FLIP ROBO

RATING PREDICTION

Submitted by:

SUDHANSU MANDAL

INTRODUCTION

➢ Business Problem Framing

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

➢ Conceptual Background of the Domain Problem

These days where lot of information generating every day in different form, some are like in text, audio, video. We must be aware that E-commerce industry has grown and keep on growing day by day. There are many e-commerce retailers like Amazon, Flipkart, eBay etc are present in market and giving competition to each other to gain maximum market share. So, customer review or sentiment plays very important role to know more about the product or service. Customer review provide very useful information for the company which they can work on and improve their product or services. Here machine learning model could help them to get deeper insight on customer review and provide some meaningful information to build business strategy about the product or services.

➢ Review of Literature

This project observed customer reviews and build a very effective model which can predict the rating of product or services. To complete the project, we have followed below process:

1. **Understand the problem:** Before getting the data, we need to understand the problem statement and the objective of the project. We need to understand the problem we are trying to solve with the help of data science.
2. **Collect Data:** In this step, we have collected the review and rating data from Amazon website through web scrapping for the electronic products.
3. **Data Pre-Processing:** Since our feature or independent variable is in text format. So we have used some NLP technique to get the correct required format text for our machine learning algorithm.
4. **Feature Engineering:** In this step we have used various NLP algorithm to convert our data into numerical and in vectorized form into different dimension to predict the customer rating.
5. **Data Exploration:** In this section, we will need to deep dive into the dataset and tries to understand the text review. In this project we have used some NLP algorithm to get more insight to our review. This step will involve to creates some chart or diagram to understand the text data
6. **Model Training:** Using various algorithm which is suitable for the dataset and we train the model on the given training dataset.
7. **Model Evaluation:** This is very important steps to know that how the model has been trained in provided dataset. We will evaluate the model performance based on various technique.

➢ Motivation for the Problem Undertaken

We are in era where company don't deal customer directly. They provide the product or services to required customer remotely and create challenging for company to get the direct feedback from customer about the product and services. Such situation Customer Review or Rating provide very useful information about the customer's sentiment about the used product or services. These pieces of information could help company to improve their product or services time to time. Also, could help them in product development or new product launch. So, Machine learning can provide solution by helping them to get into deeper insight of their customer review and rating.

ANALYTICAL PROBLEM FRAMING

➢ Mathematical/ Analytical Modeling of the Problem

In this project we have scrapped data about the Customer Review and Rating of electronic products from Amazon website. There are 29994 customer review and corresponding rating. The rating are in 5 star to 1 star.
To understand data, we have done basic data exploration.

- **Statistic Summary**

Since Our Dataset contains text data so we would not helpful more with statistical analysis. Mention below are the observation made from this summary:

1. **Data Type:** As we have seen that Dataset having 2 columns, customer review and rating and both are object type.
2. **Null Values:** We have seen that there are null values in dataset, which we need deal with.

- **Feature Selection**

As our independent variable is in text format, So we have applied various NLP algorithm to deal with text data and get rating prediction with the help of machine learning algorithm.

- **Data Visualization**

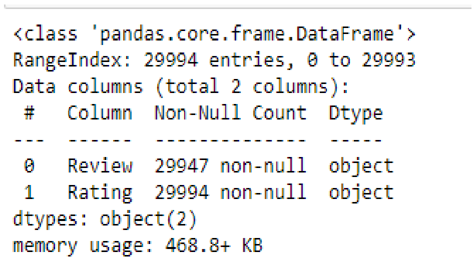
While doing EDA, we have found mentioned below important points for our dataset:

a. Get the context word of the token in the dataset.
b. Similarity between the related words.
c. Add or subtract word vectors together. To add is to combine the meaning of the components and to subtract is to take out the context of one token from another**.**
d. Topic Modelling with the help of Latent Dirichlet Allocation (LDA)
e. Principal Component Analysis.

➢ Data Sources and their formats

The data is in .csv format. We have imported and converted into Pandas dataframe for analysing purpose. The dataset having 2 columns customer review and rating both are object datatype as shown in Fig 1.

Fig 1:



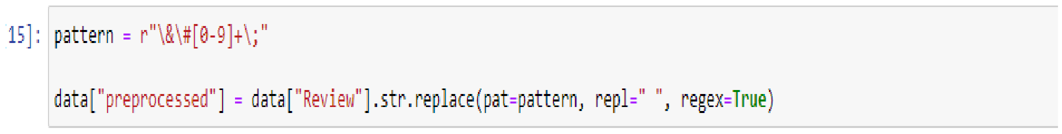
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29994 entries, 0 to 29993
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Review  29947 non-null  object
 1   Rating  29994 non-null  object
dtypes: object(2)
memory usage: 468.8+ KB
```

➢ Data Pre-processing

As we know Pre-Processing of our data is an important step for our model to perform better but in this project we will apply some NLP based algorithm to pre-processing our text data.

- **Removing Extra character:**
  As shown in Fig 2, we have applied replace method to replace unwanted character from the dataset. This method will remove all unwanted character and keep alphabet character.

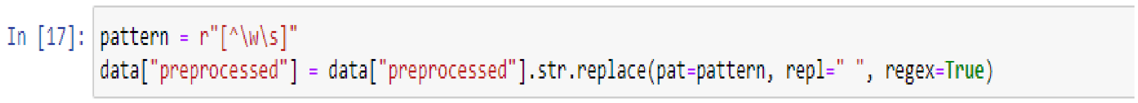
Fig 2.



```
15]: pattern = r"\&\#[0-9]+\;"

     data["preprocessed"] = data["Review"].str.replace(pat=pattern, repl=" ", regex=True)
```

- **Removing Punctuations:**
  As shown in Fig 3, we have removed punctuations sign from the text by applied method.

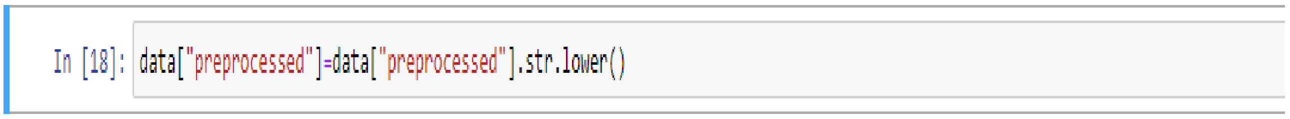
Fig 3.



```
In [17]: pattern = r"[^\w\s]"
         data["preprocessed"] = data["preprocessed"].str.replace(pat=pattern, repl=" ", regex=True)
```

- **Converting to lower case:**
  As shown in Fig 4, we have converted all alphabet character into lower case to avoid treating same word differently.

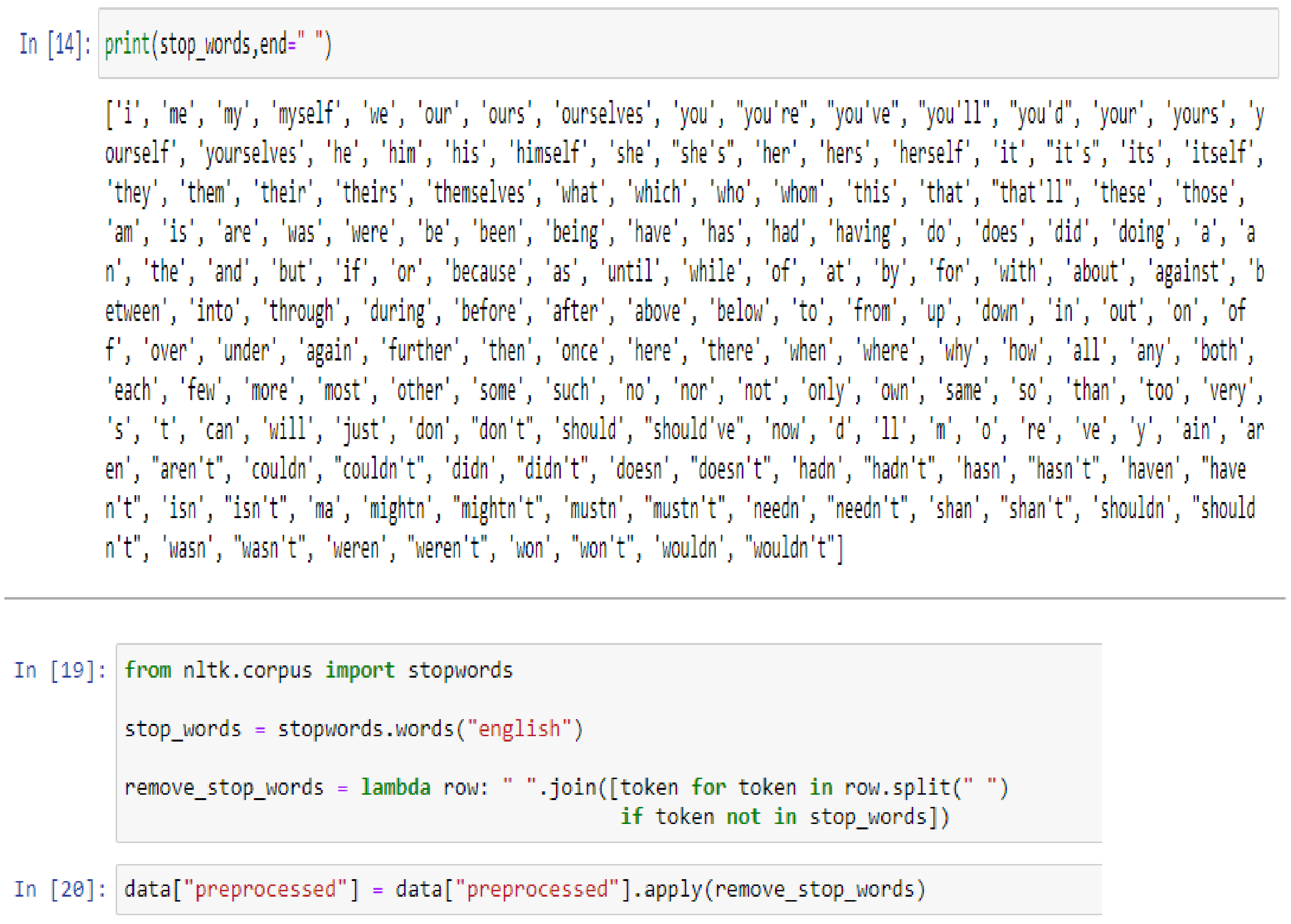
Fig 4.



