### A

MINI-PROJECT REPORT

## REVIEW CLASSIFICATION USING NLTK AND SPACY

Submitted in partial fulfilment of the requirement for the award of the degree of

## BACHELOR OF TECHNOLOGY IN

**COMPUTER SCIENCE AND ENGINEERING**

### By

**B.THARUN KUMAR (16R91A0510)**

**K.SRIKRUTHI (16R91A0530)**

**C.RITHESH RAJ (16R91A0511)**

Under the Esteemed Guidance Of

## Ms.K.Tejasvi Asst.Professor Department of CSE



**Department of Computer Science and Engineering TEEGALA KRISHNA REDDY ENGINEERING COLLEGE**

(Affiliated to JNTUH, Hyderabad, Approved by AICTE, Accredited by NBA) Medbowli, Meerpet, Saroornagar, Hyderabad – 500097.

**2016-2020**

## DECLARATION

We, hereby declare that the mini project report entitled **“REVIEW CLASSIFICATION USING NLTK AND SPACY”** is done under the guidance of Ms.K.Tejasvi , (Asst.Professor), Department of Computer Science and Engineering, Teegala Krishna Reddy Engineering College, is submitted in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering.**

This is a record of bonafide work carried out by us in Teegala Krishna Reddy Engineering college and the results embodied in this project have not been reproduced or copied from any source. The results embodied in this project report have not been submitted to any other University or Institute for the award of any other degree or diploma.

Submitted by:

**B.Tharun Kumar (16R91A0510) K.Srikruthi (16R91A0530) C.Rithesh Raj (16R91A0511)**

## ACKNOWLEDGEMENT

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance have crowned our efforts with success.

We indebted to my guide**, Ms.K.Tejasvi,** Assistant Professor, Department of Computer Science, Teegala Krishna Reddy Engineering College, Meerpet, for her constant support and guidance throughout my project.

We indebted to the project coordinator, **Mrs.J.Rachana** Assistant Professor, Computer Science Engineering, Teegala Krishna Reddy Engineering College, Meerpet for her support and guidance throughout my project.

We indebted to **Dr. Ch. V. Phani Krishna**, Professor and Head of the Department, Computer Science, Teegala Krishna Reddy Engineering College, Meerpet, for his support and guidance throughout my project.

We extend my deep sense of gratitude to Principal **Dr.K.M.V.Madan Kumar,** Teegala Krishna Reddy Engineering College, Meerpet, for permitting me to undertake this project.

Finally, We express thanks to one and all who have helped me in successfully completing this project report. Furthermore We would like thank my family and friends for their moral support and encouragement.

By

**B.Tharun Kumar (16R91A0510) K.Srikruthi (16R91A0530) C.Rithesh Raj (16R91A0511)**

## CONTENTS

|  |  |  |
| --- | --- | --- |
| **NAME OF THE TOPIC** | | **PAGE NO.** |
| **LIST OF FIGURES** | | **I** |
| **LIST OF TABLES** | | **II** |
| **ABSTRACT** | | **III** |
| **1.INTRODUCTION** | | **1** |
| **2. LITERATURE SURVEY** | | **4** |
| **3. SYSTEM ANALYSIS** | | **8** |
| 3.1 | Existing System | 9 |
| 3.2 | Problem Statement | 9 |
| 3.2 | Proposed System | 9 |
| 3.3 | Feasibility Study | 9 |
|  | 3.4.1 Economical Feasibility | 9 |
|  | 3.4.2 Technical Feasibility | 10 |
|  | 3.4.3 Social Feasibility | 10 |
| 3.5 | Software Requirements Specifications | 11 |
|  | 3.5.1 Software Requirements | 11 |
|  | 3.5.2 Hardware Requirements | 11 |
| **4. SYSTEM DESIGN** | | **12** |
| 4.1 | System Architecture | 13 |
|  | 4.1.1 Data Collection | 13 |
|  | 4.1.2 Data Visualisation | 14 |
|  | 4.1.3 Data Pre-Processing | 14 |
|  | 4.1.4 Extracting Features | 14 |
|  | 4.1.5 Data Splitting | 14 |
|  | 4.1.6 Model Building | 15 |
|  | 4.1.7 Predicting Target | 15 |
| **5. SYSTEM IMPLEMENTATION** | | **16** |
| 5.1 | Dataset | 17 |
| 5.2 | Data Pre-Processing | 18 |

