SENTIMENT ANALYSIS OF REVIEWS USING NATURAL LANGUAGE PROCESSING

A Project-II Report

Submitted in partial fulfillment of requirement of the

Degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

BY

Vaishnavi Meharwal (EN16CS301282)
Varun Gokhale (EN16CS301283)
Shruti Mundi (EN16CS301252)
Siddhesh Gupta (EN16CS301262)

Under the Guidance of
Prof. (Dr.) Ruchi Patel
Prof. Sachin Solanki
Dr. Ravi Changle

Department of Computer Science and Engineering
Faculty of Engineering
MEDI-CAPS UNIVERSITY, INDORE- 453331
May, 2020

SENTIMENT ANALYSIS OF REVIEWS USING NATURAL LANGUAGE PROCESSING

A Project-II Report

Submitted in partial fulfillment of requirement of the

Degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

BY

Vaishnavi Meharwal (EN16CS301282)
Varun Gokhale (EN16CS301283)
Shruti Mundi (EN16CS301252)
Siddhesh Gupta (EN16CS301262)

Under the Guidance of

Prof. (Dr.) Ruchi Patel
Prof. Sachin Solanki
Dr. Ravi Changle

Department of Computer Science and Engineering
Faculty of Engineering
MEDI-CAPS UNIVERSITY, INDORE- 453331
May, 2020

REPORT APPROVAL

The project work **“Sentiment Analysis of Reviews using Natural Language Processing”** is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite for the Degree for which it has been submitted.

It is to be understood that by this approval the undersigned do not endorse or approved any statement made, opinion expressed, or conclusion drawn there in; but approve the “Project Report” only for the purpose for which it has been submitted.

Internal Examiner
Name:
Designation:
Affiliation:

External Examiner
Name:
Designation:
Affiliation:

DECLARATION

We hereby declare that the project entitled "**Sentiment Analysis of Reviews using Natural Language Processing"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in 'Department of Computer Science & Engineering' completed under the supervision of **Prof. Sachin Solanki,** Faculty of Engineering, Medi-Caps University Indore, **Prof (Dr.) Ruchi Patel,** Faculty of Engineering, Medi-Caps University Indore and **Dr. Ravi Changle**, Global Data Science Trainer, Talent Sprint Pvt. Ltd. is an authentic work.

Further, I declare that the content of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for the award of any degree or diploma.

Shruti Mundi	**Siddhesh Gupta**
(EN16CS301252)	**(EN16CS301262)**
Vaishnavi Meharwal	**Varun Gokhale**
(EN16CS301282)	**(EN16CS301283)**

CERTIFICATE

We **Prof. Sachin Solanki, Prof (Dr.) Ruchi Patel and Dr. Ravi Changle** certify that the project entitled **"Customer Review Analysis – a Sentimental Approach"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology by **Shruti Mundi (EN16CS301252), Siddhesh Gupta (EN16CS301262), Vaishnavi Meharwal (EN16CS301282) and Varun Gokhale (EN16CS301283),** is the record carried out by her under guidance and that the work has not formed the basis of award of any other degree elsewhere.

Prof. Sachin Solanki	Prof. (Dr.) Ruchi Patel
Department of Computer Science & Engineering	Department of Computer Science & Engineering
Medi-Caps University, Indore	Medi-Caps University, Indore

Dr. Ravi Changle	Dr. Suresh Jain
Global Data Science Trainer	Head of the Department
Talent Sprint Pvt. Ltd.	Computer Science and Engineering
	Medi-Caps University, Indore

ACKNOWLDGEMENTS

We would like to express my deepest gratitude to Honorable Chancellor, **Shri R C Mittal,** who has provided me with every facility to successfully carry out this project, and my profound indebtedness to **Prof. (Dr.) Sunil K Somani,** Vice Chancellor, Medi-Caps University, whose unfailing support and enthusiasm has always boosted up my morale. We also thank **Prof. (Dr.) D K Panda,** Dean, Faculty of Engineering, Medi-Caps University, for giving me a chance to work on this project. We would also like to thank my Head of the Department **Prof. (Dr.) Suresh Jain** for his continuous encouragement for betterment of the project.

We express our heartfelt gratitude to my External Guide**, Dr. Ravi Changle,** Global Data Science Trainer, Talent Sprint Pvt. Ltd. as well as to our Internal Guides, **Prof. (Dr.) Ruchi Patel and Prof. Sachin Solanki,** Professors, Department of Computer Science & Engineering Medi-Caps University, without whose continuous help and support, this project would ever have reached to the completion.

We would also like to thank the team of Tata Consultancy Services for their help and support, due to which we became able to complete the design and technical report. Without their support this report would not have been possible.

Shruti Mundi (EN16CS30152)
Siddhesh Gupta (EN16CS301262)
Vaishnavi Meharwal (EN16CS301282)
Varun Gokhale (EN16CS301283)
B.Tech. IV Year
Department of Computer Science & Engineering
Faculty of Engineering
Medi-Caps University, Indore

ABSTRACT

User reviews and comments on hotels on the web are an important information source in travel planning. Therefore, knowing about these comments is important for quality control to the hotel management, too. We present a system that collects such comments from the web and creates classified and structured overviews of such comments and facilitates access to that information.

We consider the problem of classifying documents by the overall sentiment, e.g., determining whether a review is positive or negative. Using hotel reviews as data, we find that standard machine learning techniques definitively outperform human-produced baselines. We will explore wide range of machine learning models including Naive Bayes (NB), Support vector machine (SVM), Logistic Regression and Neural Network to classify a review. To extract the frequent words from the reviews we have used Term Frequency (TF) and Inverse Document Frequency (IDF) approach. We conclude by comparing accuracy of different strategic models and discuss about scope for future work.

Customer reviews on social media websites reflect the customer's opinions concerning various aspects of the hotel’s place or service (e.g., “comfortable room” and “terrible service”). We determine the positive and negative aspects of the hotel services from the text reviews using the N-Gram model.

Keywords:

Machine Learning, Natural Language Processing, Sentiment Analysis, VADER, TF-IDF, Naïve Bayes, Logistic Regression, Neural Network, N-Gram model.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

Abbreviation	Description
NLP	Natural Language Processing
NLTK	Natural Language Tool Kit
VADER	Valence Aware Dictionary and sEntiment Reasoner
POS	Part-Of-Speech
SVM	Support Vector Machine
TF	Term Frequency
IDF	Inverse Document Frequency
LDA	Latent Dirichlet Allocation
PCA	Principal Component Analysis

CHAPTER 1

INTRODUTION

INTRODUCTION

1.1 INTRODUCTION

Travel planning and hotel booking on website has become one of an important commercial use. Sharing on web has become a major tool in expressing customer thoughts about a particular product or Service. Recent years have seen rapid growth in online discussion groups and review sites where a crucial characteristic of a customer's review is their sentiment or overall opinion, for example if the review contains words like 'great', 'best', 'nice', 'good', 'awesome' is probably a positive comment. Whereas if reviews contain words like 'bad', 'poor', 'awful', 'worse' is probably a negative review. However, most hotel's star rating does not express the exact experience of the customer. We seek to turn words and reviews into quantitative measurements. We extend this model with a supervised sentiment component that is capable of classifying a review as positive or negative with accuracy. We also determine the polarity of the review that evaluates the review as recommended or not recommended using semantic VADER. A phrase has a positive semantic orientation when it has good associations (e.g., "excellent, awesome") and a negative semantic orientation when it has bad associations (e.g., "terrific, bad"). Next step is to assign the given review to a class, positive or negative, based on the average semantic orientation of the phrases extracted from the review. If the average is positive, the prediction is that the review posted is positive. Otherwise, the prediction is that the item is negative.

Many people use Internet as a source of information according to the growth of technology in the world. People use online transactions to get more useful information for shopping and booking hotel or ticket in their business traveling plan. Many travelers collect and share their views on experience about hotels, travel guideline on the Internet as a review. The most of users are allowed to share their views about interesting places, good restaurants, and grate hotels on many websites such as TravelToBagan.com and MyanmarHotel.com etc. The expression for users on the hotel reviews that is the need of mining to generate the contents in present reviews. By using internet, user also read the reviews information and make decision about hotel, restaurants, products etc. The more important and related information of hotels should be fetched on reviews and then presented the summarization of reviews for users. The main idea of opinion mining is classification on opinion of user's expression into negative or positive.

1.2 LITERATURE REVIEW

Nibedita Panigrahi and Asha. T. [1] in 2018 proposed a method based on aspect level Sentiment analysis for rating the hotels, they used RHALSA (Ranking Hotels Using Aspect Level Sentiment Analysis) algorithm. The data was taken from some trip advisor's reviews. The proposed work was for two major aspects, i.e. Cleanliness and hotel service. The Standard Core-NLP sentiment approach was used in the proposed method for sentiment analysis to generate scores. The Sentiment Score is calculated based on the Stanford Core-NLP Sentiment Levels and depicted as: 0 for "Very Negative", 1 for Negative, 2 for Neutral, 3 for "Positive" and 4 for "Very Positive". The result shows with respect to each of the aspects that were considered on the same hotel are of mixed opinions. The RHALSA presented was limited to handle negative comments but can be extended to handle discourse relation which may change the orientation of the sentence.

The paper titled "Sentiment Analysis of Hotel Reviews in Greek: A Comparison of Unigram Features Article" by George Markopoulos, George K. Mikros and Anastasia Iliadi [2] created a sentiment classifier that uses Support Vector Machine algorithm. Using Unigram language model, a machine learning algorithm is trained which calculates the frequency of individual words. The TF-IDF was applied as weighing scheme and also for finding the occurrence of selected polarity words. The tenfold cross validation was applied on dataset that divided the data into ten equal size folds that contain 180 reviews each. The cross-validation process was applied then. The proposed work excludes Extremely short (i.e. less than 30 words) and very lengthy (i.e. more than 250 words) reviews from the corpus. The authors got accuracy of TF-IDF as 95.78% and accuracy of TO-approach as 71.76%.22.

People give reviews and rate hotels based on their opinions on a numerical score. There lie many aspects behind their reviews. People generally see the overall rating of the hotel but do not read all the reviews. So, aspect- specific sentiment analysis provides a good solution. A paper was presented on this concept, by Wei Xue, Tao Li and Naphtali Rishe, [7] named "Aspect identification and ratings inference for hotel reviews", that used ILDA (Interdependent Latent Dirichlet Allocation) algorithm. The proposed method separates the vocabulary of reviews into header and modifier terms. For example, if a review is given as nice service in this service is the header term and nice is the modifier term. Header terms does not have sentiment polarity. The modifier terms lead to the sentiment polarity. The modifier terms can be changed by using the terms like nice, worst, good and so on but the header term cannot be changed. Modifier term gives the polarity while the header term gives the aspects. The algorithm used is ARIH (Aspect and Rating Inference Using Hotel Specific Aspect Rating Priors).

A. Jeyapriya and C.S. KanimozhiSelvi, [3] in their paper titled "Extracting Aspects and Mining Opinions in Product Reviews using Supervised Learning Algorithm" proposed supervised learning algorithm for extracting aspects and mining opinions in product reviews. The system uses customer reviews to extract aspect and mine whether given review is positive or negative opinion. First of all, the method removes Stop words. Stop words are words which are most frequently used in English and not useful in text mining. After removal stop words stemming is performed to form root word of a word. Porter stemmer algorithm is used for stemming process. After this POS tagging is done. POS tagging is as explained before. Here Stanford tagger is used for POS tagging. After POS tagging aspect extraction is performed. Nouns are extracted and then its frequency is compared to minimum support count. The word which has higher count than minimum support count is extracted. The Naïve Bayesian algorithm using supervised term counting based approach is used for sentence and aspect orientation. Finally, the system identifies the number of positive and negative opinions of each extracted aspect in customer reviews.

Gaurav Dubey, Ajay Rana and Naveen Kumar Shukla [4] in their paper, "User Reviews Data Analysis using Opinion Mining on Web" proposed a simple method for opinion mining. The method exploits the publicly available pool of online product reviews by automatic extracting marketing intelligence from the vast repository of user generated opinions through main three steps: Part-Of-Speech (POS) tagging, Sentiment Analysis through rule mining, and summarizing and displaying the output. Part-of-Speech Tagging (POST) or lexical set is used to find out the grammatical words in any document or user speech: like noun, verb, adjective etc. This can be done either on basis of definition, e.g. all names are noun like India, or on the basis of context which depends upon the relationship with neighboring or similar words.

Sumbal Riaz, Mehvish Fatima, M. Kamran and M. Wasif Nisar [5], paper titled "Opinion mining on large scale data using sentiment analysis and k-means clustering" offered research as sentiment classification is divided into two ways. First approach is lexicon based which used for calculating orientation of a document of words or phrases and second approach is machine learning techniques such as clustering. For this applying TF-IDF gives the term frequency of words in a document. By calculating sentiment strength calculation then we will get certain values and those values clustering technique will be applied. By calculating Euclidean distance, we classify the words/text into positive, negative and neutral.

Joscha Markle-Huß, Stefan Feuerriegel and Helmut Prendinge, [8] in their paper "Improving Sentiment Analysis with Document-Level Semantic Relationships from Rhetoric Discourse Structures" devised and compared various techniques like Bag of words models, n-grams for using semantic information to improve the performance of sentiment analysis. The earlier approaches did not consider the semantic associations between sentences or documents parts.

Abdullah Aziz Sharfuddin, Md. Nafis Tihami and Md. Saiful Islam [6] in their paper titled "A Deep Recurrent Neural Network with BiLSTM model for Sentiment Classification" proposed a new way of sentiment classification of Bengali text using Recurrent Neural Network (RNN). Using deep recurrent neural network with BiLSTM, the accuracy 85.67% is achieved.

Vijay B. Raut and D.D. Londhe [8], in their paper titled "Opinion Mining and Summarization of Hotel Reviews", presented machine learning and SentiWordNet based method for opinion mining from hotel reviews and sentence relevance score-based method for opinion summarization of hotel reviews. They obtained about 87% of accuracy of hotel review classification as positive or negative review by machine learning method.

Merin Thomas and Latha C.A [9] in their paper named , "Sentimental analysis using recurrent neural network" implemented sentimental analysis of tweets in South Indian language Malayalam. The model used was Recurrent Neural Networks Long Short-Term Memory, a deep learning technique to predict the sentiments analysis. Achieved accuracy was found increasing with quality and depth of the datasets

Bo Pang, Lillian Lee and Shivakumar Vaithyanathan [9] in their paper titled "Thumbs up? Sentiment Classification using Machine Learning Techniques" considered the problem of classifying documents not by topic, but by overall sentiment, e.g., determining whether a review is positive or negative. Using movie reviews as data, they found that standard machine learning techniques definitively outperform human-produced baselines. However, the three machine learning methods they employed (Naive Bayes, maximum entropy classification, and support vector machines) do not perform as well on sentiment classification as on traditional topic-based categorization. they concluded by examining factors that make the sentiment classification problem more challenging.

1.3 OBJECTIVE

Travel planning and hotel booking on website has become one of an important commercial use. Sharing reviews on web has become a major tool in expressing customer thoughts about a particular product or Service. The main objective of this project is to analyze the reviews given by the customers to the various hotels and turn them into quantitative measurements. The model built is extended with a supervised sentiment component that is capable of classifying review as positive or negative with accuracy. In this project, a system is built which is able to perform sentiment analysis of the review and use it to the customer satisfaction. The customer reviews were categorized as negative, positive and neutral. Also these customer reviews reflect the customer's opinions concerning various aspects of the hotel's place or the services offered by the hotel (eg:"comfortable bed" and "terrible service") therefore the project also aims at determining the positive and negative aspects of the hotel that are being frequently mentioned in the reviews and thereby enabling the hotels to know their negative aspect. The hotels can analyze the negative aspects and work upon their services to increase the number of visitors and hence increase their business value.

1.4 SIGNIFICANCE

Recent years have seen rapid growth in collecting customer reviews through the internet. These reviews are extracted in their raw form which involves usage of slangs and spelling mistakes. The preprocessing of the reviews and their analysis is itself quite time consuming when attempted manually. This project aims at preprocessing the reviews collected and further classifying them into negative, positive and neutral. The text classification model thus created is capable of classifying a significant amount of text data with acceptable accuracy. The time consumed in preprocessing is reduced by this approach and the hotels would be capable of knowing the number of negative and positive reviews received by them. Further this project's approach analyzes the negative aspects, thereby enabling hotels to know where they are lacking in providing good services. The model can further be used for rating hotels based on their reviews and hence distinguishing the top hotels which are most loved by the people and frequently visited.

1.5 RESEARCH DESIGN

The customer review analysis worked upon the reviews received by the various hotels in Europe. The customer reviews may involve usage of slangs or the spelling error. The Vader in the NLTK library dealt with these challenges and enabled the categorization of reviews into sentiments. Many research papers were studied to analyze the approach which could improvise the accuracy in the classification. The sentimental analysis approach requires the identification of word that depict some meaningful insights about the hotels like “good”, ”terrible”, ”worst”, ”superb” and so on. The research work involved the analysis of existing works on customer reviews sentimental analysis. The feature selection was done based on the suggested research. The dataset was also studied and visualized to know the hotels that were most reviewed and visited. The balancing of dataset was done to avoid any ill effects of imbalancing on the predictions. The predictions done by the model were improved based on the research papers studied. The proven approaches were tried on the model to analyze their influence on the built model.

1.6 SOURCE OF DATA

The dataset used in the project is taken from Kaggle. The dataset depicts the reviews given by the customer to the various hotels in Europe. It consist of 515k rows. In order to avoid complexity the model is applied on the balanced 20k Hotel Reviews dataset. The model analysed 484 hotels and their reviews were classified into positive and negative sentiments.

CHAPTER 2

SYSTEM REQUIREMENT ANALYSIS

SYSTEM REQUIREMENT ANALYSIS

2.1 INFORMATION GATHERING

The dataset used for this task was collected from Kaggle. The dataset contained 500,000 training examples scraped from Booking.com. The dataset contains 515,000 customer reviews and scoring of 1493 luxury hotels across Europe. For better performance of the model we selected 20,000 rows from the dataset, in the process trying our best to keep the data balanced.