```
In [18]: data["preprocessed"]=data["preprocessed"].str.lower()
```

- **Removed Stop words:**
  As shown in Fig 5, we have applied NLP algorithm to removed stop words from the dataset. For this purpose, we have downloaded 'nltk' library. Stop words are those words which repeat in sentences very frequently with no meaning of words.

Fig 5.



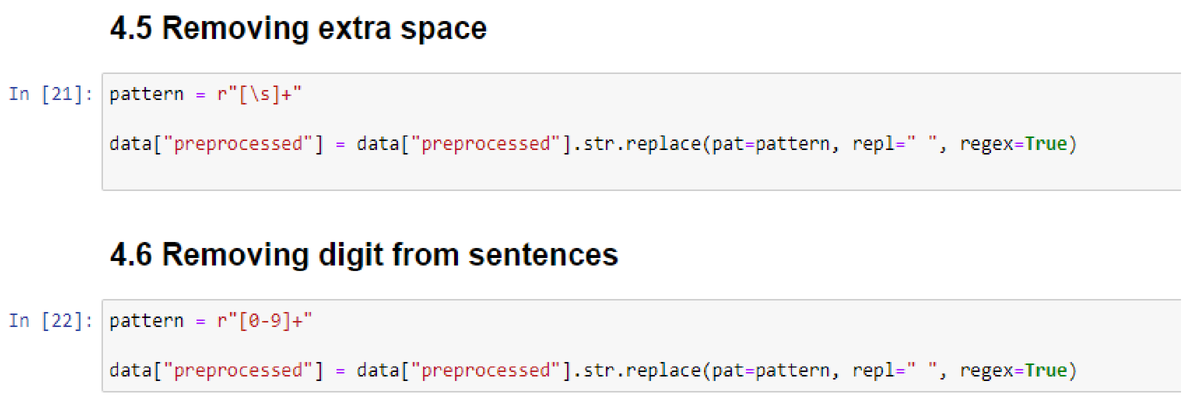
```
In [14]: print(stop_words,end=" ")

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'y
ourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself',
'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those',
'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'a
n', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'b
etween', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'of
f', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'ar
en', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "have
n't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "should
n't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

```
In [19]: from nltk.corpus import stopwords

         stop_words = stopwords.words("english")

         remove_stop_words = lambda row: " ".join([token for token in row.split(" ")
                                                   if token not in stop_words])

In [20]: data["preprocessed"] = data["preprocessed"].apply(remove_stop_words)
```

- **Removing extra spaces and digit:**

  As shown in Fig 6, we have removed extra spaces and digit from the dataset by applying below technique.

Fig 6.



4.5 Removing extra space

```
In [21]: pattern = r"[\s]+"

         data["preprocessed"] = data["preprocessed"].str.replace(pat=pattern, repl=" ", regex=True)
```

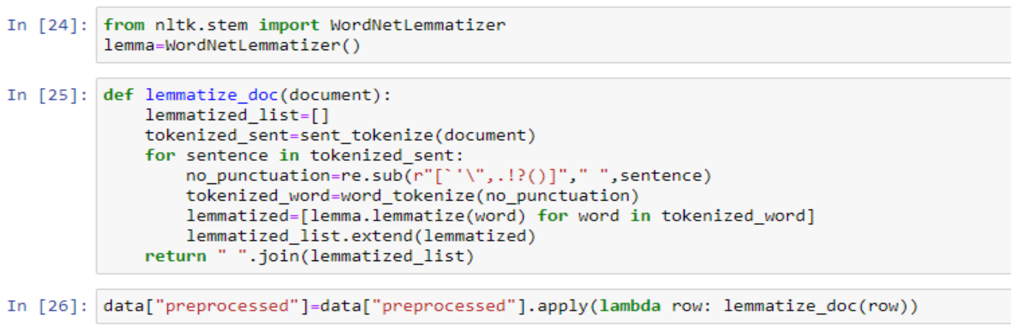
4.6 Removing digit from sentences

```
In [22]: pattern = r"[0-9]+"

         data["preprocessed"] = data["preprocessed"].str.replace(pat=pattern, repl=" ", regex=True)
```

- **Apply Tokenization and Lemmatization algorithm:**

  As shown in Fig 7, We have applied Tokenization and Lemmatization technique to our dataset. Tokenization convert the text in required token basically its split the document into sentences or token. Lemmatization usually provide the base words by keeping the meaning of words.

Fig 7.

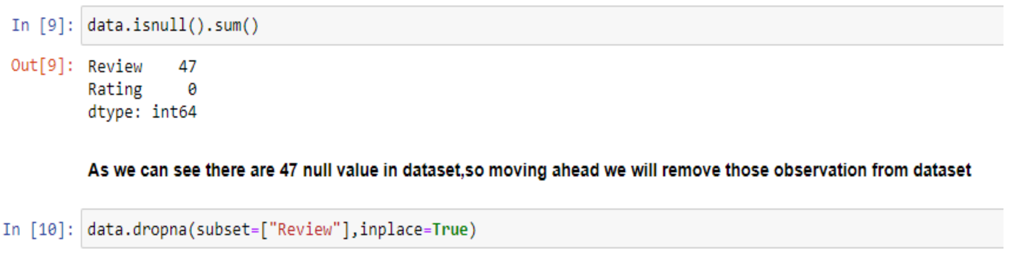


```
In [24]: from nltk.stem import WordNetLemmatizer
         lemma=WordNetLemmatizer()

In [25]: def lemmatize_doc(document):
             lemmatized_list=[]
             tokenized_sent=sent_tokenize(document)
             for sentence in tokenized_sent:
                 no_punctuation=re.sub(r"[`'\",.!?()]"," ",sentence)
                 tokenized_word=word_tokenize(no_punctuation)
                 lemmatized=[lemma.lemmatize(word) for word in tokenized_word]
                 lemmatized_list.extend(lemmatized)
             return " ".join(lemmatized_list)

In [26]: data["preprocessed"]=data["preprocessed"].apply(lambda row: lemmatize_doc(row))
```

- **Dealing with null value**

  As we have seen Fig 8, there are 47 null value in Review columns, we will remove these observations from the dataset, since we have enough sample in the dataset.

Fig 8.



```
In [9]: data.isnull().sum()

Out[9]: Review    47
        Rating     0
        dtype: int64
```

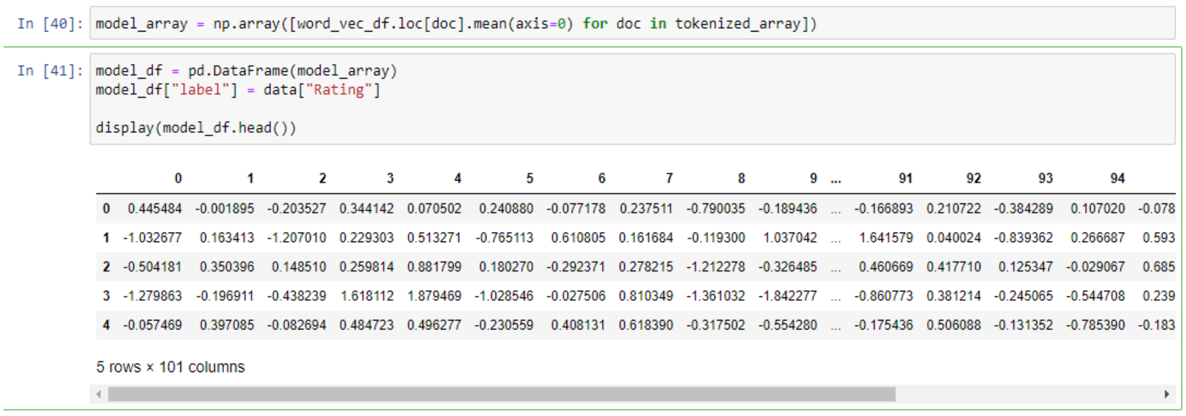
As we can see there are 47 null value in dataset,so moving ahead we will remove those observation from dataset

```
In [10]: data.dropna(subset=["Review"],inplace=True)
```

➢ Feature Engineering:

Since we have seen that our features columns "Review" has text data. So, we have used NLP algorithm to make our features columns into vector. We have applied word embedding technique with the help of NLP. Word embedding is a learned representation for text where words that have the same meaning have a similar representation. We have use Word2vec, t-SNE, Word algebra, LDA etc to explore the words in dataset and their relationship.
Here we have created dataframe with observations corresponding to the customer reviews. The word_vec_model is used to get all the unique token in the corpora which help us to generate features of every word in dimensions as shown in Fig 9.

