RATINGS PROJECT

Submitted by:

ANEESHA B SOMAN

ACKNOWLEDGMENT

The success and final outcome of this project required a lot of guidance and assistance from Keshav Bansal Sir and I am Extremely fortunate to have got this all along the completion of my project work.I am also grateful to Fliprobo Company for assigning this project to me.

Various references were used like:

Anlyticsvidhya, Medium, Data trained Reference materials and Github which helped me in completion of the project

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

The project addresses the following central questions:

1.A model to predict the rating based on review

2.Problems associated with sentiment analyzers

3.A model which can quickly gain insights using large volumes of text data

4.Importance of each word in the reviews

Conceptual Background of the Domain Problem

Till a few years ago, analyzing a customer's sentiments was not something that was taken seriously. Today, thanks to advancements in technology and also in the thinking of businesses, sentiment analysis is emerging as a viable tool. What makes it interesting and rather different from the other forms of data analytics is that this one deals with emotions, and as we all know, emotions are never black or white.

Ratings tells a brand or an enterprise what the world or the consumer feels about it. The ratings could be 1,2,3,4,5 where 1 being the lowest/poorest rating and 5 being the highest/best rating.

The model should be able to accept any review of the customer and predict the rating that could be given for the review.

Review of Literature

Rating analysis being an area of wide interest in the current Data Science Community, various papers have been published in the regard using various datasets. Some of the papers which are widely open and used by the community are:

Trend	Dataset	Best Model	Paper Title	P
	SST-2 Binary classification	SMART-RoBERTa Large	SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization	
	IMDb	NB-weighted-BON + dv-cosine	Sentiment Classification Using Document Embeddings Trained with Cosine Similarity	
	SST-5 Fine-grained classification	RoBERTa-large+Self-Explaining	Self-Explaining Structures Improve NLP Models	
	Yelp Binary classification	XLNet	XLNet: Generalized Autoregressive Pretraining for Language Understanding	
	Yelp Fine-grained classification	XLNet	XLNet: Generalized Autoregressive Pretraining for Language Understanding	
	MR	EFL	Entailment as Few-Shot Learner	
	Amazon Review Polarity	BERT large	Unsupervised Data Augmentation for Consistency Training	
	Amazon Review Full	BERT large	Unsupervised Data Augmentation for Consistency Training	
	User and product information	BILSTM+CHIM	Rethinking Attribute Representation and Injection for Sentiment Classification	
	CR	EFL	Entailment as Few-Shot Learner	

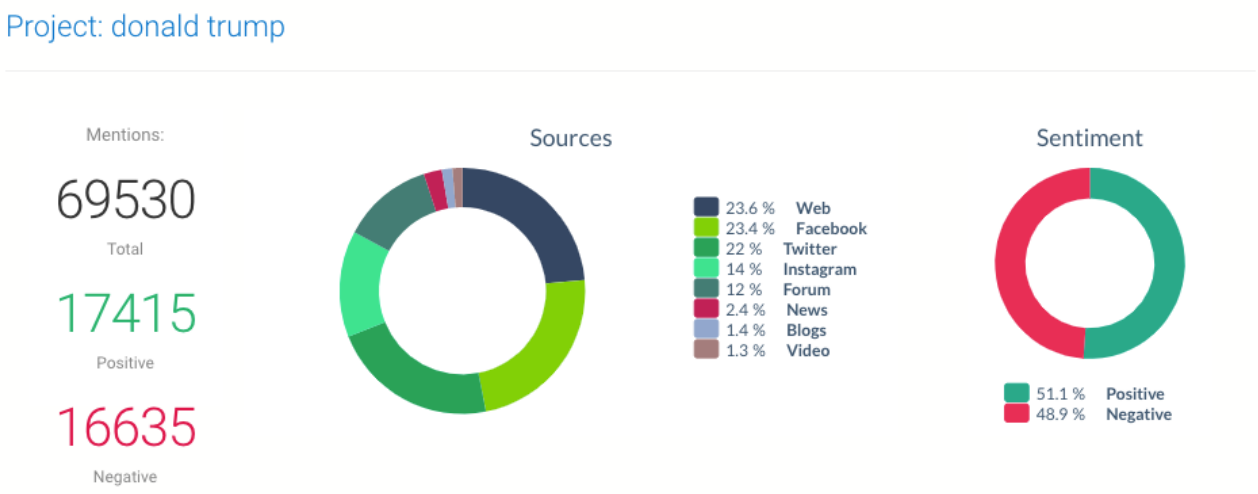
	SemEval 2014 Task 4 Subtask 1+2	GRACE	GRACE: Gradient Harmonized and Cascaded Labeling for Aspect-based Sentiment Analysis
	TweetEval	BERTweet	BERTweet: A pre-trained language model for English Tweets
	Multi-Domain Sentiment Dataset	UDALM: Unsupervised Domain Adaptation through Language Modeling	UDALM: Unsupervised Domain Adaptation through Language Modeling
	MPQA	EFL	Entailment as Few-Shot Learner
	SemEval 2017 Task 4-A	LSTMs+CNNs ensemble with multiple conv. ops	BB_twtr at SemEval-2017 Task 4: Twitter Sentiment Analysis with CNNs and LSTMs
	DBRD	RobBERT v2	RobBERT: a Dutch RoBERTa-based Language Model
	Twitter	AEN-BERT	Attentional Encoder Network for Targeted Sentiment Classification
	FiQA	FinBERT	FinBERT: Financial Sentiment Analysis with Pre-trained Language Models
	RuSentiment	RuBERT-RuSentiment	Deep Transfer Learning Baselines for Sentiment Analysis in Russian
	SemEval	LSTMs+CNNs ensemble with multiple conv. ops	BB_twtr at SemEval-2017 Task 4: Twitter Sentiment Analysis with CNNs and LSTMs
	Sogou News	fastText, h=10, bigram	Bag of Tricks for Efficient Text Classification
	ASTD	CNN-LSTM	Mazajak: An Online Arabic Sentiment Analyser
	ArSAS	CNN-LSTM	Mazajak: An Online Arabic Sentiment Analyser

Motivation for the Problem Undertaken

Data is currently gold in every business firm/government/social cause. Any product, social works or service provided cannot be on the constant improvement if there is no understanding about the current value of it in the market. Hence the reviews given by the customer/client is essential in improving the product.

The project in hand solves the issue of taking in large sets of data and analyzing it to easy terms for a third person to understand (in the form of ratings)

One such example can be seen in the below example where Trumps works were reviewed for through various social media websites and this was rated on two classes making it easy for the policy makers and the voters to understand.



Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem

a.**Statistical models used:**

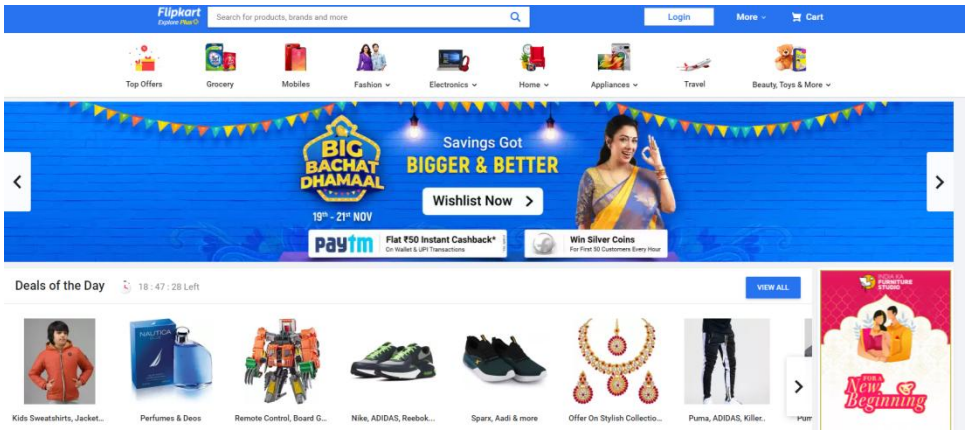
- Pipeline for applying classification
- TF-IDF