The dataset contained 17 fields:

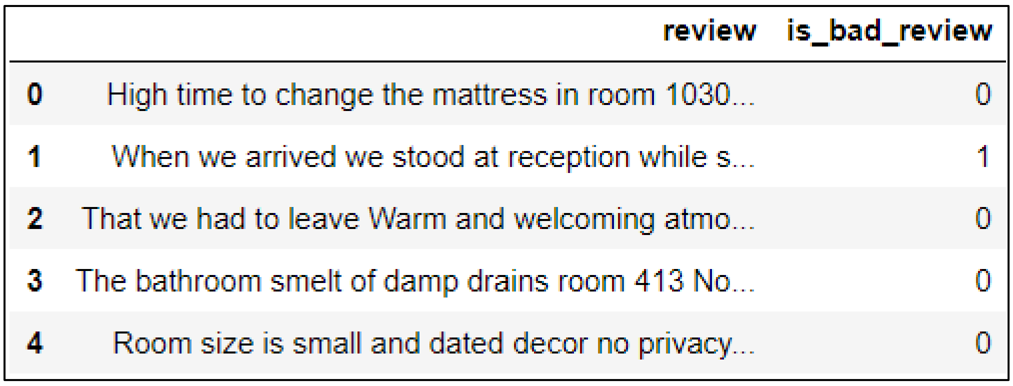
1. **Hotel_Address:** Address of hotel.
2. **Review_Date:** Date when reviewer posted the corresponding review.
3. **Average_Score:** Average Score of the hotel, calculated based on the latest comment in the last year.
4. **Hotel_Name:** Name of Hotel
5. **Reviewer_Nationality:** Nationality of Reviewer
6. **Negative_Review:** Negative Review the reviewer gave to the hotel.
7. **Review_Total_Negative_Word_Counts:** Total number of words in the negative review.
8. **Positive_Review:** Positive Review the reviewer gave to the hotel.
9. **Review_Total_Positive_Word_Counts:** Total number of words in the positive review.
10. **Reviewer_Score:** Score the reviewer has given to the hotel, based on his/her experience
11. **Total_Number_of_Reviews_Reviewer_Has_Given:** Number of Reviews the reviewers has given in the past.
12. **Total_Number_of_Reviews:** Total number of valid reviews the hotel has.
13. **Tags:** Tags reviewer gave the hotel.
14. **days_since_review:** Duration between the review date and scrape date.
15. **Additional_Number_of_Scoring:** This number indicates how many valid scores without review in there.
16. **lat:** Latitude of the hotel
17. **lng:** longitude of the hotel

As sentiments are usually bipolar like good/bad or happy/sad or like/dislike, we categorized these ratings as either 1 (like) or 0 (dislike) based on the ratings. If the rating (i.e., Reviewer_Score) was above 5, we deduced that the person liked the hotel otherwise he did not.

Reviewer_Score > 5 → Positive Review (0)

Reviewer_Score < 5 → Negative Review (1)

Figure 2.1 Binarization of Reviewer_Score



	review	is_bad_review
0	High time to change the mattress in room 1030...	0
1	When we arrived we stood at reception while s...	1
2	That we had to leave Warm and welcoming atmo...	0
3	The bathroom smelt of damp drains room 413 No...	0
4	Room size is small and dated decor no privacy...	0

Figure 2.2 dataset

2.2 FEASIBILITY STUDY

A feasibility study is a preliminary study which investigates the information of prospective users and determines the resources requirements, costs, benefits and feasibility of proposed system. A feasibility study takes into account various constraints within which the system should be implemented and operated. In this stage, the resource needed for the implementation such as computing equipment, manpower and costs are estimated. The estimated are compared with available resources and a cost benefit analysis of the system is made. The feasibility analysis activity involves the analysis of the problem and collection of all relevant information relating to the project. The main objectives of the feasibility study are to determine whether the project would be feasible in terms of economic feasibility, technical feasibility and operational feasibility and schedule feasibility or not. It is to make sure that the input data which are required for the project are available. Thus, we evaluated the feasibility of the system in terms of the following categories:

- Technical Feasibility
- Operational Feasibility
- Economic Feasibility

2.2.1 Technical Feasibility

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, at the point in time there is no any detailed designed of the system, making it difficult to access issues like performance, costs (on account of the kind of technology to be deployed) etc. A number of issues have to be considered while doing a technical analysis; understand the different technologies involved in the proposed system. Before commencing the project, we have to be very clear about what are the technologies that are to be required for the development of the new system. Is the required technology available? Our system "Customer Review Analysis" is technically feasible since all the required tools are easily available. Python and flask can be easily handled. Although all tools seems to be easily available there are challenges too.

2.2.2 Operational Feasibility

Proposed project is beneficial only if it can be turned into information systems that will meet the operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to Implementation? The proposed was to make a simplified web application. It is simpler to operate and can be used in any webpages. It is free and not costly to operate.

2.2.3 Economic Feasibility

Economic feasibility attempts to weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system. A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be useful point of reference to compare actual costs as the project progresses. There could be various types of intangible benefits on account of automation. These could increase improvement in product quality, better decision making, and timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information.

2.3 REQUIREMENT ANALYSIS

After the extensive analysis of the problems in the system, we are familiarized with the requirement that the current system needs. The requirement that the system needs is categorized into the functional and non-functional requirements. These requirements are listed below:

2.3.1 Functional Requirements

A Functional Requirement (FR) is a description of the service that the software must offer. It describes a software system or its component. A function is nothing but inputs to the software system, its behavior, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform. Functional Requirement of our System are:

- System should be able to classify reviews according to polarity.
- System should be able to list top negative and positive qualities of the Hotel.
- System should be able to extract subjective information from the reviews.

2.3.2 Non-Functional Requirements

Non-functional requirements is a description of features, characteristics and attribute of the system as well as any constraints that may limit the boundaries of the proposed system. The non-functional requirements are essentially based on the performance, information, economy, control and security efficiency and services. Based on these the non-functional requirements are as follows:

- System is available 24 hours and 7 days a week.
- System should be User friendly.
- System should provide a good accuracy.

2.4 PLATFORM SPECIFICATION

For any project various different platforms and services are required throughout the process of development and deployment. We have used google COLAB and Anaconda Spyder as our development platform.

GOOGLE COLAB

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs.

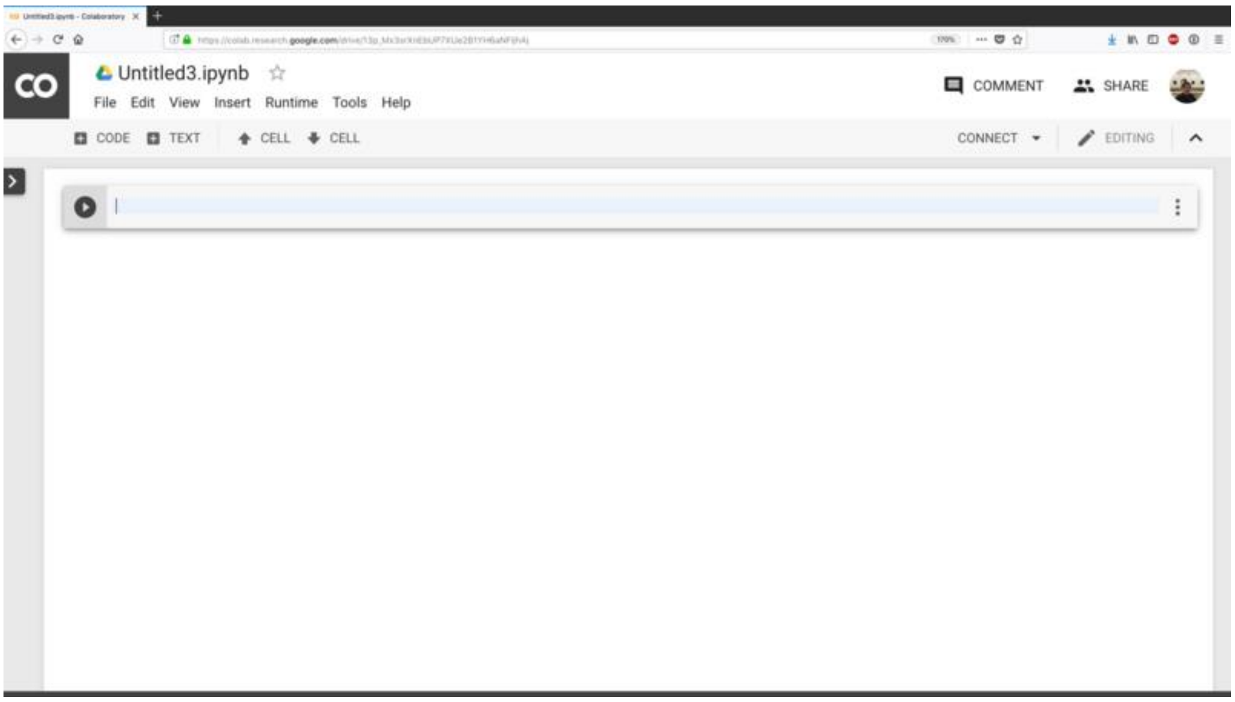
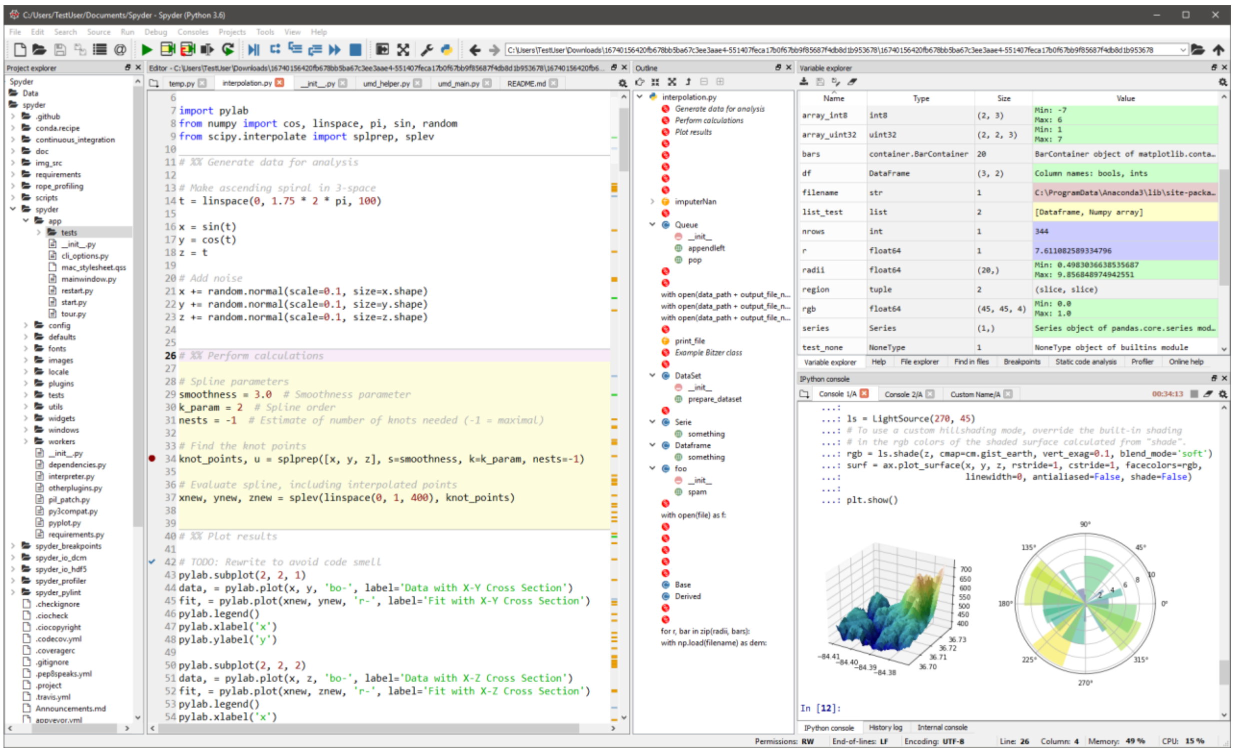


Figure 2.3 Google COLAB interface

SPYDER

Spyder, the Scientific Python Development Environment, is a free integrated development environment (IDE) that is included with Anaconda. It includes editing, interactive testing, debugging, and introspection features.



```

import pylab
from numpy import cos, linspace, pi, sin, random
from scipy.interpolate import splprep, splev

# %% Generate data for analysis

# Make ascending spiral in 3-space
t = linspace(0, 1.75 * 2 * pi, 100)

x = sin(t)
y = cos(t)
z = t

# Add noise
x += random.normal(scale=0.1, size=x.shape)
y += random.normal(scale=0.1, size=y.shape)
z += random.normal(scale=0.1, size=z.shape)


# %% Perform calculations

# Spline parameters
smoothness = 3.0  # Smoothness parameter
k_param = 2  # Spline order
nests = -1  # Estimate of number of knots needed (-1 = maximal)

# Find the knot points
knot_points, u = splprep([x, y, z], s=smoothness, k=k_param, nests=-1)

# Evaluate spline, including interpolated points
xnew, ynew, znew = splev(linspace(0, 1, 400), knot_points)


# %% Plot results

# TODO: Rewrite to avoid code smell
pylab.subplot(2, 2, 1)
data, = pylab.plot(x, y, 'bo-', label='Data with X-Y Cross Section')
fit, = pylab.plot(xnew, ynew, 'r-', label='Fit with X-Y Cross Section')
pylab.legend()
pylab.xlabel('x')
pylab.ylabel('y')

pylab.subplot(2, 2, 2)
data, = pylab.plot(x, z, 'bo-', label='Data with X-Z Cross Section')
fit, = pylab.plot(xnew, znew, 'r-', label='Fit with X-Z Cross Section')
pylab.legend()
pylab.xlabel('x')
```

Name	Type	Size	Value
array_int8	int8	(2, 3)	Min: -7 Max: 6
array_uint32	uint32	(2, 2, 3)	Min: 1 Max: 7
bars	container.BarContainer	20	BarContainer object of matplotlib.conta…
df	DataFrame	(3, 2)	Column names: bools, ints
filename	str	1	C:\ProgramData\Anaconda3\lib\site-packa…
list_test	list	2	[Dataframe, Numpy array]
nrows	int	1	344
r	float64	1	7.611082589334796
radii	float64	(20,)	Min: 0.4983036638535687 Max: 9.856848974942551
region	tuple	2	(slice, slice)
rgb	float64	(45, 45, 4)	Min: 0.0 Max: 1.0
series	Series	(1,)	Series object of pandas.core.series mod…
test_none	NoneType	1	NoneType object of builtins module

```
   ...:
   ...: ls = LightSource(270, 45)
   ...: # To use a custom hillshading mode, override the built-in shading
   ...: # in the rgb colors of the shaded surface calculated from "shade".
   ...: rgb = ls.shade(z, cmap=cm.gist_earth, vert_exag=0.1, blend_mode='soft')
   ...: surf = ax.plot_surface(x, y, z, rstride=1, cstride=1, facecolors=rgb,
   ...:                        linewidth=0, antialiased=False, shade=False)
   ...:
   ...: plt.show()

In [12]:
```

Figure 2.4 SYPDER interface

2.4.1 Hardware Specification

The project simply needs a Personal Computer with a RAM of 2 GB or above, hard disk 10GB and a basic processor with an internet connection.

2.4.2 Software Implementation Language & Technology

We have used various different software's technologies and computer programming languages for developing our project. A brief description of each is given below:

Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combines with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form either without for all major platforms, and can be freely distributed

Tableau

Tableau is a powerful and fastest growing data visualization tool used in the Business Intelligence Industry. It helps in simplifying raw data into the very easily understandable format. Data analysis is very fast with Tableau and the visualizations created are in the form of dashboards and worksheets. The data that is created using Tableau can be understood by professional at any level in an organization. It even allows a non-technical user to create a customized dashboard.

Flask

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website. Flask is part of the categories of the micro-framework.

Microsoft Azure

Microsoft Azure is a cloud computing platform created by **Microsoft** which developers and IT professionals use to build, deploy and manage applications through their global network of datacenters.

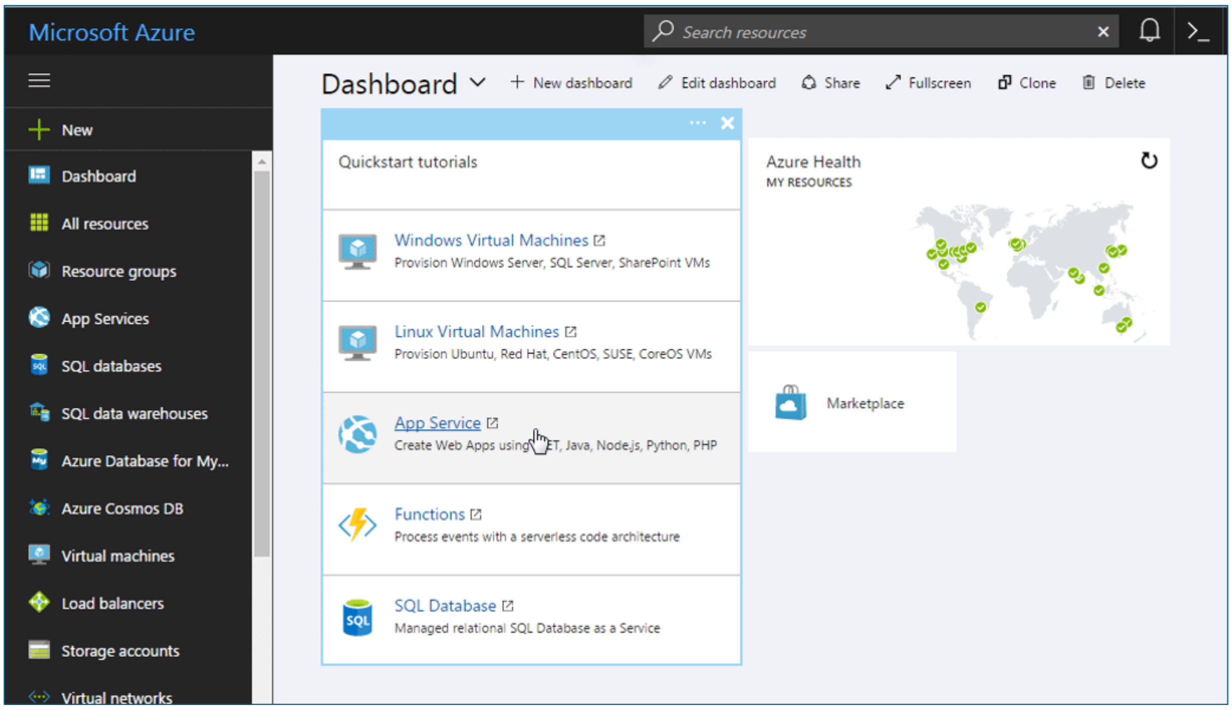


Figure 2.5 Microsoft Azure Dashboard

CHAPTER 3

SYSTEM ANALYSIS

SYSTEM ANALYSIS

It is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components. System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose.

