FLIP ROBO

2022

RATING PREDICTION PROJECT

Submitted By:

Bhushan Kumar Sharma

Intern Data Scientist

ACKNOWLEDGMENT

*****

I would like to express my special gratitude to “Flip Robo” team, who has given me this opportunity to deal with this dataset and it has helped me to improve my model building skills.

A huge thanks to my academic team “Datatrained” who is the reason behind what I am today. Last but not least my parents who have been my backbone in every step of my life. And also thank to many other persons who has helped me directly or indirectly to complete the project.

Following are the external references which I used:

www.geeksforgeeks.org

www.stackoverflow.com

www.w3school.com

www.google.com

Datatrained Lectures

INTRODUCTION

➢ Business Problem Framing

The rise in E-commerce has brought a significant rise in the importance of customer reviews. There are hundreds of review sites online and massive amounts of reviews for every product. Customers have changed their way of shopping and according to a recent survey, 70 percent of customers say that they use rating filters to filter out low rated items in their searches. The ability to successfully decide whether a review will be helpful to other customers and thus give the product more exposure is vital to companies that support these reviews, companies like Google, Amazon and Yelp.

There are two main methods to approach this problem. The first one is based on review text content analysis and uses the principles of natural language process (the NLP method). This method lacks the insights that can be drawn from the relationship between costumers and items. The second one is based on recommender systems, specifically on collaborative filtering, and focuses on the reviewer's point of view.

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 rating. So we, we have to build an application which can predict the rating by seeing the review.

➢ Conceptual Background of the Domain Problem

Recommendation systems are an important units in today's e-commerce applications, such as targeted advertising, personalized marketing and information retrieval. In recent years, the importance of contextual information has motivated generation of personalized recommendations according to the available contextual information of users. Compared to the traditional systems which mainly utilize user's rating history, review-based recommendation hopefully provide more relevant results to users. We introduce a review-based recommendation approach that obtains contextual information by mining user reviews. The proposed approach relate to features obtained by analysing textual reviews using methods developed in Natural Language Processing (NLP) and information retrieval discipline to compute a utility function over a given item. An item utility is a measure that shows how much it is preferred according to user's current context. In our system, the context inference is modelled as similarity between the user's reviews history and the item reviews history. As an example application, we used our method to mine contextual data from customer's reviews of technical products and use it to produce review-based rating prediction. The predicted ratings can generate recommendations that are item-based and should appear at the recommended items list in the product page. Our evaluations (surprisingly) suggest that our system can help produce better prediction rating scores in comparison to the standard prediction methods.

- **Review of Literature**

  - Some websites do not always offer structured information, and all do not leverage user's unstructured information, i.e. reviews, explicit social networks information is not always available and it is difficult to provide a good prediction for each user. For this problem the sentiment factor term is used to improve social recommendation.

- **Motivation for the Problem Undertaken**

The project was first provided to me by FlipRobo as a part of the internship program. The exposure to real world data and the opportunity to deploy my skillset in solving a real time problem has been the primary objective.

Many product reviews are not accompanied by a scale rating system, consisting only of a textual evaluation. In this case, it becomes daunting and time-consuming to compare different products in order to eventually make a choice between them.

Therefore, models able to predict the user rating from the text review are critically important. Getting an overall sense of a textual review could in turn improve consumer experience. However, the motivation for taking this project was that it is relatively a new field of research. Here we have many options but less concrete solutions.

The main motivation is to build a prototype of online hate and abuse review classifier which can used to classify hate and good comments so that it can be controlled and corrected according to the reviewer's choice..

- **Mathematical/ Analytical Modelling of the Problem**

The data was collected by using web scrapping for extracting review data. In web scrapping I have used selenium. In this dataset problem the Ratings can be 1, 2, 3, 4 or 5, which represents the likely ness of the product to the customer.

As it is containing multi class So clearly it is a multi-classification problem and I have to use all classification algorithms while building the model. We would perform one type of supervised learning algorithms: Classification. Here, we will only perform classification. Since there only 1 feature in the dataset, filtering the words is needed to prevent over fit.

In order to determine the regularization parameter, throughout the project in classification part, we would first remove email, phone number, web address, spaces and stops words etc. In order to further improve our models, we also performed TFID in order to convert the tokens from the train documents into vectors so that machine can do further processing. I have used all the classification algorithms while building model then tuned the best model and saved the best model.

- **Data Sources and their formats**

Scraped dataset is having 33285 records and 3 columns, in this dataset "Ratings" column is my target column and I have to apply machine learning algorithm accordingly.

- **Review_Title**: Title of the Review.
- **Review_Text**: Review content
- **Ratings**: Rating of review

	Review_Title	Reiew_Text	Ratings
0	Japanese efficiency in design, functionality a...	This is possibly going to be a game changer. M...	5 star
1	A good lightweight laptop with some future usa...	This is really cool laptop with top notch perf...	5 star
2	Compact and powerful	I have been using Fujitsu UH-X 11th (i7) gener...	5 star
3	Super light feature packed well built laptop	If you have the budget and want a no nonsense ...	5 star
4	Excellent Product	I brought this Laptop 3 days back. Excellent p...	5 star