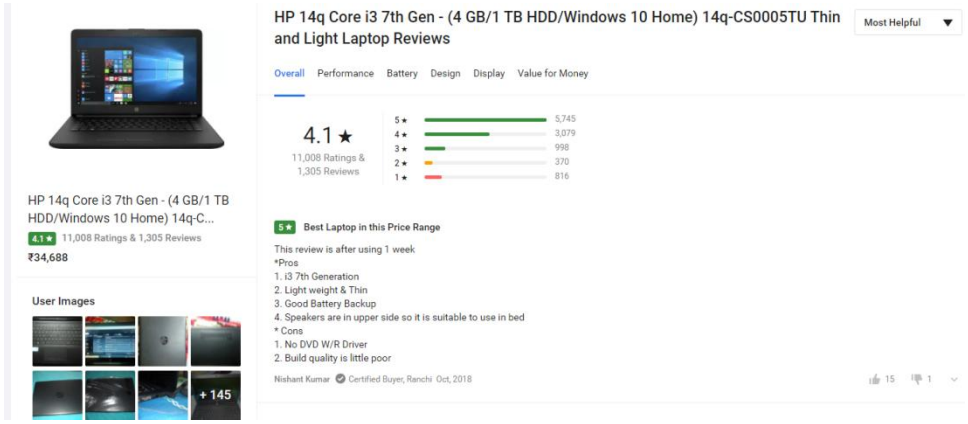
b. **Analytical models:**

Descriptive Analytics	**Diagnostic Analytics**	**Predictive Analytics**	**Prescriptive Analytics**
Data visualization done through matplotlib and seaborn between Features and Label and features and features. Also heatmap and wordcloud made to analyse the features and most common words	The reason for change is Understood by tf-idf and analyzing each words weightage. Each word is vectorized to be fed into the model	Prediction is done through various classification techniques	Prescriptive analysis is done through the Model created. Through accuracy score,recall,f1 score and precision

Data Sources and their formats

To undertake the experiments we have Mined Data from **flipkart.com** of the reviews of different laptops, Phones, Headphones, smart watches, Professional Cameras, Printers, Monitors, Home theater, Router





Scrapping was done through:

- Jupyter notebook
- Selenium

Data is scrapped for 35,267 reviews

```
import pandas as pd
import selenium
from selenium import webdriver
import time
from selenium.common.exceptions import StaleElementReferenceException, NoSuchElementException
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import datetime
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.common.keys import Keys


driver=webdriver.Chrome(r"C:\chromeDriver.exe")
```

```
<ipython-input-1-c537a1cd45d3>:13: DeprecationWarning: executable_path has been deprecated, please pass in a Service object
  driver=webdriver.Chrome(r"C:\chromeDriver.exe")
```

```
from selenium.common.exceptions import NoSuchElementException
```

FROM FLIPKART

```
Ratings=[]
Full_review=[]
Object=[]
```

Laptops

In [44]:
```
url_laptops={"https://www.flipkart.com/hp-14q-core-i3-7th-gen-4-gb-1-tb-hdd-windows-10-home-14q-cs0005tu-thin-light-laptop/produc
             "https://www.flipkart.com/hp-pavilion-gaming-ryzen-5-quad-core-3550h-8-gb-1-tb-hdd-windows-10-home-4-gb-graphics-nvi
             "https://www.flipkart.com/lenovo-ideapad-s145-ryzen-3-dual-core-3200u-4-gb-1-tb-hdd-windows-10-home-s145-15api-thin-
            "https://www.flipkart.com/apple-macbook-air-core-i5-5th-gen-8-gb-128-gb-ssd-mac-os-sierra-mqd32hn-a/product-reviews/i
            "https://www.flipkart.com/asus-celeron-dual-core-4-gb-1-tb-hdd-windows-10-home-x543ma-gq1015t-laptop/product-reviews/
```

In [ ]:
```
#running a loop to extract all the required information from url


for laptops_url in url_laptops:
    driver.get(laptops_url)

    for i in range(0,100):
        for j in driver.find_elements_by_xpath("//div[@class='_3LWZlK _1BLPMq' or @class='_3LWZlK _32lA32 _1BLPMq' or @class='_3L
            Ratings.append(j.text)
            Object.append("Laptop")


        for k in driver.find_elements_by_xpath("//div[@class='t-ZTKy']/div/div"):
            Full_review.append(k.text.replace('\n',' '))



        button=driver.find_element_by_xpath("//*[contains(text(), 'Next')]")
        driver.execute_script("arguments[0].click();", button)

        time.sleep(2)
```

In [47]:
```
print(len(Ratings),len(Full_review),len(Object))
```

```
2951 2944 2951
```

MOBILE PHONES

In [7]:
```
urls_mobile_phones={'https://www.flipkart.com/redmi-8a-dual-sky-white-32-gb/product-reviews/itmf5568beb3c3a5?pid=MOBFP7FMS6GNPQFY
                    'https://www.flipkart.com/redmi-9-sporty-orange-64-gb/product-reviews/itm4fb151383983b?pid=MOBFV5FPCJC9ZKRB&
                   'https://www.flipkart.com/samsung-galaxy-f12-celestial-black-64-gb/product-reviews/itmaae10b8db3909?pid=MOBGF
```

In [49]:
```
for mobile_url in urls_mobile_phones:
    driver.get(mobile_url)

    for i in range(0,110):
        for j in driver.find_elements_by_xpath("//div[@class='_3LWZlK _1BLPMq' or @class='_3LWZlK _32lA32 _1BLPMq' or @class='_3
            Ratings.append(j.text)
            Object.append("Mobile phone")

        for k in driver.find_elements_by_xpath("//div[@class='t-ZTKy']/div/div"):
            Full_review.append(k.text.replace('\n',' '))

        try:
            button=driver.find_element_by_xpath("//*[contains(text(), 'Next')]")
            driver.execute_script("arguments[0].click();", button)
        except NoSuchElementException:
            pass
        time.sleep(2)
```

```
<ipython-input-49-967bcf4b2f30>:5: DeprecationWarning: find_elements_by_* commands are deprecated. Please use find_elements() i
nstead
  for j in driver.find_elements_by_xpath("//div[@class='_3LWZlK _1BLPMq' or @class='_3LWZlK _32lA32 _1BLPMq' or @class='_3LWZlK
_1rdVr6 _1BLPMq']"):
<ipython-input-49-967bcf4b2f30>:9: DeprecationWarning: find_elements_by_* commands are deprecated. Please use find_elements() i
nstead
  for k in driver.find_elements_by_xpath("//div[@class='t-ZTKy']/div/div"):
<ipython-input-49-967bcf4b2f30>:13: DeprecationWarning: find_element_by_* commands are deprecated. Please use find_element() in
stead
  button=driver.find_element_by_xpath("//*[contains(text(), 'Next')]")
```

In [50]:
```
print(len(Ratings),len(Full_review),len(Object))
```

```
5868 5861 5868
```

The data scrapped was stored into a CSV file with the name Reviews

:
```
#creating a dataframe
Reviews=pd.DataFrame({})
Reviews['Ratings']=x
Reviews['Full_review']=z
Reviews['Object']=y

Reviews
```

:

	Ratings	Full_review	Object
0	5	I was a bit skeptical about buying this laptop...	Laptop
1	5	Highly recommended, the packing was fantastic,...	Laptop
2	4	I wanted to buy a laptoo fir mild use which wi...	Laptop
3	5	apple is best among all, even it is 2017 model...	Laptop
4	5	A decent purchase at very low rate thanks to f...	Laptop
...	...	...	...
35262	1	Everything is good but the configuration is ve...	Router
35263	4	Nice product	Router
35264	1	Superb quality	Router
35265	5	Aswme product Thanks	Router
35266	1	Worst product keeps getting disconnected .afte...	Router

35267 rows × 3 columns

:
```
#converting into csv format
Reviews.to_csv("Reviews.csv")
```

:
```
#closing the driver
driver.close()
```

The features and labels are:

