Sentiment Analysis on Hotel Reviews

Siva Teja Segu , Pavansai Pottimuthi, Ajith Reddy Busipally, Sreehari Revuri
Kent State University

I. Introduction

Sentiment analysis plays a key role in the field of Natural Language Processing, and it is a technique that extract emotions from the raw texts. Most of the E-Commerce sites like Amazon, Flipkart and Google etc., has wide range of applications built on this. Say, for example think about google translator application. It performs brilliantly in understanding, analyzing and translating the data. Also, it is effectively used on social media post and customer reviews in order to know the opinion of the customers whether they are happy or not with product, service and other factors which will play key role in enhancing their businesses.

A. Background

Sentiment analysis, commonly referred to as opinion mining, is a process that uses machine learning, statistical methods, and natural language processing to recognize and extract subjective information from textual data, such as views, attitudes, and emotions. To ascertain the general attitude or sentiment that is being represented in a text is the main objective of sentiment analysis.

A hotel review is a written assessment of a guest's stay at a hotel that is often published on an internet platform such as a travel website or social media. Hotel evaluations may include details about a guest's experience with the rooms, staff, and facilities, as well as their overall happiness with their stay.

B. Problem Statement

The objective of the project to perform sentimental analysis on a hotel review dataset. Given a review by its customers, we need to predict whether the review is good or bad review in other words say it is positive or negative. For each textual review, we want to predict if it corresponds to a good review (the customer is happy) or to a bad one (the customer is not satisfied).

C. Why is it important

As it enables organizations to analyse massive volumes of unstructured data effectively and economically, sentiment analysis is proven to be a useful tool. As a way to divide reviews, it is becoming more and more popular. It is easy to use and can be occasionally simply adjusted. It gives facts and quantifiable data for upcoming decision-making, and when done well, it delivers value to a business. Sentiment research should be used by businesses that want to improve their goods and services, increase sales, and outwit their rivals.

D. Plans to implement

To swiftly determine whether a review is good or negative, sentiment analysis is required. This paper offers a solution by utilizing the Random Forest Classifier, Word2Vec approach to categorize positive and negative opinion reviews, and by comparing models using preprocessing, feature extraction, and feature selection. Due to their communities' stronger inclination toward data science, such as NLP and Deep Learning for Sentiment Analysis, open-source libraries in programming languages like Python and Java are particularly well suited to creating specialized Sentiment Analysis solutions. But, this demanded a lot of time, money up front, and resources.

II. Literature Review

One of the main tools that a machine may use to comprehend human psychology is sentiment analysis. In order to be used in domains where people were previously required to identify mood or emotion, this technology is currently the subject of substantial research. It plays a key role in chat bot assistants, and when paired with speech recognition technology, it may also be used to replace people in contact centers.

For NLP, machine learning methods were introduced. Several of the systems created during this time period employed machine learning methods such as decision trees to build systems of hard if-then rules comparable to existing hand-written rules. In NLP, the hidden Markov models employed part-of-speech tagging, and certain statistical models that make probabilistic judgments were developed. For sentiment categorization, there are five different machine learning classifiers: Nave Bayes, K-Nearest Neighbor, Support Vector Machine, Logistic Regression, and Random Forest.[1]

Nandal in this paper classified amazon product reviews for sentiment analysis using SVM(Support Vector Machine) Tool. The study examined how words can shift in meaning depending on the context in which they're used, and how this impacts the overall evaluation of a product and its specific features.[2].

Humera Shaziya classified movie reviews for sentiment analysis using WEKA Tool. They enhanced the earlier work done in sentiment categorization which analyzes opinions which express either positive or negative sentiment [3].

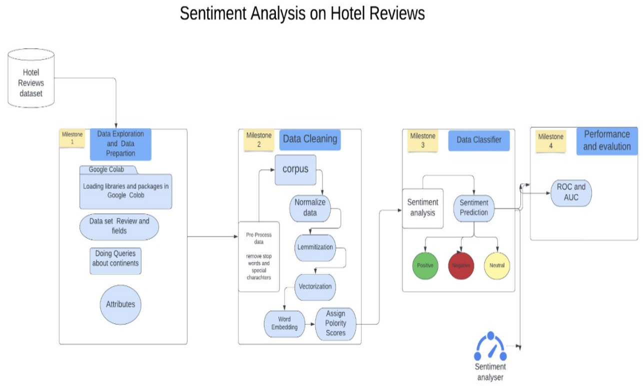
Ahmad Kamal designed an opinion mining framework that facilitates objectivity or subjectivity analysis, feature extraction and review summarizing.He used supervised machine learning approach for subjectivity and objectivity classification of reviews. The various techniques used by him were Naive Bayes, Decision Tree, Multi layer Perception and Bagging. He also improved mining performance by preventing irrelevant extraction and noise.[4].

Orestes Appel used natural language processing (NLP)essential techniques, a sentiment lexicon enhanced with the assistance of SentiWordNet,and fuzzy sets to estimate the semantic orientation polarity and its intensity for sentences, which provides a foundation for computing with sentiments. The proposed hybrid method is applied to three different data-sets and the results achieved are compared to those obtained using Naive Bayes and Maximum Entropy techniques[5].

III. Proposed Model

The research began with an examination of many research and review articles on sentiment analysis, and each publication's summary was prepared by reading and comprehending the document. Examine popular classification techniques such as Nave Bayes, Random Forest, k-nearest neighbor, Decision Tree Induction, and Support Vector Machine.

The information was obtained from Booking.com. This dataset includes 515,000 customer reviews and ratings for 1493 premium hotels throughout Europe. In the meanwhile, the geographical location of hotels is supplied for additional examination.



A. *Data preparation and data exploration*

Pandas describe method will give some statistical parameters of the data set like count, mean and standard deviation.

B. *Data cleaning is the fundamental step in any NLP techniques*

- Data Cleaning and preprocessing steps.
- Using gensim module to convert the text to vectors using DOC2VEC function
- By using SentimentIntensityAnalyzer we find the polarity scores and plot word cloud.

C. *Classification*

We are using Random forest and Transformers Using RoBERTa

D. *Performance Evaluations*

A receiver operating characteristic (ROC) and area under the curve (AUC). Precision Recall curve(PR) we are using for performance and evaluation.