Fig 9.



```
In [40]: model_array = np.array([word_vec_df.loc[doc].mean(axis=0) for doc in tokenized_array])

In [41]: model_df = pd.DataFrame(model_array)
         model_df["label"] = data["Rating"]

         display(model_df.head())
```

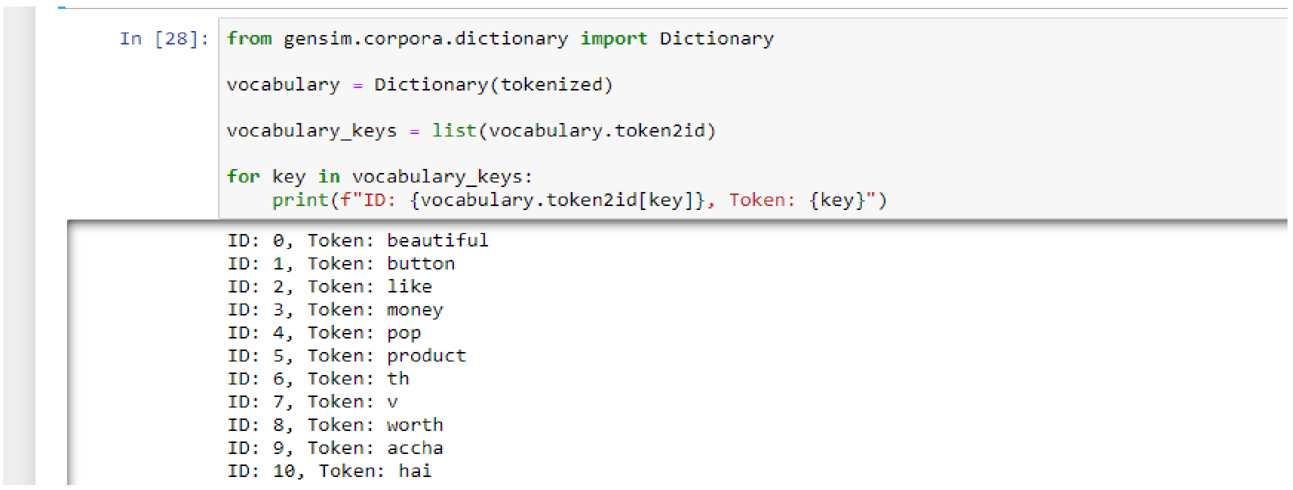
	0	1	2	3	4	5	6	7	8	9	...	91	92	93	94	
0	0.445484	-0.001895	-0.203527	0.344142	0.070502	0.240880	-0.077178	0.237511	-0.790035	-0.189436	...	-0.166893	0.210722	-0.384289	0.107020	-0.078
1	-1.032677	0.163413	-1.207010	0.229303	0.513271	-0.765113	0.610805	0.161684	-0.119300	1.037042	...	1.641579	0.040024	-0.839362	0.266687	0.593
2	-0.504181	0.350396	0.148510	0.259814	0.881799	0.180270	-0.292371	0.278215	-1.212278	-0.326485	...	0.460669	0.417710	0.125347	-0.029067	0.685
3	-1.279863	-0.196911	-0.438239	1.618112	1.879469	-1.028546	-0.027506	0.810349	-1.361032	-1.842277	...	-0.860773	0.381214	-0.245065	-0.544708	0.239
4	-0.057469	0.397085	-0.082694	0.484723	0.496277	-0.230559	0.408131	0.618390	-0.317502	-0.554280	...	-0.175436	0.506088	-0.131352	-0.785390	-0.183

5 rows × 101 columns

- **Creating Vocabulary:**

The vocabulary is the key-value pairs of all the unique tokens in our customer review data. Each token is assigned a lookup ID as shown in Fig 10

Fig 10.

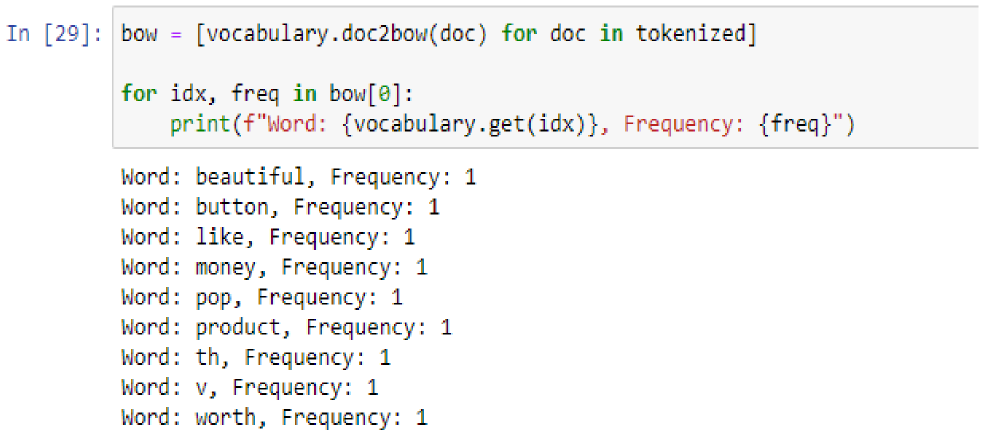


```
In [28]: from gensim.corpora.dictionary import Dictionary

         vocabulary = Dictionary(tokenized)

         vocabulary_keys = list(vocabulary.token2id)

         for key in vocabulary_keys:
             print(f"ID: {vocabulary.token2id[key]}, Token: {key}")

         ID: 0, Token: beautiful
         ID: 1, Token: button
         ID: 2, Token: like
         ID: 3, Token: money
         ID: 4, Token: pop
         ID: 5, Token: product
         ID: 6, Token: th
         ID: 7, Token: v
         ID: 8, Token: worth
         ID: 9, Token: accha
         ID: 10, Token: hai
```

- **Bags of Words Model:**

The classical approach in expressing text as a set of features is getting the token frequency. Each entry to the dataframe is a document while each column corresponds to every unique token in the entire corpora. Here we will know that how many times a word appears in the document.

Fig 11.

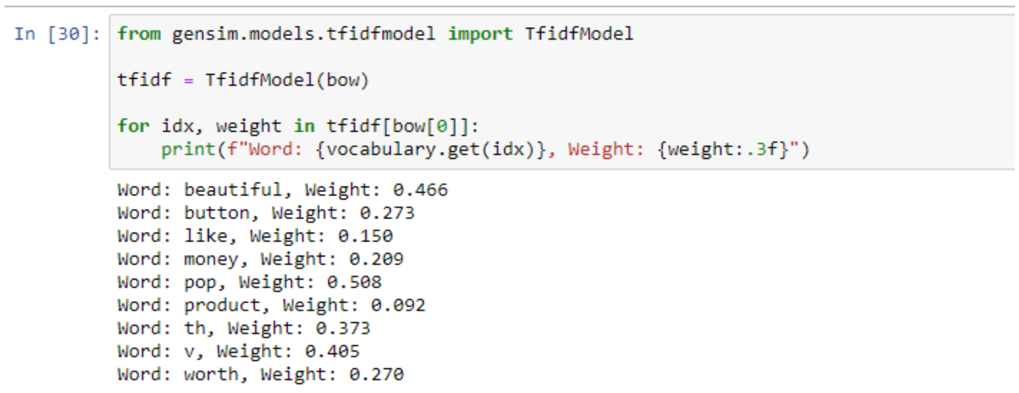


```
In [29]: bow = [vocabulary.doc2bow(doc) for doc in tokenized]

         for idx, freq in bow[0]:
             print(f"Word: {vocabulary.get(idx)}, Frequency: {freq}")

         Word: beautiful, Frequency: 1
         Word: button, Frequency: 1
         Word: like, Frequency: 1
         Word: money, Frequency: 1
         Word: pop, Frequency: 1
         Word: product, Frequency: 1
         Word: th, Frequency: 1
         Word: v, Frequency: 1
         Word: worth, Frequency: 1
```

- **TF-IDF model:**

The Term Frequency-Inverse Document Frequency (TF-IDF) approach assigns continuous values instead of binary numbers for the token frequency. Words that appear frequently overall tend to not establish saliency in a document, and are thus weighted lower. Words that are unique to some documents tend to help distinguish it from the rest and are thus weighted higher. The Tf-idf weighting is based on our bow variable as shown in fig 12.

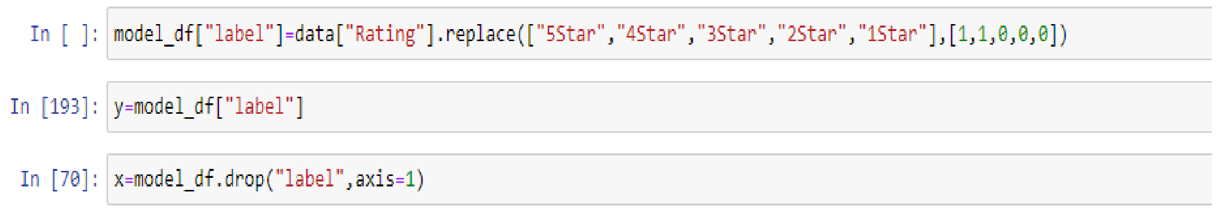
Fig 12.


```
In [30]: from gensim.models.tfidfmodel import TfidfModel

         tfidf = TfidfModel(bow)

         for idx, weight in tfidf[bow[0]]:
             print(f"Word: {vocabulary.get(idx)}, Weight: {weight:.3f}")

         Word: beautiful, Weight: 0.466
         Word: button, Weight: 0.273
         Word: like, Weight: 0.150
         Word: money, Weight: 0.209
         Word: pop, Weight: 0.508
         Word: product, Weight: 0.092
         Word: th, Weight: 0.373
         Word: v, Weight: 0.405
         Word: worth, Weight: 0.270
```

➢ Data Inputs- Logic- Output Relationships

In this dataset, we have 2 columns where both are object type data. Here target variable also is object type having rating class. So, we have converted the rating into int like 5 stars to 5, 4 stars to 4 etc. Also, while applying in model we have converted 5 and 4 stars with 1 as good rating and 3,2 1 star with 0 as bad rating, as shown in Fig 13.