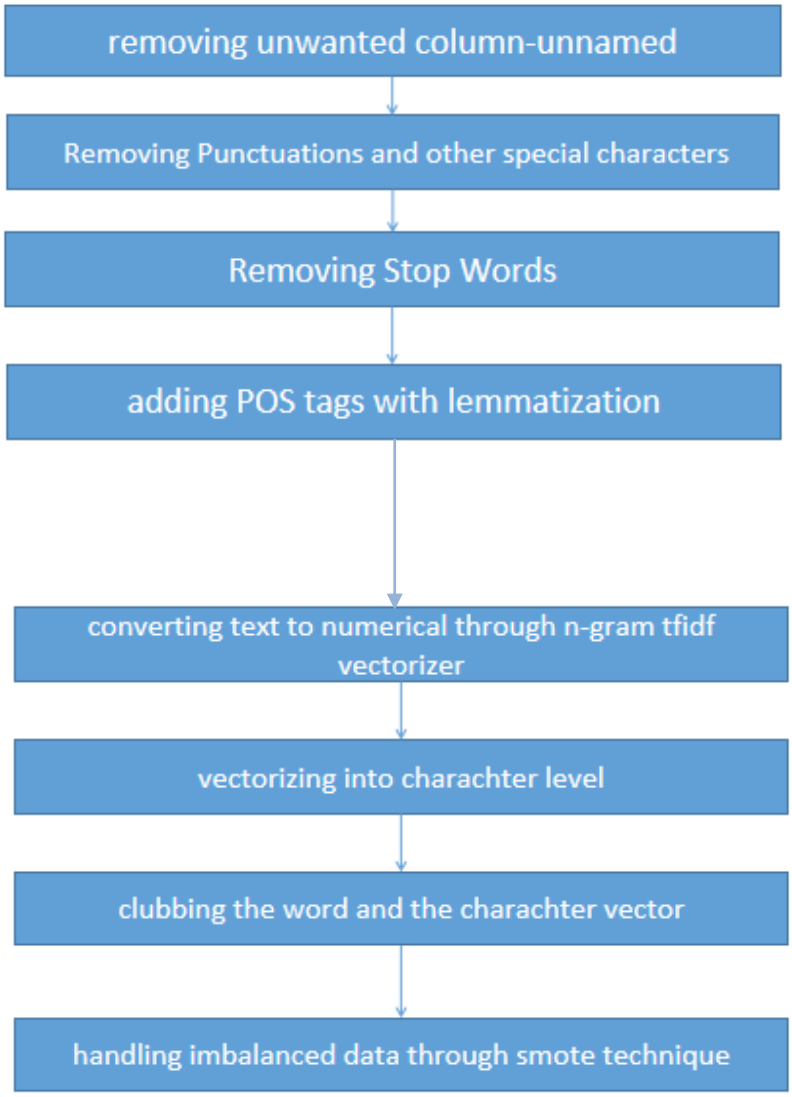
```
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Unnamed: 0    35267 non-null  int64
 1   Ratings       35267 non-null  int64
 2   Full_review   35267 non-null  object
 3   Object        35267 non-null  object
dtypes: int64(2), object(2)
memory usage: 1.1+ MB
```

The dataframe has 35267 rows and 4 columns where label is Ratings and the other three are features

Train data

	Ratings	Full_revie	Object
0	5	I was a bit	Laptop
1	5	Highly rec	Laptop
2	4	I wanted t	Laptop
3	5	apple is be	Laptop
4	5	A decent p	Laptop
5	5	this is an a	Laptop
6	5	I bought th	Laptop
7	5	2017 mod	Laptop
8	5	I used to a	Laptop

Data Preprocessing Done

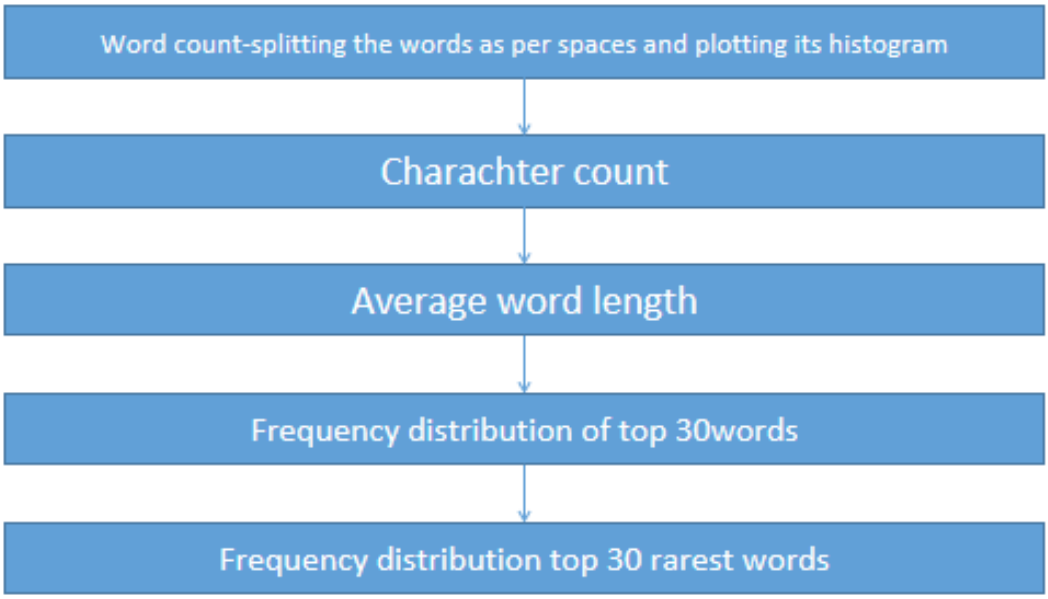


Feature extraction

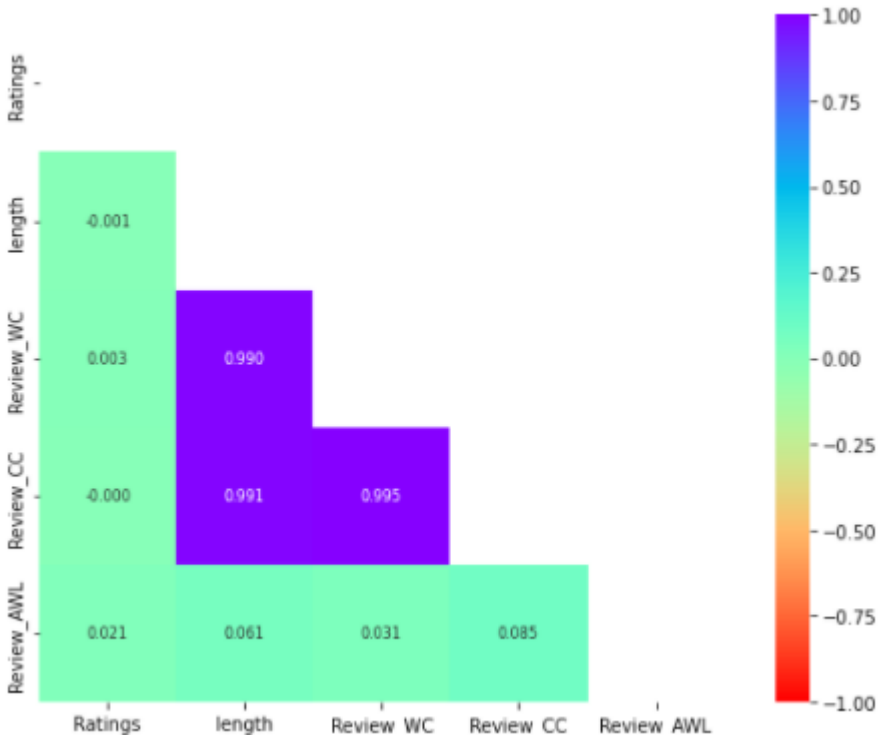
	words	tf	idf	tfidf
0	I	4	-5.058225	-20.232902
1	laptop	3	-2.839728	-8.519184

Converted textual data into numerical form

Exploratory Data Analysis



Data Inputs- Logic- Output Relationships



The set of assumptions

1.The Stopwords of spacy contains most of the stopwords

2.The reviewer has communicated their intend in proper English format

3.There is proper grammer in the sentence like active/passive, tenses and other forms of English grammer