IV. Key Concepts

A. *Overview of Dataset*

As we have taken dataset from booking.com. The CSV file contains 17 fields. The description of each field is as below

- Hotel_Address: Address of hotel.
- Review_Date: Date when reviewer posted the corresponding review.
- Average_Score: Average Score of the hotel, calculated based on the latest comment in the last year.
- Hotel_Name: Name of Hotel
- Reviewer_Nationality: Nationality of Reviewer
- Negative_Review: Negative Review the reviewer gave to the hotel. If the reviewer does not give the negative review, then it should be: 'No Negative'
- ReviewTotalNegativeWordCounts: Total number of words in the negative review.
- Positive_Review: Positive Review the reviewer gave to the hotel. If the reviewer does not give the negative review, then it should be: 'No Positive'
- ReviewTotalPositiveWordCounts: Total number of words in the positive review.
- Reviewer_Score: Score the reviewer has given to the hotel, based on his/her experience
- TotalNumberofReviewsReviewerHasGiven: Number of Reviews the reviewers has given in the past.
- TotalNumberof_Reviews: Total number of valid reviews the hotel has.
- Tags: Tags reviewer gave the hotel.
- dayssincereview: Duration between the review date and scrape date.
- AdditionalNumberof_Scoring: There are also some guests who just made a scoring on the service rather than a review. This number indicates how many valid scores without review in there.
- lat: Latitude of the hotel
- lng: longitude of the hotel

B. *Data Cleaning*

In Data Cleaning we follow set of processes like Normalizing the data, Stemming and lemmatization, Word2Vec, Wordembeding, and Vectorization.

V. Libraries

A. *NLTK*

The popular open-source library Natural Language Toolkit (nltk) makes working with human language data in Python straightforward. It provides a wide range of NLP methods, including as sentiment analysis, part-of-speech tagging, tokenization, stemming, lemmatization, and named entity recognition, among others.

B. *SentimentIntensityAnalyzer*

It is a rule-based sentiment analyzer, and depending on their semantic orientation, sentences are frequently labeled as either positive or negative. Lastly, we use the polarity scores technique to determine the emotion.

C. *Gensim*

The gensim library is an open-source Python library for topic modeling and similarity detection in large and complex text datasets.The Gensim algorithms Word2Vec, FastText, Latent Semantic Indexing where it automatically recognize the semantic structure of documents by examining statistical occurrences patterns within a corpus of training texts. There is no need for human input because these algorithms are unsupervised.

D. Google Colab Software

Google Colab is a free to use Jupyter notebook environment where we can create and run Python code in a web browser. By making GPU-accelerated computing resources and pre-installed machine learning libraries accessible, it is made to encourage collaborative work and study.It is an effective tool for projects involving data science and machine learning, especially those that call for access to huge datasets and substantial processing capacity.

VI. Infrastructure

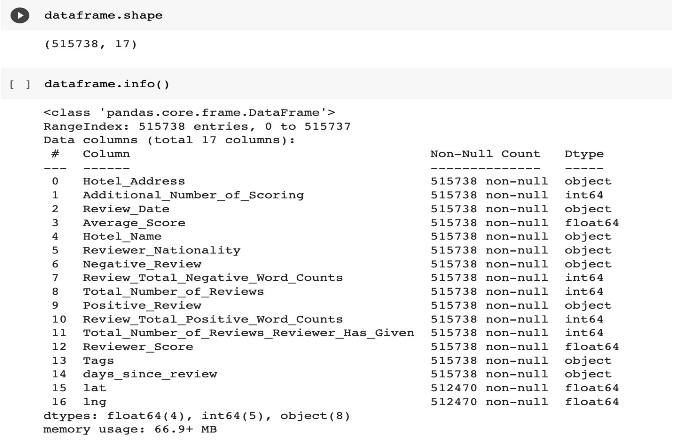
A. GPU machine

we are using GPU machines which are available from a variety of cloud computing providers, such as Amazon Web Services, Microsoft Azure, and Google Cloud. By using GPU machine we can greatly accelerate the training and inference of machine learning models, reducing the time required to train and test models.It is expensive for long term project.

VII. Research Methods

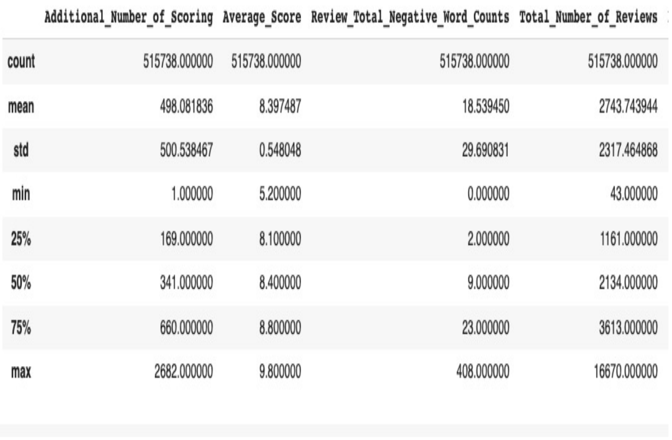
A. Data Preparation and Data Exploration

We have used Google Colob space for our project, in first step we have to give path of our dataset so that it will read and then we can do explotary data analysis. To get a quick overview of the data set we use the dataframe.info() function.Python is an excellent language for data analysis, due to the solid ecosystem of data-centric Python tools.