Data Information

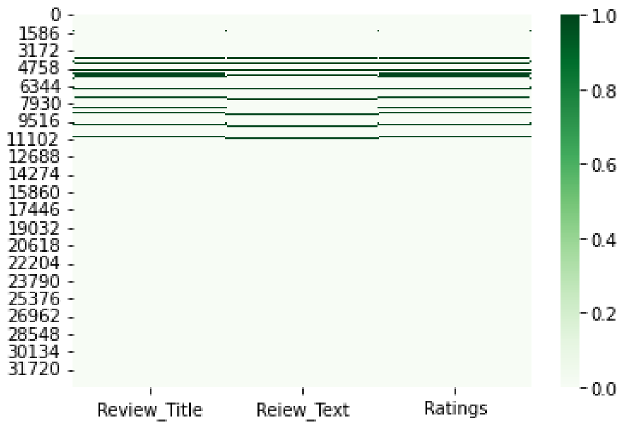
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33285 entries, 0 to 33284
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Review_Title  31496 non-null  object
 1   Reiew_Text    31445 non-null  object
 2   Ratings       31496 non-null  object
dtypes: object(3)
memory usage: 780.2+ KB
```

As we can see in the above output not-null values are not equal for every column, which is indicating that dataset is having null values.

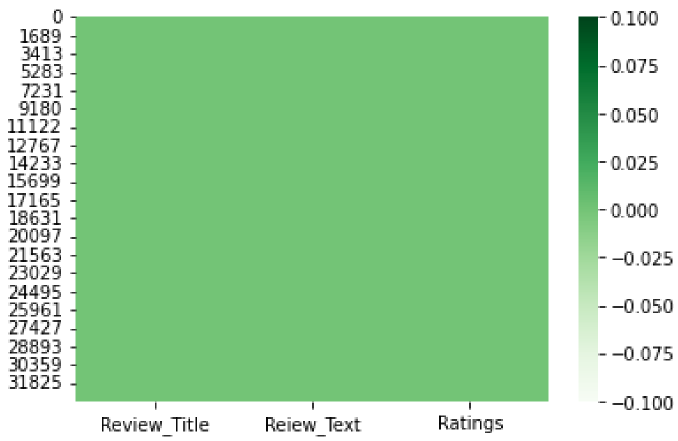
- **Data Pre-processing:**
  - **Null values identification:**

```
sns.heatmap(df.isnull(), cmap = 'Greens')
```



After found null values, I dropped the all null values from the dataset

```
df.dropna(inplace = True)
```



Note: **As heatmap is now clear, which indicating Now, no null values are present in the dataset.**

- *Converted all target values into [5,4,3,2,1] class only*

```
df['Ratings'].replace( {'2.0 out of 5 stars': 2, '3.0 out of 5 stars':3, '1.0 out of 5 stars':1, '5.0 out of 5 stars':5,
                        '4.0 out of 5 stars':4, '1':1, '2':2, '5':5, '3':3, '4':4 }, inplace = True )
df.Ratings.unique()
```

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

- *Created New columns for length to get amount of cleaned data records*

```
# New column for length of message
df['Length of Title'] = df['Review_Title'].str.len()
df['Length of Text'] = df['Reiew_Text'].str.len()
```

- *Converted all text into lower case:*

```
# Converting all msges into lower case

df['Review_Title'] = df['Review_Title'].apply(lambda x:x.lower())
df['Reiew_Text'] = df['Reiew_Text'].apply(lambda x:x.lower())
df.head()
```

- *Various operations perform to clean the review content and review title content*

```
# Replace email address with email:[
df['Review_Title'] = df['Review_Title'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$', 'email_address')
df['Reiew_Text'] = df['Reiew_Text'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$', 'email_address')
```

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

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

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

```
# Replace numbers with 'number'
df['Review_Title'] = df['Review_Title'].str.replace(r'\d+(\.\d+)?', 'number')
df['Reiew_Text'] = df['Reiew_Text'].str.replace(r'\d+(\.\d+)?', 'number')
```

```
# Remove punctuation
df['Review_Title'] = df['Review_Title'].str.replace(r'[^\w\d\s]', ' ')
df['Reiew_Text'] = df['Reiew_Text'].str.replace(r'[^\w\d\s]', ' ')
```

```
# Replace whitespace between terms with a single space:
df['Review_Title'] = df['Review_Title'].str.replace(r'\s+', ' ')
df['Reiew_Text'] = df['Reiew_Text'].str.replace(r'\s+', ' ')
```

```
# Remove leading and trailing whitespace
df['Review_Title'] = df['Review_Title'].str.replace(r'^\s+|\s+?$', ' ')
df['Reiew_Text'] = df['Reiew_Text'].str.replace(r'^\s+|\s+?$', ' ')
```

```
df['Review_Title'] = df['Review_Title'].str.replace(r'^\s+|\s+?$', ' ')
df['Reiew_Text'] = df['Reiew_Text'].str.replace(r'^\s+|\s+?$', ' ')
```

- *Now, Applied operation to remove stopwords*

```
# Remove stopwords
import nltk
nltk.download('stopwords')