Fig:13


```
In [ ]: model_df["label"]=data["Rating"].replace(["5Star","4Star","3Star","2Star","1Star"],[1,1,0,0,0])

In [193]: y=model_df["label"]

In [70]: x=model_df.drop("label",axis=1)
```

➢ Hardware and Software Requirements and Tools Used

In this project the mention below hardware, software and tools used to complete this project:

✓ Hardware:

Processor	Intel(R) Pentium(R) CPU 3825U @ 1.90GHz 1.90 GHz
Installed RAM	4.00 GB
System type	64-bit operating system, x64-based processor

✓ Software:

Edition	Windows 10 Home Single Language
Anaconda	

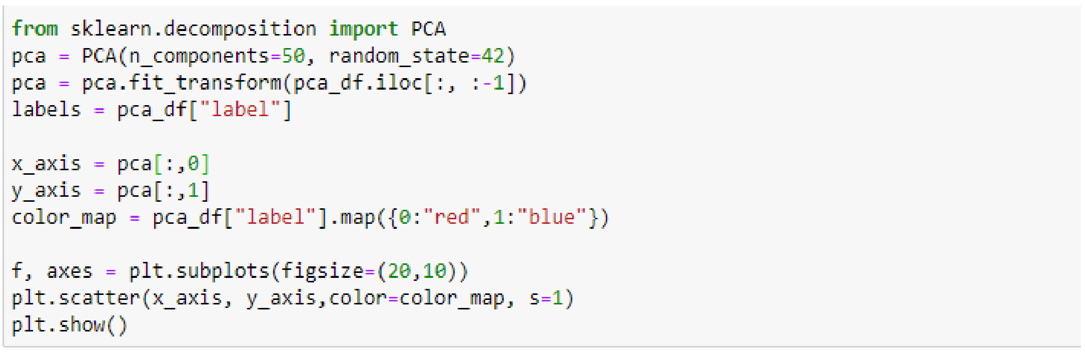
✓ Library & Tools:
Jupyter Notebook
Pandas
NumPy
Matplotlib
Seaborn
Nltk
genism

Model/s Development and Evaluation

➢ Identification of possible problem-solving approaches (methods)

As we have seen that customer Review is in text and our target variable Rating also in string. So, we have used NLP technique to explore our customer Review columns and applied word2vec model to create our dataframe into vector with many dimensions.
Here Rating columns has converted into int by replacing 5Star, 4Star, 3Star, 2Star, 1Star into 1 and 0, where 1 represent good rating i.e. 5Star and 4Star. And 0 represent as bad rating i.e. 3Star,2Star and 1Star.
Also, we have applied PCA **(Principal component Analysis)** for dimension reduction of our feature's dataset, as shown in fig 14, Principal Component Analysis (PCA) is a dimensionality reduction technique that we can use on our model to reduce it. This will help in visualize if there is a clear decision boundary along the rating classifications. The more datapoints belonging to the same class are clustered together, the higher the likelihood that our machine learning model is simpler and more effective.

Fig 14:



```
from sklearn.decomposition import PCA
pca = PCA(n_components=50, random_state=42)
pca = pca.fit_transform(pca_df.iloc[:, :-1])
labels = pca_df["label"]

x_axis = pca[:,0]
y_axis = pca[:,1]
color_map = pca_df["label"].map({0:"red",1:"blue"})

f, axes = plt.subplots(figsize=(20,10))
plt.scatter(x_axis, y_axis,color=color_map, s=1)
plt.show()
```

➢ Testing of Identified Approaches (Algorithms)

As we know our target variable having binary class type of data, so we need apply classification algorithm for model building. Also, the training data are negative in numbers so we would apply GaussianNB instead of MultinomialNB.
we have also used ensemble technique for prediction like Random Forest and Gradient Boosting classifier.
We will train our dataset on below mentioned algorithm and observe the performance each of them.

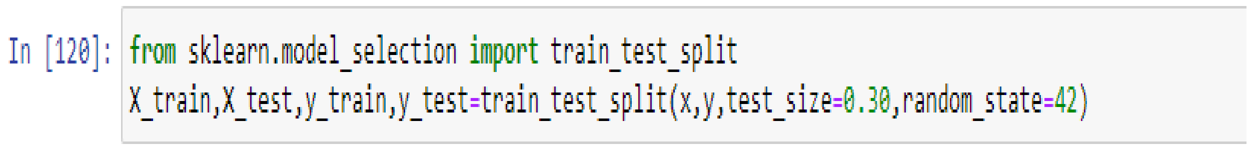
1. GaussianNB
2. Logistic Regression
3. Random Forest Classifier
4. Gradient Boosting Classifier

We have applied mentioned above model in our training and test dataset. Here in project we have split 70% of data in training and 30% of data for testing.

➢ Run and Evaluate selected models

As shown in Fig 15, we have separated the training and test data set into 70% and 30%.

Fig: 15



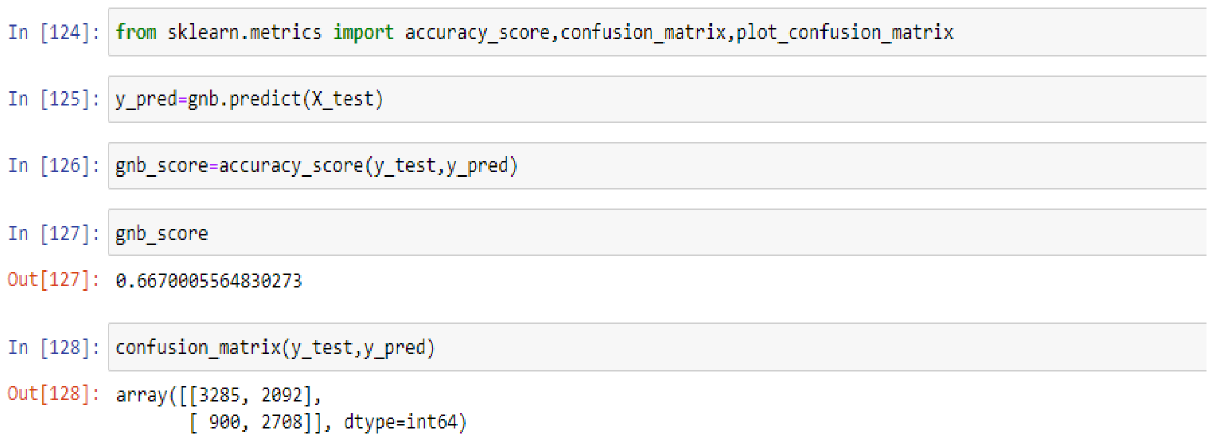
```
In [120]: from sklearn.model_selection import train_test_split
          X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42)
```

In this section we have evaluated our model performance based on Accuracy score,Confusion matrix and score on difference cross validation . Also have used ROC AUC Curve to understand the best performed model.

1. GaussianNB:

Evaluation Matrix:

Fig: 16

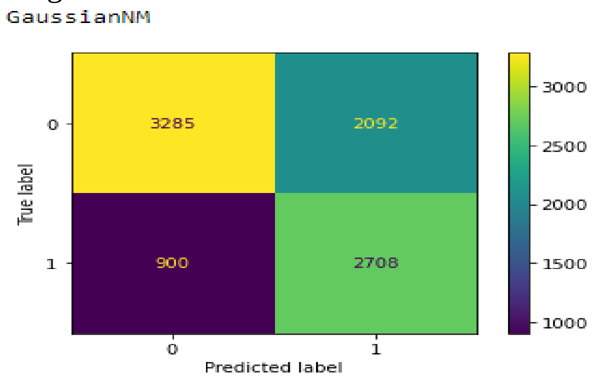


```
In [124]: from sklearn.metrics import accuracy_score,confusion_matrix,plot_confusion_matrix

In [125]: y_pred=gnb.predict(X_test)

In [126]: gnb_score=accuracy_score(y_test,y_pred)

In [127]: gnb_score
Out[127]: 0.6670005564830273

In [128]: confusion_matrix(y_test,y_pred)
Out[128]: array([[3285, 2092],
                 [ 900, 2708]], dtype=int64)
```

After training the model we have got model Accuracy Score is 66.70%.

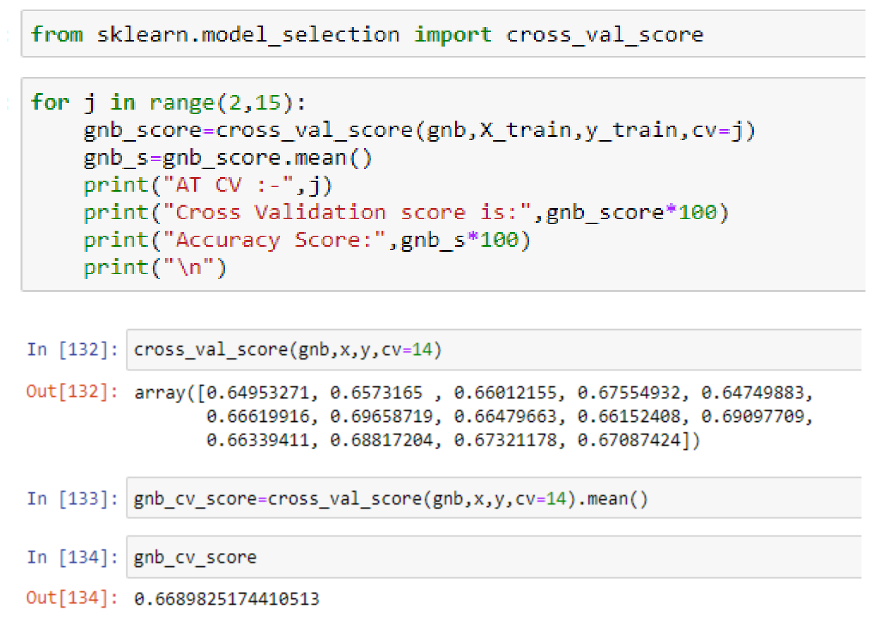
Confusion Matrix:

Fig 17.



Also, as shown in Fig 18, we have observed Score of models on different fold by using Cross Validation to understand our model performance on various fold and ultimate have found the difference with Accuracy score and Cross Validation Score to understand model performance in context of overfitting or underfitting.

Fig 18.