4.It is a Multiclass classification type of problem

5.Dataset if imbalanced which needs to be balanced

Hardware and Software Requirements and Tools Used

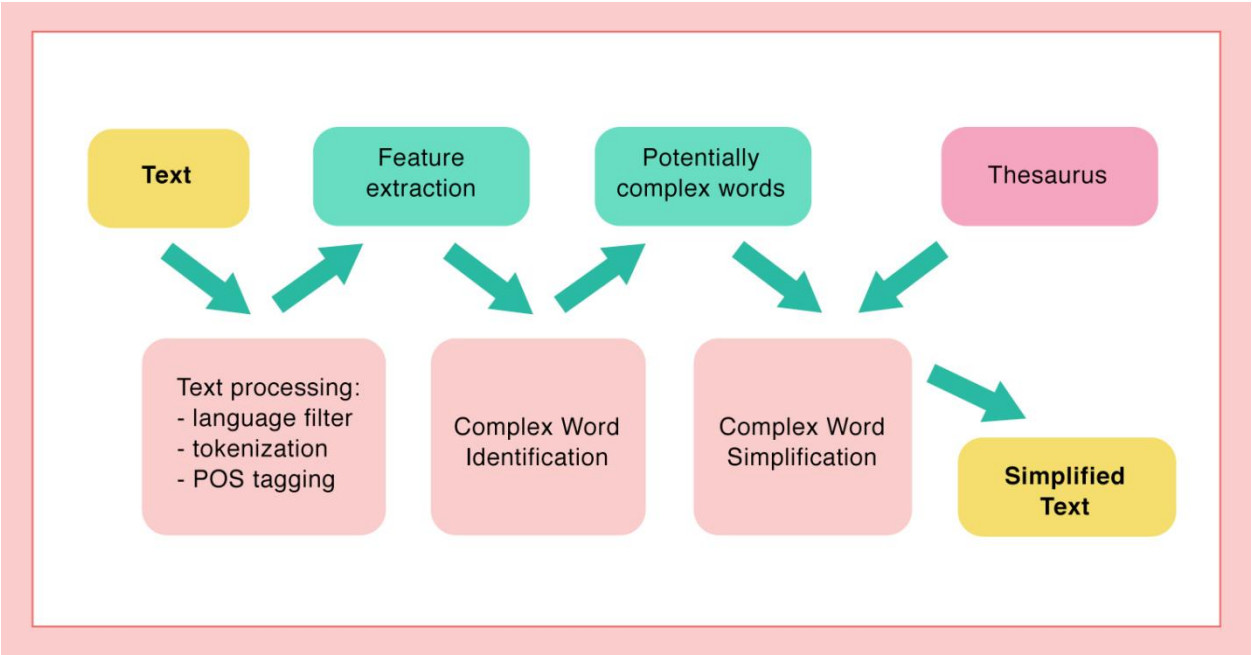
Hardware technology being used.

- RAM : 8 GB
- CPU :Intel® Core™ i7-10510U CPU @ 1.80GHz
- GPU: NVIDA-Cuda , performed in tf2.4 enviroment

Software technology being used.

- Programming language : Python
- Distribution : Anaconda Navigator
- Browser based language shell : Jupyter Notebook
- Libraries/Packages specifically being used.: Pandas , NumPy, matplotlib, seaborn, scikit-learn, pandas profiling, missingno

Model/s Development and Evaluation



The problem at Hand is a CLASSIFICATION problem

Identified Approaches (Algorithms) for classification

KNeighborsClassifier	RandomForestClassifier	GradientBoostingClassifier
DecisionTreeClassifier	AdaBoostClassifier	BaggingClassifier
XGBClassifier	MultinomialNB	ExtraTreesClassifier

MODELLING

```
#converting text to numerical through n-gram tfidf vectorizer
from sklearn.feature_extraction.text import TfidfVectorizer


word_vectorizer=TfidfVectorizer(sublinear_tf=True,
                                strip_accents='unicode',
                                analyzer='word',
                                token_pattern=r'\w{1,}',
                                stop_words='english',
                                ngram_range=(1,3))

word_vectorizer.fit(x)
train_word_features=word_vectorizer.transform(x)
```

```
#vectorizing into charachter level
char_vectorizer=TfidfVectorizer(sublinear_tf=True,
                                strip_accents='unicode',
                                analyzer='char',
                                stop_words='english',
                                ngram_range=(2,6),
                                 max_features=50000)

char_vectorizer.fit(x)
train_char_features=char_vectorizer.transform(x)
```

```
C:\Users\Vimal\anaconda3\envs\tensorflow\lib\site-packages\sklearn\feature_extraction\text.py:539: UserWarning: The parameter
'stop_words' will not be used since 'analyzer' != 'word'
  "The parameter 'stop_words' will not be used"
```

```
#clubbing the word and the charachter vector
train_features=hstack([train_char_features,train_word_features])
```

Headstack is a library which will horizontally club the word and charachter matrix

Run and Evaluate selected models using key metrics

```
x_train, x_test, y_train, y_test = train_test_split(train_features, y, test_size=0.25, random_state=9)

print(y_train.shape)
print(y_test.shape)

print(x_train.shape)
print(x_test.shape)
```

```
(26450,)
(8817,)
(26450, 213068)
(8817, 213068)
```

```
from imblearn import under_sampling, over_sampling
from imblearn.over_sampling import SMOTE
```

```
#handling imbalanced data through smote technique

from imblearn.over_sampling import SMOTE
smt = SMOTE(random_state=0)
oversample = SMOTE()
x_train_SMOTE, y_train_SMOTE= oversample.fit_resample(x_train, y_train)

print("The number of classes before fit{}".format(Counter(y_train)))
print("The number of classes after fit {}".format(Counter(y_train_SMOTE)))
```

```
The number of classes before fitCounter({5: 14351, 4: 5504, 1: 3670, 3: 2068, 2: 857})
The number of classes after fit Counter({5: 14351, 1: 14351, 3: 14351, 4: 14351, 2: 14351})
```

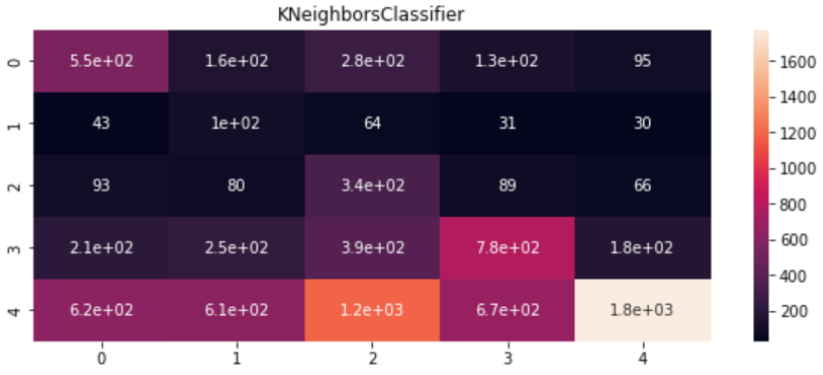
```
KNN=KNeighborsClassifier(n_neighbors=6)
DT=DecisionTreeClassifier(random_state=6)
XGB=XGBClassifier()
RF=RandomForestClassifier()
ADA=AdaBoostClassifier()
MNB=MultinomialNB()
GBC=GradientBoostingClassifier()
BC=BaggingClassifier()
ETC=ExtraTreesClassifier()

models= []
models.append(('KNeighborsClassifier', KNN))
models.append(('DecisionTreeClassifier', DT))
models.append(('XGBClassifier', XGB))
models.append(('RandomForestClassifier', RF))
models.append(('AdaBoostClassifier', ADA))
models.append(('MultinomialNB', MNB))
models.append(('GradientBoostingClassifier', GBC))
models.append(('BaggingClassifier', BC))
models.append(('ExtraTreesClassifier', ETC))
```

```
import tensorflow as tf

if tf.test.gpu_device_name():
    print('Default GPU Device: {}'.format(tf.test.gpu_device_name()))
else:
    print("Please install GPU version of TF")

Model= []
score= []
cvs=[]
difference=[]
for name,model in models:
    print('****************************************',name,'****************************************************')
    print('\n')
    Model.append(name)
    model.fit(x_train_SMOTE,y_train_SMOTE)
    print(model)
    pre=model.predict(x_test)
    print('\n')
    AS=accuracy_score(y_test,pre)
    print('Accuracy_score = ',AS)
    score.append(AS*100)
    print('\n')
    sc= cross_val_score(model, x_train_SMOTE, y_train_SMOTE, cv=10, scoring='accuracy').mean()
    print('Cross_Val_Score = ',sc)
    cvs.append(sc*100)
    print('\n')
    print('classification_report\n',classification_report(y_test,pre))
    print('\n')
    cm=confusion_matrix(y_test,pre)
    print(cm)
    print('\n')
    plt.figure(figsize=(10,40))
    plt.subplot(911)
    plt.title(name)
    print(sns.heatmap(cm,annot=True))
    print('\n\n')
    diff=AS-sc
    difference.append(diff)
```