|  |  |  |
| --- | --- | --- |
|  | 5.2.1. Cleaning The Data | 18 |
|  | 5.2.2. Stemming And Lemmatization | 19 |
|  | 5.2.3 Tokenization | 20 |
|  | 5.2.4 Normalization | 21 |
|  | 5.2.5 Stop Words | 21 |
| 5.3 | Data Visualisation | 23 |
| 5.4 | Extracting Features | 23 |
|  | 5.4.1 Tf-Idf | 25 |
|  | 5.4.2 Bag Of Words | 26 |
| 5.5 | Data Splitting | 27 |
| 5.6 | 5 Model Building | 28 |
|  | 5.6.1 Logistic Regression | 28 |
|  | 5.6.2 Decision Tree Classifier | 30 |
|  | 5.6.3 Random Forest Classifier | 31 |
|  | 5.6.4 Multinomial Naïve Bayes Classifier | 33 |
|  | 5.6.5 MLP Classifier | 34 |
| 5.7 | Code | 35 |
| **6. SYSTEM TESTING** | | **41** |
| 6.1 | Model Evaluation | 42 |
| 6.2 | Predicting Target | 44 |
| **7. RESULT ANALYSIS** | | **46** |
| **8.CONCLUTION** | | **48** |
| **9.FUTURE ENHANCEMENT** | | **50** |
| **10. BIBILOGRAPHY** | | **52** |

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **FIGURE NAME** | **PAGE NO** |
| Figure 4.1 | System Architecture | 13 |
| Figure 5.1 | Word Cloud before data cleaning | 18 |
| Figure 5.2 | Word Cloud after data cleaning | 19 |
| Figure 5.3 | Before removing stop words | 23 |
| Figure 5.4 | After removing stop words | 23 |
| Figure 5.5 | Word Cloud | 24 |
| Figure 5.6 | Tf-Idf | 25 |
| Figure 5.7 | Bag of Words | 26 |
| Figure 5.8 | Relationship of a Dichotomous Outcome | 28 |
| Figure 5.9 | Decision Tree | 31 |
| Figure 5.10 | Random Forest | 33 |
| Figure 6.1 | Accuracy | 42 |
| Figure 6.2 | ROC curve | 43 |
| Figure 6.3 | Predicting Target | 44 |

Table 7.1: RESULT ANALYSIS 46

# ABSTRACT

## ABSTRACT

With the flourish of the web, online review is becoming a more and more useful and important information resource for people. As a result, automatic review mining and summarizing has become a hot research topic recently. “Text Classification” is one of the most important NLP tasks. It is the process of classifying text strings or documents into different categories, based upon the contents of the strings. Some examples of text classification are: Understanding audience sentiment from social media, Detection of spam and non-spam emails, Auto tagging of customer queries, and Classifying blog posts into different categories. This project focuses on specific domain – Movie Reviews. A multi-knowledge-based approach is used, which integrates statistical analysis and movie knowledge. The dataset used in this project is Movie Review Dataset, which is retrieved from the IMDB movie reviews dataset. The dataset is comprised of thousands of positive and negative movie reviews. The dataset is cleaned up and pre-processed in several ways like all reviews in dataset are converted to English language. The experimental results show the effectiveness of the multi-knowledge-based approach in movie review mining and summarizing. Thus enabling the audience to have a clear and summarized end review of the movie.

# INTRODUCTION

## INTRODUCTION

Review Classification (often referred as Sentiment Analysis) refers to classification of the viewpoint or review expressed in the text span; using information retrieval and computational linguistics. The review expressed on the topic is given significance rather than the topic itself [1]. Review Classification or Sentiment analysis extracts the subjective information from the source materials such as reviews using techniques such as natural language processing, and text analytics. Review plays essential part in our information-gathering behaviour before taking a decision. Online review sites, and personal blogs facilitate gathering of sentiments of products or object using information technologies. The main objective of Review Classification is to determine the polarity of comments (positive, negative or neutral) by extracting features and components of the object that have been commented on in each document [2, 3].

Studies related to Review Classification, on the implication of economic impact due to the reviews, issues about breach of privacy are given attention. Generally, the opinion expressed in a review document could either be a direct Review or comparative Review. Direct sentiment expressions on some target objects such as products, events, topics, persons. E.g.: “The climax of the movie is gripping.” Comparison opinion expresses the similarities or differences of more than one object usually stating an ordering or preference. E.g.: “movie x is better than movie y.” Different types of comparatives are Non equal Gradable (less than), Equative (same), Superlative (longest).