3.1 SYSTEM FLOW

System Flows are systems models that show the activities and decisions that systems execute. They are useful for understanding complex system interactions because they visually show the back and forth interactions between systems and complex branching. They are very similar to Process Flows in look and feel; however, Process Flows are used to document a user's actions whereas System Flows are used to document a system's actions. Below is a template of a system flow with swim lanes denoting different systems needed in the flow.

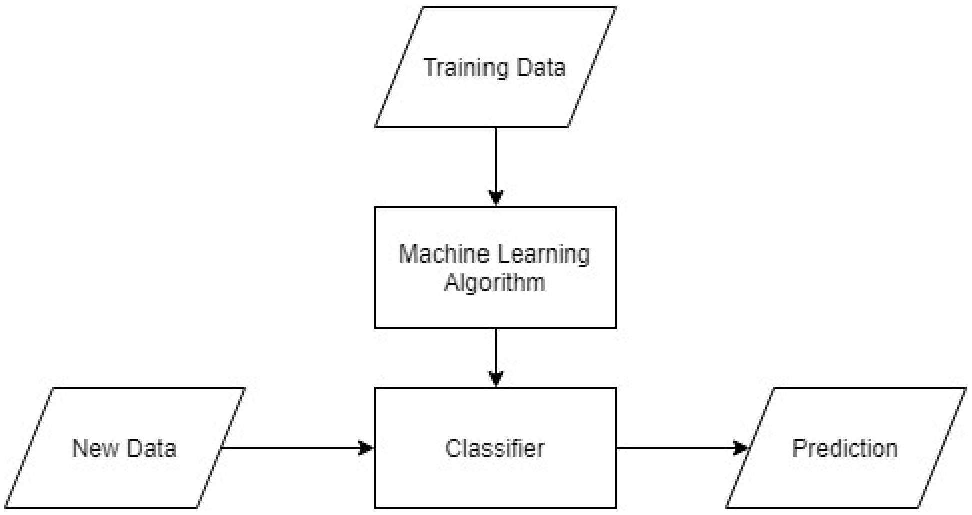


Figure 3.1 System Flow Diagram

3.2 INFORMATION FLOW REPRESENTATION

An information flow diagram (IFD) is a diagram that shows how information is communicated (or "flows") from a source to a receiver or target (e.g. A→C), through some medium. The medium acts as a bridge, a means of transmitting the information. Examples of media include word of mouth, radio, email, etc. The concept of IFD was initially used in radio transmission. The diagrammed system may also include feedback, a reply or response to the signal that was given out. The return paths can be two-way or bi-directional: information can flow back and forth.

An IFD can be used to model the information flow throughout an organization. An IFD shows the relationship between internal information flows within an organization and external information flows between organizations. It also shows the relationship between the internal departments and sub-systems.

An IFD usually uses "blobs" to decompose the system and sub-systems into elemental parts. Lines then indicate how the information travels from one system to another. IFDs are used in businesses, government agencies, television and cinematic processes [12].

3.2.1 Entity-Relationship Diagram

Entity Relationship Diagram, also known as ERD, ER Diagram or ER model, is a type of structural diagram for use in database design. An E-R model is usually the result of systematic analysis to define and describe what is important to processes in an area of a business. It does not define the business processes; it only presents a business data schema in graphical form. It is usually drawn in a graphical form as boxes (entities) that are connected by lines (relationships) which express the associations and dependencies between entities.

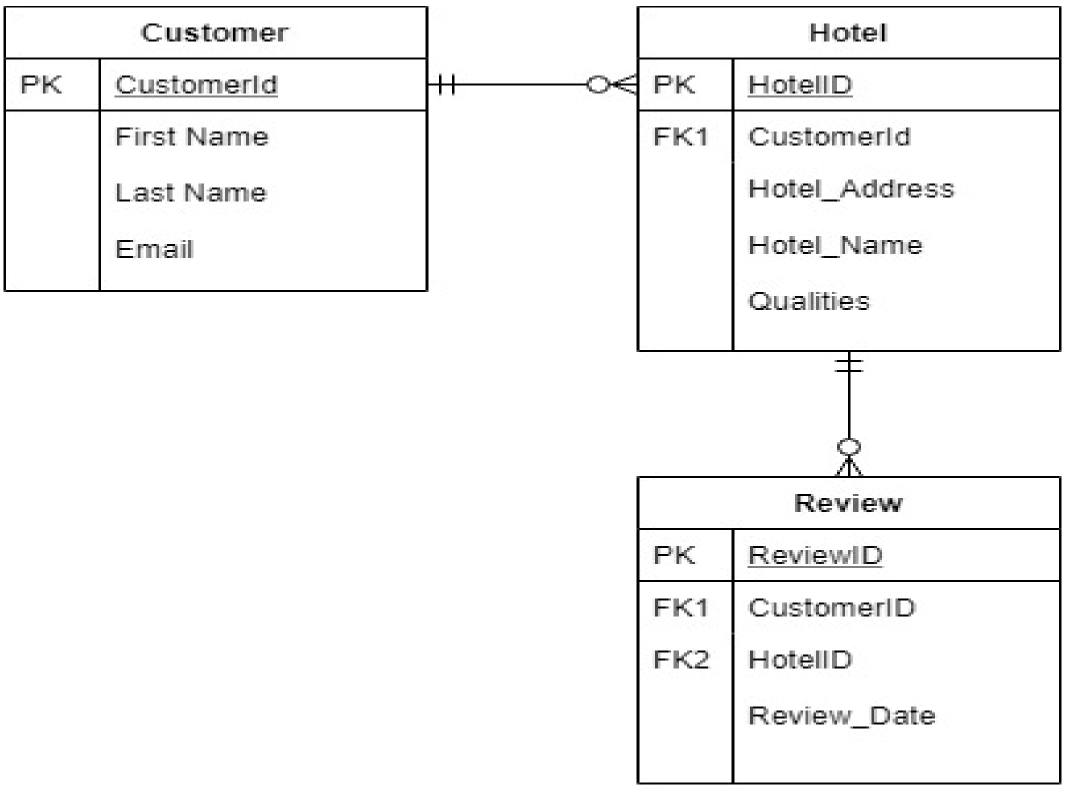


Figure 3.2 Entity Relationship Diagram

3.2.2 Activity Diagram

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc. Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

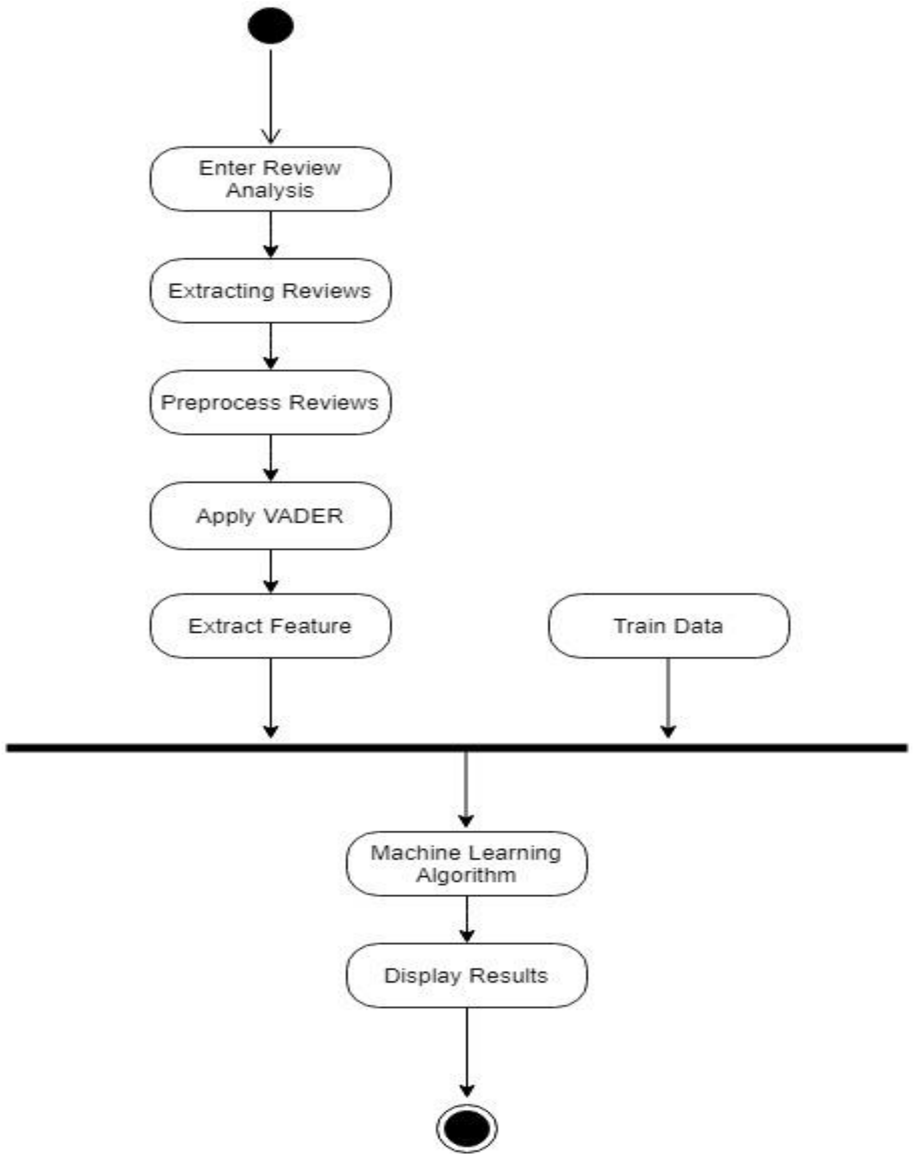


Figure 3.3 Activity Diagram

CHAPTER 4
SYSTEM DESIGN

SYSTEM DESIGN

System design is the phase that bridges the gap between problem domain and the existing system in a manageable way. This phase focuses on the solution domain. It is the phase where the SRS document is converted into a format that can be implemented and decides how the system will operate. In this phase, the complex activity of system development is divided into several smaller sub-activities, which coordinate with each other to achieve the main objective of system development. System design is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system.

4.1 ARCHITECTURAL DESIGN

The software needs the architectural design to represents the design of software. IEEE defines architectural design as “the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system.” The software that is built for computer-based systems can exhibit one of these many architectural styles.
Each style will describe a system category that consists of:

- A set of components (example: a database, computational modules) that will perform a function required by the system.
- The set of connectors will help in coordination, communication, and cooperation between the components.
- Conditions that how components can be integrated to form the system.
- Semantic models that help the designer to understand the overall properties of the system.

4.1.1 Architectural Context Diagram

The context diagram is used to establish the context and boundaries of the system to be modeled: which things are inside and outside of the system being modeled, and what is the relationship of the system with these external entities.

A context diagram sometimes called a level 0 data-flow diagram, is drawn to define and clarify the boundaries of the software system. It identifies the flows of information between the system and external entities. The entire software system is shown as a single process.

To produce the context diagram and agree on system scope, the following must be identified:

- external entities
- data-flows

4.1.2 Architectural Behavioral Diagram

Behavioral Diagrams depict the elements of a system that are dependent on time and that convey the dynamic concepts of the system and how they relate to each other. The elements in these diagrams resemble the verbs in a natural language and the relationships that connect them typically convey the passage of time.

Use Case Diagrams:
Use Case diagrams capture Use Cases and relationships among Actors and the system; they describe the functional requirements of the system; how external operators interact at the system boundary and the response of the system.

Actor:
An actor represents the roles that the users of the use cases play. An actor may be a person (e.g. student, customer), a device (e.g. workstation), or another system (e.g. bank, institution).

In the proposed work, there are two actors one is the user who share his reviews and other is admin of the system that generates prediction and analyze the sentiment of the reviewer.

- User can login to the system and generate the reviews. He is also authorized to check the reviews of other users.
- System Admin can analyze the review generating the predictions. He can also look for the top positive and negative qualities of the hotel.

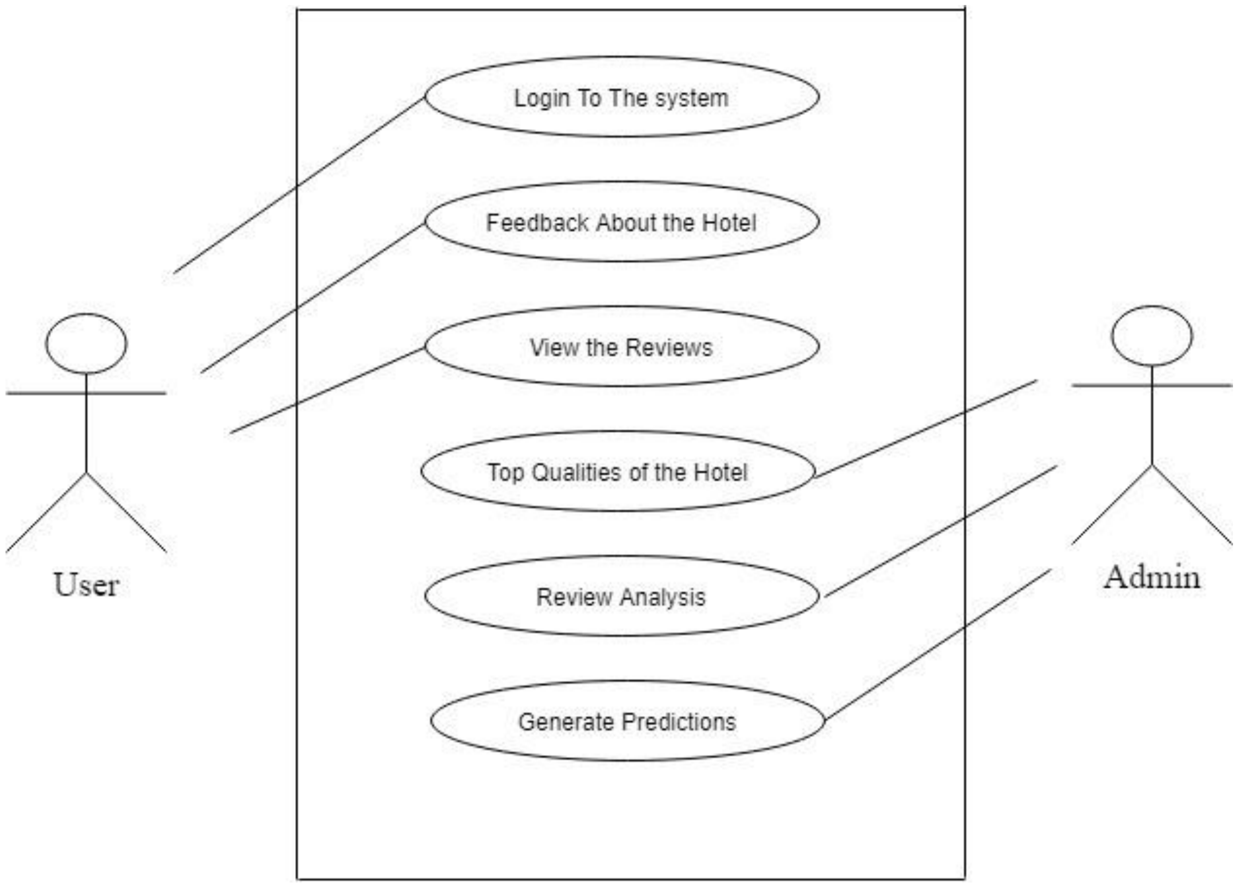


Figure 4.1 Use-Case diagram

4.2 INTERFACE DESIGN

User interface (UI) design is the process of making interfaces in software or computerized devices with a focus on looks or style. Designers aim to create designs users will find easy to use and pleasurable. UI design typically refers to graphical user interfaces but also includes others, such as voice-controlled ones. User interfaces are the access points where users interact with designs. Graphical user interfaces (GUIs) are designs' control panels and faces; voice-controlled interfaces involve oral-auditory interaction, while gesture-based interfaces witness users engaging with 3D design spaces via bodily motions.

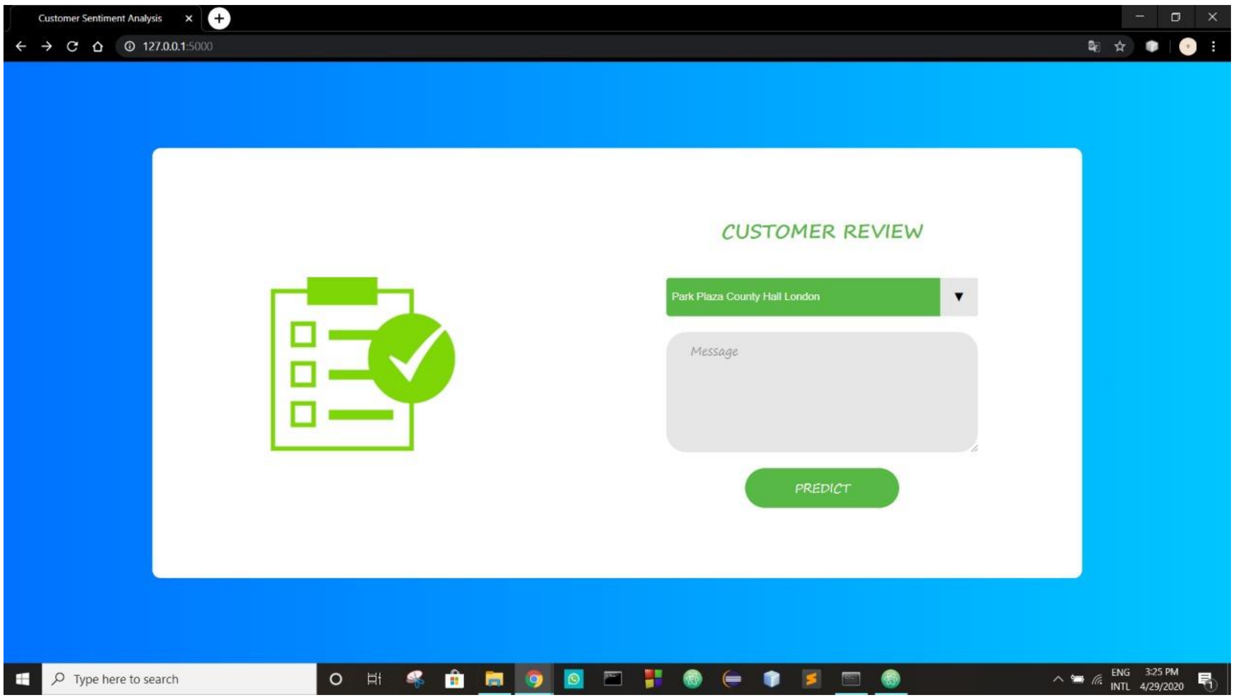


Figure 4.2 Review Analysis Home Page

CHAPTER 5

DATA VISULAIZATIONS

DATA VISUALIZATIONS

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

In the world of Big Data, data visualization tools and technologies are essential to analyse massive amounts of information and make data-driven decisions. It’s hard to think of a professional industry that doesn’t benefit from making data more understandable. Every STEM field benefits from understanding data—and so do fields in government, finance, marketing, history, consumer goods, service industries, education, sports, and so on.