import string
from nltk.corpus import stopwords
stop_words = stopwords.words('english') + ['u', 'ur', '4', '2','im','dont', 'doin', 'ure']
print(stop_words)
```

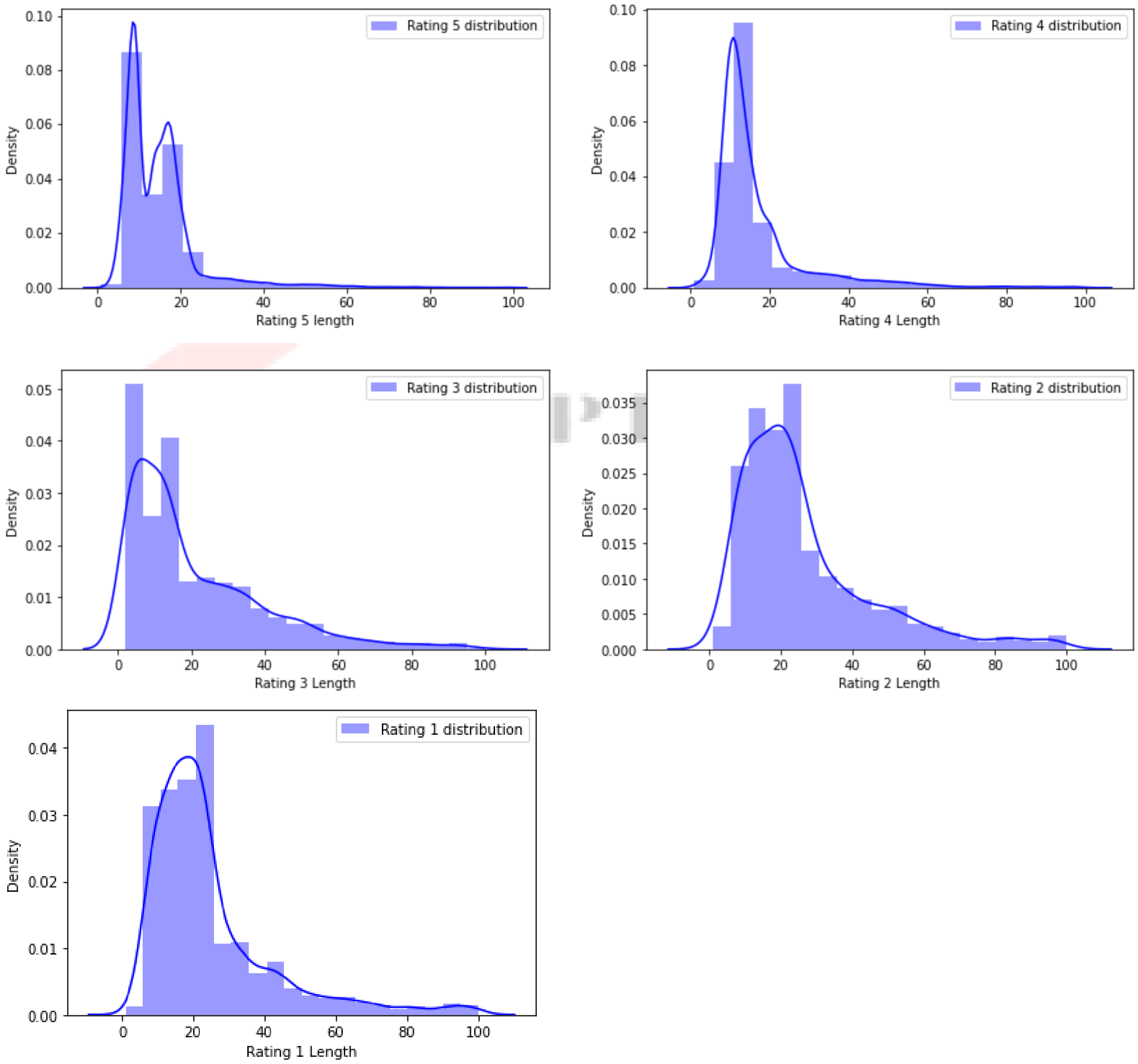
```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves',
```