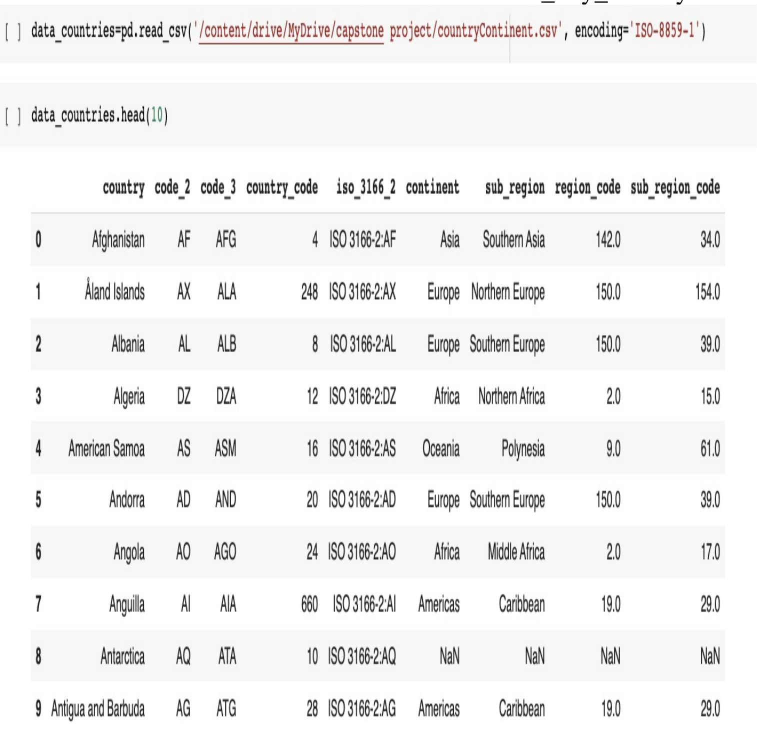
```
dataframe.shape

(515738, 17)

[ ] dataframe.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 515738 entries, 0 to 515737
Data columns (total 17 columns):
 #   Column                                      Non-Null Count   Dtype
---  ------                                      --------------   -----
 0   Hotel_Address                               515738 non-null  object
 1   Additional_Number_of_Scoring                515738 non-null  int64
 2   Review_Date                                 515738 non-null  object
 3   Average_Score                               515738 non-null  float64
 4   Hotel_Name                                  515738 non-null  object
 5   Reviewer_Nationality                        515738 non-null  object
 6   Negative_Review                             515738 non-null  object
 7   Review_Total_Negative_Word_Counts           515738 non-null  int64
 8   Total_Number_of_Reviews                     515738 non-null  int64
 9   Positive_Review                             515738 non-null  object
 10  Review_Total_Positive_Word_Counts           515738 non-null  int64
 11  Total_Number_of_Reviews_Reviewer_Has_Given  515738 non-null  int64
 12  Reviewer_Score                              515738 non-null  float64
 13  Tags                                        515738 non-null  object
 14  days_since_review                           515738 non-null  object
 15  lat                                         512470 non-null  float64
 16  lng                                         512470 non-null  float64
dtypes: float64(4), int64(5), object(8)
memory usage: 66.9+ MB
```

Pandas method will give us some of the statistical parameters of the dataset like count, mean and standard deviation.



	Additional_Number_of_Scoring	Average_Score	Review_Total_Negative_Word_Counts	Total_Number_of_Reviews
count	515738.000000	515738.000000	515738.000000	515738.000000
mean	498.081836	8.397487	18.539450	2743.743944
std	500.538467	0.548048	29.690831	2317.464868
min	1.000000	5.200000	0.000000	43.000000
25%	169.000000	8.100000	2.000000	1161.000000
50%	341.000000	8.400000	9.000000	2134.000000
75%	660.000000	8.800000	23.000000	3613.000000
max	2682.000000	9.800000	408.000000	16670.000000

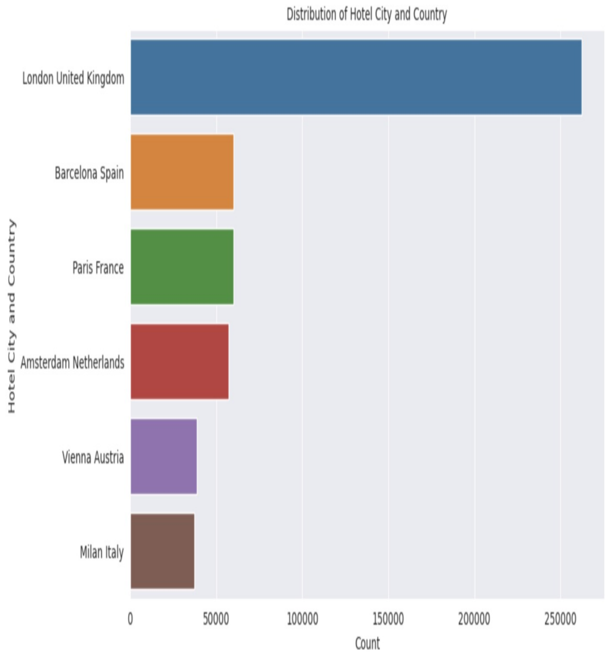
After exploring the dataset there is only Reviewer nationality data in the dataset, so in order to find from which city, country and continent customers visited the hotels.Using countrycity dataset we added new coloumn to the dataset 'Hotel_city_country'.



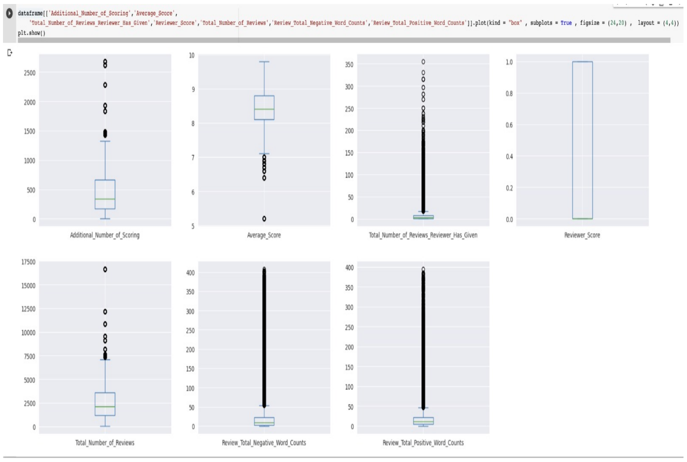
```
[ ] data_countries=pd.read_csv('/content/drive/MyDrive/capstone project/countryContinent.csv', encoding='ISO-8859-1')

[ ] data_countries.head(10)
```

	country	code_2	code_3	country_code	iso_3166_2	continent	sub_region	region_code	sub_region_code
0	Afghanistan	AF	AFG	4	ISO 3166-2:AF	Asia	Southern Asia	142.0	34.0
1	Åland Islands	AX	ALA	248	ISO 3166-2:AX	Europe	Northern Europe	150.0	154.0
2	Albania	AL	ALB	8	ISO 3166-2:AL	Europe	Southern Europe	150.0	39.0
3	Algeria	DZ	DZA	12	ISO 3166-2:DZ	Africa	Northern Africa	2.0	15.0
4	American Samoa	AS	ASM	16	ISO 3166-2:AS	Oceania	Polynesia	9.0	61.0
5	Andorra	AD	AND	20	ISO 3166-2:AD	Europe	Southern Europe	150.0	39.0
6	Angola	AO	AGO	24	ISO 3166-2:AO	Africa	Middle Africa	2.0	17.0
7	Anguilla	AI	AIA	660	ISO 3166-2:AI	Americas	Caribbean	19.0	29.0
8	Antarctica	AQ	ATA	10	ISO 3166-2:AQ	NaN	NaN	NaN	NaN
9	Antigua and Barbuda	AG	ATG	28	ISO 3166-2:AG	Americas	Caribbean	19.0	29.0

In the below visualization we can see distribution hotel city country, customers visiting hotels are highest from United Kingdom and least is from Milan Italy.