EVALUATION OF MODELS

1. K Neighbors Classifier (n_neighbors=6)

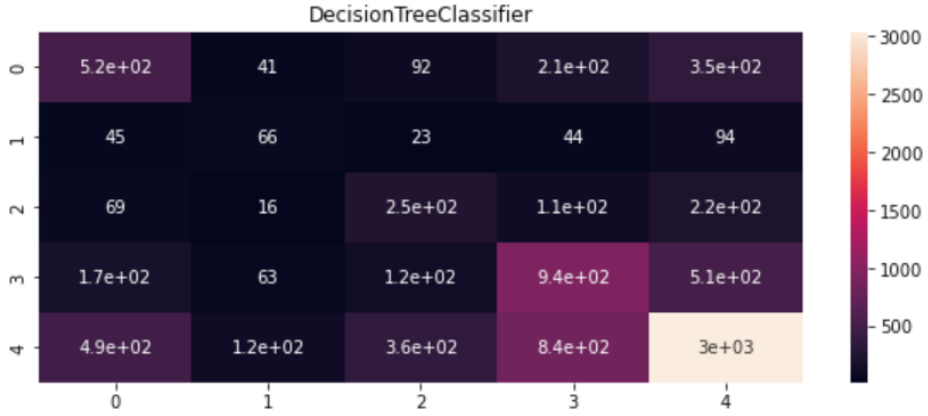
```
Accuracy_score =  0.40285811500510377

Cross_Val_Score =  0.6772496416626852

classification_report
              precision    recall  f1-score   support

           1       0.36      0.45      0.40      1217
           2       0.09      0.38      0.14       272
           3       0.15      0.51      0.23       669
           4       0.46      0.43      0.45      1809
           5       0.83      0.37      0.51      4850

    accuracy                           0.40      8817
   macro avg       0.38      0.43      0.35      8817
weighted avg       0.61      0.40      0.45      8817

[[ 473   30   63  138  513]
 [  34   57   17   35  129]
 [  67   10  221   67  304]
 [ 127   32  104  764  782]
 [ 320   64  278  545 3643]]
```



2. Decision Tree Classifier (random_state=6)

```
Accuracy_score =  0.5459906997845072


Cross_Val_Score =  0.6935144946958307


classification_report
              precision    recall  f1-score   support

           1       0.40      0.43      0.41      1217
           2       0.21      0.24      0.23       272
           3       0.30      0.38      0.33       669
           4       0.44      0.52      0.47      1809
           5       0.72      0.63      0.67      4850

    accuracy                           0.55      8817
   macro avg       0.41      0.44      0.42      8817
weighted avg       0.57      0.55      0.56      8817
```

```
[[ 520   41   92  213  351]
 [  45   66   23   44   94]
 [  69   16  251  112  221]
 [ 170   63  124  940  512]
 [ 486  122  360  845 3037]]
```



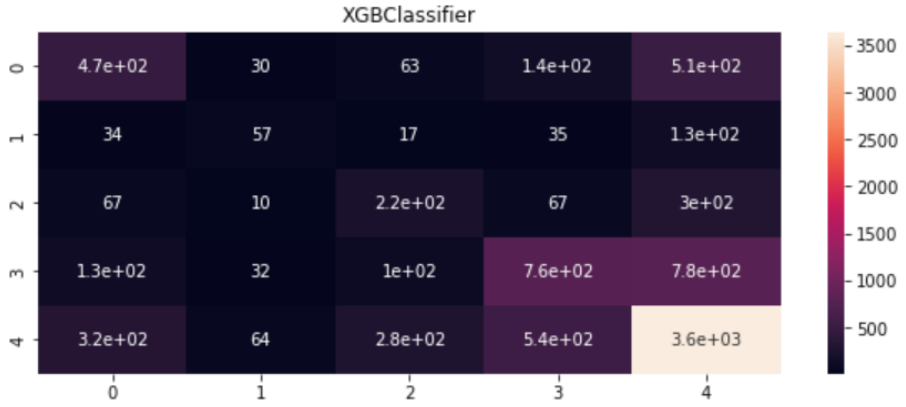
3. XGB Classifier

```
Accuracy_score =  0.5850062379494159

Cross_Val_Score =  0.7019051581151264
```

classification_report

	precision	recall	f1-score	support
1	0.46	0.39	0.42	1217
2	0.30	0.21	0.25	272
3	0.32	0.33	0.33	669
4	0.49	0.42	0.46	1809
5	0.68	0.75	0.71	4850
accuracy			0.59	8817
macro avg	0.45	0.42	0.43	8817
weighted avg	0.57	0.59	0.58	8817

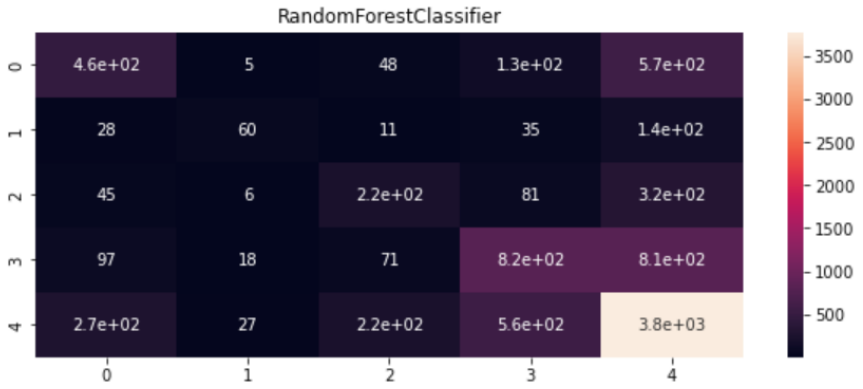
```
[[ 473   30   63  138  513]
 [  34   57   17   35  129]
 [  67   10  221   67  304]
 [ 127   32  104  764  782]
 [ 320   64  278  545 3643]]
```



4. RandomForest Classifier()

```
Accuracy_score =  0.6050810933424067


Cross_Val_Score =  0.7859261630910623


classification_report
              precision    recall  f1-score   support

           1       0.52      0.38      0.44      1217
           2       0.52      0.22      0.31       272
           3       0.38      0.33      0.36       669
           4       0.50      0.45      0.48      1809
           5       0.67      0.78      0.72      4850

    accuracy                           0.61      8817
   macro avg       0.52      0.43      0.46      8817
weighted avg       0.59      0.61      0.59      8817
```

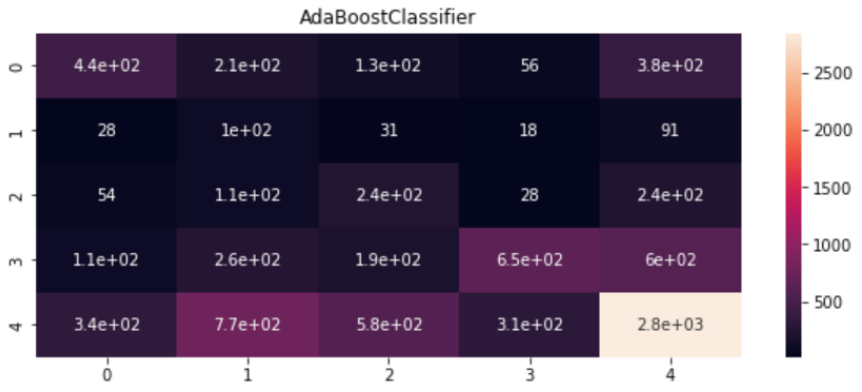
```
[[ 463    5   48  132  569]
 [  28   60   11   35  138]
 [  45    6  221   81  316]
 [  97   18   71  815  808]
 [ 266   27  224  557 3776]]
```



5. Ada Boost Classifier()

```
Accuracy_score =  0.4862198026539639


Cross_Val_Score =  0.5090112143070786


classification_report
              precision    recall  f1-score   support

           1       0.46      0.36      0.41      1217
           2       0.07      0.38      0.12       272
           3       0.21      0.36      0.26       669
           4       0.61      0.36      0.45      1809
           5       0.69      0.59      0.63      4850

    accuracy                           0.49      8817
   macro avg       0.41      0.41      0.38      8817
weighted avg       0.58      0.49      0.52      8817
```