```
df['Review_Title'] = df['Review_Title'].apply(lambda x : ' '.join(word for word in x.split() if word not in stop_words ))
df['Reiew_Text'] = df['Reiew_Text'].apply(lambda x : ' '.join(word for word in x.split() if word not in stop_words ))
```

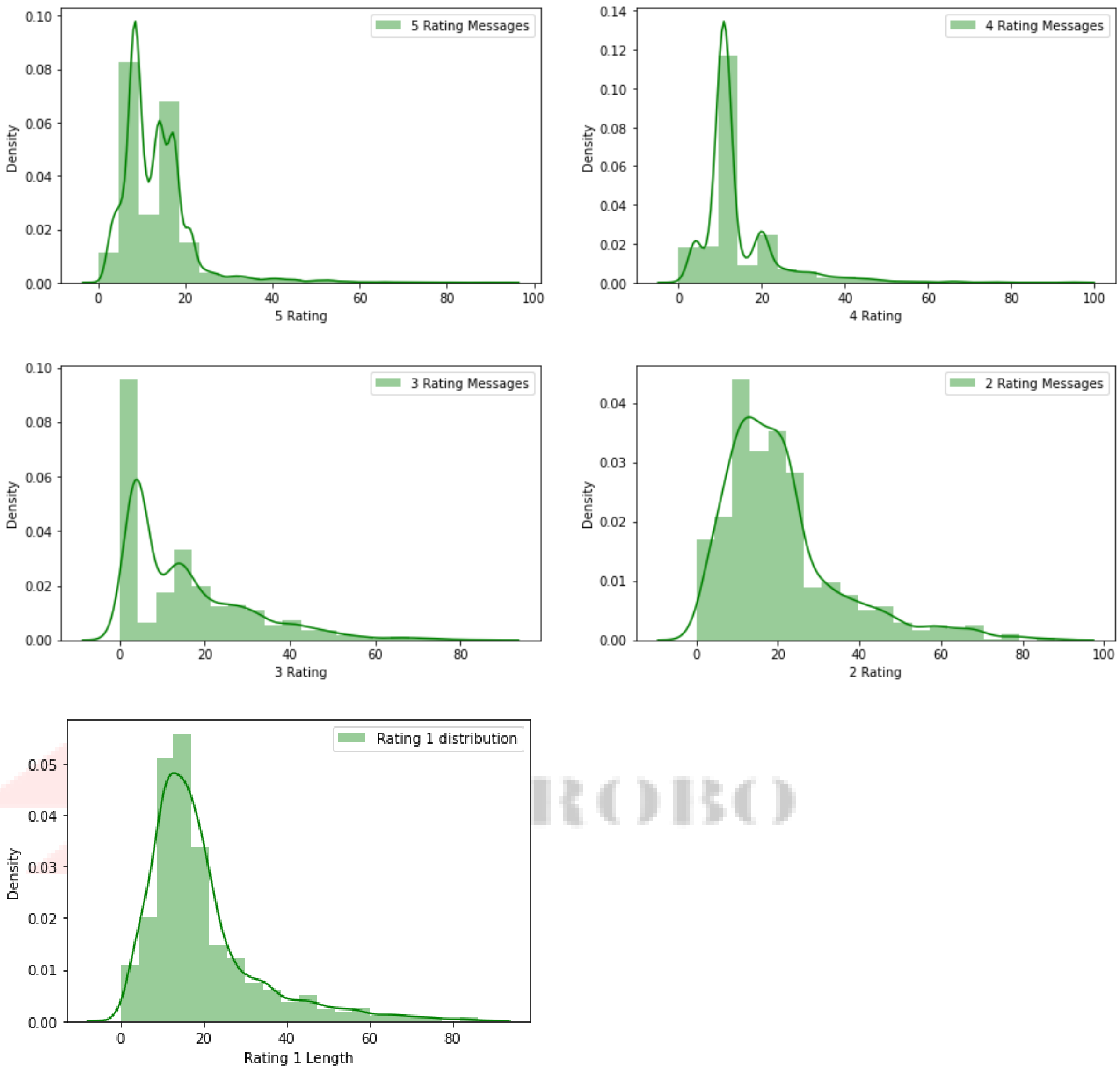
```python
# New column length after removing unwanted crupt data
df['Len of clean Review_Title'] = df.Review_Title.str.len()
df['Len of clean Reiew_Text'] = df.Reiew_Text.str.len()
```

By finding this Len of clean Review and Text we can see the content weightage, how much data cleaned by processing various operation to convert into this desired form.

- ***Review text Distribution before cleaning***

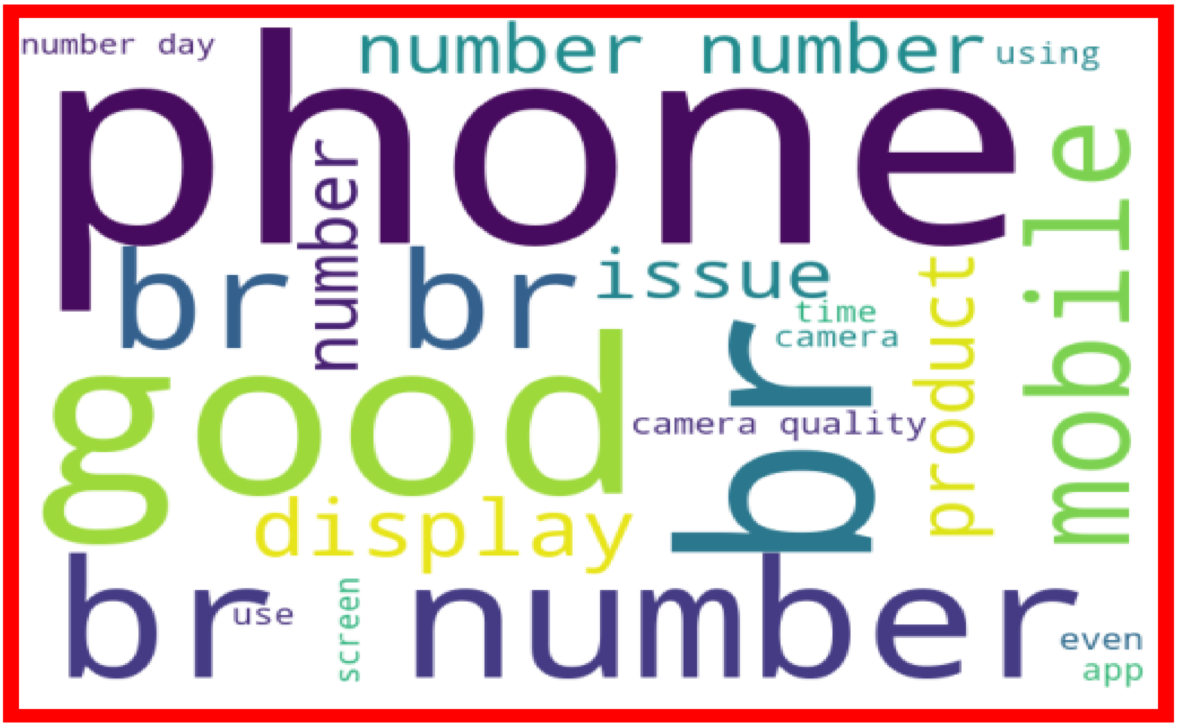