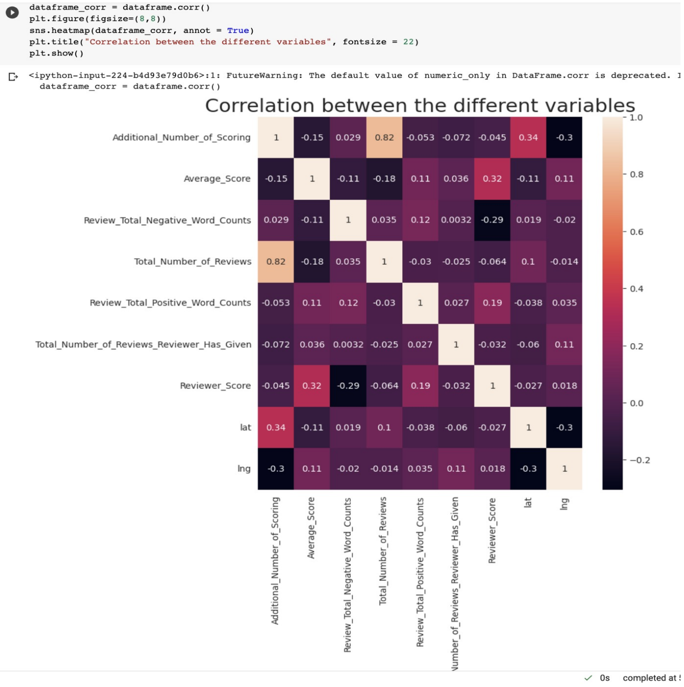


In order to see if there is any outliers in the dataset we are using Boxplot will be useful one, after plotting there are outliers in our dataset in some of the coloumns there are outliers those should be cleaned or manipulated to bring under standard normal distribution.



```
dataframe[['Additional_Number_of_Scoring','Average_Score',
    'Total_Number_of_Reviews_Reviewer_Has_Given','Reviewer_Score','Total_Number_of_Reviews','Review_Total_Negative_Word_Counts','Review_Total_Positive_Word_Counts']].plot(kind = "box" , subplots = True , figsize = (24,20) ,  layout = (4,4))
plt.show()
```

To find correlation among other variables in the dataset we plotted correlation using heatmaps. The further away the correlation coefficient is from zero, the stronger the relationship between the two variables.



```
dataframe_corr = dataframe.corr()
plt.figure(figsize=(8,8))
sns.heatmap(dataframe_corr, annot = True)
plt.title("Correlation between the different variables", fontsize = 22)
plt.show()
```

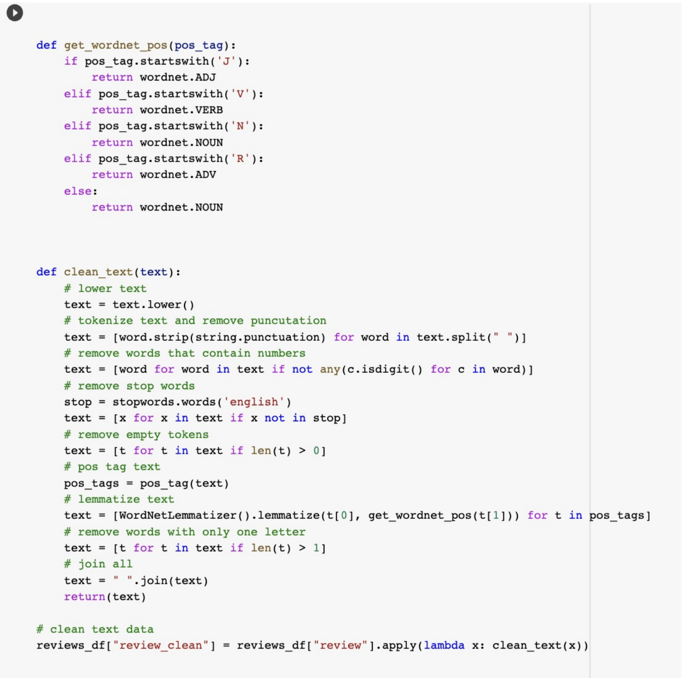
```
<ipython-input-224-b4d93e79d0b6>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. I
  dataframe_corr = dataframe.corr()
```

Correlation between the different variables
Additional_Number_of_Scoring 1 -0.15 0.029 0.82 -0.053 -0.072 -0.045 0.34 -0.3
Average_Score -0.15 1 -0.11 -0.18 0.11 0.036 0.32 -0.11 0.11
Review_Total_Negative_Word_Counts 0.029 -0.11 1 0.035 0.12 0.0032 -0.29 0.019 -0.02
Total_Number_of_Reviews 0.82 -0.18 0.035 1 -0.03 -0.025 -0.064 0.1 -0.014
Review_Total_Positive_Word_Counts -0.053 0.11 0.12 -0.03 1 0.027 0.19 -0.038 0.035
Total_Number_of_Reviews_Reviewer_Has_Given -0.072 0.036 0.0032 -0.025 0.027 1 -0.032 -0.06 0.11
Reviewer_Score -0.045 0.32 -0.29 -0.064 0.19 -0.032 1 -0.027 0.018
lat 0.34 -0.11 0.019 0.1 -0.038 -0.06 -0.027 1 -0.3
lng -0.3 0.11 -0.02 -0.014 0.035 0.11 0.018 -0.3 1
Additional_Number_of_Scoring
Average_Score
Review_Total_Negative_Word_Counts
Total_Number_of_Reviews
Review_Total_Positive_Word_Counts
Number_of_Reviews_Reviewer_Has_Given
Reviewer_Score
lat
lng
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
✓ 0s completed at !