5.1 TYPES OF VISUALIZATIONS

The different types of Visualizations performed in the proposed work are:

- Bar Plot
- Pie Chart
- Geospatial Plot
- Line Plot
- Packed Bubble Chart

5.1.1 Bar Plot

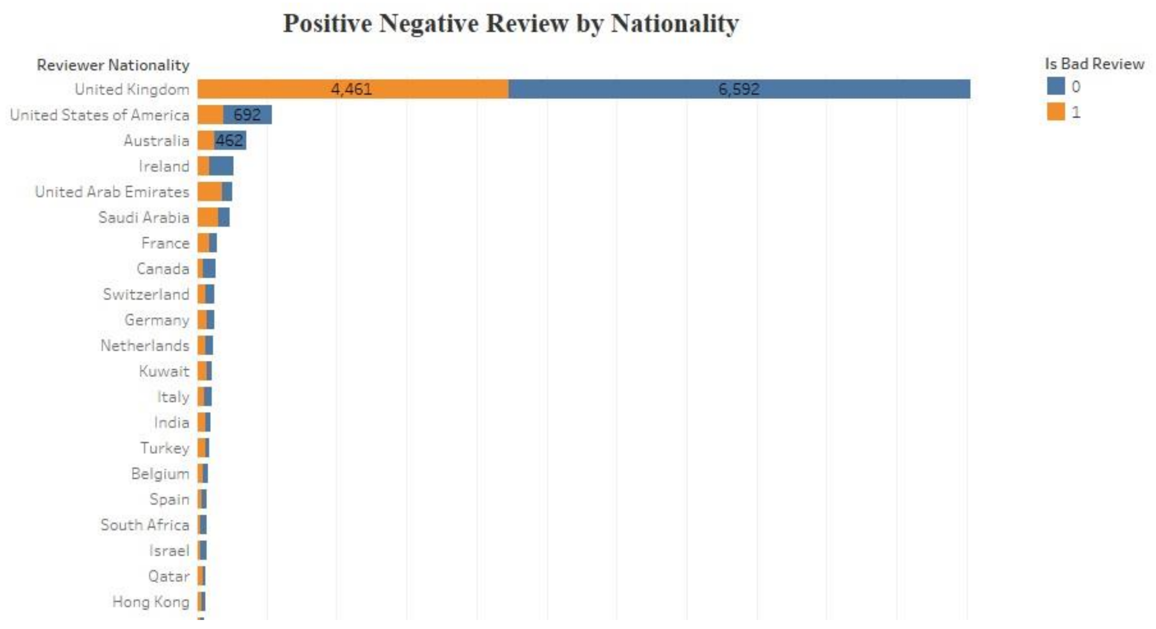


Figure 5.1 Count of reviews by Nationality

From the fig 5.1 it is clear that maximum number of reviewers are from United Kingdom (U.K.) and of all reviewer from U.K 4,461 are not satisfied i.e. they have given negative reviews.

Negative reviews are colored as ‘orange’ while positive are colored ‘Blue’. Similar insights for every nationality can be drawn from the figure.

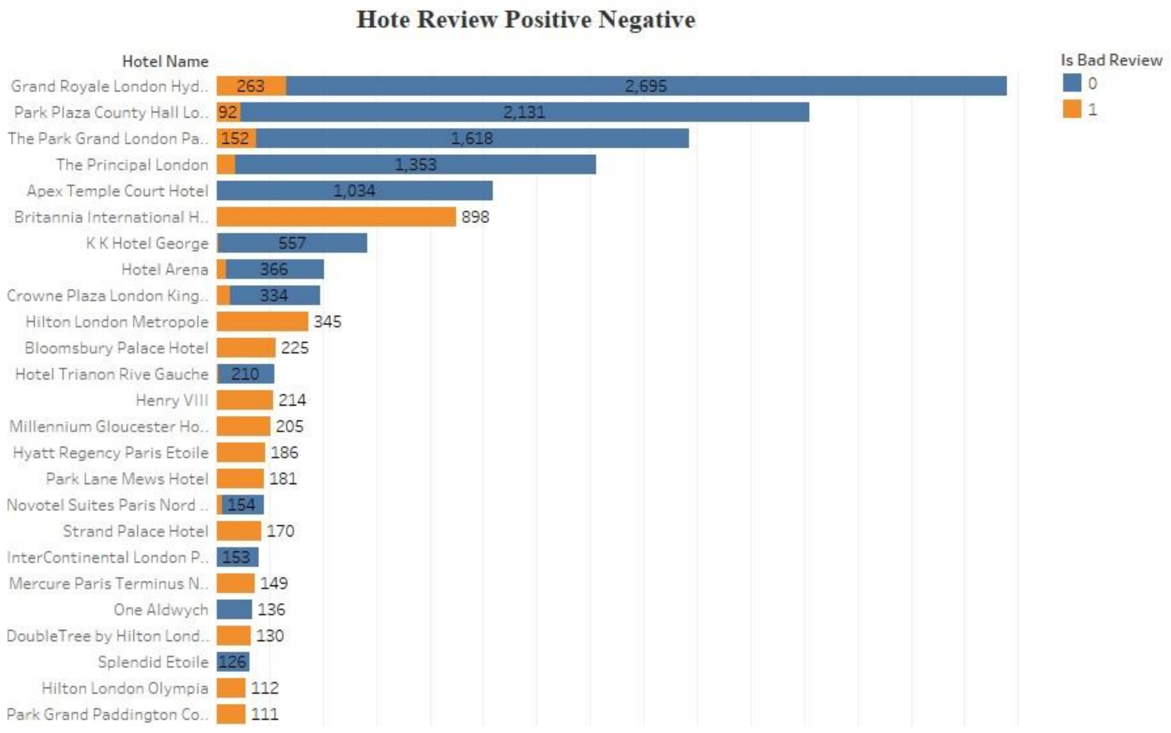


Figure 5.2 Count of Review by Hotel

From the fig 5.2 it is clear that hotel “Grand Royale London Hyde Park” has maximum reviews. Also, this hotel has very less negative reviews of the total. One of the possibility is that this hotel has a brand name associated with and are more preferred by customers.

Also, from the above figure it can be noted that there are some hotels that are unanimously rated or reviewed negative by the customers. One of such hotels is “Britannia International”

5.1.2 Pie Chart

A pie chart (or a circle chart) is a circular statistical graphic, which is divided into slices to illustrate numerical proportion.

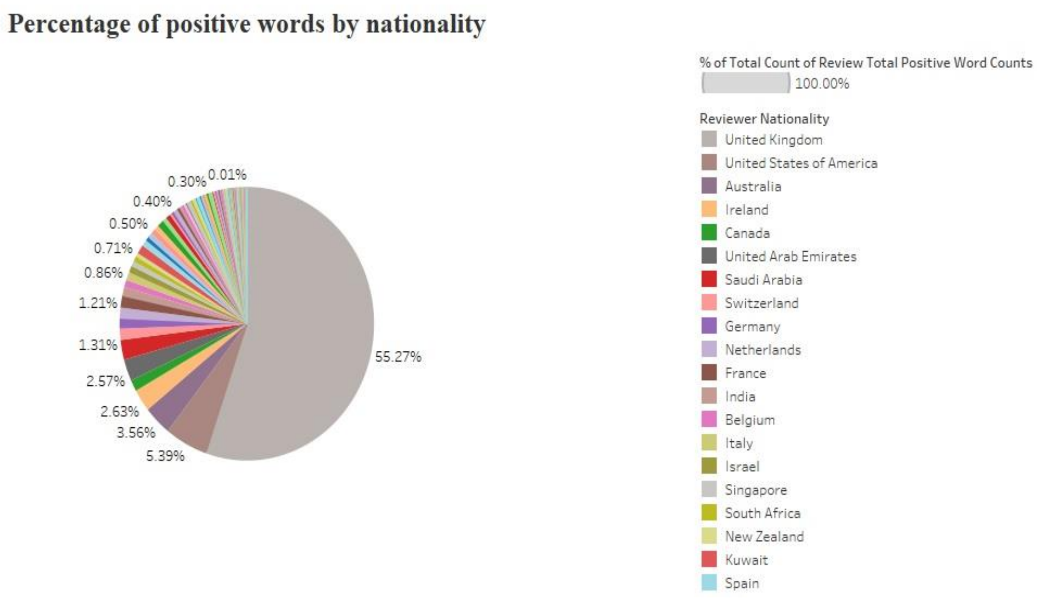


Figure 5.3 Percentage of Positive Words

From the above figure it can be inferred that maximum percentage of positive words are reviewed by United Kingdom reviewer. In above figure the legend for the plot is shown at the left.

5.1.3 Map Plot

Geo-visualization (short for geographic visualization), refers to a set of tools and techniques supporting the analysis of geospatial data through the use of interactive visualization.

From the figure below (Figure 5.4) it is clear that hotel named “Portobello” situated in U.K has got only 2 reviews and the reviews are given by reviewers who have a nationality of U.K. It means that this hotel is not preferred by any other nationalists.

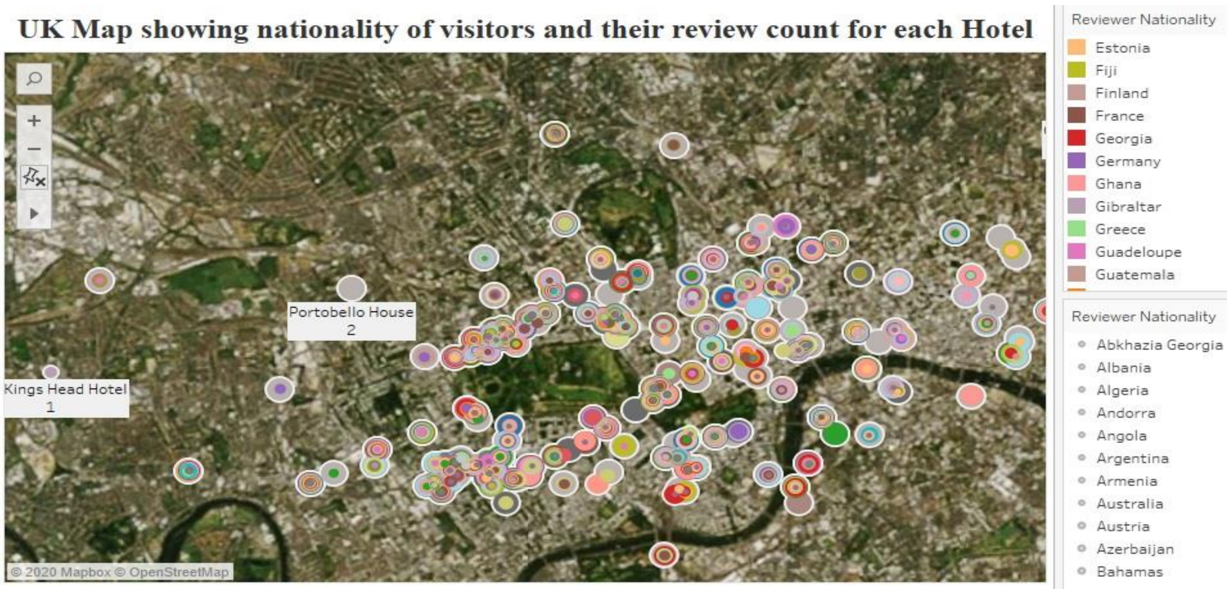


Figure 5.4 United Kingdom geo-spatial data

5.1.4 Line Plots

A line chart or line plot or line graph or curve chart is a type of chart which displays information as a series of data points called 'markers' connected by straight line segments. It is a basic type of chart common in many fields. It is similar to a scatter plot except that the measurement points are ordered (typically by their x-axis value) and joined with straight line segments. A line chart is often used to visualize a trend in data over intervals of time – a time series – thus the line is often drawn chronologically. In these cases they are known as run charts.

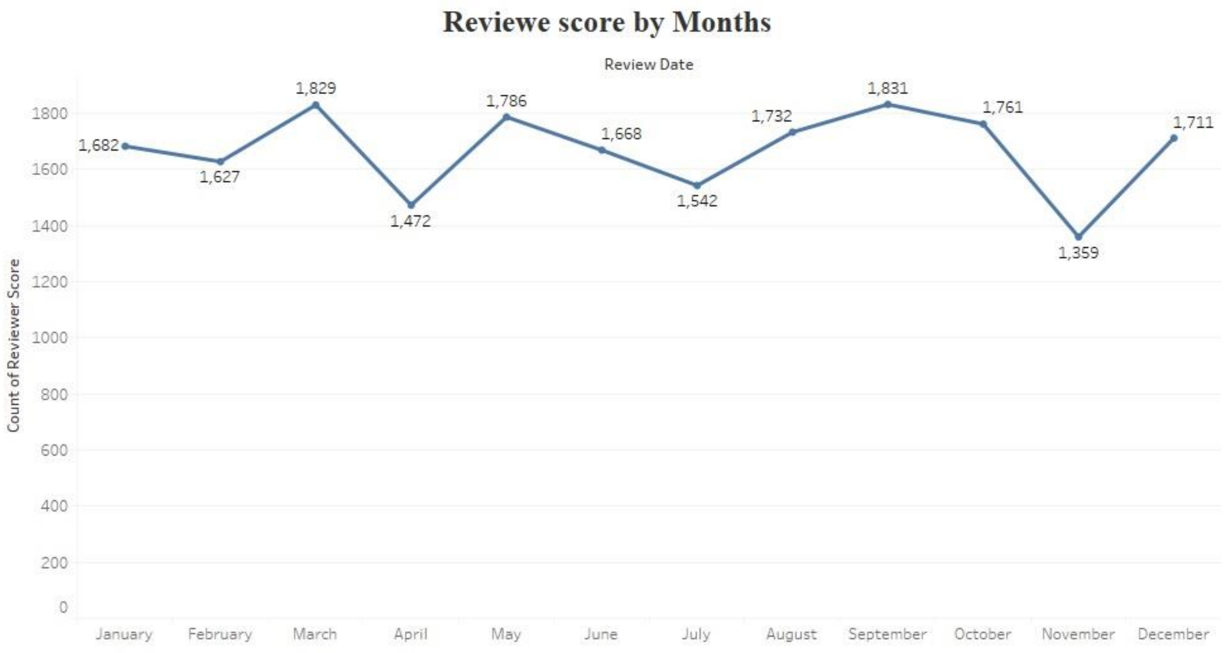


Figure 5.5 Line Chart showing number of reviews by months.

From the figure 5.5 it is clear that number of reviews are maximum in the month of March probably Europe has a peak season of travelling in the month of March.

5.1.5 Packed Bubble Chart

A bubble chart is a data visualization that displays multiple circles (bubbles) in a two-dimensional plot. It is a generalization of the scatter plot, replacing the dots with bubbles. Most commonly, a bubble chart displays the values of three numeric variables, where each observation's data is shown by a circle ("bubble"), while the horizontal and vertical positions of the bubble show the values of two other variables.

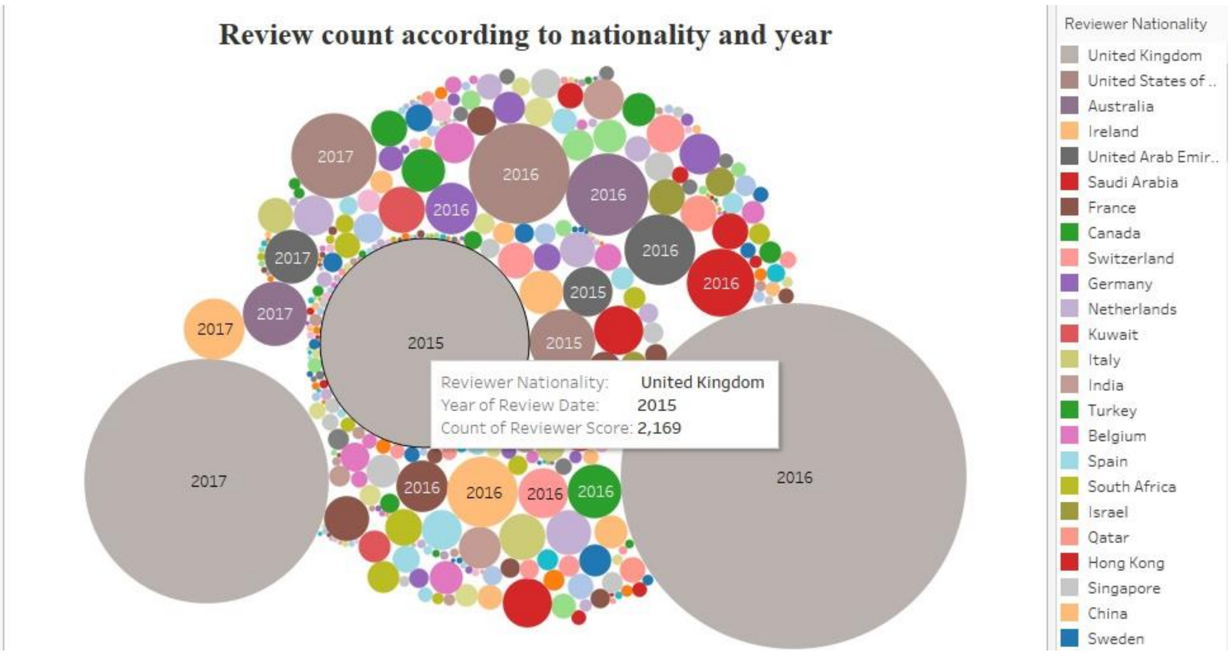


Figure 5.6 Bubble Chart

In the bubble chart above each bubble represent the year and colors represents Nationalities. It can also be noted that in the year 2015 U.K. Nationals have reviewed 2,169 hotels. Similar insights can be drawn for every nationality and for every year.

CHAPTER 6

METHODOLOGY

METHODOLOGY

The following graphic presents the workflow of the project. Each step in the workflow corresponds to a task completed in order to achieve accuracy. The experiments must run in order because the output of one experiment is the input to the next.

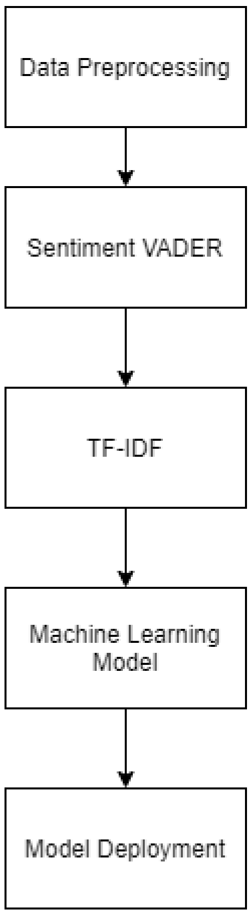


Figure 6.1 Workflow

6.1 DATA PREPROCESSING

Data Preprocessing is needed for transferring text from human language to machine-readable format for further processing. To process text simply means to bring your text into a form that is predictable and analyzable for the task.