- ***Review Distribution after cleaning***



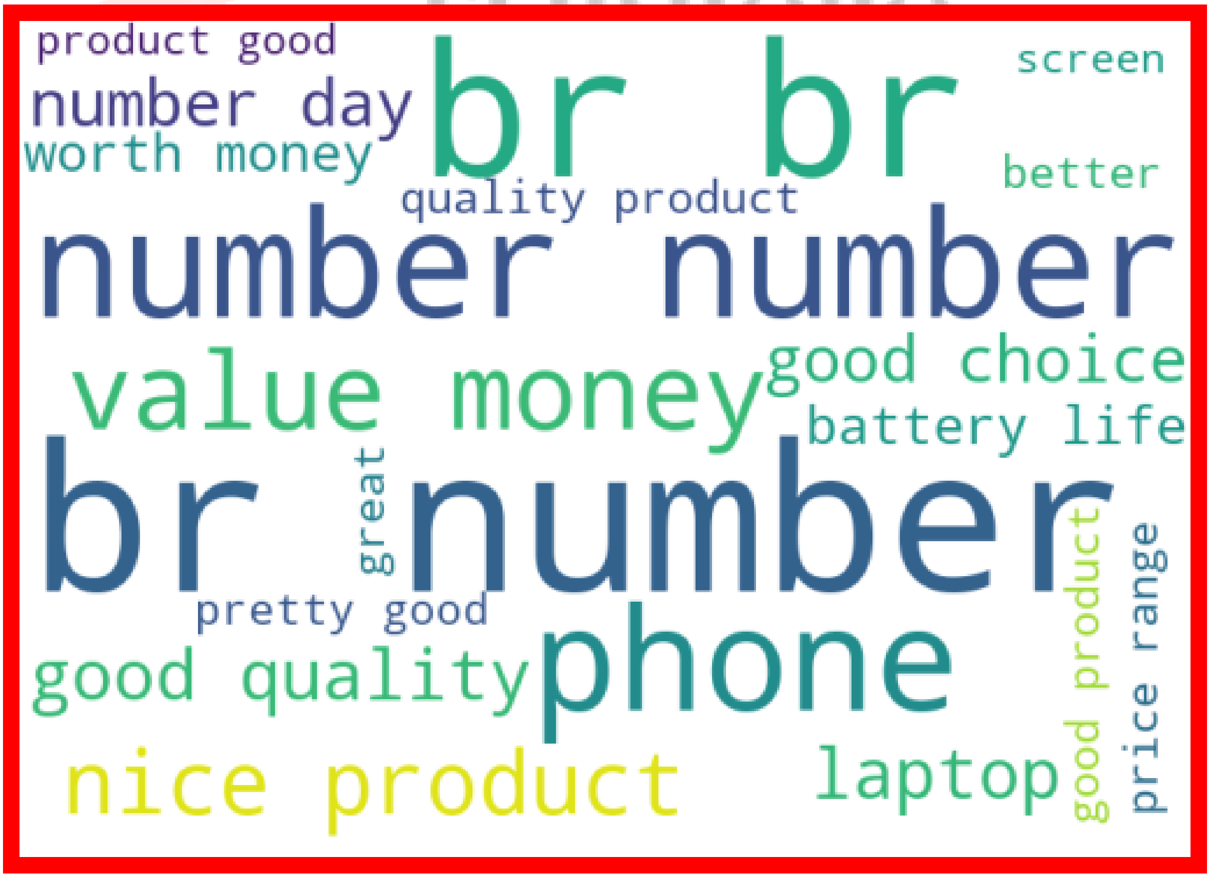
- ***Visualization:***
  - **Word Cloud for Rating 1:**

  - **Word Cloud for Rating 2:**

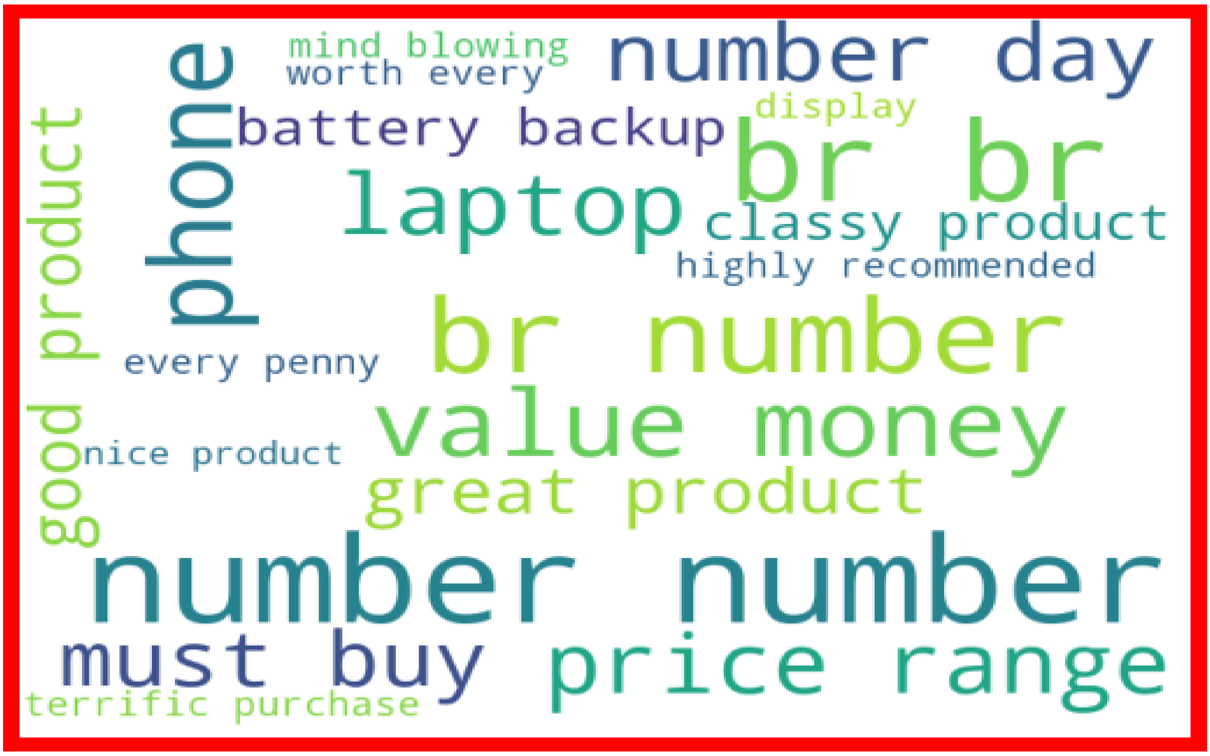


- **Word Cloud for Rating 3:**

- **Word Cloud for Rating 4:**



- **Word Cloud for Rating 5:**



➢ Apply TfidVectorizer to the Combined Review column

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
tf_vec = TfidfVectorizer()
df.columns
```

```
Index(['Ratings', 'Length of Title', 'Length of Text',
       'Len of clean Review_Title', 'Len of clean Reiew_Text', 'Reivew'],
      dtype='object')
```

```
tf_vec = TfidfVectorizer()
features = tf_vec.fit_transform(df['Reivew'])
x = features
y = df['Ratings']
```