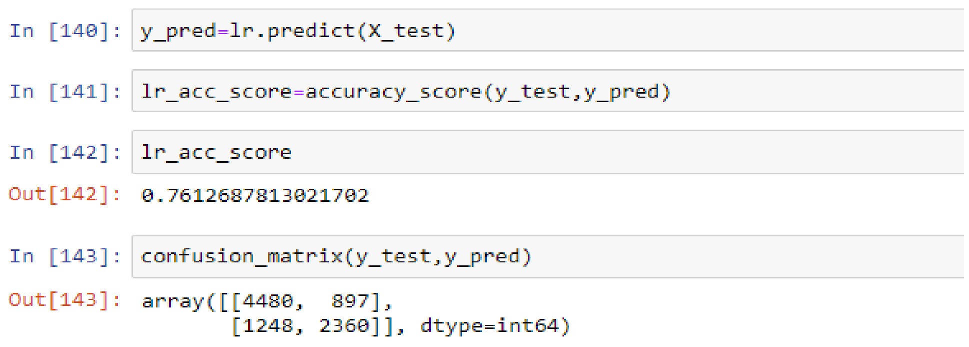


```
from sklearn.model_selection import cross_val_score

for j in range(2,15):
    gnb_score=cross_val_score(gnb,X_train,y_train,cv=j)
    gnb_s=gnb_score.mean()
    print("AT CV :-",j)
    print("Cross Validation score is:",gnb_score*100)
    print("Accuracy Score:",gnb_s*100)
    print("\n")

In [132]: cross_val_score(gnb,x,y,cv=14)

Out[132]: array([0.64953271, 0.6573165 , 0.66012155, 0.67554932, 0.64749883,
                 0.66619916, 0.69658719, 0.66479663, 0.66152408, 0.69097709,
                 0.66339411, 0.68817204, 0.67321178, 0.67087424])

In [133]: gnb_cv_score=cross_val_score(gnb,x,y,cv=14).mean()

In [134]: gnb_cv_score

Out[134]: 0.6689825174410513
```

2. Logistic Regression:

Evaluation Matrix:

Fig: 19

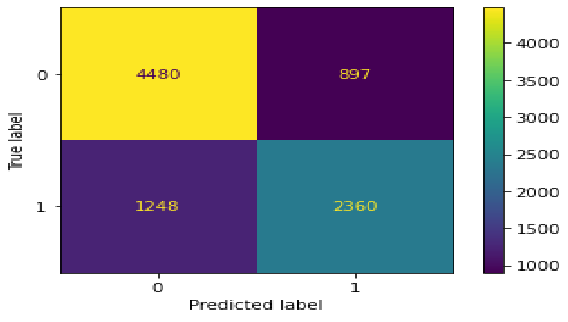


```
In [140]: y_pred=lr.predict(X_test)

In [141]: lr_acc_score=accuracy_score(y_test,y_pred)

In [142]: lr_acc_score

Out[142]: 0.7612687813021702

In [143]: confusion_matrix(y_test,y_pred)

Out[143]: array([[4480,  897],
                 [1248, 2360]], dtype=int64)
```

As shown in Fig 19, the Logistic Regression Accuracy Score is 76.13%

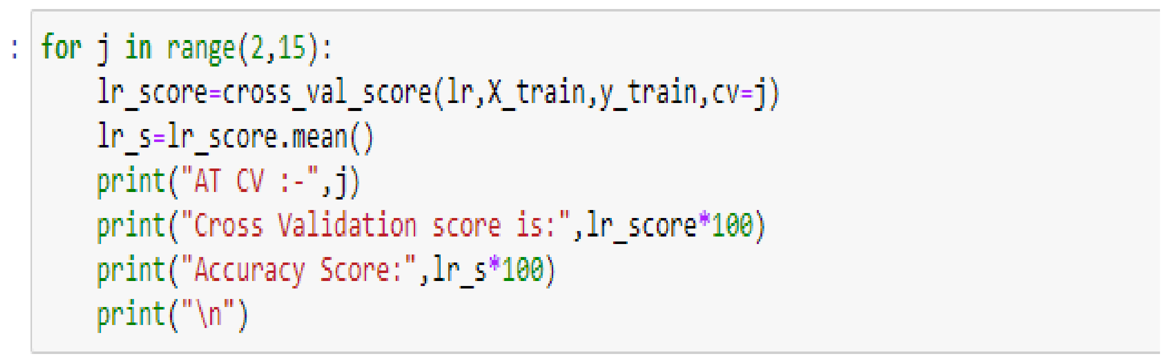
Confusion Matrix:

Fig 20.

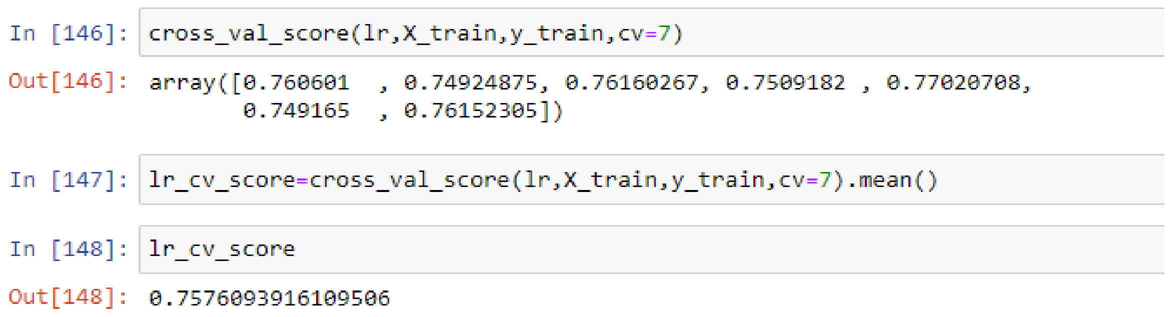


Also, as shown in Fig 21, we have observed Score of models on different fold by using Cross Validation to understand our model performance on various fold and ultimate have found the difference with Accuracy score and Cross Validation Score to understand model performance in context of overfitting or underfitting.

Fig 21.



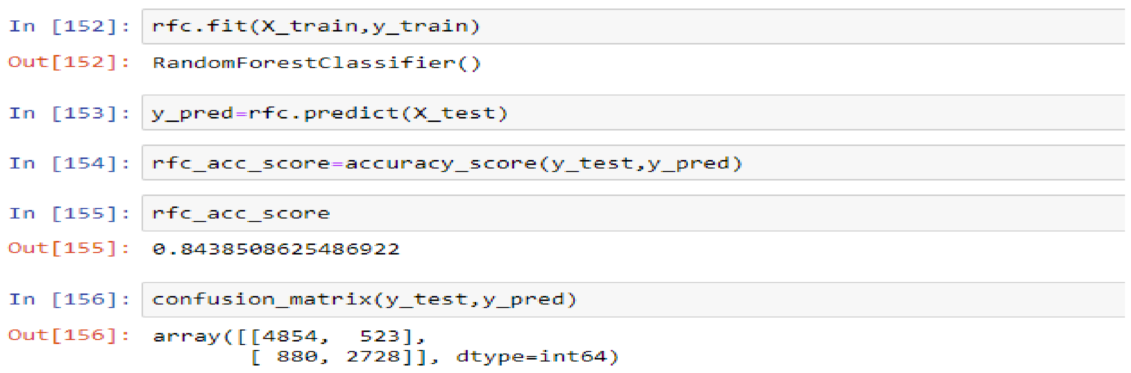
```
: for j in range(2,15):
      lr_score=cross_val_score(lr,X_train,y_train,cv=j)
      lr_s=lr_score.mean()
      print("AT CV :-",j)
      print("Cross Validation score is:",lr_score*100)
      print("Accuracy Score:",lr_s*100)
      print("\n")
```



```
In [146]: cross_val_score(lr,X_train,y_train,cv=7)

Out[146]: array([0.760601  , 0.74924875, 0.76160267, 0.7509182 , 0.77020708,
                 0.749165  , 0.76152305])

In [147]: lr_cv_score=cross_val_score(lr,X_train,y_train,cv=7).mean()

In [148]: lr_cv_score

Out[148]: 0.7576093916109506
```

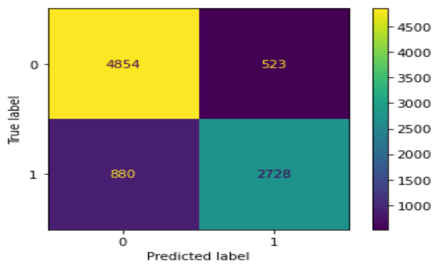
3. Random Forest Classifier

Evaluation Matrix:
Fig 22

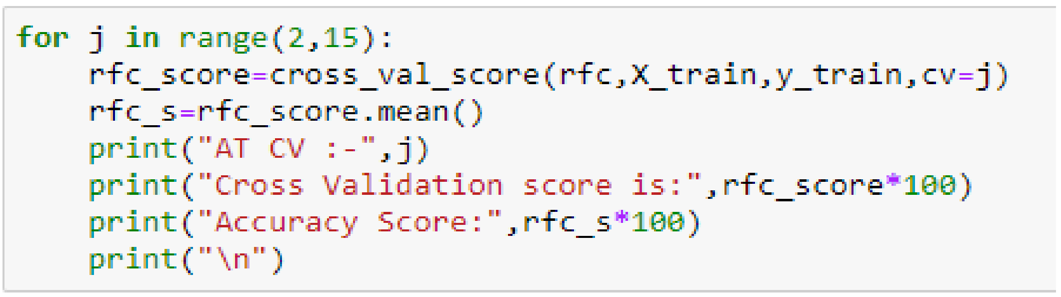


```
In [152]: rfc.fit(X_train,y_train)
Out[152]: RandomForestClassifier()

In [153]: y_pred=rfc.predict(X_test)

In [154]: rfc_acc_score=accuracy_score(y_test,y_pred)

In [155]: rfc_acc_score
Out[155]: 0.8438508625486922

In [156]: confusion_matrix(y_test,y_pred)
Out[156]: array([[4854,  523],
                 [ 880, 2728]], dtype=int64)
```