In NLP, most of the text and documents contain many words that are redundant for text classification, such as stopwords, mis-spellings, slangs, etc. We have used many methods for text cleaning and pre-processing text

6.1.1 Preprocessing Methods

Capitalization

Sentences can contain a mixture of uppercase and lower case letters. Multiple sentences make up a text document. To reduce the problem space, the most common approach is to reduce everything to lower case. This brings all words in a document in same space

WashingTon → washington

Noise Removal

Another issue of text cleaning as a pre-processing step is noise removal. Text documents generally contains characters like punctuations or special characters and they are not necessary for text mining or classification purposes. Although punctuation is critical to understand the meaning of the sentence, but it can affect the classification algorithms negatively.

[!”#$%&’()*+,-./:;<=>?@[\]^_`{|}~]:

Number Removal

Remove numbers if they are not relevant to your analyses. Usually, regular expressions are used to remove numbers.

nice 2 view→ nice view

Stop Words

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. We would not want these words taking up space in our database, or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to be stop words. NLTK in python has a list of stopwords stored in 16 different languages.

I liked the view → liked view

POS Tagging

The process of classifying words into their parts of speech and labeling them accordingly is known as part-of-speech tagging, POS-tagging, or simply tagging. Parts of speech are also known as word classes or lexical categories. The collection of tags used for a particular task is known as a tagset. Our emphasis in this chapter is on exploiting tags, and tagging text automatically.

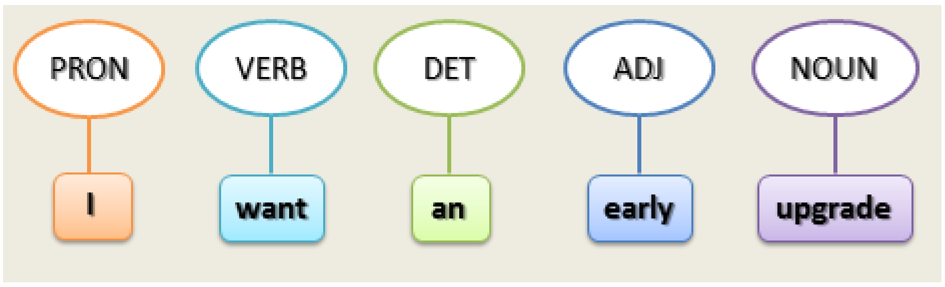


Figure 6.2 POS-Tagging

Lemmatization

Text lemmatization is the process of eliminating redundant prefix or suffix of a word and extract the base word (lemma). The aim of lemmatization, is to reduce inflectional forms to a common base form. As opposed to stemming, lemmatization does not simply chop off inflections. Instead it uses lexical knowledge bases to get the correct base forms of words. Lemmatization tool is present in NLTK library – WordNetLemmatizer.

WORD	LEMMATIZATION
was	be
studying	study
Studies	study

Table 6.1 Lemmatization

6.2 SENTIMENT VADER

VADER **(**Valence Aware Dictionary and sEntiment Reasoner**)** is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. VADER uses a combination of A sentiment lexicon i.e., a list of lexical features (e.g., words) which are generally labelled according to their semantic orientation as either positive or negative.

VADER has been found to be quite successful when dealing with social media texts, NY Times editorials, movie reviews, and product reviews. This is because VADER not only tells about the Positivity and Negativity score but also tells us about how positive or negative a sentiment is.

It is fully open-sourced under the MIT License. The developers of VADER have used Amazon's Mechanical Turk to get most of their ratings

VADER has a lot of advantages over traditional methods of Sentiment Analysis, including:

- It works exceedingly well on social media type text, yet readily generalizes to multiple domains
- It doesn't require any training data but is constructed from a generalizable, valence-based, human-curated gold standard sentiment lexicon
- It is fast enough to be used online with streaming data, and
- It does not severely suffer from a speed-performance tradeoff.

6.3 TF-IDF

Conventionally, histogram of words are the features for the text classification problems. In general, we first build the vocabulary of the corpus and then we generate word count vector from each file which is nothing but frequency of words present in the vocabulary. Most of them will be zero as a single file won't contain all the words in the vocabulary.

For example, suppose we have 500 words in vocabulary. So, each word count vector will contain the frequency of 500 vocab words in the text file. Suppose text in a file was "Get the work done, work done". So, a fixed length encoding will be generated as [0,0,0,0,0,…….0,0,2,0,0,0,……,0,0,1,0,0,…0,0,1,0,0,……2,0,0,0,0,0]. Here, all the word counts are placed at 296th, 359th, 415th, 495th index of 500 length word count vector and the rest are zero.

There are limitations in this conventional approach of extracting features as listed below:

- Frequently occurring words present in all files of corpus irrespective of the sentiment, like in this case, ‘movie’, ‘acting’, etc. will be treated equally like other distinguishing words in the document.
- Stop words will be present in the vocab if not processed properly.
- Rare words or key words which can be distinguishing will not get special weight.

6.3.1 Term Frequency

It increases the weight of the terms (words) that occur more frequently in the document. So it can be defined as tf(t,d) = F(t,d) where F(t,d) is number of occurrences of term ‘t’ in document ‘d’. But practically, it seems unlikely that thirty occurrences of a term in a document truly carry thirty times the significance of a single occurrence. So, in order to make it more pragmatic, we scale tf in logarithmic way so that as the frequency of terms increases exponentially, we will be increasing the weights of terms in additive manner.

$$tf(t,d) = \log\,(F(t,d)$$

6.3.2 Inverse Document Frequency

It diminishes the weight of the terms that occur in all the documents of corpus and similarly increases the weight of the terms that occur in rare documents across the corpus. Basically, the rare keywords get special treatment and stop words/non-distinguishing words get punishment. We define idf as:

$$idf(t,D) = \log\,(\frac{N}{N_{t\in d}})$$

Here, ‘N’ is the total number of files in the corpus ‘D’ and ‘$N_{t \in d}$‘ is number of files in which term ‘t’ is present. By now, we can agree to the fact that tf is a intra-document factor which depends on individual document and idf is a per corpus factor which is constant for a corpus. Finally, we calculate tf-idf as:

$$tf - idf(t,d,D) = tf(t,d)\,.\,idf(t,D)$$

6.4 MACHINE LEARNING MODEL

The machine learning based text classifiers are a kind of supervised machine learning paradigm, where the classifier needs to be trained on some labeled training data before it can be applied to actual classification task. The training data is usually an extracted portion of the original data hand labeled manually. After suitable training they can be used on the actual test data. Different Classifiers can then be used for sentiment classification problem as it can be visualized as a 2-class text classification problem: in positive and negative classes.

This approach needs

- A good classifier, i.e., classifier with maximum accuracy
- A training set for each class

There are various training sets available on Internet such for hotel review dataset, twitter dataset, etc. Class can be Positive, negative. For both the classes we need training data sets. Our dataset contains both positive and negative reviews.

6.4.1 Logistic Regression

Logistic regression is one of the most important analytic tools in the social and natural sciences. In natural language processing, logistic regression is the baseline supervised machine learning algorithm for classification, and also has a very close relationship with neural networks. Logistic regression can be used to classify an observation into one of two classes (like 'positive sentiment' and 'negative sentiment').

The goal of binary logistic regression is to train a classifier that can make a binary decision about the class of a new input observation.

Consider a single input observation x, which we will represent by a vector of features [x_1, x_2,..., x_n]. The classifier output y can be 1 (meaning the observation is a member of the class) or 0 (the observation is not a member of the class). We want to know the probability $P(y = 1|x)$ that this observation is a member of the class. So perhaps the decision is "positive sentiment" versus "negative sentiment", the features represent counts of words in a document, and $P(y = 1|x)$ is the probability that the document has positive sentiment, while and $P(y = 0|x)$ is the probability that the document has negative sentiment.

Logistic regression solves this task by learning, from a training set, a vector of weights and a bias term. Each weight w_i is a real number, and is associated with one of the input features x_i . The weight w_i represents how important that input feature is to the classification decision, and can be positive (meaning the feature is associated with the class) or negative (meaning the feature is not associated with the class). Thus we might expect in a sentiment task the word awesome to have a high positive bias term weight, and abysmal to have a very negative weight. The bias term, also called the intercept intercept, is another real number that's added to the weighted inputs.

$$z = (\sum_{i=1}^{n} w_i x_i) + b$$

$$z = w \cdot x + b$$

Since weights are real-valued, the output might even be negative; z ranges from −∞ to ∞.

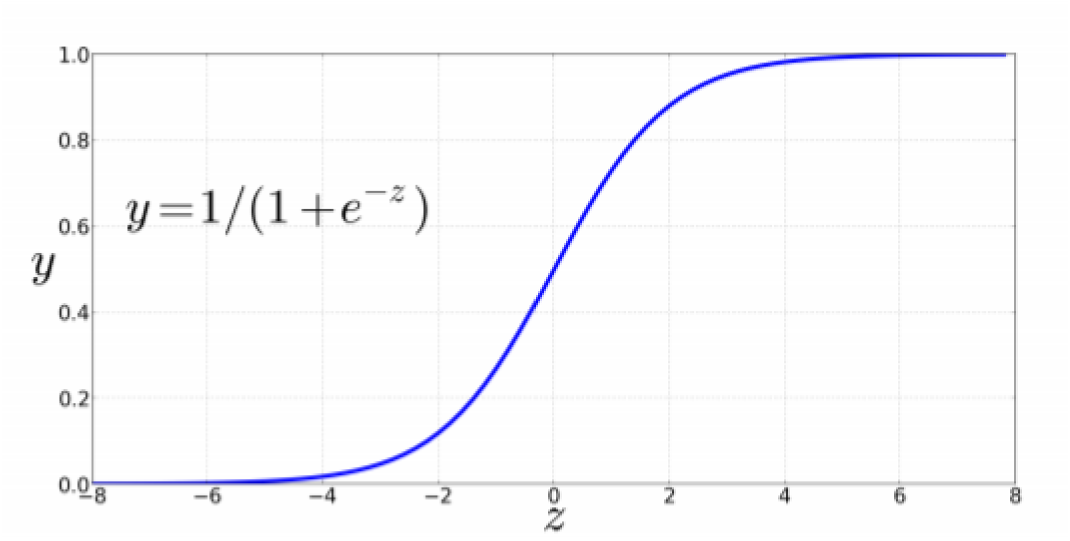


Figure 6.3 The Sigmoid function

To create a probability, we'll pass z through the sigmoid function, σ(z). The sigmoid function (named because it looks like an s) is also called the logistic function, and gives logistic regression its name. The sigmoid has the following equation:

$$y = \sigma(z) = \frac{1}{1 + e^{-z}}$$

The sigmoid has a number of advantages; it takes a real-valued number and maps it into the range [0,1], which is just what we want for a probability. Because it is nearly linear around 0 but has a sharp slope toward the ends, it tends to squash outlier values toward 0 or 1. And it's differentiable.

If we apply the sigmoid to the sum of the weighted features, we get a number between 0 and 1. To make it a probability, we just need to make sure that the two cases, p(y = 1) and p(y = 0), sum to 1. We can do this as follows:

$$P(y = 1) = \sigma(w \cdot x + b)$$

$$P(y = 1) = \frac{1}{1 + e^{-(w \cdot x + b)}}$$

$$P(y = 0) = 1 - \sigma(w \cdot x + b)$$

$$P(y = 0) = 1 - \frac{1}{1 + e^{-(w \cdot x + b)}}$$

$$P(y = 0) = \frac{e^{-(w \cdot x + b)}}{1 + e^{-(w \cdot x + b)}}$$

Now we have an algorithm that given an instance x computes the probability P(y = 1|x). For a test instance x, we say yes if the probability P(y = 1|x) is more than .5, and no otherwise. We call .5 the decision boundary:

$$\hat{y} = \begin{cases} 1 \ if \ P(y = 1|x) > 0.5 \\ 0 \qquad\qquad otherwise \end{cases}$$

6.4.2 Neural Network

Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated.

Neural networks help us cluster and classify. You can think of them as a clustering and classification layer on top of the data you store and manage. They help to group unlabeled data according to similarities among the example inputs, and they classify data when they have a labeled dataset to train on. (Neural networks can also extract features that are fed to other algorithms for clustering and classification; so you can think of deep neural networks as components of larger machine-learning applications involving algorithms for reinforcement learning, classification and regression.)

A typical neural network has anything from a few dozen to hundreds, thousands, or even millions of artificial neurons called **units** arranged in a series of layers, each of which connects to the layers on either side. Some of them, known as **input units**, are designed to receive various forms of information from the outside world that the network will attempt to learn about, recognize, or otherwise process. Other units sit on the opposite side of the network and signal how it responds to the information it's learned; those are known as **output units**. In between the input units and output units are one or more layers of **hidden units**, which, together, form the majority of the artificial brain. Most neural networks are **fully connected**, which means each hidden unit and each output unit is connected to every unit in the layers either side. The connections between one unit and another are represented by a number called a **weight**, which can be either positive (if one unit excites another) or negative (if one unit suppresses or inhibits another). The higher the weight, the more influence one unit has on another. (This corresponds to the way actual brain cells trigger one another across tiny gaps called synapses.)

Information flows through a neural network in two ways. When it's learning (being trained) or operating normally (after being trained), patterns of information are fed into the network via the input units, which trigger the layers of hidden units, and these in turn arrive at the output units. This common design is called a **feedforward network**. Not all units "fire" all the time. Each unit receives inputs from the units to its left, and the inputs are multiplied by the weights of the connections they travel along. Every unit adds up all the inputs it receives in this way and (in the simplest type of network) if the sum is more than a certain **threshold** value, the unit "fires" and triggers the units it's connected to (those on its right).

Neural networks learn things by a feedback process called **backpropagation** (sometimes abbreviated as "backprop"). This involves comparing the output a network produces with the output it was meant to produce, and using the *difference* between them to modify the weights of the connections between the units in the network, working from the output units through the hidden units to the input units—going backward, in other words. In time, backpropagation causes the network to learn, reducing the difference between actual and intended output to the point where the two exactly coincide, so the network figures things out exactly as it should.

6.5 DEPLOYMENT

We deployed our model in two different environments:

- Flask
- Azure

6.5.1 Flask Development

As a Python developer and data scientist, we always have a desire to build web apps to showcase our work. As much as we like to design the front-end, it becomes very overwhelming to take both machine learning and app development.

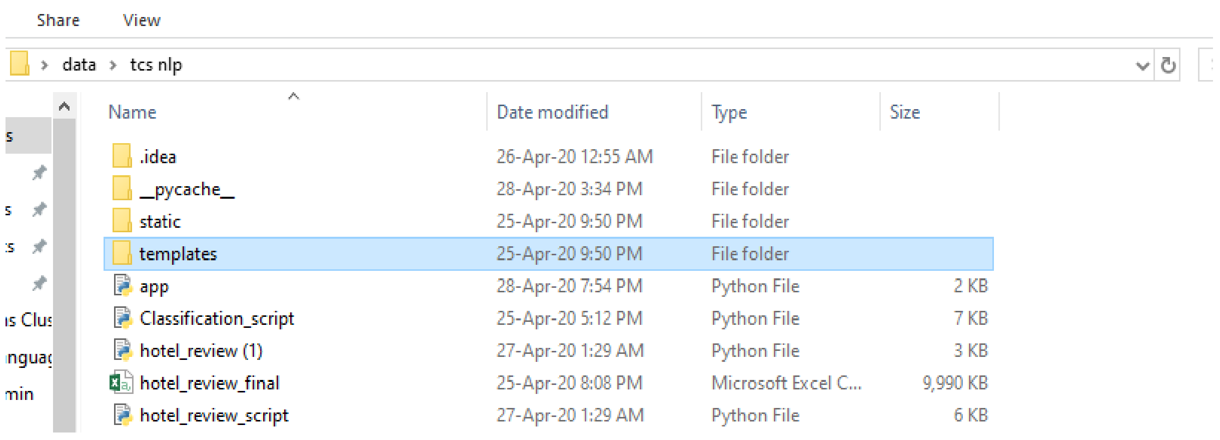
By building a REST API for my model, I could keep my code separate from other developers. There is a clear division of labor here which is nice for defining responsibilities and prevents me from directly blocking teammates who are not involved with the machine learning aspect of the project. Another advantage is that my model can be used by multiple developers working on different platforms, such as web or mobile.

We have built a simple Scikit-Learn model and deploy it as a REST API using Flask Restful.

REST API Guide:

Having prepared the code for classifying Sentiments in the previous section, we will develop a web application that consists of a simple web page with a form field that lets us enter a message. After submitting the message to the web application, it will render it on a new page which gives us a result of either the customer is satisfied or not.