- **Dataset is highly imbalanced applied SMOTE technique to overcome from it:**

```
df.Ratings.value_counts()
```

```
5    15888
4     6290
1     3597
3     3002
2     2003
Name: Ratings, dtype: int64
```

```
# Ratio:
print('5 :' ,  round((df[df.Ratings == 5].shape[0] / df.shape[0])*100, 2), '%'  )
print('4 :' ,  round((df[df.Ratings == 4].shape[0] / df.shape[0])*100, 2), '%'  )
print('3 :' ,  round((df[df.Ratings == 3].shape[0] / df.shape[0])*100, 2), '%'  )
print('2 :' ,  round((df[df.Ratings == 2].shape[0] / df.shape[0])*100, 2), '%'  )
print('1 :' ,  round((df[df.Ratings == 1].shape[0] / df.shape[0])*100, 2), '%'  )
```

```
5 : 51.62 %
4 : 20.44 %
3 : 9.75 %
2 : 6.51 %
1 : 11.69 %
```

To balance the dataset apply SMOTE

```
[ ]  from imblearn.over_sampling import SMOTE
     smote = SMOTE()
```

```
[ ]  x, y = smote.fit_resample(x,y)
```

```
[ ]  y.value_counts()
```

```
5    15888
4    15888
3    15888
1    15888
2    15888
Name: Ratings, dtype: int64
```

➢ Machine Learning:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.model_selection import cross_val_score
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.naive_bayes import MultinomialNB
from xgboost import XGBClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB, GaussianNB,BernoulliNB
from xgboost import XGBClassifier
from sklearn.svm import LinearSVC
from lightgbm import LGBMClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier, AdaBoostClassifier
from sklearn.naive_bayes import MultinomialNB, GaussianNB,BernoulliNB
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV
```

Note: ML_Model is a function which give accuracy and metrics with CV values at difference cross fold.

```
lr = LogisticRegression()
sgd = SGDClassifier()
random = RandomForestClassifier()
dtc = DecisionTreeClassifier()
gbc = GradientBoostingClassifier()
naive = MultinomialNB()
xgb = XGBClassifier()
bernoulli = BernoulliNB()
```

Linear Regression Model Performance:

```
models = [lr ]
ML_Model(models, x, y)
```

```
Training accuracy is :  0.9279959718026183
Testing accuracy is : 0.909197717354817
-------------------------------------------------------------------------------
Classification Report:
               precision    recall  f1-score   support

           1       0.96      0.95      0.95      4819
           2       0.95      0.92      0.93      4918
           3       0.89      0.89      0.89      4786
           4       0.87      0.87      0.87      4820
           5       0.88      0.93      0.90      4489

    accuracy                           0.91     23832
   macro avg       0.91      0.91      0.91     23832
weighted avg       0.91      0.91      0.91     23832

Confusion Matrix:
 [[4559   99  108   20   33]
 [ 148 4513  170   59   28]
 [  43  113 4252  262  116]
 [  15   28  195 4180  402]
 [   3    7   51  264 4164]]
-------------------------------------------------------------------------------
Cross value score
cv score 0.8450906344410876 at 2 cross fold
cv score 0.880022658610272 at 3 cross fold
cv score 0.8766364551863042 at 4 cross fold
cv score 0.8905211480362538 at 5 cross fold
cv score 0.8941339375629406 at 6 cross fold
cv score 0.8966025496491831 at 7 cross fold
-------------------------------------------------------------------------------
```

DecisionTreeClassifier:

```
Training accuracy is :  0.9971227161559488
Testing accuracy is : 0.9125545485062101
-------------------------------------------------------------------------------
Classification Report:
               precision    recall  f1-score   support

           1       0.93      0.93      0.93      4768
           2       0.93      0.91      0.92      4830
           3       0.90      0.91      0.90      4735
           4       0.90      0.89      0.90      4828
           5       0.91      0.92      0.91      4671

    accuracy                           0.91     23832
   macro avg       0.91      0.91      0.91     23832
weighted avg       0.91      0.91      0.91     23832
```

```
Confusion Matrix:
 [[4438  147   89   30   64]
 [ 160 4405  155   72   38]
 [  89  110 4299  151   86]
 [  47   61  147 4312  261]
 [  34   37   86  220 4294]]
--------------------------------------------------------------------------------
Cross value score
cv score 0.833257804632427 at 2 cross fold
cv score 0.8760699899295066 at 3 cross fold
cv score 0.873904833836858 at 4 cross fold
cv score 0.8863922457200403 at 5 cross fold
cv score 0.8932150050352469 at 6 cross fold
cv score 0.8975340018395882 at 7 cross fold
--------------------------------------------------------------------------------
```

Naïve Bayes

```
--------------------------------------------------------------------------------
Training accuracy is :  0.8604157675154654
Testing accuracy is : 0.8404665995300437
--------------------------------------------------------------------------------
Classification Report:
              precision    recall  f1-score   support

           1       0.86      0.91      0.89      4524
           2       0.90      0.83      0.86      5156
           3       0.82      0.78      0.80      4991
           4       0.79      0.81      0.80      4702
           5       0.83      0.88      0.85      4459

    accuracy                           0.84     23832
   macro avg       0.84      0.84      0.84     23832
weighted avg       0.84      0.84      0.84     23832

