RATINGS PREDICTION

Submitted by:

ASHOK KUMAR SHARMA

ACKNOWLEDGMENT

I would like to express my deep sense of gratitude to my SME (Subject Matter Expert) **Mr. Shubham Yadav** as well as **Flip Robo Technologies** who gave me the golden opportunity to do this data scraping and analysis project on **Ratings Prediction**, which also helped me in doing lots of research and I came to know about so many new things.

I am very much thankful to **Mr. Shankargouda Tegginmani, Trainer (DataTrained)**, for their valuable guidance, keen interest and encouragement at various stages of my training period which eventually helped me a lot in doing this project.

I also acknowledge with thanks for suggestion and timely guidance, which I have received from my SME Mr. Shubham Yadav during this project, which immensely helped me in the evaluation of my ideas on the project.

ASHOK KUMAR SHARMA

INTRODUCTION

- Business Problem Framing

  Websites and online stores increasingly rely on rating systems and interactive elements for visitors and customers: users leave ratings on websites or give their opinions on products and companies using the comment boxes embedded on the page. The added value for users is clear: Customers and website visitors often gain important information through ratings and can read other user's experiences before investing in a product, service, or company. Since it's not possible to take a closer look at products online, these ratings and reviews fill information gaps. Online shopping is convenient, practical, and fast, but nonetheless there's a distance between the provider and the customer. However, if a website contains **ratings or a comment box**, this can help to close the distance between the provider and the customer: Customers can then use the feedback to help each other decide whether to go ahead with the purchase by providing information on the function, range, and value of a product.

- ## Conceptual Background of the Domain Problem

  This project is related a website where people write different reviews for technical products. Now a new feature has been added to 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 the requirement is to predict ratings for the reviews which were written in the past for which website don't have a rating. So, we have to build an application which can predict the rating by seeing the review.

- ## Review of Literature

  1. What is rating?

     Rating is a classification or ranking of someone or something based on a comparative assessment of their quality, standard, or performance.

- ## Motivation for the Problem Undertaken

  This model will be used by the websites where reviews are available but not the corresponding rating(s). This will help in assigning particular rating(s) for the given reviews.

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

  For checking datatypes and null values, pandas.DataFrame.info() method has been used. To change the column name pandas.DataFrame.rename() method has been used and to drop the null values pandas.DataFrame.dropna() method has been used. To replace and remove the certain terms and punctuations, pandas.Series.str.replace() method with regular expression has been used. To get rid of stop words, nltk.corpus.stopwords() method has been used.

- ## Data Sources and their formats

  The dataset is being created by scrapping different e-commerce websites and saved in .CSV (Comma Separated Values) format. Dataset consists of 31585 rows with 3 columns as explained below:

  1. **`Ratings:`** Rating of the product i.e., 1, 2, 3, 4, & 5.
  2. **`Review Title:`** Title for the review.
  3. **`Review Description:`** Description of review.

- ## Data Pre-processing Done

  The following pre-processing pipeline is required to perform model prediction:

  1. Load Dataset
  2. Merge feature `Review Titles` and `Review Descriptions` to `review_text`
  3. Drop columns `Review Titles and Review Descriptions`
  4. Treat Null Values by dropping null value rows using pandas dropna() method.
  5. Convert review_text to lower-case.
  6. Remove punctuations, leading whitespaces, trailing whitespaces and replace money symbols with 'dollars', numbers with 'numbr', white space between terms with single space.
  7. Remove Stop Words
  8. Convert Text into Vectors using TfidfVectorizer
  9. Load Serialized Model
  10. Predict Output by Supplying Input.

- Data Inputs- Logic- Output Relationships

Input	Logic (algorithm)	Output
`Review_text (object)`	MultinomialNB **SGDClassifier** KNeighborsClassifier DecisionTreeClassifier	rating

There is 1 input variable needs to be provided to the logic to get the output i.e. rating. Logic highlighted in green i.e. SGDClassifier is the best performing algorithm among all other logics on this dataset.