We create a folder for this project and the directory tree inside the folder

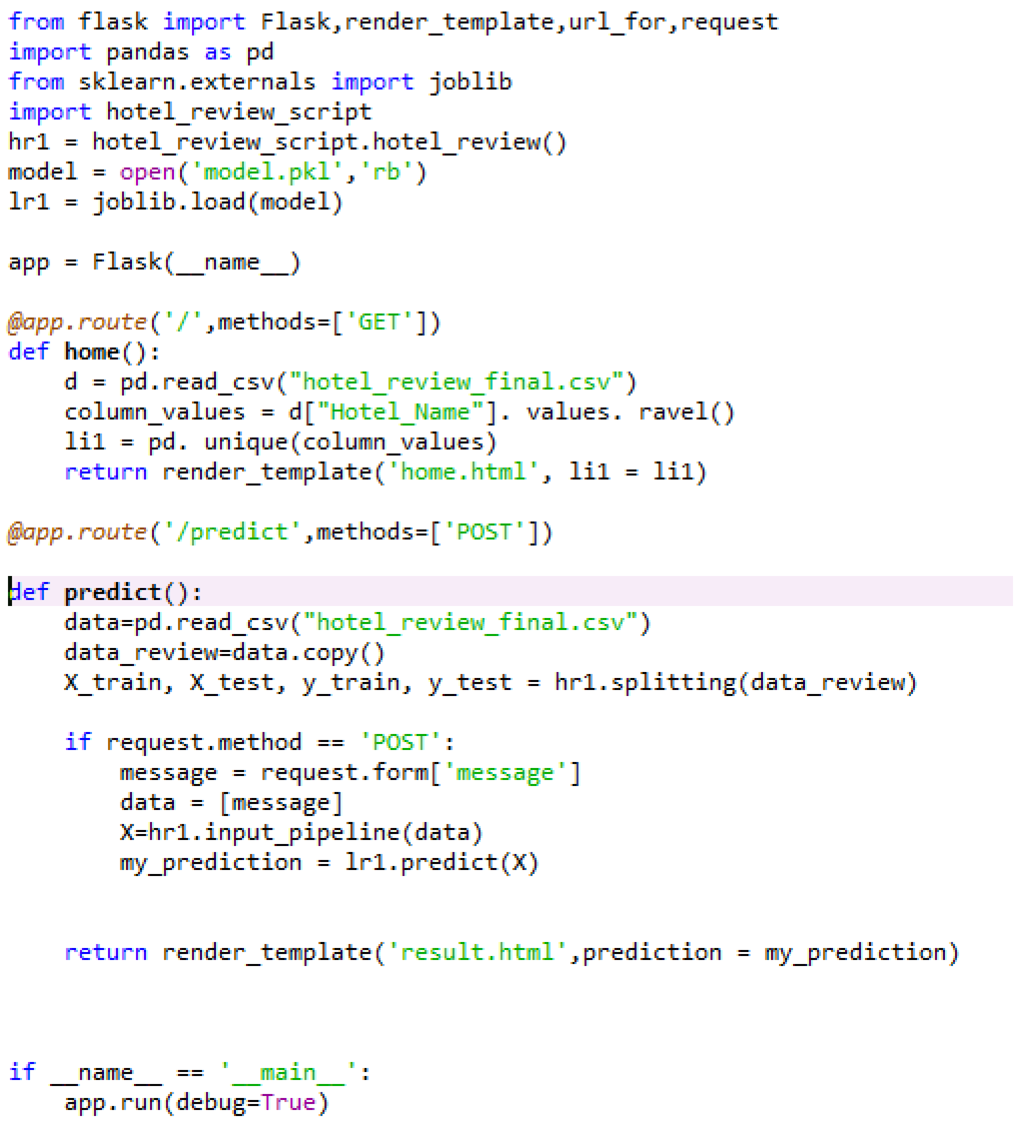


Name	Date modified	Type	Size
.idea	26-Apr-20 12:55 AM	File folder	
__pycache__	28-Apr-20 3:34 PM	File folder	
static	25-Apr-20 9:50 PM	File folder	
templates	25-Apr-20 9:50 PM	File folder	
app	28-Apr-20 7:54 PM	Python File	2 KB
Classification_script	25-Apr-20 5:12 PM	Python File	7 KB
hotel_review (1)	27-Apr-20 1:29 AM	Python File	3 KB
hotel_review_final	25-Apr-20 8:08 PM	Microsoft Excel C...	9,990 KB
hotel_review_script	27-Apr-20 1:29 AM	Python File	6 KB

Figure 6.4 Project Directory

app.py:

The app.py file contains the main code that will be executed by the Python interpreter to run the Flask web application, it included the ML code for classifying SMS messages



```python
from flask import Flask,render_template,url_for,request
import pandas as pd
from sklearn.externals import joblib
import hotel_review_script
hr1 = hotel_review_script.hotel_review()
model = open('model.pkl','rb')
lr1 = joblib.load(model)

app = Flask(__name__)

@app.route('/',methods=['GET'])
def home():
    d = pd.read_csv("hotel_review_final.csv")
    column_values = d["Hotel_Name"]. values. ravel()
    li1 = pd. unique(column_values)
    return render_template('home.html', li1 = li1)

@app.route('/predict',methods=['POST'])

def predict():
    data=pd.read_csv("hotel_review_final.csv")
    data_review=data.copy()
    X_train, X_test, y_train, y_test = hr1.splitting(data_review)

    if request.method == 'POST':
        message = request.form['message']
        data = [message]
        X=hr1.input_pipeline(data)
        my_prediction = lr1.predict(X)


    return render_template('result.html',prediction = my_prediction)



if __name__ == '__main__':
    app.run(debug=True)
```

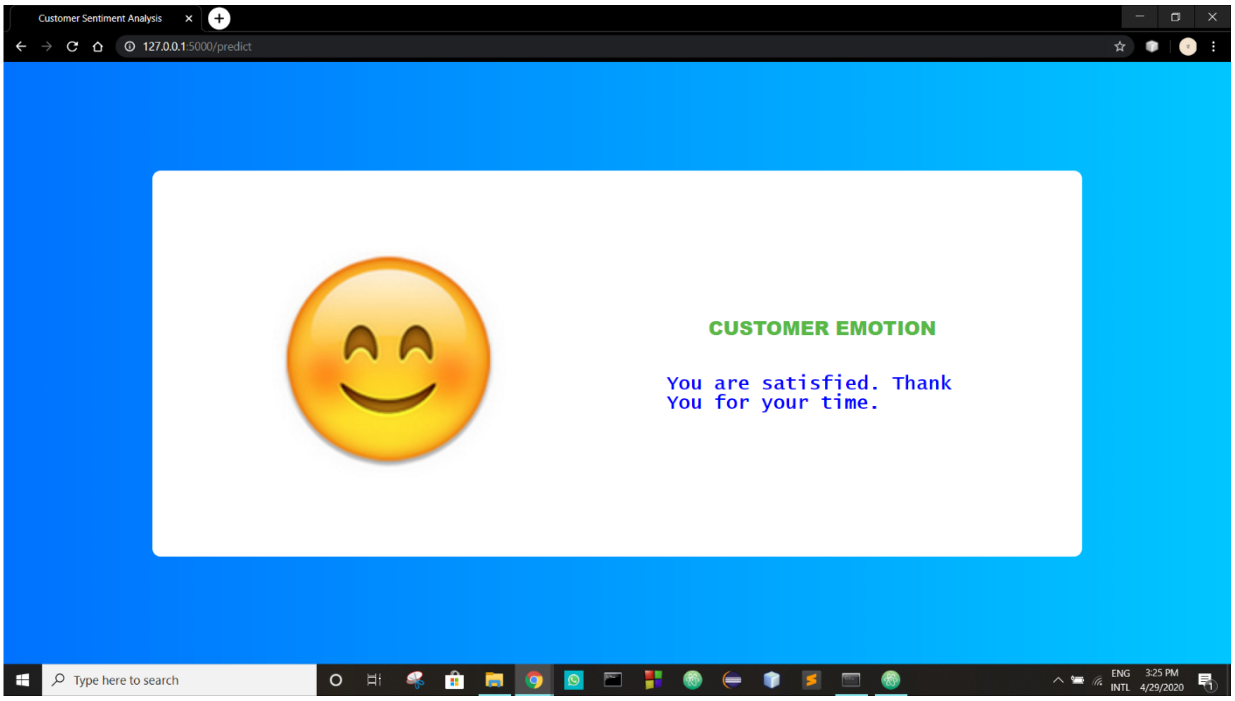
Figure 6.5 app.py

Steps to follow:

- We ran our application as a single module; thus we initialized a new Flask instance with the argument __name__ to let Flask know that it can find the HTML template folder (templates) in the same directory where it is located.

- Next, we used the route decorator (@app.route('/')) to specify the URL that should trigger the execution of the home function.

- Our home function simply rendered the home.html HTML file, which is located in the templates folder.
- Inside the predict function, we access the review data set, pre-process the text, and make predictions, then store the model. We access the new message entered by the user and use our model to make a prediction for its label.
- we used the POST method to transport the form data to the server in the message body. Finally, by setting the debug=True argument inside the app.run method, we further activated Flask's debugger.
- Lastly, we used the run function to only run the application on the server when this script is directly executed by the Python interpreter, which we ensured using the if statement with __name__ == '__main__'.



CUSTOMER EMOTION

You are satisfied. Thank You for your time.

Figure 6.6 Output of the flask deployment

6.5.2 Azure Deployment

Set Up the Development Environment

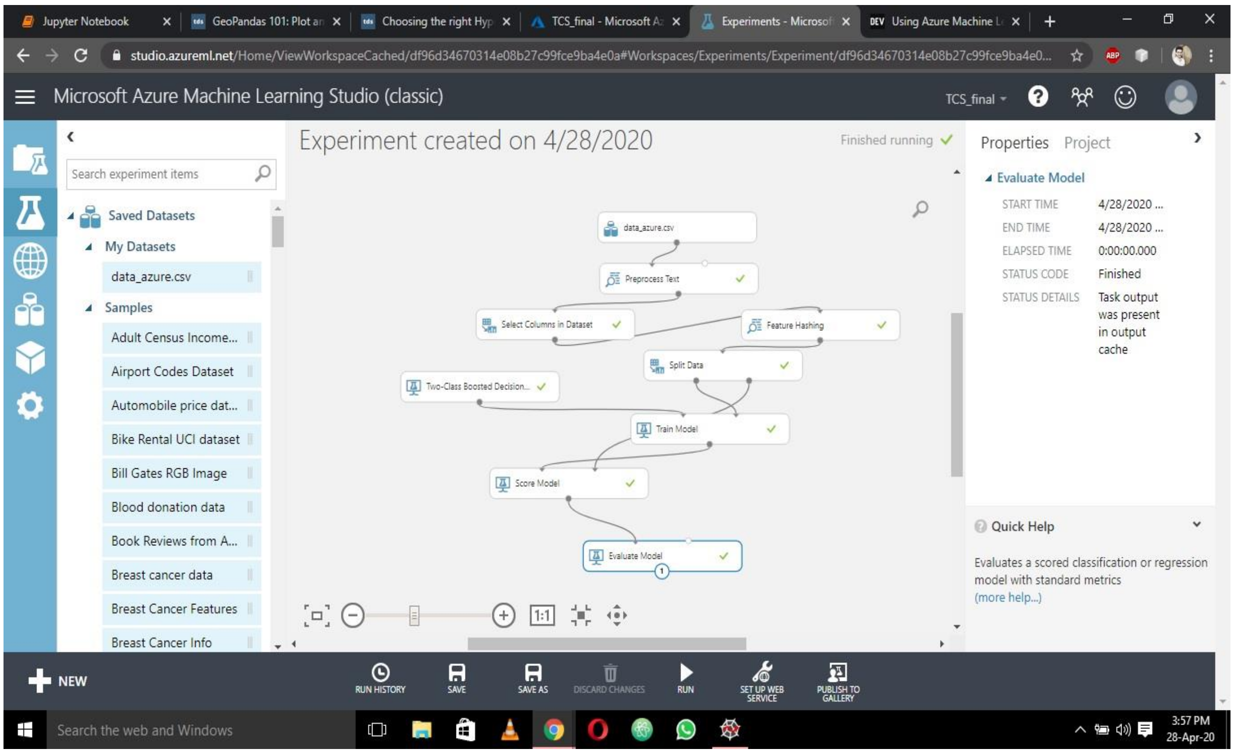
We'll begin by setting up our environment and getting it ready to train an experiment. There are five essential steps to getting our environment ready for an experiment:

1. Initialize the Workspace:

In Azure platform we first have to initialize the workspace for our project development we have used machine Learning Classis Workspace

2. Create Experiment:

After creating the workspace, we create an Experiment. First, we have to import our hotel review Dataset. After importing dataset, we can apply machine learning techniques simply by drag and drop mechanism



Evaluate Model

START TIME	4/28/2020 ...
END TIME	4/28/2020 ...
ELAPSED TIME	0:00:00.000
STATUS CODE	Finished
STATUS DETAILS	Task output was present in output cache

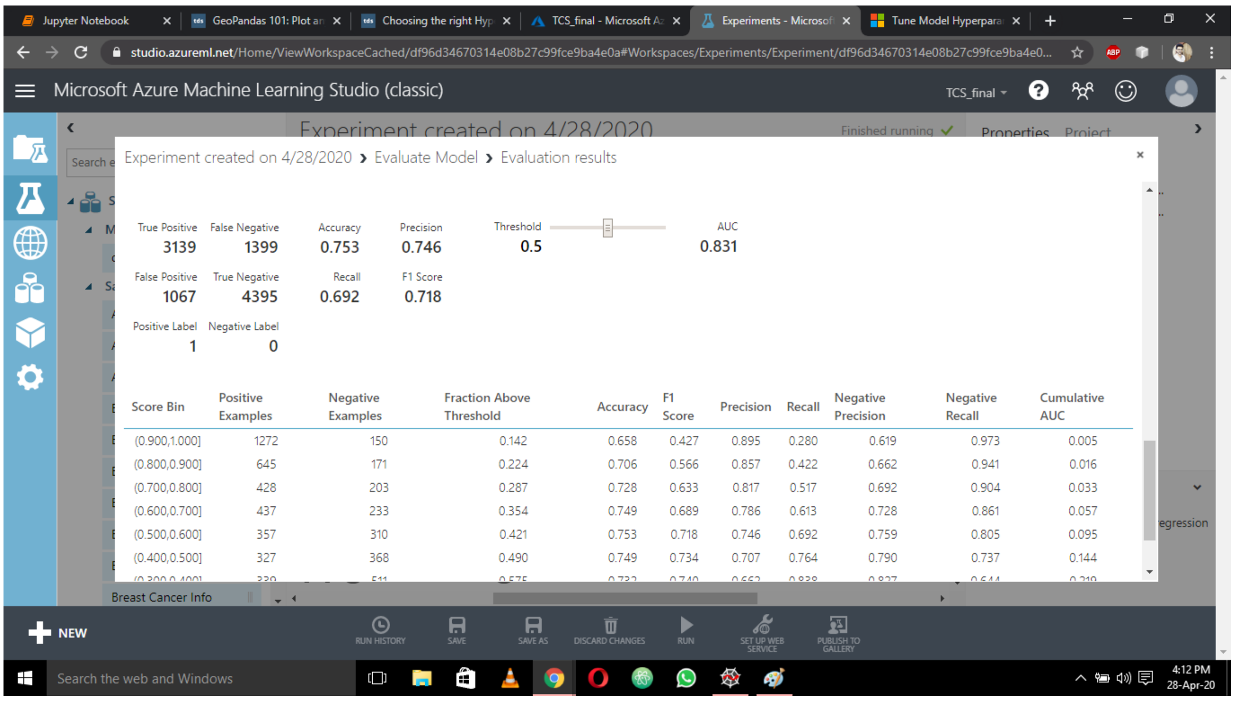
Quick Help

Evaluates a scored classification or regression model with standard metrics (more help...)

Figure 6.7 Azure Model

3. Checking Accuracy:

After creating the entire model, we check the accuracy of the model


Experiment created on 4/28/2020 › Evaluate Model › Evaluation results

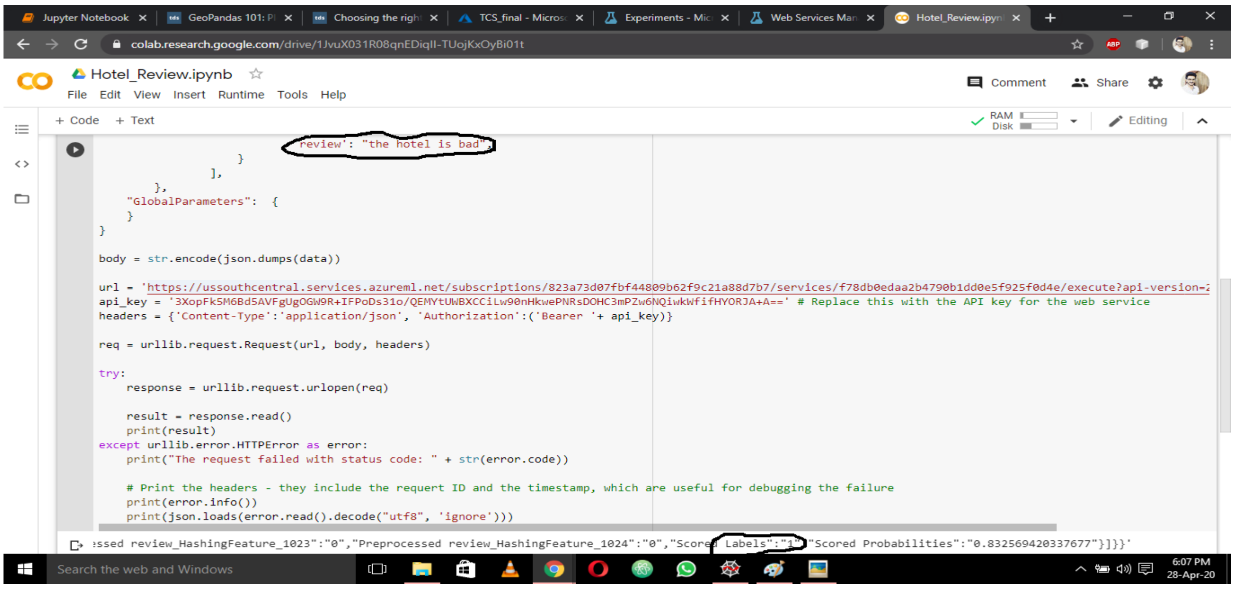
True Positive	False Negative	Accuracy	Precision	Threshold	AUC
3139	1399	0.753	0.746	0.5	0.831
False Positive	**True Negative**	**Recall**	**F1 Score**		
1067	4395	0.692	0.718		
Positive Label	**Negative Label**				
1	0				

Score Bin	Positive Examples	Negative Examples	Fraction Above Threshold	Accuracy	F1 Score	Precision	Recall	Negative Precision	Negative Recall	Cumulative AUC
(0.900,1.000]	1272	150	0.142	0.658	0.427	0.895	0.280	0.619	0.973	0.005
(0.800,0.900]	645	171	0.224	0.706	0.566	0.857	0.422	0.662	0.941	0.016
(0.700,0.800]	428	203	0.287	0.728	0.633	0.817	0.517	0.692	0.904	0.033
(0.600,0.700]	437	233	0.354	0.749	0.689	0.786	0.613	0.728	0.861	0.057
(0.500,0.600]	357	310	0.421	0.753	0.718	0.746	0.692	0.759	0.805	0.095
(0.400,0.500]	327	368	0.490	0.749	0.734	0.707	0.764	0.790	0.737	0.144

Figure 6.8 Accuracy of the model

4. Web Service:

After creating web service we deploy our model in azure platform.