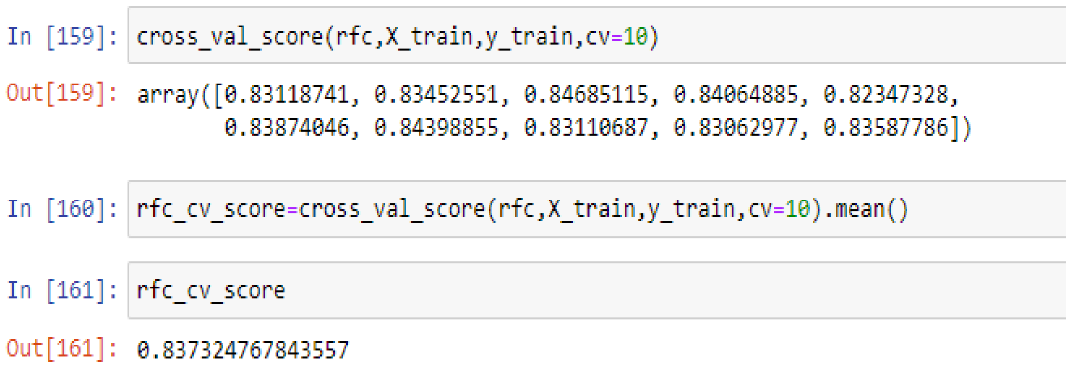
As shown in Fig 22, the Random Forest Classifier, Accuracy Score is 84.38%
Confusion Matrix:
Fig 23.



Also, as shown in Fig 24, we have observed Score of models on different fold by using Cross Validation to understand our model performance on various fold and ultimate have found the difference with Accuracy score and Cross Validation Score to understand model performance in context of overfitting or underfitting.
Fig 24.



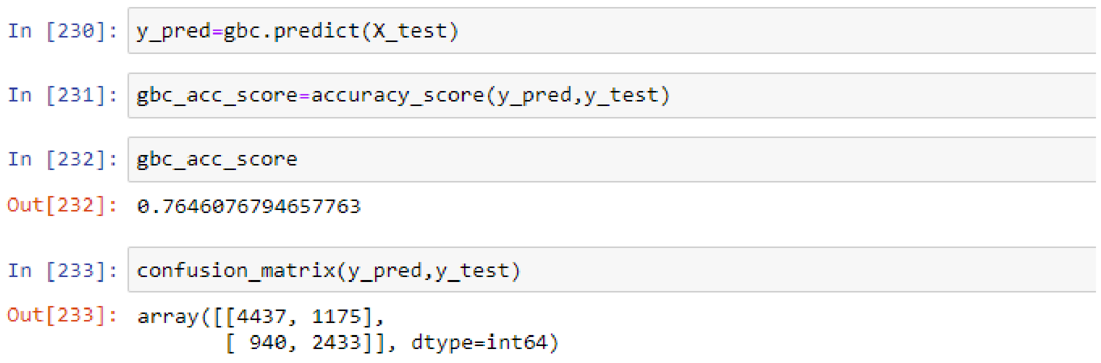
```
for j in range(2,15):
    rfc_score=cross_val_score(rfc,X_train,y_train,cv=j)
    rfc_s=rfc_score.mean()
    print("AT CV :-",j)
    print("Cross Validation score is:",rfc_score*100)
    print("Accuracy Score:",rfc_s*100)
    print("\n")
```



```
In [159]: cross_val_score(rfc,X_train,y_train,cv=10)

Out[159]: array([0.83118741, 0.83452551, 0.84685115, 0.84064885, 0.82347328,
                 0.83874046, 0.84398855, 0.83110687, 0.83062977, 0.83587786])

In [160]: rfc_cv_score=cross_val_score(rfc,X_train,y_train,cv=10).mean()

In [161]: rfc_cv_score

Out[161]: 0.837324767843557
```

4. Gradient Boosting Classifier

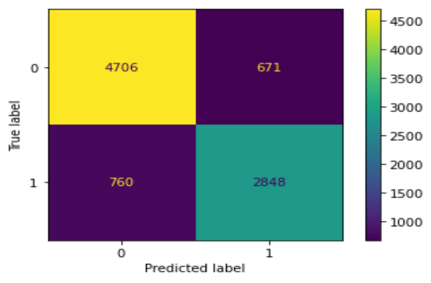
Evaluation Matrix:
Fig: 25



```
In [230]: y_pred=gbc.predict(X_test)

In [231]: gbc_acc_score=accuracy_score(y_pred,y_test)

In [232]: gbc_acc_score

Out[232]: 0.7646076794657763

In [233]: confusion_matrix(y_pred,y_test)

Out[233]: array([[4437, 1175],
                 [ 940, 2433]], dtype=int64)
```

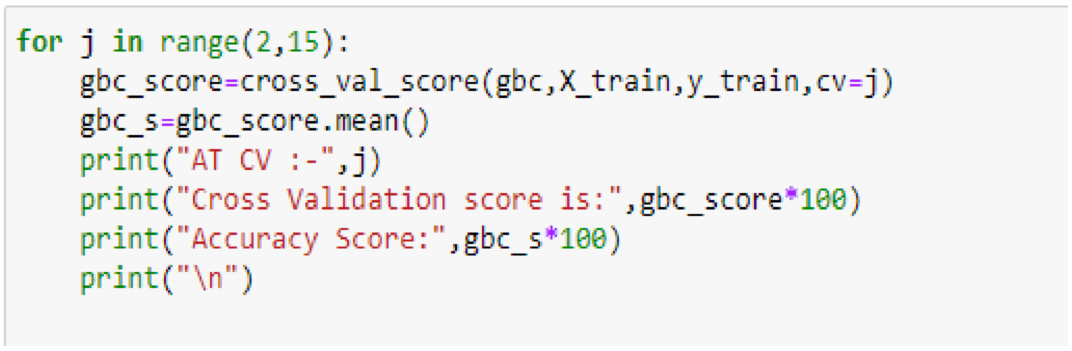
As shown in Fig 25, the Gradient Boosting Classifier, Accuracy Score is 76.46%

Confusion Matrix:
Fig 26.

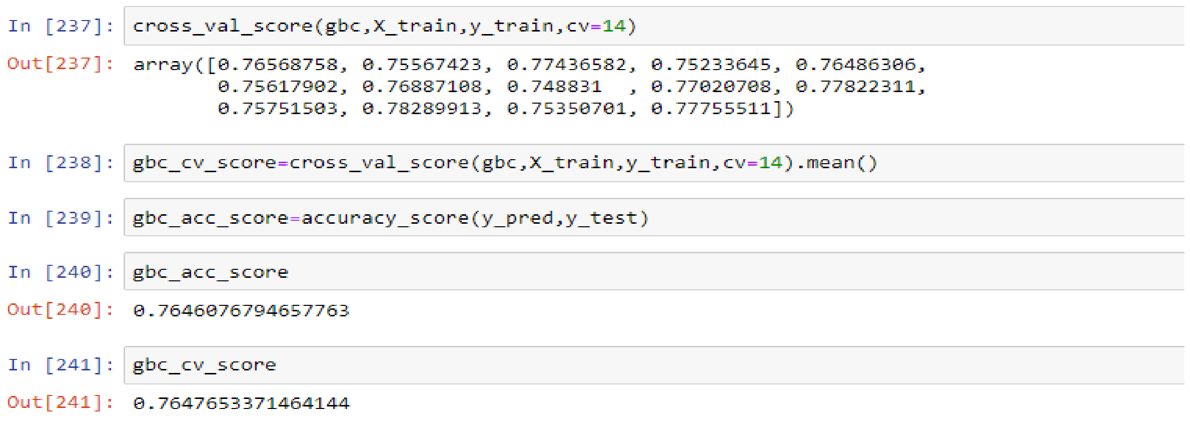


Also, as shown in Fig 27, we have observed Score of models on different fold by using Cross Validation to understand our model performance on various fold and ultimate have found the difference with Accuracy score and Cross Validation Score to understand model performance in context of overfitting or underfitting.

Fig 27.



```
for j in range(2,15):
    gbc_score=cross_val_score(gbc,X_train,y_train,cv=j)
    gbc_s=gbc_score.mean()
    print("AT CV :-",j)
    print("Cross Validation score is:",gbc_score*100)
    print("Accuracy Score:",gbc_s*100)
    print("\n")
```