- Hardware and Software Requirements and Tools Used

During this project, following set of hardware is being used:

RAM: 8 GB

CPU: AMD A8 Quad Core 2.2 Ghz

GPU: AMD Redon R5 Graphics

and the following software and tools is being used:

a. Python
b. Jupyter Notebook
c. Anaconda

With following libraries and packages:

- Pandas
- Numpy
- Matplotlib
- Seaborn
- nltk
- wordcloud
- sys

- Ipython
- timeit
- sklearn

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  To solve this problem following steps are used:

  1. Load Dataset using pandas
  2. Merge feature **Review Titles and Review Descriptions** to ***review_text***.
  3. Drop columns Review Titles and Review Descriptions.
  4. Treat Null Values by dropping null value rows using pandas dropna() method.
  5. Convert review_text to lower-case.
  6. Remove punctuations, leading whitespaces, trailing whitespaces and replace money symbols with 'dollars', numbers with 'numbr', white space between terms with single space.
  7. Remove Stop Words.
  8. Convert Text into Vectors using TfidfVectorizer.
  9. Separate Input and Output Variables.
  10. Train & Test the Model by supplying Input and Output Variables.

- Testing of Identified Approaches (Algorithms)

  Following are the list of algorithms used for training and testing:

  1. MultinomialNB
  2. SGDClassifier

3. KNeighborsClassifier
4. DecisionTreeClassifier

- Run and Evaluate selected models

  A total of 4 algorithm has been used on this dataset for training testing purpose, these are MultinomialNB, SGDDClassifier, KNeighborsClassifier and DecisionTreeClassifier. To perform training and testing operation(s) following functions has been defined for which codes are as follows:

```
#function to get best random state
def get_best_random_state(model,X,Y,t_size=0.25,rs_range=range(1,301,50)):
    best_rstate = 0
    best_accuracy_score = 0
    random_state_message = "\r"

    for i in tqdm.tqdm(rs_range,desc=f"Best_Random_State => {model}"):
        X_train, X_test, Y_train, Y_test =
train_test_split(X,Y,test_size=t_size,random_state=i)
        model.fit(X_train, Y_train)
        y_pred = model.predict(X_test)
        a_score = accuracy_score(Y_test,y_pred)

        if a_score > best_accuracy_score:
            best_accuracy_score = a_score
            best_rstate = i

        random_state_message += f"[{i}: {round(a_score*100,2)}]<--->"
        sys.stdout.write(random_state_message)

    sys.stdout.write(f"\n\nBest Random State: {best_rstate} found with
Accuracy: {best_accuracy_score}")
    return best_rstate, best_accuracy_score
#End of function
```

```
#function to get best cv score
def get_best_cv(model,X_train,Y_train,parameters,cv_range=range(5,25,5)):
    best_cv_score = 0
    best_cv = 0

    cv_message = "\r"
    for i in tqdm.tqdm(cv_range,desc=f"Best_CV => {model}"):
        gscv = GridSearchCV(model,parameters)
        gscv.fit(X_train,Y_train)

        cv_score =
cross_val_score(gscv.best_estimator_,X_train,Y_train,cv=i).mean()

        if cv_score > best_cv_score:
            best_cv_score = cv_score
            best_cv = i

        cv_message += f"[{i}:{round(cv_score*100,2)}]<--->"
        sys.stdout.write(cv_message)

    sys.stdout.write(f"\n\nBest CV: {best_cv} found with Cross Val Score:
{best_cv_score}")

    return best_cv, best_cv_score
#End of function

#function to build models
def
build_models(models,X,Y,t_size=0.25,rs_range=range(1,301,50),cv_range=range
(5,25,5)):
    for i in tqdm.tqdm(models,desc="Building Models"):

sys.stdout.write("\n=============================================================
================================\n")
        sys.stdout.write(f"Current Model in Progress: {i} ")

sys.stdout.write("\n=============================================================
================================\n")
```

```
        #start time
        start_time = timeit.default_timer()

        #Find the best random state
        best_random_state, best_accuracy_score =
get_best_random_state(models[i]['name'],X,Y,t_size,rs_range)
        sys.stdout.write("\n")

        #Spliting train and test data using train_test_split method with
best random state value
        X_train,X_test,Y_train,Y_test =
train_test_split(X,Y,test_size=t_size,random_state=best_random_state)

        #Find the best CV
        best_cv, best_cv_score =
get_best_cv(models[i]['name'],X_train,Y_train,models[i]['parameters'],cv_ra
nge)
        sys.stdout.write("\n\nBuilding Model...")

        #Training the model using best CV
        gscv =
GridSearchCV(models[i]['name'],models[i]['parameters'],cv=best_cv)
        gscv.fit(X_train,Y_train)

        #Testing model
        y_pred = gscv.best_estimator_.predict(X_test)

        #Recording model performance
        model_accuracy_score = accuracy_score(Y_test,y_pred)
        model_confusion_matrix = confusion_matrix(Y_test,y_pred)
        model_classification_report = classification_report(Y_test,y_pred)

        #end time
        end_time = timeit.default_timer()
        sys.stdout.write(f"Completed in [{end_time-start_time} sec.]")
```

```
        #storing model specifications