Hotel_Review.ipynb

```
                    review': "the hotel is bad",
                }
            ],
        },
    "GlobalParameters":  {
    }
}

body = str.encode(json.dumps(data))

url = 'https://ussouthcentral.services.azureml.net/subscriptions/823a73d07fbf44809b62f9c21a88d7b7/services/f78db0edaa2b4790b1dd0e5f925f0d4e/execute?api-version=2
api_key = '3XopFk5M6Bd5AVFgUgOGW9R+IFPoDs31o/QEMYtUWBXCCiLw90nHkwePNRsDOHC3mPZw6NQiwkWfifHYORJA+A==' # Replace this with the API key for the web service
headers = {'Content-Type':'application/json', 'Authorization':('Bearer '+ api_key)}

req = urllib.request.Request(url, body, headers)

try:
    response = urllib.request.urlopen(req)

    result = response.read()
    print(result)
except urllib.error.HTTPError as error:
    print("The request failed with status code: " + str(error.code))

    # Print the headers - they include the requert ID and the timestamp, which are useful for debugging the failure
    print(error.info())
    print(json.loads(error.read().decode("utf8", 'ignore')))
```

```
ssed review_HashingFeature_1023":"0","Preprocessed review_HashingFeature_1024":"0","Scored Labels":"1","Scored Probabilities":"0.832569420337677"}]}}'
```

Figure 6.9 Web service

CHAPTER 7

N-GRAM

N-GRAM

Models that assign probabilities to sentences and sequences of words are called Statistical Language Models. One such model is n-gram. N-gram is the sequence of N words, by that notion, a 2-gram (or bigram) is a two-word sequence of words like “please turn”, “turn your”, or ”your homework”, and a 3-gram (or trigram) is a three-word sequence of words like “please turn your”, or “turn your homework”.

N-gram models are widely used in probability, communication theory and computational linguistics (for instance, statistical natural language processing) Two benefits of *n*-gram models are simplicity and scalability – with larger *n*, a model can store more context with a well-understood space-time trade-off, enabling small experiments to scale up efficiently. N-grams can also be used for efficient approximate matching. By converting a sequence of items to a set of *n*-grams, it can be embedded in a vector space, thus allowing the sequence to be compared to other sequences in an efficient manner.

Intuitive Formulation

The Intuitive formulae is given by the equation P(w | h), the probability of word *w*, given some history, *h*. For example,

P(the| its water is so transparent

Here,
w → The
h → its water is so transparent that

Now it would be very time consuming if we follow this approach for a significantly large corpus. This shortcoming and ways to decompose the probability function using the chain rule serves as the base intuition of the N-gram model. The chain rule is applied to compute the joint probability of words in sentence.

$$\mathbf{P(w_1\, w_2\, \ldots\, w_n) = P(w_i \mid w_1\, w_2 \ldots w_{i-1})}$$

P(“its water is so transparent”) = P(its) × P(water | its) × P(is | its water) × P(so | its water is) × P(transparent |Its water is so)

A model that simply relies on how often a word occurs without looking at previous words is called unigram. If a model considers only the previous word to predict the current word, then it's called bigram. If two previous words are considered, then it's a trigram model.

The Bigram Model

the bigram model approximates the probability of a word given all the previous words by using only the conditional probability of one preceding word. In other words, you approximate it with the probability: P(the | that)

And so, when we use a bigram model to predict the conditional probability of the next word, we are thus making the following approximation:

$$\mathbf{P(w_i \mid w_1\ w_2 \dots w_{i-1}) = P(w_i \mid w_i\text{-}1)}$$

This assumption that the probability of a word depends only on the previous word is also known as **Markov** assumption.

Estimating bigram probabilities

The probabilities of the bigram are estimated using the maximum likelihood estimate given by the formulae:

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1},\ w_i)}{count(w_{i-1})}$$

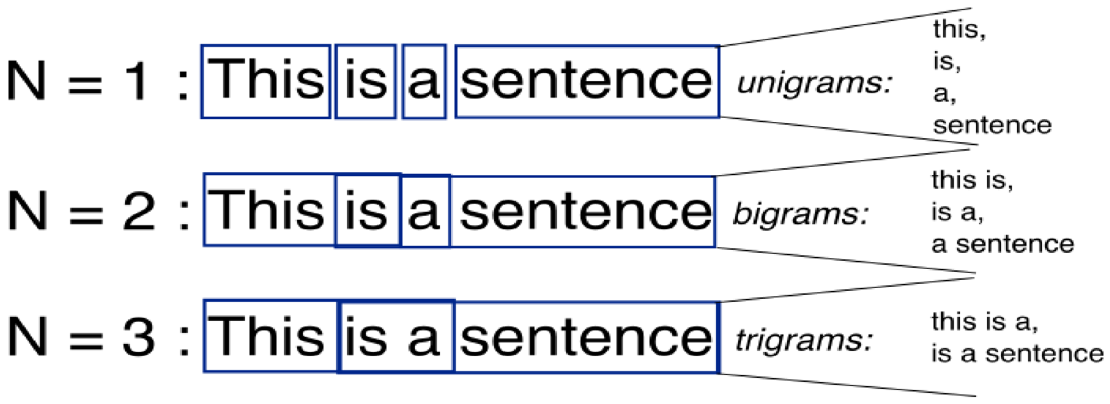


Figure 7.1 N-gram

The bigram can be generalized to the **trigram model** which looks two words into the past and can thus be further generalized to the **N-gram model.**

The N-gram model is used in the form of trigrams in the project. The reviews of the customers were analyzed thereby to know the various positive and negative aspects which were frequently talked about. N-gram models extract the words in the sequence form and can help in getting the meaningful insights from the text. N-grams has been widely used The conflicts faced while dealing with the text data specially with the customer reviews and feedbacks is that often a single word isn't enough to depict the positive or negative sentiments. The challenge is distinguishing sentences like "The hotel bed was not very good", here bigrams can extract "very good", depicting that the customer is satisfied. The trigram can help in this scenario where we can look at past two words "not very good", thereby giving negative sentiment.

CHAPTER 8

TESTING

TESTING

Software testing is a process, to evaluate the functionality of a software application with an intent to find whether the developed software met the specified requirements or not and to identify the defects to ensure that the product is defect-free to produce the quality product. Testing is conducted at the phase level in the software development life cycle or the module level in the program code. Software testing comprises of Validation and Verification.

8.1 TESTING OBJECTIVES

Software testing is a critical phase under software quality assurance. It demonstrates the ultimate review of specification, design, and code generation. Once the source code has been created, software must be tested to correct maximum possible errors, before they can be delivered. To accomplish this, various software testing methods are used. These methods provide systematic guidance for designing tests that must do the following –

- Exercise the internal logic of the software components.
- Exercise program's input and output domains, thus uncovering errors in program function, behaviour, and performance.

The primary objectives of software testing are:

1. **Verification:** The main objective of testing is verification, which allows testers to confirm that the software meets the requirements stated in synopsis before the inception of the whole project. These requirements guide the design and development of the software, thus are required to be followed rigorously. Moreover, compliance with these requirements and specifications is important for the success of the project.
2. **Validation:** It usually confirms that the software performs as expected by the system. Validation involves comparing the final output with the expected output and then making necessary changes if there is a difference between the two.
3. **Defects:** The most important purpose of testing is to find different defects in the software which in turn is used to prevent failure, crash during implementation. Defects if left undetected or unattended can harm the functioning of the software and can lead to loss of resources, money, and reputation of the client. Therefore, software testing is executed regularly during each stage of software development to find defects of all kinds. The ultimate source of these defects can be traced back to a fault introduced during the specification, design, development, or programming phases of the software.

4. **Providing Information:** With the assistance of reports generated during the process of software testing, testers can accumulate a variety of information related to the software and the steps taken to prevent its failure. These, then can be shared with all the stakeholders of the project for better understanding of the project as well as to establish transparency between members.
5. **Preventing Defects:** During the process of testing the aim of testes to identify defects and prevent them from occurring aging in the future. To accomplish this goal, software is tested rigorously by independent testers, who are not responsible for software development.
6. **Analysis:** Testing helps improve the quality of the software by constantly measuring and verifying its design and coding. Additionally, various types of testing techniques are used by testers, which help them achieve the desired software quality.
7. **Compatibility:** It helps validate application's compatibility with the implementation environment, various devices, Operating Systems, user requirements, among other things.
8. **For Optimum User Experience:** Easy software and application accessibility and optimum user experience are the most important requirements that are need to be accomplished for the success of any project as well as to increase the revenue of the client. Therefore, to ensure this software is tested again and again by the testers with the assistance of stress testing, load testing, spike testing, etc.
9. **Verifying Performance & Functionality:** It ensures that the software has superior performance and functionality. This is mainly verified by placing the software under extreme stress to identify and measure it's all possible failure modes. To ensure this, performance testing, usability, functionality testing, etc. is executed by the testers.

8.2 TESTING SCOPE

In this highly competitive world, every industry is struggling hard to work under strict deadlines and high work pressure. The Industry of Information Technology is not an exception in any way. In such a scenario, the testing teams need to adopt the methods of agile testing for keeping up well with the requirements. The testing teams must be alert that they would have to invest good time in developing the features and that they would not be blessed with unlimited space for the quality assurance processes. Hence, it is very clear that establishing test scope within a limited time can be a bit challenging. So, it is very important to be conscious about building a brilliant test scope strategy for making sure that everyone on the team is working collectively for the same goal. It also ensures to provide a good quality product to the customer which creates a good image of the organization in the market.

8.3 TESTING PRINCIPLES

Software testing is one of the major phases of software development. Testing of software is exceptionally imaginative and an intellectual task for the testers to perform. Testing of software or applications follow some principles that are mentioned in this chapter. These principles also play a significant role for a software tester to test the project.

There are seven principles in testing:

1. **Testing shows presence of defects:** Testing reduces the presence of defects. Testing ensures that defects are present but it can’t prove that software is defects free. Even multiple testing can never ensure that software is 100% bug-free. Testing can reduce the number of defects but not remove all defects.
2. **Exhaustive testing is not possible:** It is the process of testing the functionality of a software in all possible inputs (valid or invalid) and pre-conditions which is known as exhaustive testing. The software can never test for every test case. It can test only some test cases and assume that software is correct and it will produce the correct output in every test case. In testing every test case, it will take more cost, effort, etc. and which is impractical.
3. **Early Testing**: To find the defect in the software, early test activity shall be started. The defect detected in early phases of SDLC will be very less expensive. For better performance of software, software testing will start at the initial phase i.e. testing will perform at the requirement analysis phase.
4. **Defect clustering:** In a project, a small number of the module can contain most of the defects. Pareto Principle to software testing states that 80% of software defects come from 20% of modules.
5. **Pesticide paradox:** Repeating the same test cases again and again will not find new bugs. So, it is necessary to review the test cases and add or update test cases to find new bugs.
6. **Testing is context dependent:** Testing approach depends on context of software developed. Different types of software need to perform different types of testing. For example, the testing of an e-commerce site is different from the testing of an Android application.
7. **Absence of errors fallacy:** If a built software is 99% bug-free but it does not follow the user's requirement then it is unusable. It is not only necessary that software is 99% bug-free but it is also mandatory to fulfil all the customer requirements.

8.4 TESTING METHODS

In this project, many methods for testing are used. These are

- Black-Box Testing
- White Box Testing
- Unit Testing
- Integration testing
- System testing

8.4.1 Black-Box Testing:

The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon. Black-box testing focuses on the functional requirements of the software. That is, testing enables the software engineer to derive sets of Input conditions that will fully exercise all functional requirements for a program.

It finds errors in following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structures
- Behaviour or performance errors
- Initialization and termination errors

Black box testing is applied on all forms for example login form, update, add, delete form, information filling forms etc.

8.4.2 White Box Testing

White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also known as glass testing or open-box testing. To perform white box testing on an application, the tester needs to possess knowledge of the internal working of the code. The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

8.4.3 Unit Testing

This type of testing is performed by developers before the setup is handed over to the testing team to formally execute the test cases. Unit testing is performed by the respective developers on the individual units of source code assigned areas. The developers use test data which is different from the test data of the quality assurance team. The main goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality. Testing cannot find each and every bug in an application. It is impossible to evaluate every execution path in software application. The same is the case with unit testing. There is a limit to the number of scenarios and test data that a developer can use to verify a source code. After having exhausted all the options, there is no choice but to stop unit testing and merge the code segment with other units.

8.4.4 Integration Testing

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. Integration testing is conducted to evaluate the compliance of a system or component with specified functional requirements. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

8.4.5 System Testing

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input. The main goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behaviour of a component or a system when it is tested. System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both. System testing tests the design and behaviour of the system and also the expectations of the customer.

8.5 TEST CASES

To test the project, several test cases were defined. We reiterated this whole process many times until satisfactory results were obtained.

Case 1: Checking for the correct test size of data (as per the requirement).

Case 2: Using Logistic Regression algorithm with the dataset.

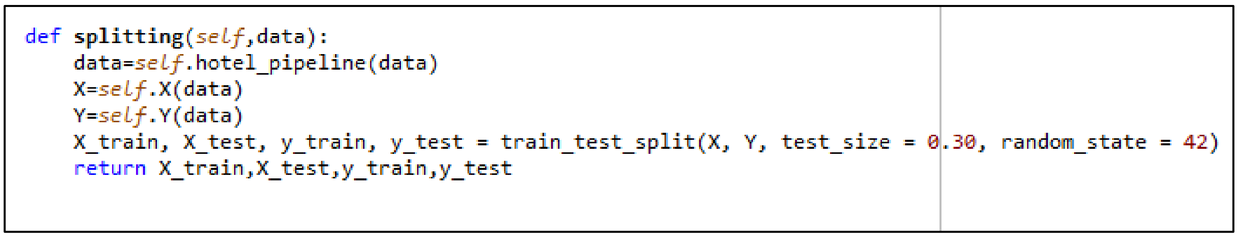
Case 3: Applying dimensionality reduction and hyper-parameter tuning

Using the combination of cases in Case 1, Case 2 and Case 3 we tested the software for different algorithms with different data set combination.

It is important here to note that given any set of conditions the algorithm reached its maximum efficiency in its third run

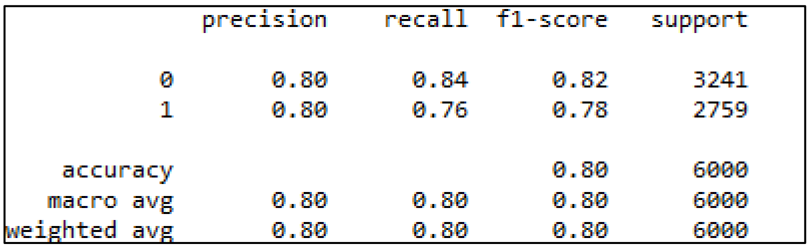
Case 1: Checking for the correct test size of data:

We have tested our Model with different test size of dataset



```
def splitting(self,data):
    data=self.hotel_pipeline(data)
    X=self.X(data)
    Y=self.Y(data)
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.30, random_state = 42)
    return X_train,X_test,y_train,y_test
```

Figure 8.1 code for splitting



```
              precision    recall  f1-score   support

           0       0.80      0.84      0.82      3241
           1       0.80      0.76      0.78      2759

    accuracy                           0.80      6000
   macro avg       0.80      0.80      0.80      6000
weighted avg       0.80      0.80      0.80      6000
```

Figure 8.2 Confusion matrix

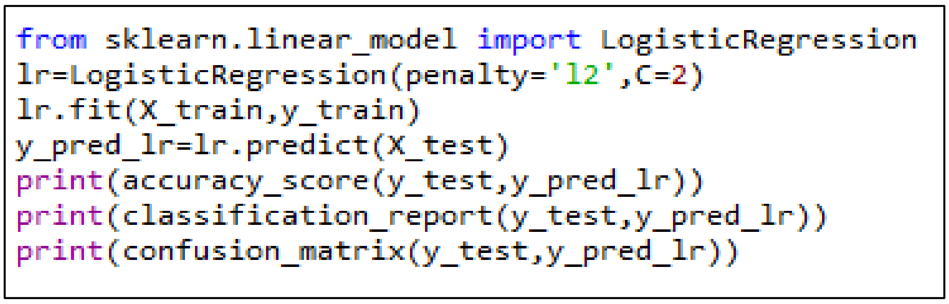
This is the best size of the dataset on which we are getting the best results.

Case 2: Using Logistic Regression algorithm with the dataset:

We have applied different Machine Learning algorithms on our datasets including

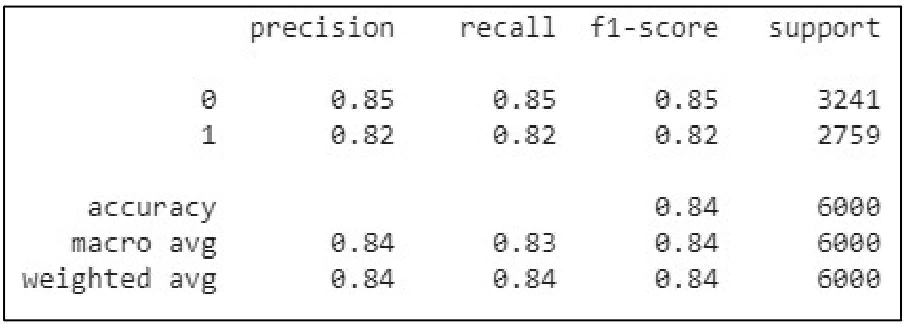
- Naïve Bayes
- Support vector Machine
- Decision Tree
- Random Forest
- Logistic Regression
- XGBoost

Among all the algorithms the best result was given by Logistic Regression



```python
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression(penalty='l2',C=2)
lr.fit(X_train,y_train)
y_pred_lr=lr.predict(X_test)
print(accuracy_score(y_test,y_pred_lr))
print(classification_report(y_test,y_pred_lr))
print(confusion_matrix(y_test,y_pred_lr))
```

Figure 8.3 Code for Logistic regression

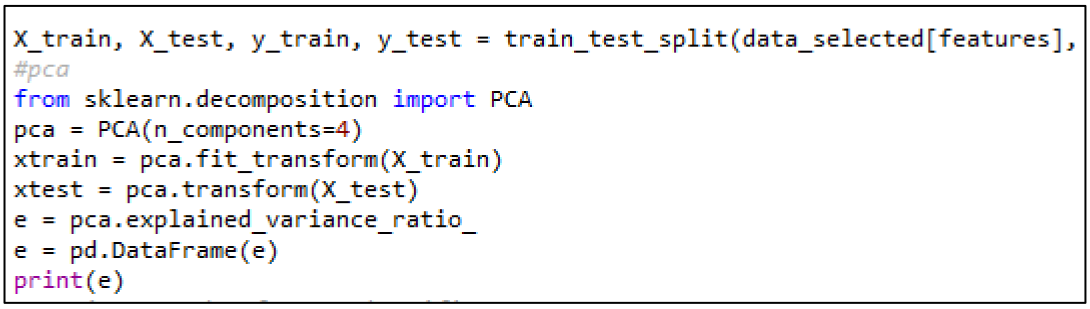


```
              precision    recall  f1-score   support

           0       0.85      0.85      0.85      3241
           1       0.82      0.82      0.82      2759

    accuracy                           0.84      6000
   macro avg       0.84      0.83      0.84      6000
weighted avg       0.84      0.84      0.84      6000
```