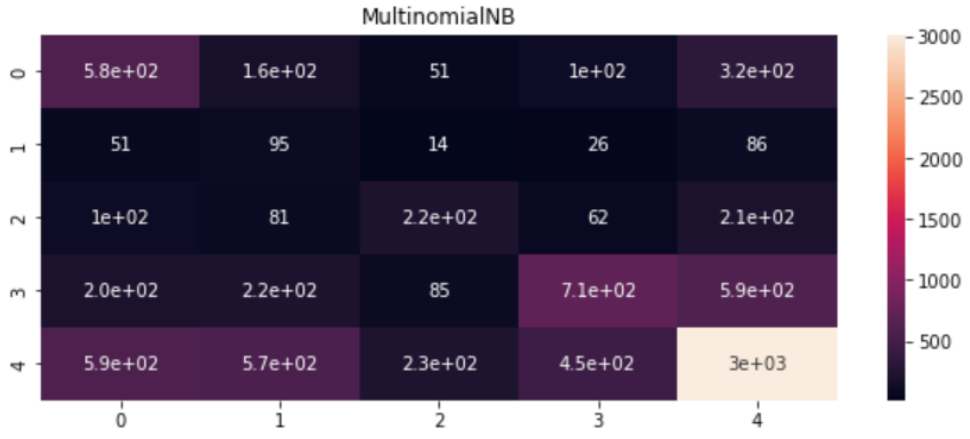
```
[[ 444  209  128   56  380]
 [  28  104   31   18   91]
 [  54  106  243   28  238]
 [ 106  259  193  653  598]
 [ 338  772  584  313 2843]]
```



6. Multinomial NB()

```
Accuracy_score =  0.5236474991493706


Cross_Val_Score =  0.6076936847175448


classification_report
              precision    recall  f1-score   support

           1       0.38      0.48      0.42      1217
           2       0.08      0.35      0.14       272
           3       0.37      0.33      0.35       669
           4       0.53      0.39      0.45      1809
           5       0.71      0.62      0.66      4850

    accuracy                           0.52      8817
   macro avg       0.41      0.43      0.40      8817
weighted avg       0.58      0.52      0.55      8817
```

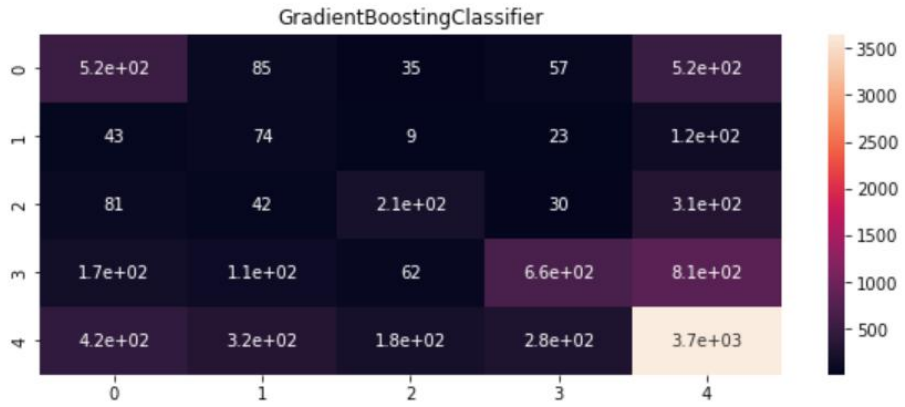
```
[[ 580  163   51  100  323]
 [  51   95   14   26   86]
 [ 101   81  219   62  206]
 [ 205  222   85  708  589]
 [ 586  573  227  449 3015]]
```



7. Gradient Boosting Classifier

```
Accuracy_score =  0.5799024611545878


Cross_Val_Score =  0.602036595076892


classification_report
              precision    recall  f1-score   support

           1       0.42      0.43      0.43      1217
           2       0.12      0.27      0.16       272
           3       0.42      0.31      0.36       669
           4       0.63      0.36      0.46      1809
           5       0.67      0.75      0.71      4850

    accuracy                           0.58      8817
   macro avg       0.45      0.43      0.42      8817
weighted avg       0.59      0.58      0.58      8817
```

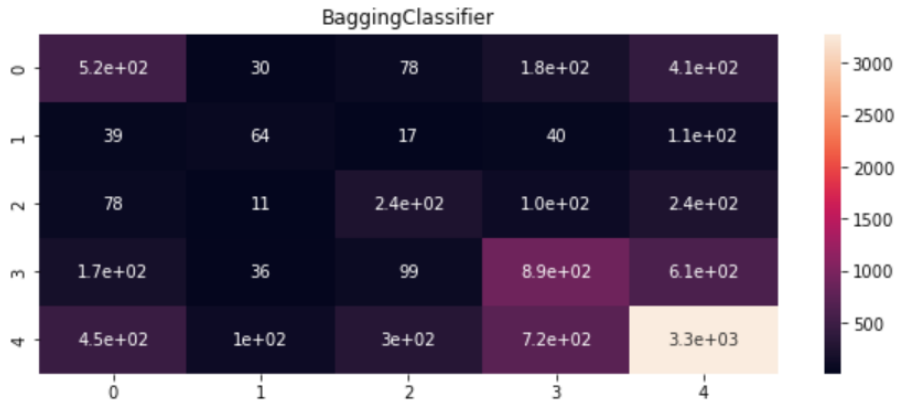
```
[[ 523   85   35   57  517]
 [  43   74    9   23  123]
 [  81   42  207   30  309]
 [ 166  113   62  655  813]
 [ 425  316  180  275 3654]]
```



8. Bagging Classifier

```
Accuracy_score =  0.5657253033911761


Cross_Val_Score =  0.7318534060495885


classification_report
              precision    recall  f1-score   support

           1       0.41      0.43      0.42      1217
           2       0.26      0.24      0.25       272
           3       0.32      0.35      0.34       669
           4       0.46      0.49      0.48      1809
           5       0.70      0.68      0.69      4850

    accuracy                           0.57      8817
   macro avg       0.43      0.44      0.43      8817
weighted avg       0.57      0.57      0.57      8817
```

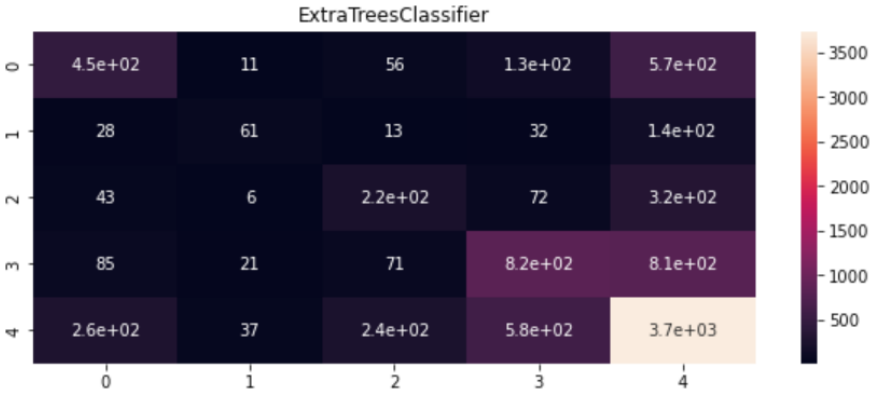
```
[[ 519   30   78  178  412]
 [  39   64   17   40  112]
 [  78   11  235  105  240]
 [ 174   36   99  889  611]
 [ 446  101  302  720 3281]]
```



9. Extra Trees Classifier

```
Accuracy_score =  0.6008846546444369


Cross_Val_Score =  0.7937021430319414


classification_report
              precision    recall  f1-score   support

           1       0.52      0.37      0.43      1217
           2       0.45      0.22      0.30       272
           3       0.37      0.33      0.35       669
           4       0.51      0.46      0.48      1809
           5       0.67      0.77      0.72      4850

    accuracy                           0.60      8817
   macro avg       0.50      0.43      0.46      8817
weighted avg       0.59      0.60      0.59      8817
```

```
[[ 451   11   56  127  572]
 [  28   61   13   32  138]
 [  43    6  223   72  325]
 [  85   21   71  825  807]
 [ 265   37  235  575 3738]]
```



MODEL PERFORMANCES

	Model	Accuracy_score	Cross_val_score
0	KNeighborsClassifier	40.285812	67.724964
1	DecisionTreeClassifier	54.599070	69.351449
2	XGBClassifier	58.500624	70.190516
3	RandomForestClassifier	60.508109	78.592616
4	AdaBoostClassifier	48.621980	50.901121
5	MultinomialNB	52.364750	60.769368
6	GradientBoostingClassifier	57.990246	60.203660
7	BaggingClassifier	56.572530	73.185341
8	ExtraTreesClassifier	60.088465	79.370214

The model chosen is of **Gradient Boosting Classifier**. It has the good accuracy score and least difference with the cross val score which shows it has less over fitting.