B. Data Cleaning

After exploring dataset the next step is Data Cleaning. In this step data is cleaned in many steps.Data cleaning is the fundamental step in any NLP techniques. The below code represents that we are doing the following processes

- Convert the sentences to lower case letter as a step for preprocessing.
- Remove special symbols like “”, !,@,#,$ in our data.
- Remove stop words and remove tokens like tags.
- Tokenization, lemmatization.
- Remove no negatives and no positives from data.
- Clean textual data using parts of speech.



```
def get_wordnet_pos(pos_tag):
    if pos_tag.startswith('J'):
        return wordnet.ADJ
    elif pos_tag.startswith('V'):
        return wordnet.VERB
    elif pos_tag.startswith('N'):
        return wordnet.NOUN
    elif pos_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN



def clean_text(text):
    # lower text
    text = text.lower()
    # tokenize text and remove puncutation
    text = [word.strip(string.punctuation) for word in text.split(" ")]
    # remove words that contain numbers
    text = [word for word in text if not any(c.isdigit() for c in word)]
    # remove stop words
    stop = stopwords.words('english')
    text = [x for x in text if x not in stop]
    # remove empty tokens
    text = [t for t in text if len(t) > 0]
    # pos tag text
    pos_tags = pos_tag(text)
    # lemmatize text
    text = [WordNetLemmatizer().lemmatize(t[0], get_wordnet_pos(t[1])) for t in pos_tags]
    # remove words with only one letter
    text = [t for t in text if len(t) > 1]
    # join all
    text = " ".join(text)
    return(text)

# clean text data
reviews_df["review_clean"] = reviews_df["review"].apply(lambda x: clean_text(x))
```

1) Bag Of Words (BOW): Raw text is initially reprocessed and after processing and cleaning techniques.

2) Cleaned or preprocessed : Remove all unnecessary special characters, if there are words of other accent like polish, German, Spanish etc. Remove or replace them or add the right Unicode to make them readable for machine.

3) Normalize all Data: Make the data properly in a single case letter, either upper or lower. Preferred lower using .lower() function.

4) Stemming and lemmatization: In this methods used by search engines and chat bots to analyze the meaning behind a word. Stemming uses the stem of the word, while lemmatization uses the context in which the word is being used.

5) Term Frequency and Inverse Dense Frequency(TF-IDF): Inverse document frequency looks at how common (or uncommon) a word is amongst the document.

IDF =Log [(# Number of documents) / (Number of documents containing the word)]

Term frequency works by looking at the frequency of a particular term you are concerned with relative to the document.

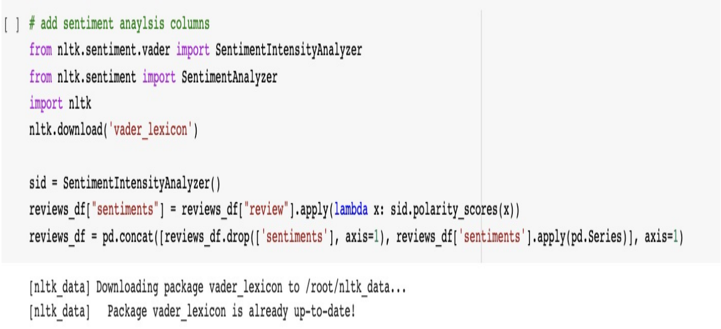
TF = (Number of repetitions of word in a document) / (# of words in document).

For combined TF – IDF = TF(t, d) * IDF(t)

So, using TF and IDF machine makes sense of important words in a document and important words throughout all documents.

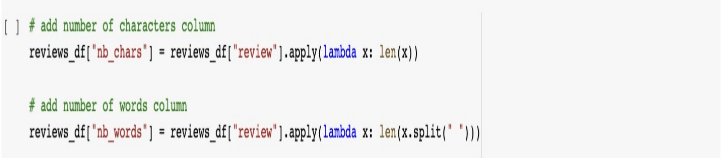
6) Word2vec: It is a combination of models where it represents distributed of words in a corpus C. Here, it is an algorithm which accepts input corpus and outputs as vector representation for each word.Word embeddings are numerical representations of words that capture their semantic meaning, used in various NLP tasks such as text classification, sentiment analysis, and machine translation.Word2Vec creates a dense vector representation for each word, allowing us to explore semantic relationships between words.

7) *SentimentIntensityAnalyzer:*

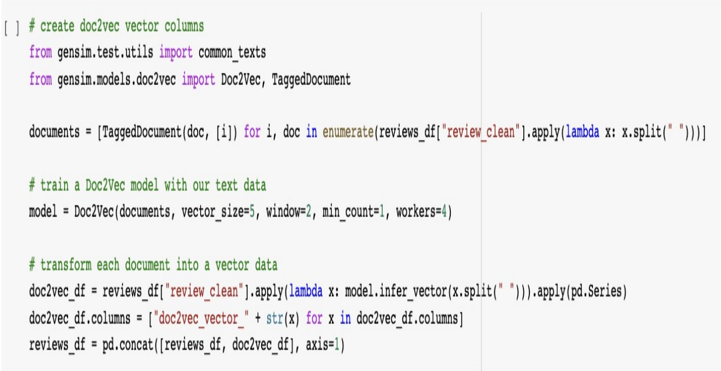


```
[ ] # add sentiment anaylsis columns
    from nltk.sentiment.vader import SentimentIntensityAnalyzer
    from nltk.sentiment import SentimentAnalyzer
    import nltk
    nltk.download('vader_lexicon')

    sid = SentimentIntensityAnalyzer()
    reviews_df["sentiments"] = reviews_df["review"].apply(lambda x: sid.polarity_scores(x))
    reviews_df = pd.concat([reviews_df.drop(['sentiments'], axis=1), reviews_df['sentiments'].apply(pd.Series)], axis=1)
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```



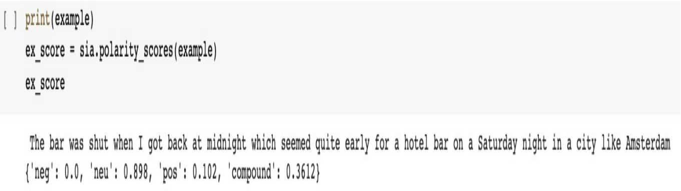
```
[ ] # add number of characters column
    reviews_df["nb_chars"] = reviews_df["review"].apply(lambda x: len(x))

    # add number of words column
    reviews_df["nb_words"] = reviews_df["review"].apply(lambda x: len(x.split(" ")))
```



```
[ ] # create doc2vec vector columns
    from gensim.test.utils import common_texts
    from gensim.models.doc2vec import Doc2Vec, TaggedDocument

    documents = [TaggedDocument(doc, [i]) for i, doc in enumerate(reviews_df["review_clean"].apply(lambda x: x.split(" ")))]

    # train a Doc2Vec model with our text data
    model = Doc2Vec(documents, vector_size=5, window=2, min_count=1, workers=4)