Figure 8.4 Confusion matrix for Logistic Regression

Case 3: Applying dimensionality reduction and hyper-parameter tuning

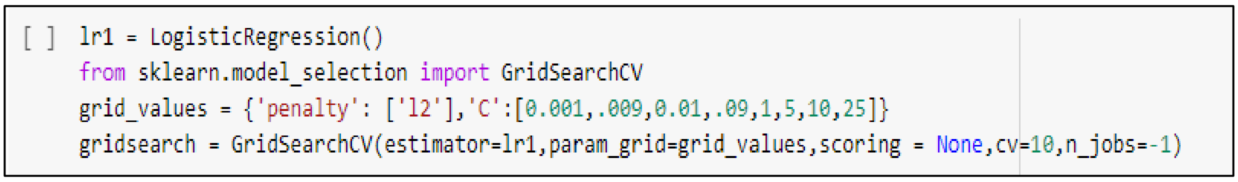
We have applied principal component analysis for dimensionality reduction



```
X_train, X_test, y_train, y_test = train_test_split(data_selected[features],
#pca
from sklearn.decomposition import PCA
pca = PCA(n_components=4)
xtrain = pca.fit_transform(X_train)
xtest = pca.transform(X_test)
e = pca.explained_variance_ratio_
e = pd.DataFrame(e)
print(e)
```

Figure 8.5 applying PCA

We have also used Grid Search method for hyper-parameter tuning in our Logistic Regression Model



```
[ ]  lr1 = LogisticRegression()
     from sklearn.model_selection import GridSearchCV
     grid_values = {'penalty': ['l2'],'C':[0.001,.009,0.01,.09,1,5,10,25]}
     gridsearch = GridSearchCV(estimator=lr1,param_grid=grid_values,scoring = None,cv=10,n_jobs=-1)
```

Figure 8.6 hyperparameter tuning

CHAPTER 9

RESULTS AND DISCUSSIONS

RESULTS AND DISCUSSIONS

The project work depicts the analysis of customer reviews and their categorization into sentiments. The preprocessing of text was done to remove noise from the data and VADER from the NLTK library was applied to the preprocessed text to determine the sentiments of the customer reviews. The reviews were labelled as positive, negative and neutral. Various models were applied to train the classifier. The several classifiers applied include Logistic Regression, Decision Trees, Random Forests, SVM RBF, Naïve Bayes and Gradient Boosting. The best accuracy achieved was in Logistic regression among the machine learning models applied for classification. The accuracy achieved by the various models are depicted below:

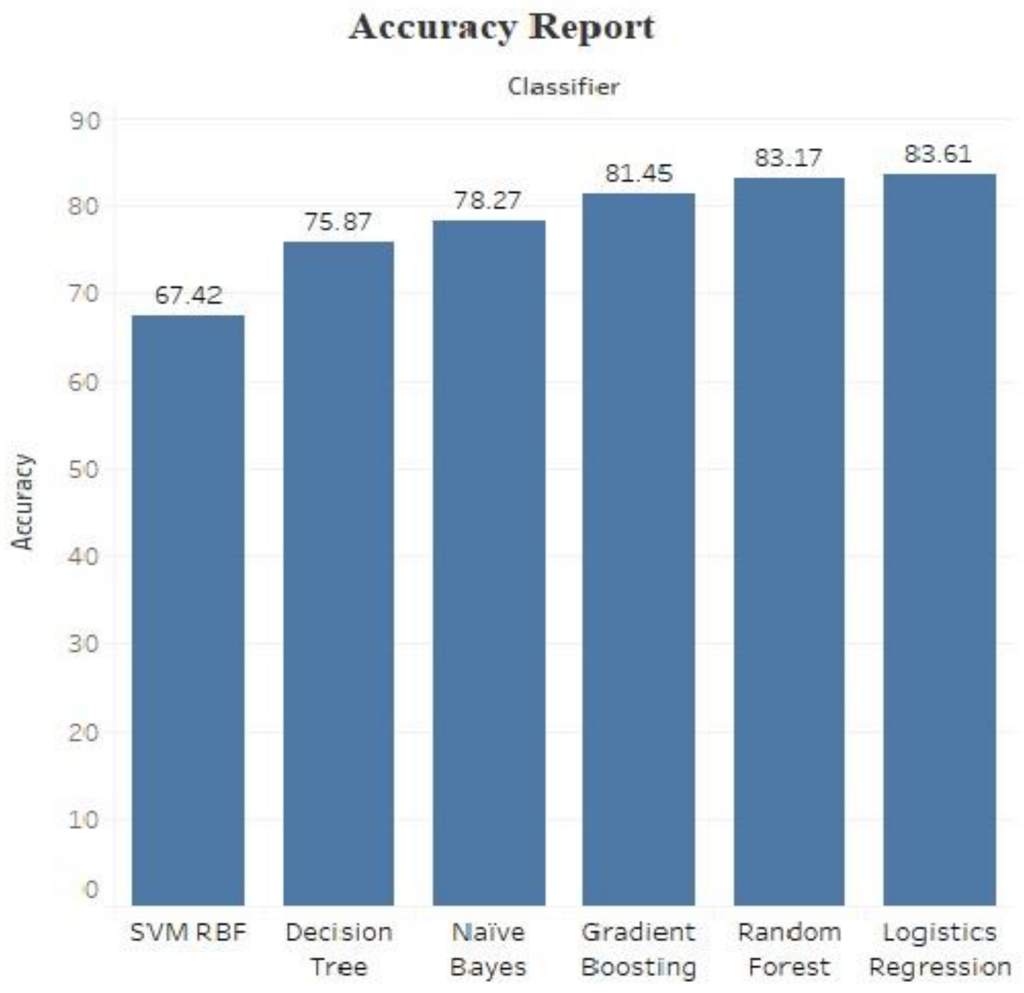
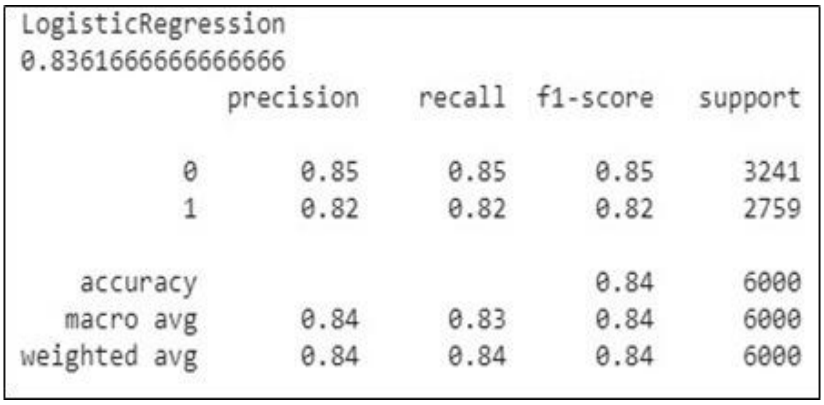


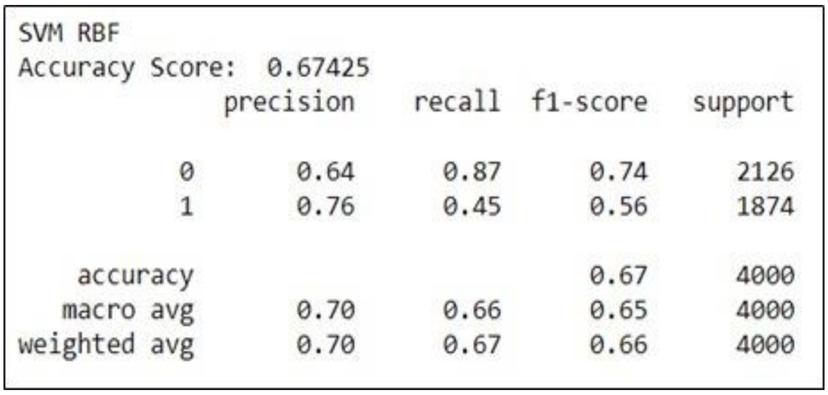
Figure 9.1 Accuracy Comparison

The precision, recall and f1-score were acceptable and the Logistic regression gave the best values among the other classifiers applied. The classification report of the several classifiers applied is shown below:



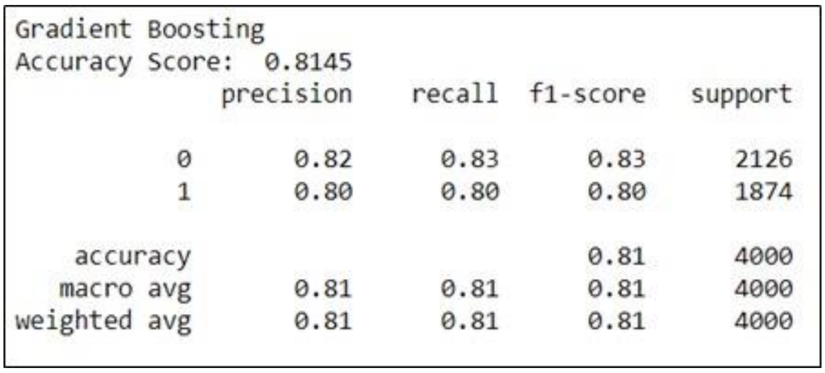
```
LogisticRegression
0.8361666666666666
              precision    recall  f1-score   support

           0       0.85      0.85      0.85      3241
           1       0.82      0.82      0.82      2759

    accuracy                           0.84      6000
   macro avg       0.84      0.83      0.84      6000
weighted avg       0.84      0.84      0.84      6000
```

Figure 9.2 Logistic Regression



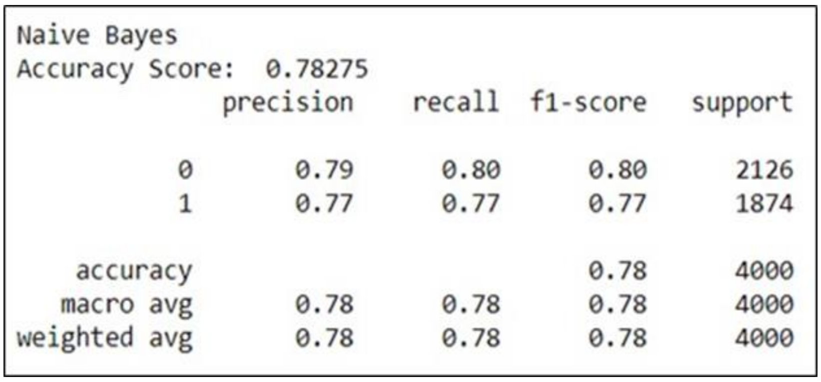
```
SVM RBF
Accuracy Score:  0.67425
              precision    recall  f1-score   support

           0       0.64      0.87      0.74      2126
           1       0.76      0.45      0.56      1874

    accuracy                           0.67      4000
   macro avg       0.70      0.66      0.65      4000
weighted avg       0.70      0.67      0.66      4000
```

Figure 9.3 Support Vector



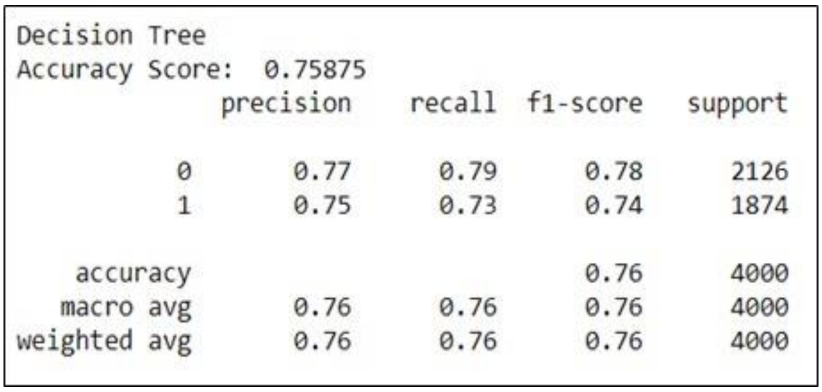
```
Gradient Boosting
Accuracy Score:  0.8145
              precision    recall  f1-score   support

           0       0.82      0.83      0.83      2126
           1       0.80      0.80      0.80      1874

    accuracy                           0.81      4000
   macro avg       0.81      0.81      0.81      4000
weighted avg       0.81      0.81      0.81      4000
```

Figure 9.4 Gradient Boosting



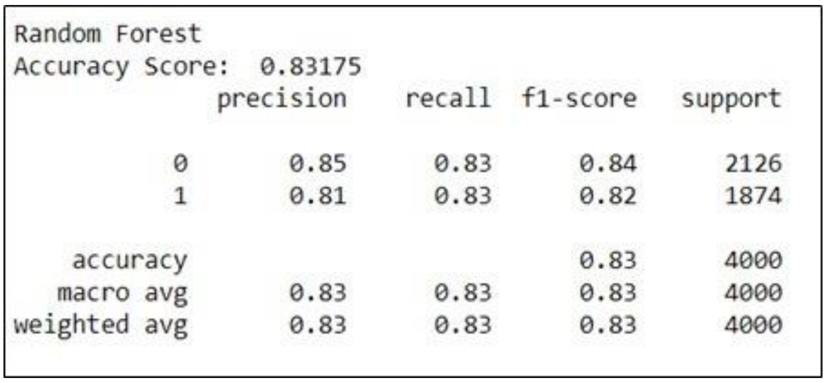
```
Naive Bayes
Accuracy Score:  0.78275
              precision    recall  f1-score   support

           0       0.79      0.80      0.80      2126
           1       0.77      0.77      0.77      1874

    accuracy                           0.78      4000
   macro avg       0.78      0.78      0.78      4000
weighted avg       0.78      0.78      0.78      4000
```

Figure 9.5 Naïve Bayes



```
Decision Tree
Accuracy Score:  0.75875
              precision    recall  f1-score   support

           0       0.77      0.79      0.78      2126
           1       0.75      0.73      0.74      1874

    accuracy                           0.76      4000
   macro avg       0.76      0.76      0.76      4000
weighted avg       0.76      0.76      0.76      4000
```

Figure 9.6 Decision Tree



```
Random Forest
Accuracy Score:  0.83175
              precision    recall  f1-score   support

           0       0.85      0.83      0.84      2126
           1       0.81      0.83      0.82      1874

    accuracy                           0.83      4000
   macro avg       0.83      0.83      0.83      4000
weighted avg       0.83      0.83      0.83      4000
```

Figure 9.7 Random Forest

These classifiers trained the model very well to classify the sentiments and predictions obtained were quite accurate. However a better accuracy than logistic regression was achieved using deep learning approach. The neural network was applied and the accuracy achieved with this model was 91 percent which was the best among all the approaches followed. The n-gram model was also applied in order to determine the negative as well as positive aspects of the hotels in the datasets. The trigram approach enabled to extract the trigrams from the customer reviews thereby retrieving the meaningful insights from the reviews

CHAPTER 10
FUTURE SCOPE

FUTURE SCOPE

The project aimed at classifying the reviews. The future scope involves the usage of these reviews to rate the hotels. The ratings could be generated based on the opinions of the customers and the number of positive and negative reviews received by the hotel. Therefore, a Universal rating system can be created that could enable customers to identify top hotels to enhance their stay experience. Further the Topic modelling approach like LDA can be applied on the hotels to recommend the hotels based on specific characteristics. The customers demand could be recognized say "A hotel with the good scenic view" and based on that aspect the hotels rated with the top scores could be recommended thereby enhancing the user experience. The customer can have a personalised recommendation generated for them while taking into considerations the top hotels rated at the specified destination. A further extension of this project includes a collaborative rating system which could enhance the recommendations to a greater extent.

CHAPTER 11
CONCLUSION

CONCLUSION

The customer reviews were pre-processed and turned into quantitative measurements. The text classification model was created to categorise the reviews into sentiment component. The model thus built was able to classify the reviews into positive, negative and neutral with an acceptable accuracy. The various classification models such as Logistic Regression, Decision trees, Random forests, Naïve Bayes, SVM, gradient Boosting and finally deep learning approach was applied and the one with the improved accuracy in classification was chosen. The deep learning approach gave the best accuracy in classifying the reviews. Furthermore, the negative and positive aspects of the hotels were extracted to determine the services for which hotels has been reviewed negatively. The trigrams were applied to get the meaningful insights about the reviews and to extract the good and bad services of the hotels. The most frequently talked aspects in the reviews were extracted. The model helps hotels to know several aspects where they are lacking and to further improve their services. This would enable hotels to increase the number of visitors and thereby increase their business value.

CHAPTER 12
BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Nibedita Panigrahi, Asha T; “RHALSA: Ranking Hotels using Aspect Level Sentiment Analysis”.

[2] George Markopoulos, George K. Mikros, Anastasia Iliadi; “Sentiment Analysis of Hotel Reviews in Greek: A Comparison of Unigram Features Article”.

[3] A.Jeyapriya, C.S.KanimozhiSelvi; “Extracting Aspects and Mining Opinions in Product Reviews using Supervised Learning Algorithm”, in 2015, IEEE.

[4] Gaurav Dubey, Ajay Rana, Naveen Kumar Shukla; “User Reviews Data Analysis using Opinion Mining on Web”; In 2015, IEEE.

[5] Sumbal Riaz, Mehvish Fatima, M. Kamran, M. Wasif Nisar; “Opinion mining on large scale data using sentiment analysis and k-means clustering”.

[6] Abdullah Aziz, Sharfuddin, Md. Nafis Tihami, Md. Saiful Islam; “A Deep Recurrent Neural Network with BiLSTM model for Sentiment Classification”; In International Conference on Bangla Speech and Language Processing(ICBSLP), 21-22 September, 2018.

[7] Wei Xue, Tao Li, Naphtali Rishe; “Aspect identification and ratings inference for hotel reviews”.

[8] Joscha Markle-Huß, Stefan Feuerriegel, Helmut Prendinger; “Improving Sentiment Analysis with Document-Level Semantic Relationships from Rhetoric Discourse Structures”; Proceedings of the 50th Hawaii International Conference on System Sciences, 2017.

[9] Vijay B. Raut, D.D. Londhe; “Opinion Mining and Summarization of Hotel Reviews”; In 2014 Sixth International Conference on Computational Intelligence and Communication Networks.

[10] Merin Thomas, Latha C.A; “Sentimental analysis using recurrent neural network”; In International Journal of Engineering & Technology, 2018.

[11] Bo Pang, Lillian Lee, Shivakumar Vaithyanathan; “Thumbs up? Sentiment Classification using Machine Learning”; In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Philadelphia, July 2002, pp. 79-86. Association for Computational Linguistics.

[12] Wikipedia