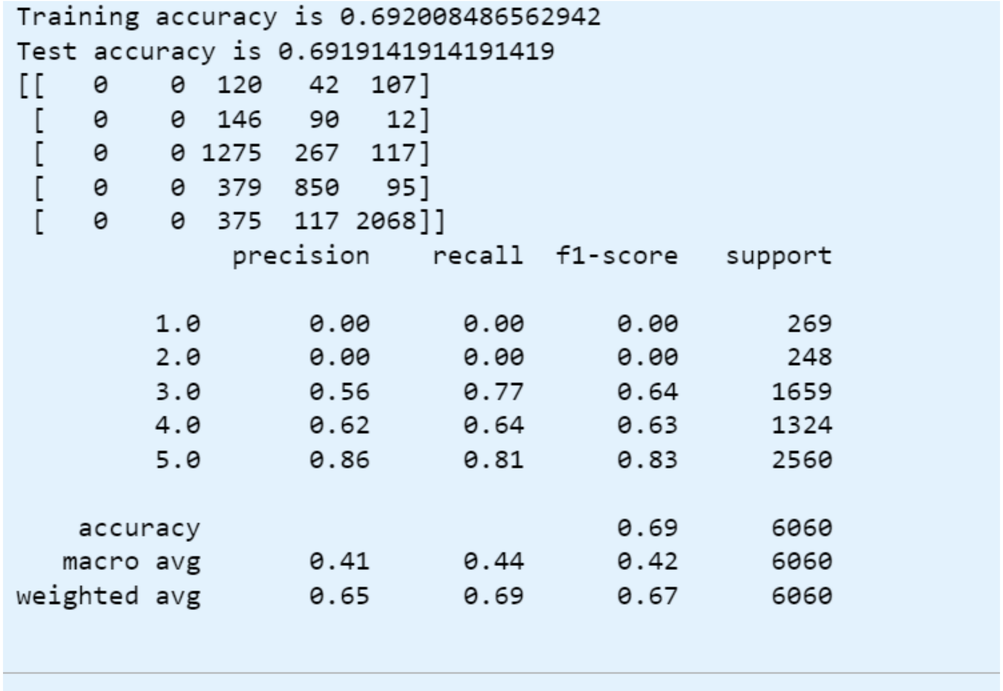
Random Forest classifier is the best algo. From all of these algorithms which is used in this data to predict because the difference between the cross val score and the accuracy score is minimum for the Random Forest algo. and it also gives BETTER ACCURACY(approx. 70%)that's why we USED THISAlgo.

- Lets hypertune this algo with the help of Grid searchCV Lets hypertune this algo with the help of GridsearchCV

- **Save the model for later predictions**

# Now my model is ready to predict

CONCLUSIONS



```
Training accuracy is 0.692008486562942
Test accuracy is 0.6919141914191419
[[   0    0  120   42  107]
 [   0    0  146   90   12]
 [   0    0 1275  267  117]
 [   0    0  379  850   95]
 [   0    0  375  117 2068]]
              precision    recall  f1-score   support

         1.0       0.00      0.00      0.00       269
         2.0       0.00      0.00      0.00       248
         3.0       0.56      0.77      0.64      1659
         4.0       0.62      0.64      0.63      1324
         5.0       0.86      0.81      0.83      2560

    accuracy                           0.69      6060
   macro avg       0.41      0.44      0.42      6060
weighted avg       0.65      0.69      0.67      6060
```

So these are the confusion matrix results of the Random algo.

Hence my model is ready to predict….