Confusion Matrix:
 [[4118  213  123   42   28]
 [ 419 4277  292  107   61]
 [ 170  208 3917  470  226]
 [  27   46  335 3792  502]
 [  34   16  109  374 3926]]
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Cross value score
cv score 0.7731873111782477 at 2 cross fold
cv score 0.8050981873111782 at 3 cross fold
cv score 0.7961354481369587 at 4 cross fold
cv score 0.8139728096676737 at 5 cross fold
cv score 0.8173841893252769 at 6 cross fold
cv score 0.8231132644008886 at 7 cross fold
--------------------------------------------------------------------------------
```

Bernoulli

```
Training accuracy is :  0.6456804776291181
Testing accuracy is : 0.6332661967103055
-------------------------------------------------------------------------------
Classification Report:
               precision    recall  f1-score   support

           1       0.78      0.78      0.78      4763
           2       0.53      0.90      0.67      2821
           3       0.56      0.81      0.66      3332
           4       0.43      0.73      0.54      2796
           5       0.86      0.41      0.55     10120

    accuracy                           0.63     23832
   macro avg       0.63      0.72      0.64     23832
weighted avg       0.72      0.63      0.63     23832

Confusion Matrix:
 [[3722  741  246   37   17]
 [ 127 2544   63   60   27]
 [  82  156 2687  220  187]
 [  17  121  211 2037  410]
 [ 820 1198 1569 2431 4102]]
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Cross value score
cv score 0.617837361530715 at 2 cross fold
cv score 0.6242321248741188 at 3 cross fold
cv score 0.6185045317220543 at 4 cross fold
cv score 0.624269889224572 at 5 cross fold
cv score 0.6268378650553877 at 6 cross fold
cv score 0.6282492814539881 at 7 cross fold
--------------------------------------------------------------------------------
```


Applied Boosting Techniques:

RandomForestClassifier

```
models = [random, gbc]
ML_Model(models, x, y)
```

```
RandomForestClassifier()
Training accuracy is :  0.9974643936124299
Testing accuracy is : 0.9762084592145015
-------------------------------------------------------------------------------
Classification Report:
               precision    recall  f1-score   support

           1       1.00      0.98      0.99      4852
           2       0.99      0.99      0.99      4763
           3       0.98      0.98      0.98      4771
           4       0.96      0.96      0.96      4781
           5       0.95      0.97      0.96      4665

    accuracy                           0.98     23832
   macro avg       0.98      0.98      0.98     23832
weighted avg       0.98      0.98      0.98     23832
```

```
Confusion Matrix:
 [[4753   18   29   16   36]
 [   5 4726   12   10   10]
 [   4    6 4684   42   35]
 [   1    7   35 4589  149]
 [   5    3   16  128 4513]]
------------------------------------------------------------------------------
Cross value score
cv score 0.9252517623363545 at 2 cross fold
cv score 0.949685297079557 at 3 cross fold
cv score 0.9475830815709969 at 4 cross fold
------------------------------------------------------------------------------
```

GradientBoostingClassifier

```
GradientBoostingClassifier()
Training accuracy is :  0.8255107178823191
Testing accuracy is : 0.8118496139644176
------------------------------------------------------------------------------
Classification Report:
              precision    recall  f1-score   support

           1       0.89      0.81      0.84      5241
           2       0.77      0.76      0.77      4796
           3       0.74      0.81      0.77      4356
           4       0.82      0.79      0.81      4952
           5       0.84      0.89      0.87      4487

    accuracy                           0.81     23832
   macro avg       0.81      0.81      0.81     23832
weighted avg       0.81      0.81      0.81     23832

Confusion Matrix:
 [[4222  671  223   53   72]
 [ 430 3665  485  143   73]
 [  71  293 3527  327  138]
 [  31  116  407 3936  462]
 [  14   15  134  326 3998]]
------------------------------------------------------------------------------
Cross value score
cv score 0.7874244712990937 at 2 cross fold
cv score 0.8016112789526687 at 3 cross fold
cv score 0.800453172205438 at 4 cross fold
------------------------------------------------------------------------------
```

#XGB Boost:

```
XGBClassifier()
Training accuracy is :  0.7849949647532729
Testing accuracy is : 0.7742950654582075
--------------------------------------------------------------------------------
Classification Report:
               precision    recall  f1-score   support

           1       0.88      0.75      0.81      5633
           2       0.71      0.73      0.72      4626
           3       0.65      0.79      0.71      3896
           4       0.80      0.74      0.77      5153
           5       0.83      0.87      0.85      4524

    accuracy                           0.77     23832
   macro avg       0.77      0.78      0.77     23832
weighted avg       0.78      0.77      0.78     23832

 Confusion Matrix:
 [[4212  908  329   73  111]
 [ 381 3384  600  185   76]
 [  93  284 3092  309  118]
 [  65  171  594 3825  498]
 [  17   13  161  393 3940]]
 --------------------------------------------------------------------------------
 Cross value score
 cv score 0.7419436052366566 at 2 cross fold
 cv score 0.7642497482376637 at 3 cross fold
 cv score 0.7598187311178247 at 4 cross fold
 --------------------------------------------------------------------------------
```

Note: By observation various outputs of the difference machine learning algorithm, random forest is giving best result, In random forest training and testing accuracy is also very close to each other and also having very close value of CV, therefore randomforest algorithm selected as final machine learning algorithm to train the dataset for final model

➢ Ensemble Technique of RandomForestClassifer:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.30, random_state = 51 )
```

er Parameter Tuning

```
# parameter = {'max_depth' : [80, 90, 100, 110],
#              'min_samples_leaf': [3, 4, 5],
#              'criterion' : ['gini', 'entropy'],
#              'min_samples_split': [8, 10, 12],
#              'n_estimators': [100, 200, 300, 1000]  }
# Tried to check with these above mentioned parameters but fail to do, due to so much time consumption, approx 9 to 12 hours or more than that

parameter = {'criterion' : ['gini', 'entropy'],
             'n_estimators': [100, 200]  }
```

```
gcv = GridSearchCV(estimator = RandomForestClassifier(), param_grid = parameter, cv = 3)
gcv.fit(x_train, y_train)
```

```
GridSearchCV(cv=3, estimator=RandomForestClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'n_estimators': [100, 200]})
```

```
gcv.best_params_
```

```
{'criterion': 'entropy', 'n_estimators': 200}
```

Note: I have applied ensemble technique by using various parameters but it was taking so much time, approx 18 to 20 hours, due to which I have reduced parameters.