        models[i]['initial_accuracy_score'] = best_accuracy_score
        models[i]['best_random_state'] = best_random_state
        models[i]['x_train'] = X_train
        models[i]['x_test'] = X_test
        models[i]['y_train'] = Y_train
        models[i]['y_test'] = Y_test
        models[i]['best_cv'] = best_cv
        models[i]['best_cv_score'] = best_cv_score
        models[i]['gscv'] = gscv
        models[i]['y_predict'] = y_pred
        models[i]['final_accuracy'] = model_accuracy_score
        models[i]['confusion_matrix'] = model_confusion_matrix
        models[i]['classification_report'] = model_classification_report
        models[i]['build_time'] = f"{end_time - start_time} (in sec.)"


sys.stdout.write("\n============================================================
==============================\n\n\n")


    return models
#End of function


#function to display model performance
def display_performance(models):
    model_names = []
    model_initial_score = []
    model_cross_val_score = []
    model_final_score = []
    model_build_time = []
    for i in models:
        model_names.append(i)
        model_initial_score.append(models[i]['initial_accuracy_score'])
        model_cross_val_score.append(models[i]['best_cv_score'])
        model_final_score.append(models[i]['final_accuracy'])
```

```
        model_build_time.append(models[i]['build_time'])


    model_df = pd.DataFrame({
        "Name": model_names,
        "Initial Score": model_initial_score,
        "Cross Val Score": model_cross_val_score,
        "Final Score": model_final_score,
        "Build Time": model_build_time,
    })


    model_df['Difference (Final Score - Cross Val Score)'] =
model_df['Final Score'] - model_df['Cross Val Score']
    display(model_df)


    for i in models:
        print("============================================================")
        print(f"for model: {i}")
        print("============================================================")
        print("CLASSIFICATION REPORT")
        print(models[i]['classification_report'])
        print("CONFUSION MATRIX")
        print(models[i]['confusion_matrix'])

print("============================================================\n\n")


    return
#End of function


#List of models for training & testing
models = {
    "MultinomialNB":{
        "name": MultinomialNB(),
        "parameters":{
            "alpha": [1.0]
        }
```

```
    },
    "SGDClassifier":{
        "name": SGDClassifier(),
        "parameters":{
            "loss":['hinge','modified_huber'],
            "alpha":[0.001,0.0001,0.00001],
            "n_jobs":[-1],
            "learning_rate":['optimal'],
            "max_iter":[100]
        }
    },
    "KNeighborsClassifier":{
        "name": KNeighborsClassifier(),
        "parameters":{
            "n_neighbors": [5,10],
            "weights": ['uniform','distance'],
            "n_jobs": [-1]
        }
    },
    "DecisionTreeClassifier":{
        "name": DecisionTreeClassifier(),
        "parameters":{
            "criterion": ['gini','entropy'],
            "splitter": ['best','random']
        }
    }
}


#building models
build_model = build_models(models,X,Y)
```

```
Building Models: 100%
4/4 [1:08:33<00:00, 1456.64s/it]

================================================================================
============
```

```
Current Model in Progress: MultinomialNB
================================================================================
============
```
Best_Random_State => MultinomialNB(): 100%
6/6 [00:00<00:00, 13.60it/s]
```
[1: 72.87]<--->[51: 73.5]<--->[101: 72.89]<--->[151: 72.65]<--->[201: 72.09
]<--->[251: 73.06]<--->

Best Random State: 51 found with Accuracy: 0.7349993657237093
```
Best_CV => MultinomialNB(): 100%
4/4 [00:02<00:00, 1.23it/s]
```
[5:72.43]<--->[10:72.43]<--->[15:72.48]<--->[20:72.41]<--->

Best CV: 15 found with Cross Val Score: 0.7247849265080627

Building Model...Completed in [4.174949800000007 sec.]
================================================================================
============




================================================================================
============
Current Model in Progress: SGDClassifier
================================================================================
============
```
Best_Random_State => SGDClassifier(): 100%
6/6 [00:05<00:00, 1.05it/s]
```
[1: 74.72]<--->[51: 75.16]<--->[101: 74.05]<--->[151: 74.88]<--->[201: 74.3
5]<--->[251: 74.78]<--->

Best Random State: 51 found with Accuracy: 0.7516174045414182
```
Best_CV => SGDClassifier(): 100%
4/4 [01:04<00:00, 16.07s/it]
```
[5:74.82]<--->[10:75.09]<--->[15:74.92]<--->[20:74.94]<--->

Best CV: 10 found with Cross Val Score: 0.7508770385951357

Building Model...Completed in [94.5427563 sec.]
================================================================================
============




================================================================================
============
Current Model in Progress: KNeighborsClassifier
================================================================================
============
```