    # transform each document into a vector data
    doc2vec_df = reviews_df["review_clean"].apply(lambda x: model.infer_vector(x.split(" "))).apply(pd.Series)
    doc2vec_df.columns = ["doc2vec_vector_" + str(x) for x in doc2vec_df.columns]
    reviews_df = pd.concat([reviews_df, doc2vec_df], axis=1)
```

Here, after finishing all the cleaning processes the next step is to assign polarity scores to the reviews in the dataset. Below is the code for SentimentIntensityAnalyzer.A sentiment intensity analyzer assigns a numerical value to the sentiment expressed to a text, typically on a scale from -1 to 1. It uses NLP and machine learning algorithms to recognize patterns and identify sentiment.Sentiment intensity analyzers can be used to monitor customer feedback and product reviews, providing insights into the sentiment of a large volume of text data to inform business decisions and improve customer satisfaction.

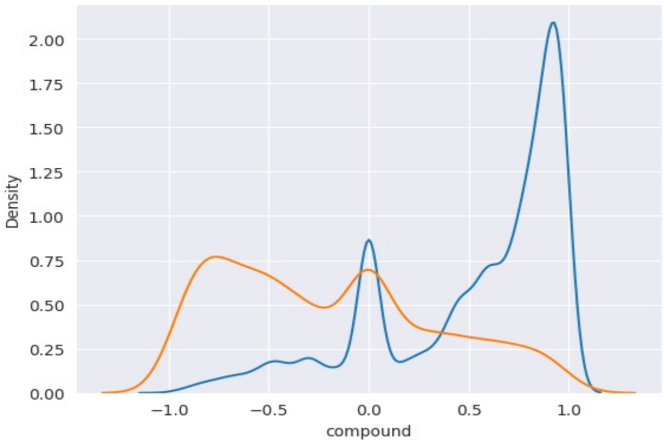
For example in below image we can see sentiment polarity scores for the review in the text is ”The bar was shut when I got back at midnight which seemed quite early for a hotel bar on a Saturday night in a city like Amsterdam”. {’neg’: 0.0, ’neu’: 0.898, ’pos’: 0.102, ’compound’: 0.3612}.



```
[ ] print(example)
    ex_score = sia.polarity_scores(example)
    ex_score
```

```
The bar was shut when I got back at midnight which seemed quite early for a hotel bar on a Saturday night in a city like Amsterdam
{'neg': 0.0, 'neu': 0.898, 'pos': 0.102, 'compound': 0.3612}
```

we can plot sentiment distribution curve using sentimental scores which we got, 1 reprsents good review and 0 as bad review.



C. *Data Classification*

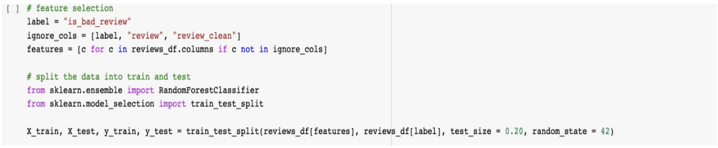
In our method we have used two methods to classify the data they are Random Forest Classifier and RoBERTa using Transformers.

1) Random Forest Classifier: It is s popular algorithm for classification and regression issues is the supervised machine learning technique known as random forest. It creates decision trees from several samples, using the majority vote for classification and the average for regression. The ”forest” it creates is an ensemble of decision trees, often trained using the ”bagging” approach.

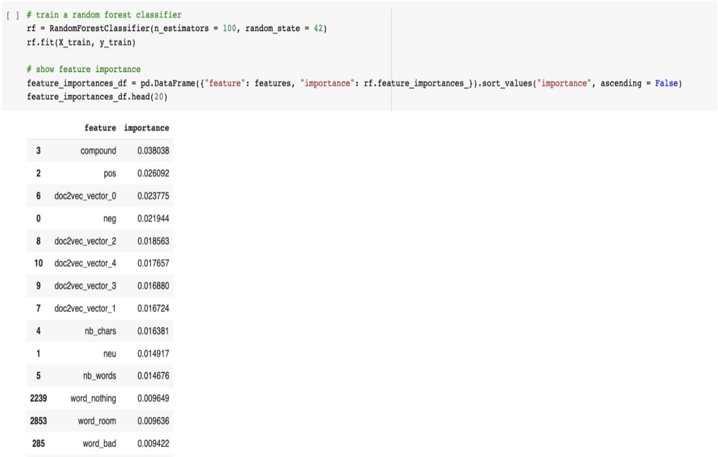
Bagging: The final result is based on majority vote and a separate training subset is created using replacement from a sample of the training data.

In Random Forest we have these advantages such as Diversity, Parallelization, Train-Test Split, and Stability are important features of a random forest, which reduces feature space, uses CPU, and is based on majority voting.

Steps involved in training our dataset are we have imported necessary libraries from Sklearn ensemble Random Forest classifier and model select to train split our dataset features selected.



```
[ ] # feature selection
    label = "is_bad_review"
    ignore_cols = [label, "review", "review_clean"]
    features = [c for c in reviews_df.columns if c not in ignore_cols]

    # split the data into train and test
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.model_selection import train_test_split

    X_train, X_test, y_train, y_test = train_test_split(reviews_df[features], reviews_df[label], test_size = 0.20, random_state = 42)
```

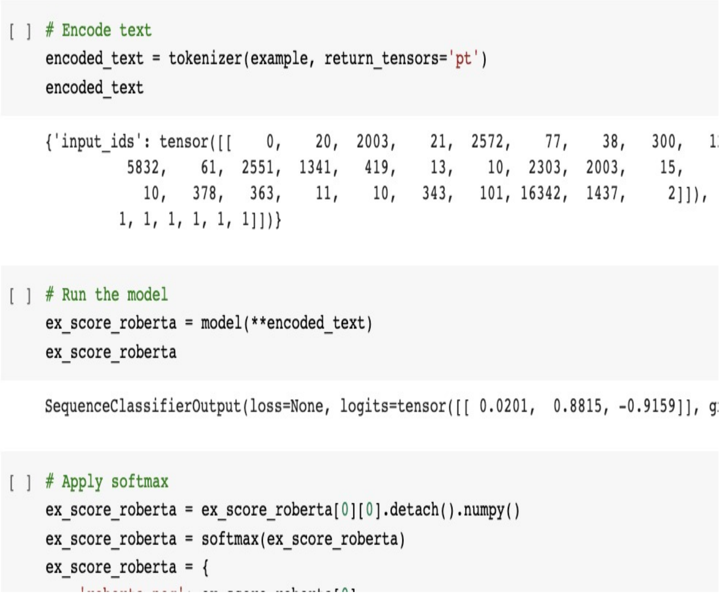