Hyper paramter tuning on the chosen model:

```
from sklearn.ensemble import GradientBoostingClassifier
GBC=GradientBoostingClassifier()
```

```
from sklearn.model_selection import GridSearchCV

parameters={'max_depth': [1, 3],
            'min_samples_leaf': [1, 3],
            'min_samples_split': [1, 3, 5],
            'n_estimators': [100,200]}
gbc=GradientBoostingClassifier()

gsv=GridSearchCV(gbc,parameters,cv=5,n_jobs=-1)
gsv.fit(x_train_SMOTE,y_train_SMOTE)
print(gsv.best_params_)
```

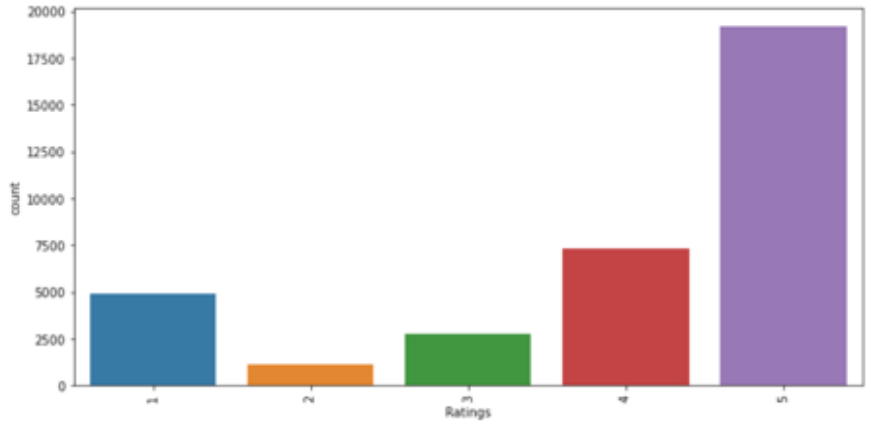
The scores of the model is:

```
#RandomForesetClassifier with best parameters

print('GradientBoostingClassifier')
rfc=GradientBoostingClassifier(max_depth=100, min_samples_leaf=3, min_samples_split=1, n_estimators=100
rfc.fit(x_train_ns,y_train_ns)
rfc.score(x_train_ns,y_train_ns)
predrfc=rfc.predict(x_test)
print(accuracy_score(y_test,predrfc))
print(confusion_matrix(y_test,predrfc))
print(classification_report(y_test,predrfc))
```

```
Accuracy_score =  0.5799024611545878


Cross_Val_Score =  0.602036595076892


classification_report
              precision    recall  f1-score   support

           1       0.42      0.43      0.43      1217
           2       0.12      0.27      0.16       272
           3       0.42      0.31      0.36       669
           4       0.63      0.36      0.46      1809
           5       0.67      0.75      0.71      4850

    accuracy                           0.58      8817
   macro avg       0.45      0.43      0.42      8817
weighted avg       0.59      0.58      0.58      8817
```

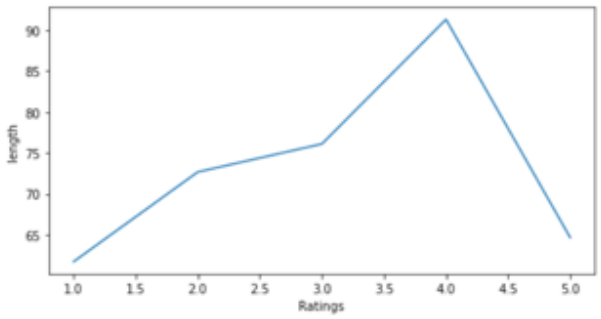
Visualizations

Ratings



The data is unbalanced with 5 rating data to be highest and 2 rating data to be the lowest

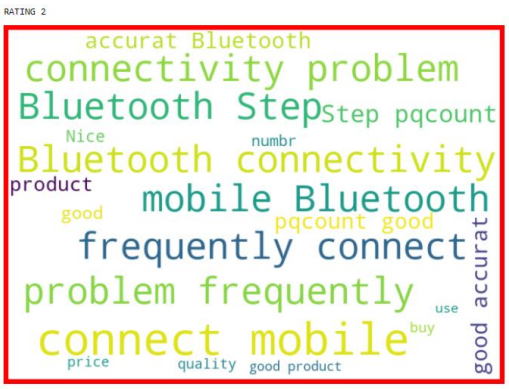
Length of Ratings



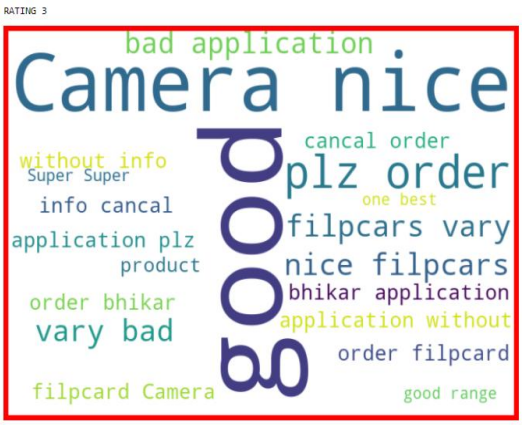
Length of reviews is highest for a rating of 4 and lowers for higher or lower than 4

WORD CLOUD OF EACH RATINGS

Largest number of word is good



Largest number of word is connect mobile

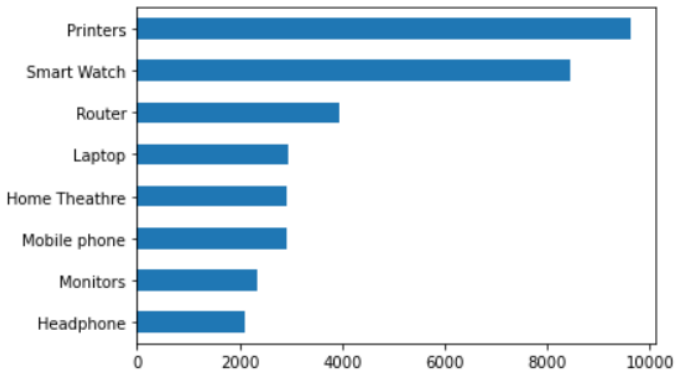


Largest number of word is good

Largest number of word is good

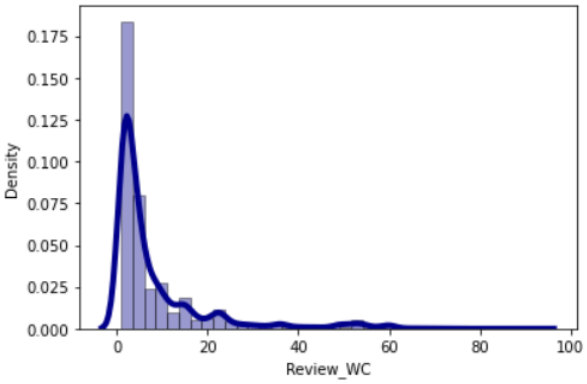
Largest number of word is good

Objects whose reviews are taken



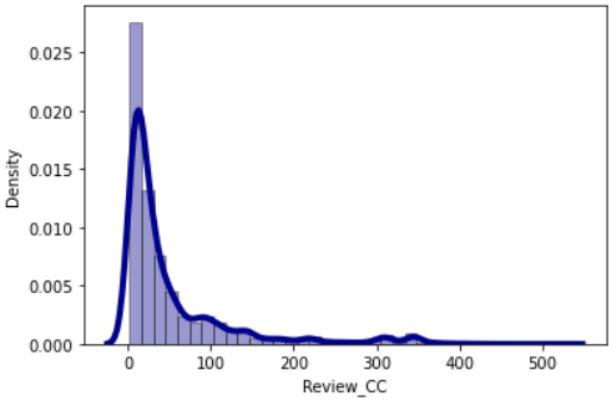
The highest reviews are of smart watch and printers