Best_Random_State => KNeighborsClassifier(): 100%
6/6 [01:12<00:00, 11.97s/it]
[1: 42.4]<--->[51: 43.27]<--->[101: 43.61]<--->[151: 42.03]<--->[201: 41.47
]<--->[251: 42.5]<--->

Best Random State: 101 found with Accuracy: 0.4361283775212483
Best_CV => KNeighborsClassifier(): 100%
4/4 [15:52<00:00, 237.30s/it]
[5:58.74]<--->[10:56.76]<--->[15:56.49]<--->[20:55.87]<--->

Best CV: 5 found with Cross Val Score: 0.58738916057952

Building Model...Completed in [1241.6716397 sec.]
==========
==========

==========
==========
Current Model in Progress: DecisionTreeClassifier
==========
==========
Best_Random_State => DecisionTreeClassifier(): 100%
6/6 [01:25<00:00, 14.49s/it]
[1: 66.7]<--->[51: 67.25]<--->[101: 66.48]<--->[151: 66.64]<--->[201: 65.75
]<--->[251: 66.31]<--->

Best Random State: 51 found with Accuracy: 0.6724597234555373
Best_CV => DecisionTreeClassifier(): 100%
4/4 [26:26<00:00, 430.60s/it]
[5:66.15]<--->[10:66.08]<--->[15:66.2]<--->[20:66.29]<--->

Best CV: 20 found with Cross Val Score: 0.6629164145759225

Building Model...Completed in [2772.7051459000004 sec.]
==========
==========

```
#displaying model performances
display_performance(models)
```