```
[ ] # train a random forest classifier
    rf = RandomForestClassifier(n_estimators = 100, random_state = 42)
    rf.fit(X_train, y_train)

    # show feature importance
    feature_importances_df = pd.DataFrame({"feature": features, "importance": rf.feature_importances_}).sort_values("importance", ascending = False)
    feature_importances_df.head(20)
```

	feature	importance
3	compound	0.038038
2	pos	0.026092
6	doc2vec_vector_0	0.023775
0	neg	0.021944
8	doc2vec_vector_2	0.018563
10	doc2vec_vector_4	0.017657
9	doc2vec_vector_3	0.016880
7	doc2vec_vector_1	0.016724
4	nb_chars	0.016381
1	neu	0.014917
5	nb_words	0.014676
2239	word_nothing	0.009649
2853	word_room	0.009636
285	word_bad	0.009422

We have used hyperparameters like n_estimators and random-state and trained the dataset with different set of values.

2) RoBERTa using Transformers: A variation of BERT that changes the pretraining procedure is called RoBERTa. The modifications includes for training models, longer, larger batch sizes. Training on longer sequences, deleting the next sentence prediction aim, adjusting the masking pattern dynamically applied to the training data. By dynamically changing the masking pattern applied to the training data and removing the objective training for next sentence prediction on longer sequences.
Here, we have used pretrained cardiffnlp/twitter-robberta analysis for training our dataset.
Steps involved in roberta process are installing necessary libraries which are required for our model. They are AutoTokenizer, AutoModelForSequenceClassification, and softmax. Here we have taken a subset of 1000 reviews which are there in the dataset. Next step is to encode the text with tokenizer, using pretrained model we running the dataset and then applying the softmax in order to get polarity scores.

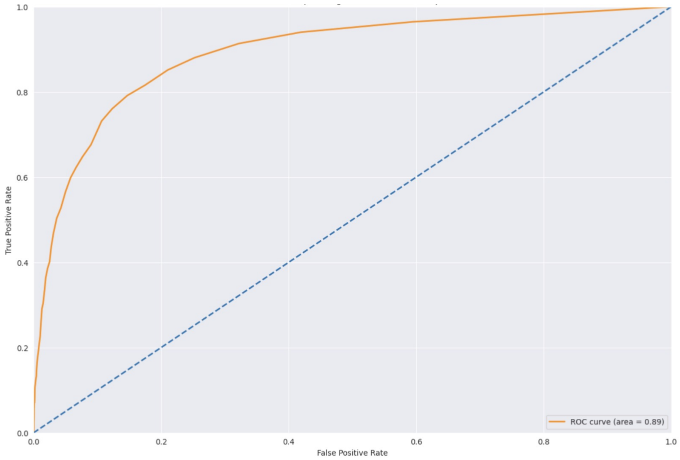


```
[ ] # Encode text
    encoded_text = tokenizer(example, return_tensors='pt')
    encoded_text

    {'input_ids': tensor([[    0,    20,  2003,    21,  2572,    77,    38,   300,   1
             5832,    61,  2551,  1341,   419,    13,    10,  2303,  2003,    15,
               10,   378,   363,    11,    10,   343,   101, 16342,  1437,     2]]),
            1, 1, 1, 1, 1, 1]])}

[ ] # Run the model
    ex_score_roberta = model(**encoded_text)
    ex_score_roberta

    SequenceClassifierOutput(loss=None, logits=tensor([[ 0.0201,  0.8815, -0.9159]], g