➢ Final Model (RandomForestClassifer)

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.30, random_state = 51)
final_model = RandomForestClassifier(criterion='entropy', n_estimators=200)
final_model.fit(x_train, y_train)
predict_train = final_model.predict(x_train)
predict_test = final_model.predict(x_test)

training = accuracy_score(predict_train, y_train)
testing = accuracy_score(predict_test, y_test)

print('the training accuracy is :', training)
print('the testing accuracy is :'  , testing)
print('___________________________________________________________________')
print('Classification Report: \n', classification_report(predict_test, y_test) )
print('Confusion Matrix: \n', confusion_matrix(predict_test, y_test) )
print('_____________________________________________________________________')
```

```
the training accuracy is : 0.9971946482520501
the testing accuracy is : 0.9770057066129574
_______________________________________________________________
Classification Report:
              precision    recall  f1-score   support

           1       1.00      0.98      0.99      4844
           2       0.99      0.99      0.99      4766
           3       0.98      0.98      0.98      4756
           4       0.96      0.96      0.96      4788
           5       0.96      0.97      0.96      4678

    accuracy                           0.98     23832
   macro avg       0.98      0.98      0.98     23832
weighted avg       0.98      0.98      0.98     23832

Confusion Matrix:
 [[4750   16   32   20   26]
 [   4 4727   15    9   11]
 [   1    5 4680   40   30]
 [   6    7   32 4597  146]
 [   7    5   17  119 4530]]
_________________________________________________________________
```

```
# perform cross-validation
print('Cross value score')
cv_score = cross_val_score(RandomForestClassifier(criterion='entropy', n_estimators=200), x, y, cv = 3 ).mean()
print('cv score',  cv_score )
print('-------------------------------------------------------------------------------------')
```

```
Cross value score
cv score 0.9507175226586102
```

Deploy the model

```
[ ]  import pickle
     filename = 'rating_project.pkl'                    # model name
     pickle.dump(final_model, open(filename, 'wb'))          # operation to deploy model
```

```
[ ]
```

Loading model

```
load_model =  pickle.load(open('rating_project.pkl', 'rb'))    # loading deployed model
result = load_model.score(x_test, y_test)
print(result)
```

```
0.9770057066129574
```

Conclusion

```
[ ] original = np.array(y_test)
    predicted = np.array(load_model.predict(x_test))
    # convert columns in to np.array
```

```
print(predicted.shape)
print(original.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(23832,)
(23832,)
(23832, 13629)
(23832,)
```

```
conclusion = pd.DataFrame({'Original fraud_reported': original, 'Predicted fraud_reported': predicted}, index = range(len(original)))
# Dataframe creation
```

```
pd.set_option('display.max_rows', None)  # To maximize the rows
conclusion.head()
```

	Original fraud_reported	Predicted fraud_reported
0	1	1
1	4	4
2	1	1
3	2	2
4	4	4

```
conclusion.sample(10)
```

	Original fraud_reported	Predicted fraud_reported
14384	4	4
6465	3	3
17223	3	3
17663	4	4
3485	3	3
11480	1	1
21600	5	5
18061	5	5
10124	5	5
3477	5	5

- **Hardware and Software Requirements and Tools Used**

All used libraries:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.model_selection import cross_val_score
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.naive_bayes import MultinomialNB
from xgboost import XGBClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB, GaussianNB,BernoulliNB
from xgboost import XGBClassifier
from sklearn.svm import LinearSVC
from lightgbm import LGBMClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier, AdaBoostClassifier
from sklearn.naive_bayes import MultinomialNB, GaussianNB,BernoulliNB
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV
```

```
# imp libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Pandas: This library used for dataframe operations

Numpy: This library gives statistical computation for smooth functioning

Matplotlib: Used for visualization

Seaborn: This library is also used for visualization

Sklearn: This library having so many machine learning module and we can import them from this library

Pickle: This is used for deploying the model

Imblearn: This library is import to get SMOTE technique for balance the data

Scipy: It is import to perform outlier removing technique using zscore

Warning: To avoid unwanted warning shows in the output

I am giving this requirement and tool used, based on my laptop configuration.

Operating System:	**Window 11**
RAM:	**8 GB**
Processor:	**i5 10th Generation**
Software:	**Jupyter Notebook,**

- **Observations from the whole problem.**

i) Review column is combine column of Review_Title and Review_Text.

ii) Dataset was having null values.

iii) Tfvectorizer applied to review column to convert this column into machine learning language.

- **Learning Outcomes of the Study in respect of Data Science**

My learnings: - the power of visualization is helpful for the understanding of data into the graphical representation its help me to understand that what data is trying to say,

Various algorithms I have used in this dataset and to get out best result and save that model. The best algorithm is RandomForestClassifier.

Ensemble operation was giving biggest challenge which I have faced while working and as this dataset is very large which have leads to take lot of time for machine learning.

for scraping the data, it had taken almost 7 to 8 hours' time for execution.

- **Limitations of this work and Scope for Future Work**

We can train machine learning model with more data, which should be in lacs but this model is also working well.