	Name	Initial Score	Cross Val Score	Final Score	Build Time	Difference (Final Score - Cross Val Score)
0	MultinomialNB	0.734999	0.724785	0.734999	4.174949800000007 (in sec.)	0.010214
1	SGDClassifier	0.751617	0.750877	0.759229	94.5427563 (in sec.)	0.008352
2	KNeighborsClassifier	0.436128	0.587389	0.550679	1241.6716397 (in sec.)	-0.036710
3	DecisionTreeClassifier	0.672460	0.662916	0.677280	2772.7051459000004 (in sec.)	0.014364

```
============================================================
for model: MultinomialNB
============================================================
CLASSIFICATION REPORT
              precision    recall  f1-score   support

           1       0.74      0.84      0.79      1595
           2       0.76      0.55      0.63      1605
           3       0.61      0.63      0.62      1545
           4       0.71      0.78      0.74      1517
           5       0.86      0.88      0.87      1621

    accuracy                           0.73      7883
   macro avg       0.74      0.73      0.73      7883
weighted avg       0.74      0.73      0.73      7883

CONFUSION MATRIX
[[1337  102  120   27    9]
 [ 274  877  365   72   17]
 [ 144  151  966  244   40]
 [  34   27  105 1185  166]
 [  11    3   28  150 1429]]
============================================================


============================================================
for model: SGDClassifier
============================================================
CLASSIFICATION REPORT
              precision    recall  f1-score   support
```

```
           1       0.76      0.87      0.81      1595
           2       0.73      0.58      0.65      1605
           3       0.65      0.63      0.64      1545
           4       0.76      0.78      0.77      1517
           5       0.87      0.93      0.90      1621

    accuracy                           0.76      7883
   macro avg       0.76      0.76      0.75      7883
weighted avg       0.76      0.76      0.75      7883

CONFUSION MATRIX
[[1394   93   82   21    5]
 [ 268  934  324   65   14]
 [ 135  209  971  191   39]
 [  30   36   99 1186  166]
 [   7    0   18   96 1500]]
============================================================


============================================================
for model: KNeighborsClassifier
============================================================
CLASSIFICATION REPORT
              precision    recall  f1-score   support

           1       0.71      0.64      0.67      1536
           2       0.32      0.75      0.45      1579
           3       0.61      0.32      0.42      1562
           4       0.76      0.43      0.55      1570
           5       0.92      0.61      0.74      1636

    accuracy                           0.55      7883
   macro avg       0.66      0.55      0.56      7883
weighted avg       0.67      0.55      0.57      7883

CONFUSION MATRIX
[[ 982  482   52   17    3]
 [ 203 1183  158   31    4]
 [ 143  820  499   87   13]
 [  50  699   84  673   64]
 [  14  516   20   82 1004]]
============================================================


============================================================
for model: DecisionTreeClassifier
```

```
=====================================================
CLASSIFICATION REPORT
              precision    recall  f1-score   support

           1       0.69      0.72      0.71      1595
           2       0.58      0.54      0.56      1605
           3       0.55      0.56      0.55      1545
           4       0.72      0.72      0.72      1517
           5       0.83      0.85      0.84      1621

    accuracy                           0.68      7883
   macro avg       0.67      0.68      0.68      7883
weighted avg       0.68      0.68      0.68      7883

CONFUSION MATRIX
[[1143  236  159   36   21]
 [ 277  866  353   76   33]
 [ 159  293  858  162   73]
 [  42   76  161 1089  149]
 [  24   27   41  146 1383]]
=====================================================
```

From the above model performance comparison it is clear that **SGDClassifier** out-performs the other models with **accuracy_score of 75.92%** and **lowest difference between accuracy_score and cross_val_score**. Therefore, continuing with **SGDClassifier** as final model.

- Key Metrics for success in solving problem under consideration

  To find out best performing model following metrices are used:

  1. Accuracy Score: It is used to check the model performance score between 0.0 to 1.0
  2. Confusion Matrix: A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