Word Count



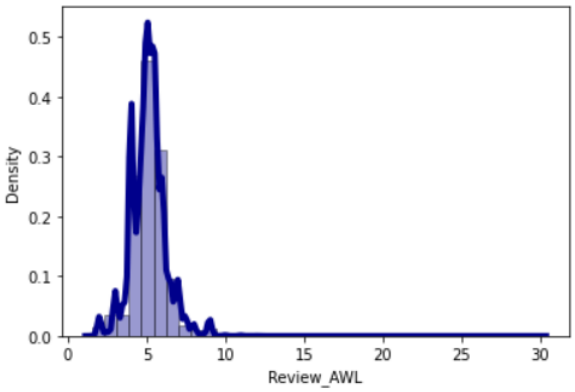
Words are highest in sentences upto 3 words

Charachter count

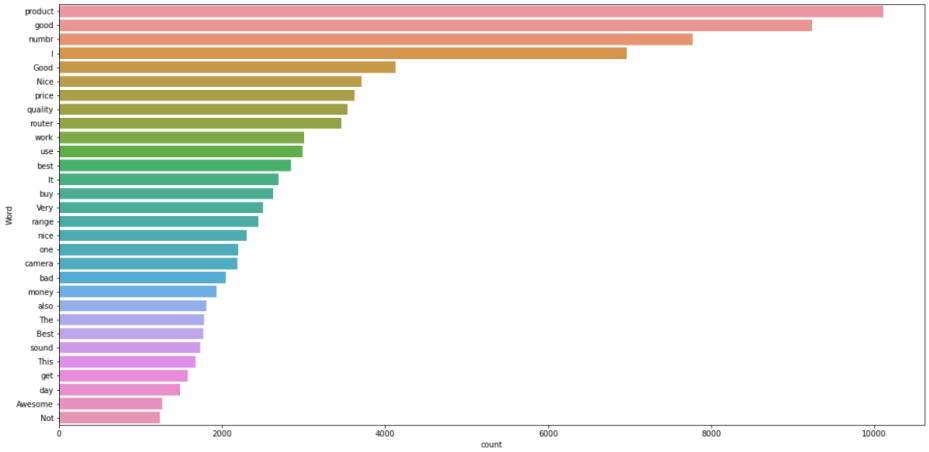


Highest Charachters seen in sentences are 14 Charachters

Average word length

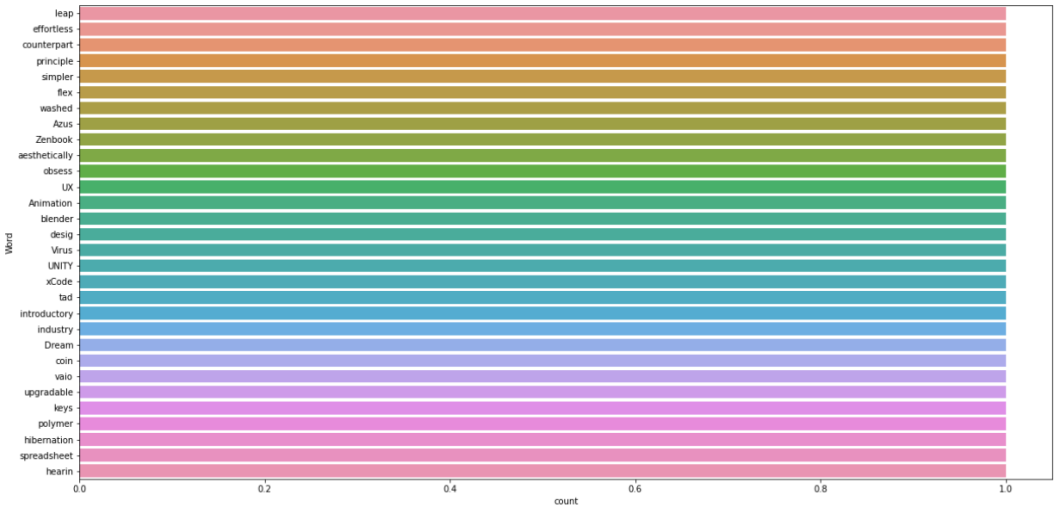


Frequency distribution of top 30words



Product word is maximum seen in the sentences

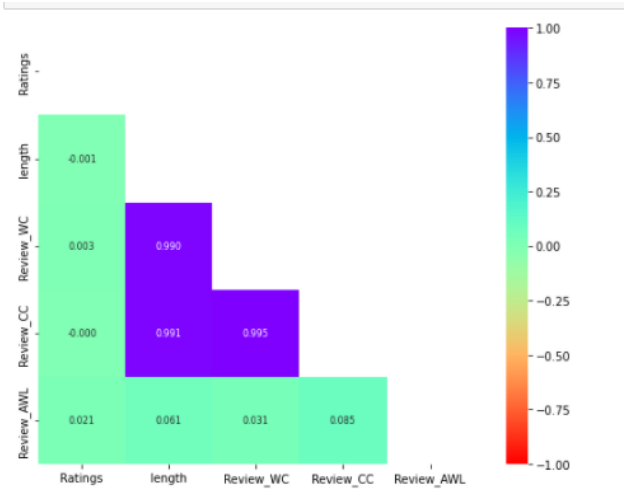
Frequency distribution of rarest 30words



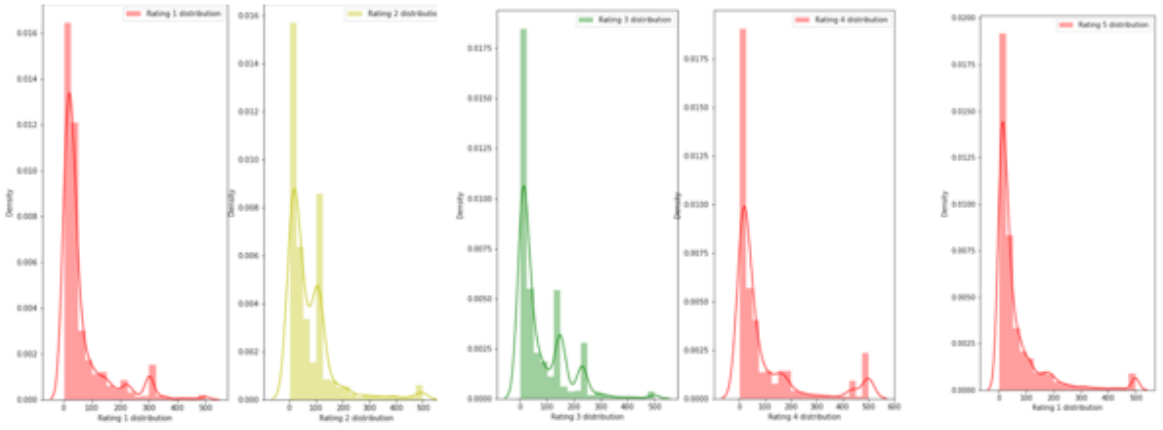
WordCloud for the Reviews taken

The highest word occurence is of product

HEAT MAP



Ratings distribution



Rating 1 Rating 2 Rating 3 Rating 4 Rating 5

Interpretation of the results:

1.The highest number of words are good for ratings 1,2,3 and 5.

It is seen that even for rating 1 good is sometimes given as rating

2.Data scrapped is for 35,367.The data shows a linearity and data is mostly normalized with less outliers in rating 1,2,3,4 and 5

3. The model will predict any review to be of rating 1,2,3,4 or 5

4.The most frequent word is Product and there are many rare words

CONCLUSION

Key Findings and Conclusions of the Study

The aim of the project was to predict the ratings of the textual body or analyse the sentiment behind each review of the product.

The purpose of the model developed is to use the model for predicting the rates of the company where no rating has been given for the previous comments and thus ratings could be predicted.

The Gradientbooster performed the best with the best accuracy and least difference with the cross validation score, showing that the least overfitting was there in it.

Best parameters are given into the XGB model and the model is built and saved as a file

Learning Outcomes of the Study in respect of Data Science

Various Algorithms were used to train the data such as KNeighborsClassifier, RandomForestClassifier , GradientBoostingClassifier , DecisionTreeClassifier , AdaBoostClassifier , BaggingClassifier , XGBClassifier , MultinomialNB and ExtraTreesClassifier

Natural language processing is one of the key learning areas in the field of Data science. The project enabled me in diving more into the world of NLP

Limitations of this work and Scope for Future Work

The limitation of the project Is that the dataset is skewed

The future scope of work is performing Keras deep learning technique and also RNN LSTM for predicting the Reviews.

****