[ ] # Apply softmax
    ex_score_roberta = ex_score_roberta[0][0].detach().numpy()
    ex_score_roberta = softmax(ex_score_roberta)
    ex_score_roberta = {
```

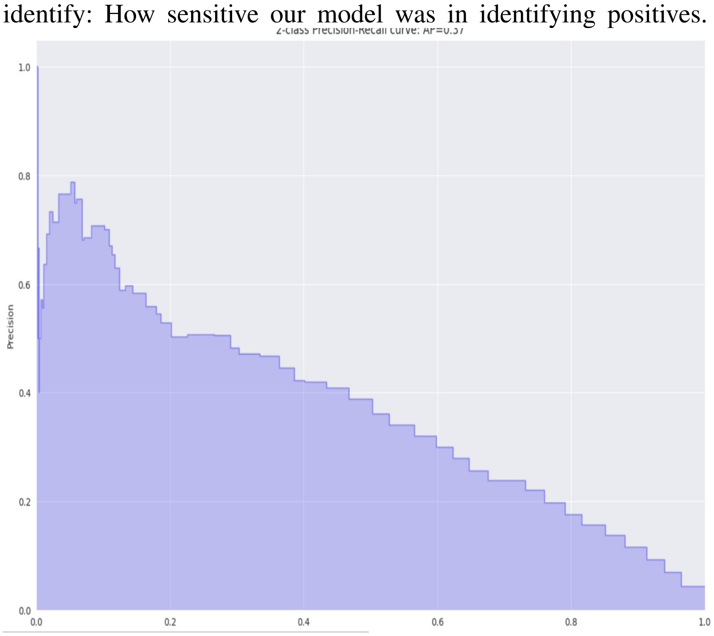
D. Performance and Evaluation

1) Random Forest Model: A receiver operating characteristic (ROC) curve shows how well a model can categorize binary outcomes. An ROC curve is created by displaying a model's false positive rate vs its true positive rate for each feasible cutoff value. The area under the curve (AUC) is frequently measured and used as a statistic to demonstrate how well a model can identify data points.

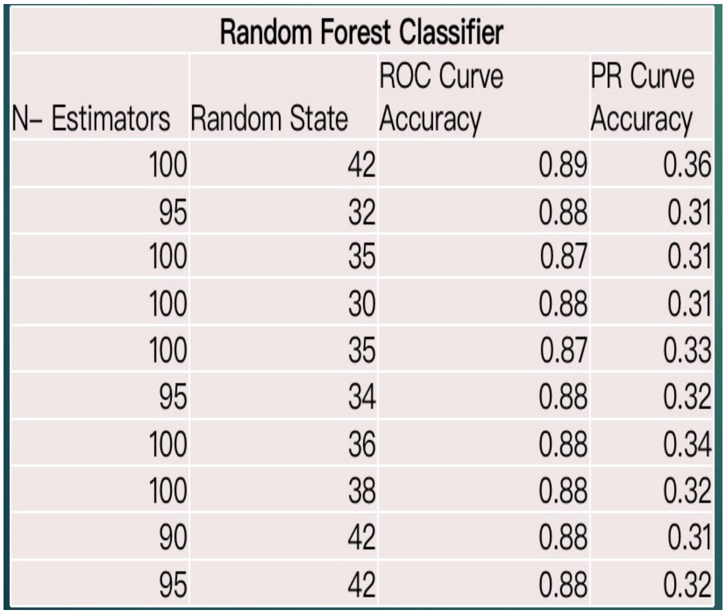
The ROC (Receiver Operating Characteristic) curve is frequently employed to summarize the quality of our classifier. The better predictions are, the higher the curve is above the diagonal baseline.
Although the AUC ROC (Area Under the Curve ROC) is excellent, we should not use it to evaluate the quality of our model here. Because our dataset is imbalanced, the Negatives corresponds to our number of positive reviews, which is very high.
This means that even if there are some instances of false positives, our FPR will remain very low. Our model will be able to forecast many false positives while maintaining a low false positive rate, while raising the genuine positive rate and therefore artificially improving the accuracy.



In order to calculate the PR Curve we have here precession and recall. Precision (also called positive predictive value) is defined as the number of true positives divided by the total number of positive predictions. Hence, precision quantifies what percentage of the positive predictions were correct: How correct our model's positive predictions.Recall (also called sensitivity) is defined as the number of true positives divided by the total number of true positives and false negatives (i.e. all actual positives). Hence, recall quantifies what percentage of the actual positives you were able to identify: How sensitive our model was in identifying positives.



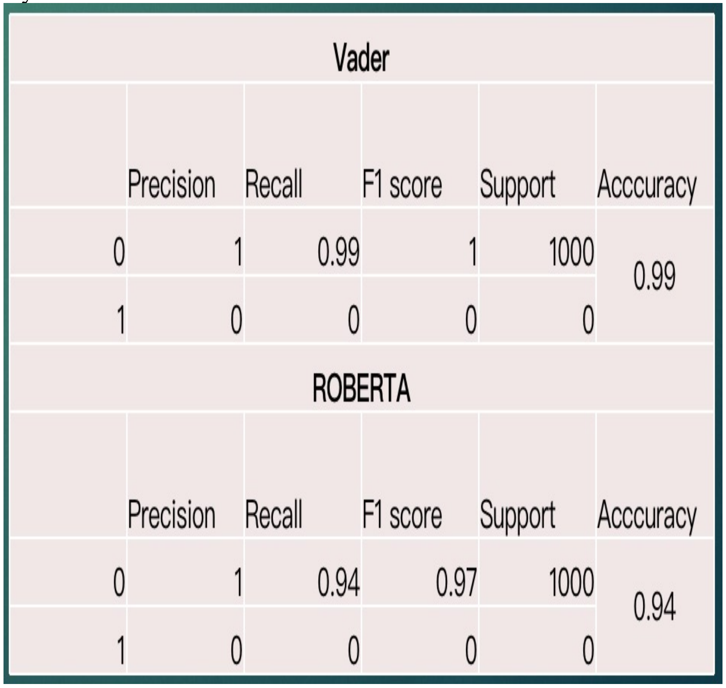
After experimenting with various N-Estimators and Random State values, the Random Forest Classifier achieved the highest accuracy of 0.89 for the ROC and 0.34 for the PR curves.



Random Forest Classifier

N– Estimators	Random State	ROC Curve Accuracy	PR Curve Accuracy
100	42	0.89	0.36
95	32	0.88	0.31
100	35	0.87	0.31
100	30	0.88	0.31
100	35	0.87	0.33
95	34	0.88	0.32
100	36	0.88	0.34
100	38	0.88	0.32
90	42	0.88	0.31
95	42	0.88	0.32

2) Transformers Model: Another strategy using Transformers yields an accuracy of 0.99 in Vader and 0.94 in Roberta.Here, at first we have taken subset of 1000 reviews and done sentimental analysis on all the dataframe subset taken from the dataset.



Vader

	Precision	Recall	F1 score	Support	Acccuracy
0	1	0.99	1	1000	0.99
1	0	0	0	0	

ROBERTA

	Precision	Recall	F1 score	Support	Acccuracy
0	1	0.94	0.97	1000	0.94
1	0	0	0	0	

VIII. Conclusion

1) Obtaining a ROC score of 0.89, which makes it very good and trust worthy in generating predictions on hotel reviews.
2) We have a precision recall curve to back this up. The chart clearly shows that as recall increases, precision drops. This demonstrates the importance of selecting a prediction threshold that is appropriate for our needs.
3) As a result, the Random Forest classifier performed admirably when generating final predictions.
4) The Gensim module generates a vector of numbers that represents every word in the corpus based on the situations in which they appear (Word2Vec), which makes it quite intriguing.
5) In general Roberta has high accuracy when compared to Vader but in our model as we have taken subset of reviews so we got vader higher accuracy.

IX. References

1) Sarah Anis, Sally Saad and Mostafa Aref [2020]. Sentiment Analysis of Hotel Reviews Using Machine Learning Techniques. Part of the Advances in Intelligent Systems and Computing book series (AISC,volume 1261).
2) Nandal, N., Tanwar, R. and Pruthi, J. Machine learning based aspect level sentiment analysis for Amazon products. Spat. Inf. Res. 28, 601–607 (2020).
3) Shaziya, Humera, G. Kavitha, and Raniah Zaheer. ”Text categorization of movie reviews for sentiment analysis.” International Journal of Innovative Research in Science, Engineering and Technology 4.11 (2015): 11255-11262.
4) Kamal, Ahmad. Review mining for feature based opinion summarization and visualization. arXiv preprint arXiv:1504.03068 (2015).
5) Appel, O., Chiclana, F., Carter, J., and Fujita, H. (2016). A hybrid approach to the sentiment analysis problem at the sentence level. Knowledge-Based Systems, 108, 110–124.
6) https://medium.com/analytics-vidhya/tf-idf-term-frequency-technique-easiest-explanation-for-text-classification-in-nlp-with-code-8ca3912e58c3
7) https://medium.com/swlh/bag-of-words-code-the-easiest-explanation-of-nlp-technique-using/