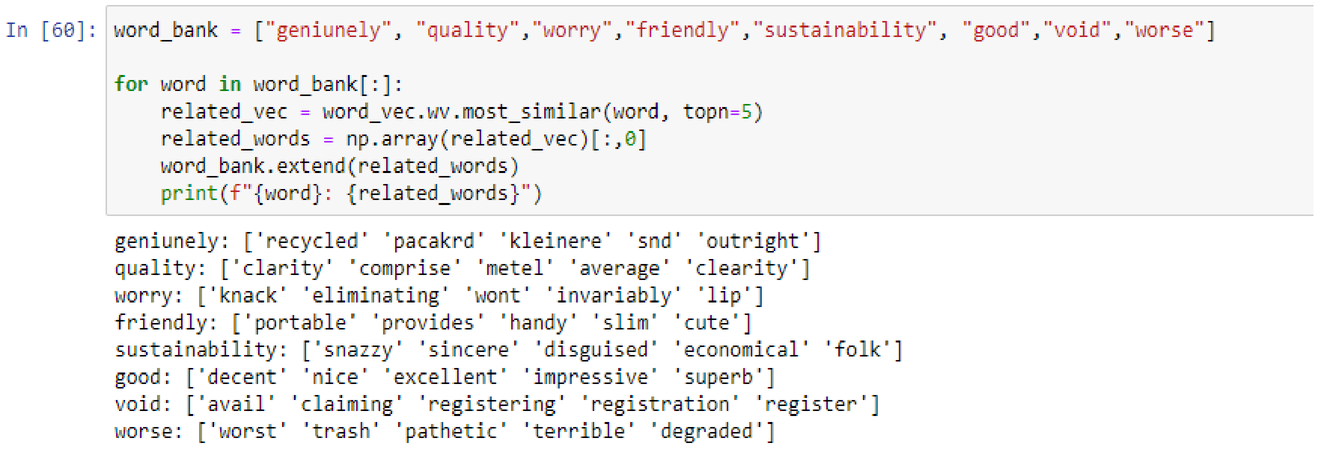
```
In [237]: cross_val_score(gbc,X_train,y_train,cv=14)
Out[237]: array([0.76568758, 0.75567423, 0.77436582, 0.75233645, 0.76486306,
                 0.75617902, 0.76887108, 0.748831  , 0.77020708, 0.77822311,
                 0.75751503, 0.78289913, 0.75350701, 0.77755511])

In [238]: gbc_cv_score=cross_val_score(gbc,X_train,y_train,cv=14).mean()

In [239]: gbc_acc_score=accuracy_score(y_pred,y_test)

In [240]: gbc_acc_score
Out[240]: 0.7646076794657763

In [241]: gbc_cv_score
Out[241]: 0.7647653371464144
```

➢ Visualizations

In this Section we have visualized the features with various graph and plot to understand or analyse our features for our model.

- **Word2vec:**

Fig 28**:**

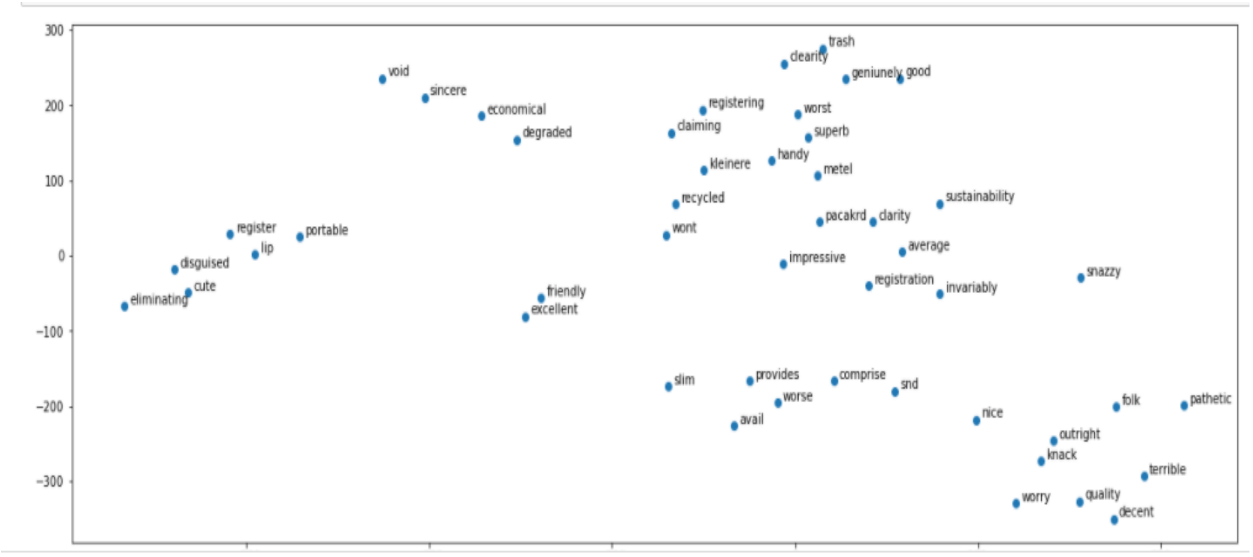


```
In [60]: word_bank = ["geniunely", "quality","worry","friendly","sustainability", "good","void","worse"]

         for word in word_bank[:]:
             related_vec = word_vec.wv.most_similar(word, topn=5)
             related_words = np.array(related_vec)[:,0]
             word_bank.extend(related_words)
             print(f"{word}: {related_words}")

         geniunely: ['recycled' 'pacakrd' 'kleinere' 'snd' 'outright']
         quality: ['clarity' 'comprise' 'metel' 'average' 'clearity']
         worry: ['knack' 'eliminating' 'wont' 'invariably' 'lip']
         friendly: ['portable' 'provides' 'handy' 'slim' 'cute']
         sustainability: ['snazzy' 'sincere' 'disguised' 'economical' 'folk']
         good: ['decent' 'nice' 'excellent' 'impressive' 'superb']
         void: ['avail' 'claiming' 'registering' 'registration' 'register']
         worse: ['worst' 'trash' 'pathetic' 'terrible' 'degraded']
```

- **T-SNE:**

Like PCA, the t-Distributed Stochastic Neighbour Embedding (t-SNE) is another dimensionality reduction technique that assists in visualizing high-dimensional datasets. To perceive the similarity between the related words in terms of spatial distance, t-SNE provided the coordinates of each word in a 2D scatterplot plane as shown in Fig 29.

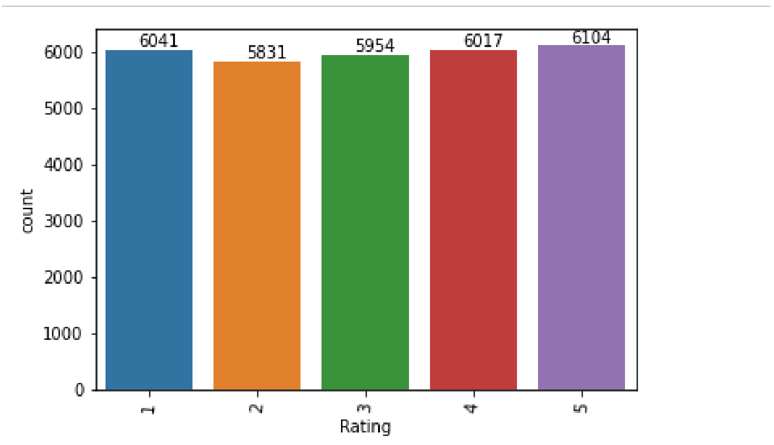
Fig. 29



- **Target label:**

As shown in Fig 30, the target label having almost equal number of observations, means we can say that our target variable is balanced well for our model.

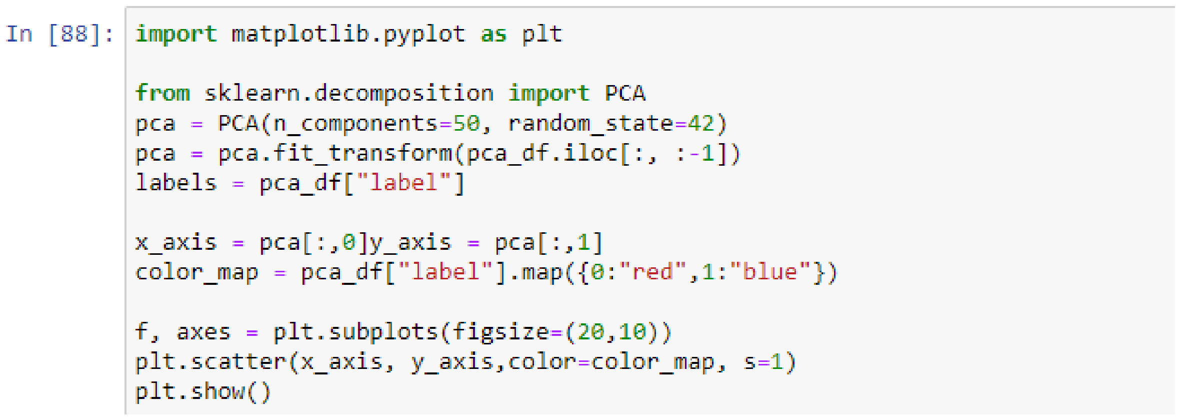
Fig 30.



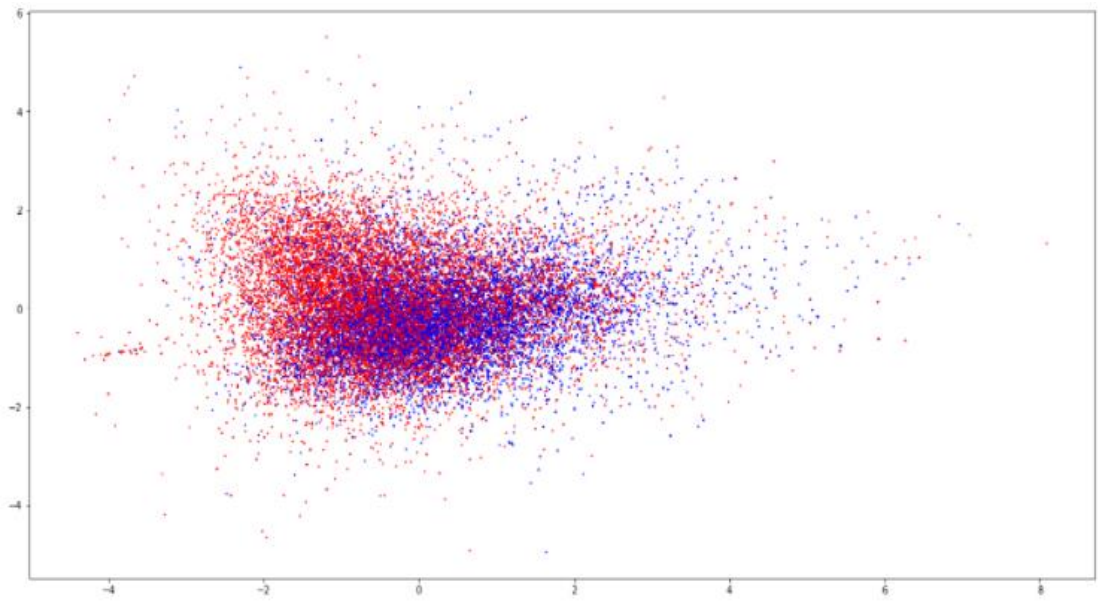
- **Principal Component Analysis:**

  Principal Component Analysis (PCA) is a dimensionality reduction technique that we can use on our model to reduce it. This will help visualize if there is a clear decision boundary along the rating classifications. The more datapoints belonging to the same class are clustered together, the higher the likelihood that our machine learning model is simpler and more effective as shown in Fig 31.