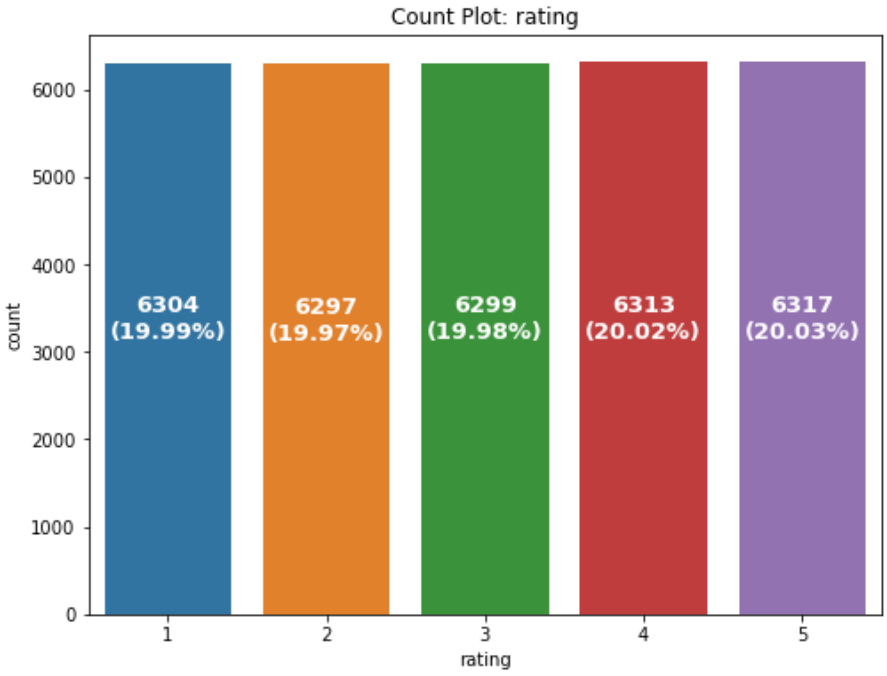
3. Classification Report: A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False.

- Visualizations

To better understand the data, following types of visualizations have been used: 1. Univariate.

1. Univariate Analysis: Univariate analysis is the simplest form of data analysis where the data being analysed contains only one variable. In this project, distribution plot, count plot and box plot has been used.

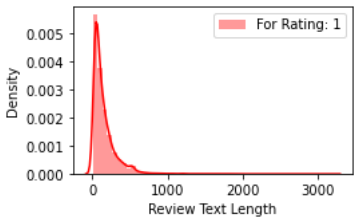
Count Plot (countplot):

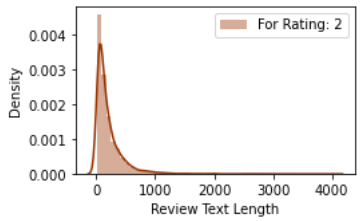


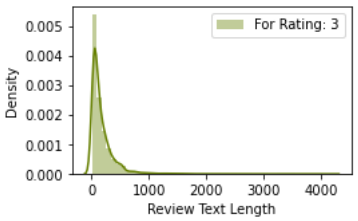
Remarks:

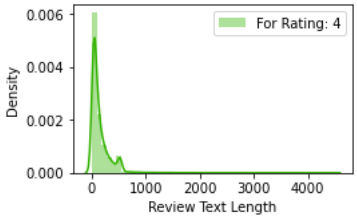
- There are almost equal number of records are availble for all ratings i.e. from 1 to 5.

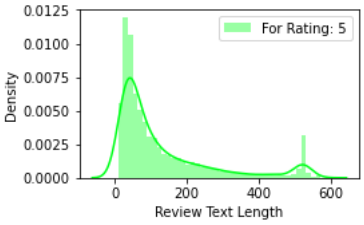
Distribution Plot (distplot):





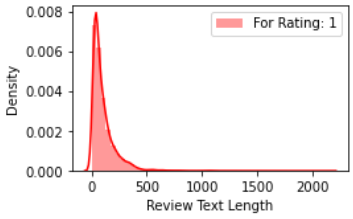


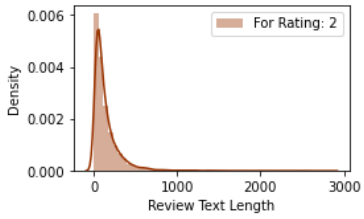


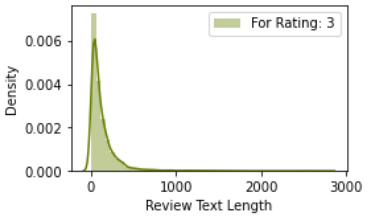


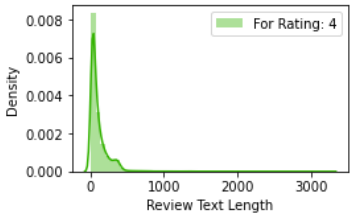
Remarks:

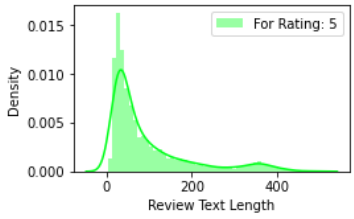
- Rating 2, 3, 4 has almost similer review text length and higher than Rating 1 and 5.
- Rating 5 has lowest review text lengths.







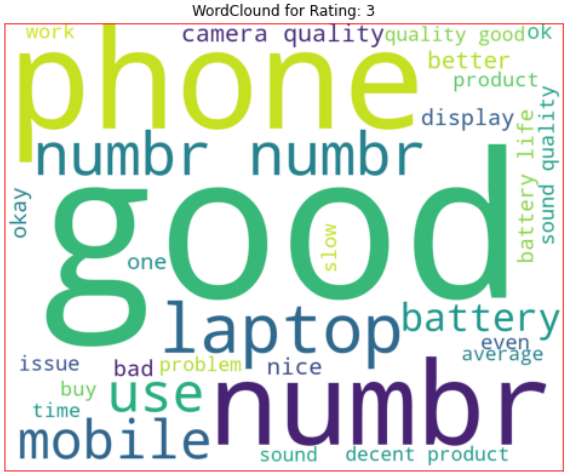




Remarks:

- Review text length reduced by almost 1000 characters for Rating 1 to 4 while for Rating 5 it is almost 200 characters

Displaying with WordCloud (wordcloud):



Remarks:

`for Rating: 1`

- It mostly consists of words like laptop, waste, money, slow, worst, issue, horrible etc.

for Rating: 2

- It mostly consists of words like phone, good, printer, product, problem, issue, bad, poor, slow etc.

for Rating: 3

- It mostly consists of words like phone, good, laptop, problem, bad, issue, slow, life, average, nice etc.

for Rating: 4

- It mostly consists of words like laptop, good, value, money, nice, performance, great, better, wonderful etc.

for Rating: 5

- It mostly consists of words like laptop, excellent, must buy, great, perfect, super, awesome, mind blowing etc.

Interpretation of the Results

Starting with univariate analysis, with the help of countplot, it was found that the data consists of almost in equal amount for each rating (i.e., from 1 to 5). Moving further with the removal and replacement of certain terms (like, punctuations, extra spaces, numbers, money symbols) as well as removal of stop words, it was evident that the length of review text decreases by a large amount. This was also dipcted by using distribution plot. With the help of wordcloud, it was found that the rating 1 consists of words like waste, money, slow, worst, issue, horrible etc, rating 2 consists of words like problem, issue, bad, poor, slow etc., rating 3 consists of word like problem, bad, issue, slow, life, average, nice etc, rating 4 consists of word like good, value, money, nice, performance, great, better, wonderful etc. and rating 5 consists of words like excellent, must buy, great, perfect, super, awesome, mind blowing etc.

CONCLUSION

- Key Findings and Conclusions of the Study

  From the model performance comparison it is clear that **SGDClassifier** out-performs the other models with **accuracy_score of 75.92%** and **lowest difference between accuracy_score and cross_val_score**. Therefore, continuing with **SGDClassifier** as final model.

- Learning Outcomes of the Study in respect of Data Science

  During the data analysis, review_text feature contains null values which I have dropped. But these values can also be replaced with some other values which might impact the model performance either in positive or negative way. As of now, I am finishing this project with my current approach which gives the **final accuracy score of 75.92% and cross_val_score: 75.08%** and this can be further improved by training with more specific data.

- Limitations of this work and Scope for Future Work

  Current model is limited to technical product rating(s) and reviews data but this can further be improved for other sectors of ecommerce rating(s) prediction by training the model accordingly. The overall score can also be improved further by training the model with more specific data.