Fig: 31



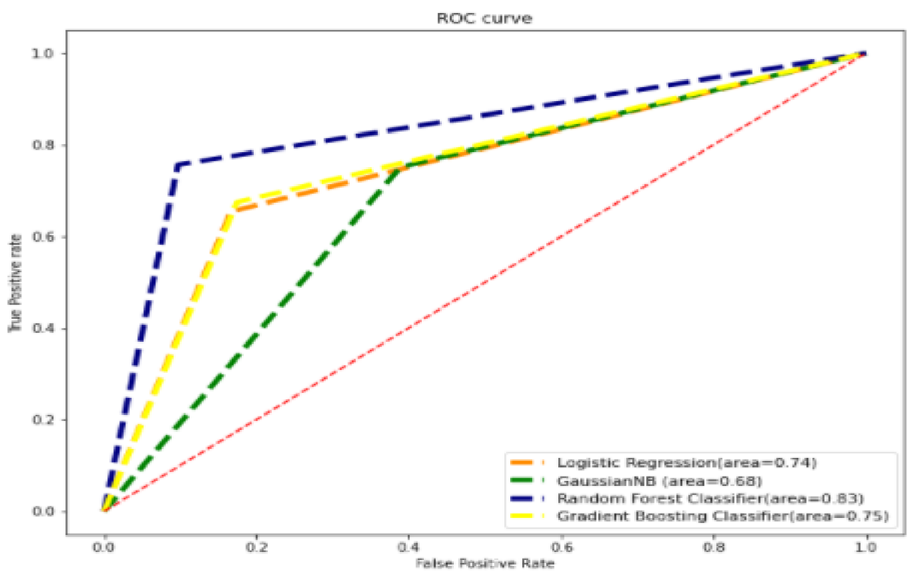
```
In [88]: import matplotlib.pyplot as plt

         from sklearn.decomposition import PCA
         pca = PCA(n_components=50, random_state=42)
         pca = pca.fit_transform(pca_df.iloc[:, :-1])
         labels = pca_df["label"]

         x_axis = pca[:,0]y_axis = pca[:,1]
         color_map = pca_df["label"].map({0:"red",1:"blue"})

         f, axes = plt.subplots(figsize=(20,10))
         plt.scatter(x_axis, y_axis,color=color_map, s=1)
         plt.show()
```



- **AUC ROC Curve:**

To check the best performing model, we have applied AUC ROC curve as shown in below Fig 31.

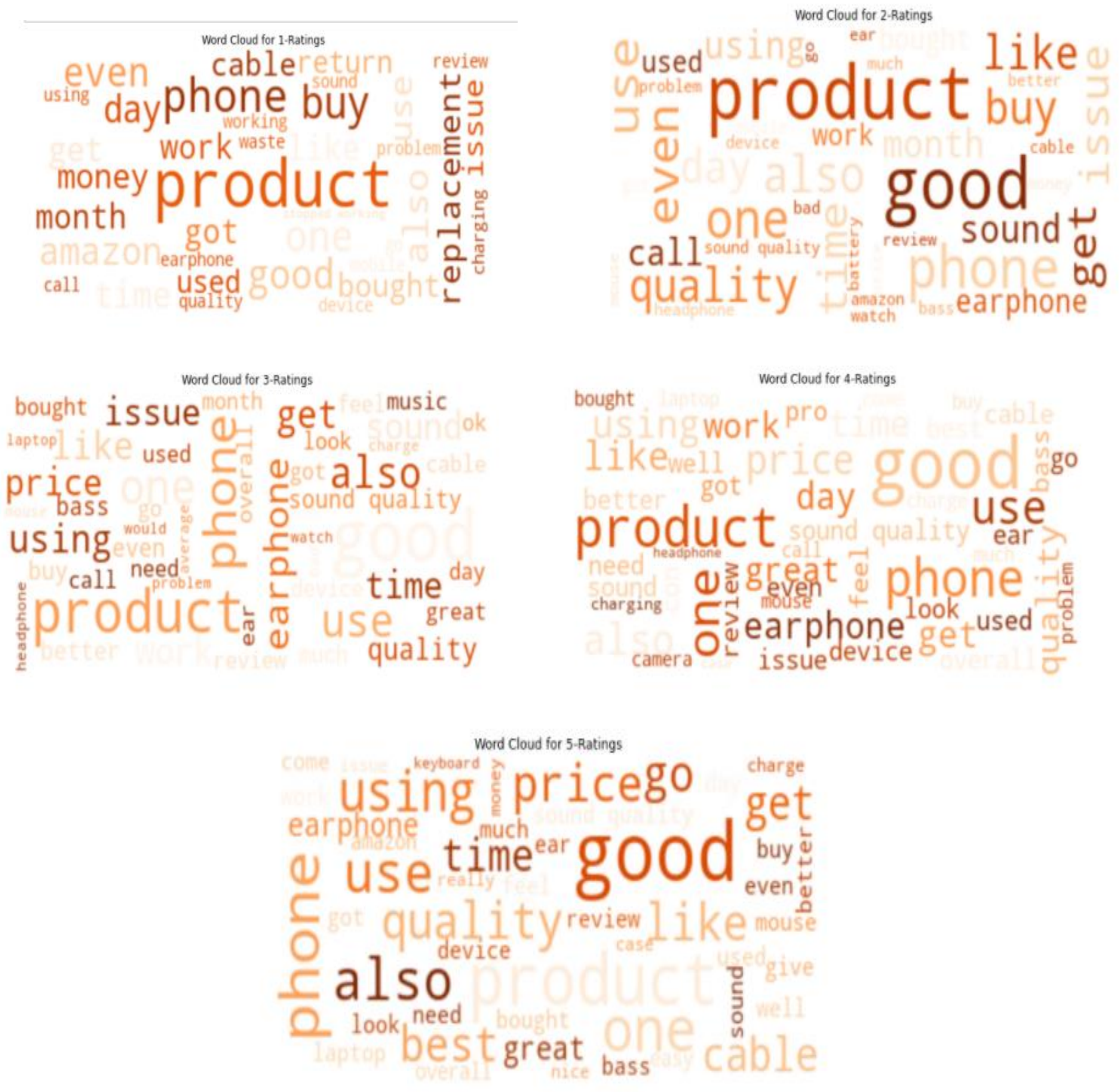
Fig: 31



- **Word cloud:**

As shown in Fig 32, word cloud helps us in visualizing the most important words in each class of target variable i.e. 5Star,4Star,3Star,2Star and 1Star.

Fig.32.

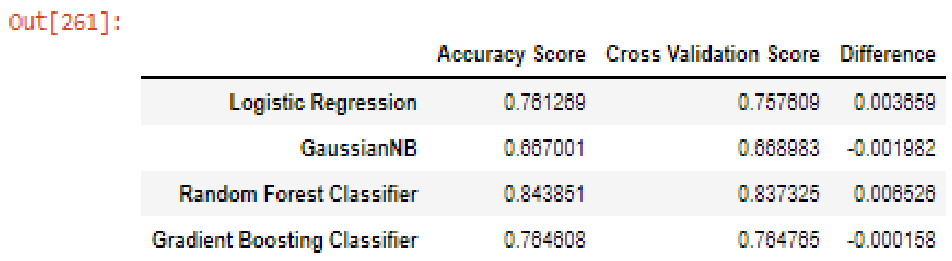


➢ Interpretation of the Results

Below are the interpreted from the visualizations and modelling:

- ✓ We have seen that Gradient Boosting Classifier Accuracy Score with compare to CV Score with different 15 k fold is providing good performance as shown in Fig 33.

Fig 33.



```
Out[261]:
```

	Accuracy Score	Cross Validation Score	Difference
Logistic Regression	0.761269	0.757609	0.003659
GaussianNB	0.667001	0.668983	-0.001982
Random Forest Classifier	0.843851	0.837325	0.006526
Gradient Boosting Classifier	0.764608	0.764765	-0.000158

CONCLUSION

➢ Key Findings and Conclusions of the Study

- ✓ We have shown in Fig 33 Gradient Boosting Classifier algorithm is performing better compared to GaussianNB, Logistic Regression and Random Forest Classifier.
- ✓ While doing Principal Components Analysis, as shown in Fig 31 we can go with 50 number of components to get more than 90% of variance explained, but we have tried the model performance without PCA applied in model.
- ✓ We have Confirmed that based on R2 Score, CV Score Gradient Boosting Classifier Model is performing better.

➢ Learning Outcomes of the Study in respect of Data Science

Below are the Learning Outcome of the Study:

- ✓ Collecting the data from website, where after scraping data would be not in proper format.
- ✓ Our feature data is completely in string.
- ✓ Dealing with huge number to text document.
- ✓ Finding the similarity words and word Algebra
- ✓ Selecting the Best Model based on R2 Score, CV Score.
- ✓ Selecting best Model based on AUC ROC Curve.

➢ Limitations of this work and Scope for Future Work

Mention below are the limitations and future scope steps:

- ✓ We could apply some more NLP based algorithm like Countvector, TF-IDF, n-grams technique and evaluate our model performance.
- ✓ Due to time constraint we could not done Hyper tuning on selected model where we could improve our model accuracy based on tuned parameter.