Review Classification is carried at either document level or sentence level as follows:

* Sentence level Review Classification is performed by two tasks subjective or Objective.

Objective: I watched a movie few days ago. Subjective: It is such a nice movie.

* For subjective sentences or clauses, classify positive or negative. Positive: It is such a nice movie.

Negative: The movie was boring.

In document level, a document (e.g., a review) is classified based on the overall sentiment expressed by Review holder.

Classes: Positive or negative

Assumption: each document focuses on a single object and contains opinions from a single opinion holder. E.g., thumbs-up or thumbs-down, star ratings (1 star, 2 stars, 3 stars…)

Review can also be made based on features as shown in example. “This movie is so misunderstood it is not even funny. If you think of seeing this one for the shootings.. stay clear. This one deals with the effects and trauma that the survivors must endure. Even the detectives are seeking the answer we all do...WHY? Fantastic acting from the two leading ladies as we see how those we ignore are affected by the very same things we are affected with. Yes the language is harsh at times, but it suits the characters well. There are some loose ends left or unanswered, but all movies have these. The major issues are dealt with and this movie makes a major statement about how all of us adults feel after such major incidents. Highly recommended for teens and adults...”

Each feature of the product is classified and overall sentiment is judged. This project presents a survey on different methods of sentiment analysis available in literature related to product reviews.

# LITERATURE SURVEY

## LITERATURE SURVEY

The main aim of this project is to identify the underlying sentiment of a movie review on the basis of its textual information. In this project, we try to classify whether a person liked the movie or not based on the review they give for the movie. This is particularly useful in cases when the creator of a movie wants to measure its overall performance using reviews that critics and viewers are providing for the movie. The outcome of this project can also be used to create a recommender by providing recommendation of movies to viewers on the basis of their previous reviews. Another application of this project would be to find a group of viewers with similar movie tastes (likes or dislikes). As a part of this project, we aim to study several feature extraction techniques used in text mining e.g. keyword spotting, lexical affinity and statistical methods, and understand their relevance to our problem. In addition to feature extraction, we also look into different classification techniques and explore how well they perform for different kinds of feature representations. We finally draw a conclusion regarding which combination of feature representations and classification techniques are most accurate for the current predictive task.

The original work on this dataset was done by researchers at Stanford University wherein they used unsupervised learning to cluster the words with close semantics and created word vectors. They ran various classification models on these word vectors to understand the polarity of the reviews. This approach is particularly useful in cases when the data has rich sentiment content and is prone to subjectivity in the semantic affinity of the words and their intended meanings. Apart from the above, a lot of work has been done by Bo Pang and Peter Turnkey towards polarity detection of movie reviews and product reviews. They have also worked on creating a multi- class classification of the review and predicting the reviewer rating of the movie/product.

These works discussed the use of Random Forest classifier, Linear Regression, Decision tree, MLP classifier and Multinomial naïve Bayes for the classification of reviews and also on the use of various feature extraction techniques. One major point to be noted in these papers was exclusion of a neutral category in classification under the assumption that neutral texts lie close to the boundary of the binary classifiers and are disproportionately hard to classify. There are many sentiment analysis tools and software existing today that are available for free or under commercial license. With the advent of micro blogging, sentiment analysis is being widely used to analyse the general public sentiments and draw inferences out of these. One famous application was use of Twitter to understand the political sentiment of the people in context of German Federal elections.

In Section 2.1, we present an overview of how sentiment analysis and classification work in general. In Section 2.2, we discuss some factors that affect sentiment analysis and classification

## General Overview of Sentiment Analysis and Classification

Sentiments can be classified at the word, sentence, or document level. This project focused on document-level sentiment classification. Document-level sentiment classification can be described as follows. Given a set of related documents containing opinions (also known as opinionated documents), we determine whether each document d  expresses a positive or negative opinion of (i.e., the sentiment toward) an object. Existing research in this area makes the assumption that the opinionated document d (e.g., a movie review) contains opinions regarding a single object . This assumption typically holds for customer reviews of products and services, but not for forum and blog posts due to the fact that such a post may contain opinions on multiple products and services.

In trying to find the document-level sentiment, one approach that could be used is to perform sentence-level sentiment classification on each sentence in the document and then sum up all the sentence-level sentiments to produce the document- level sentiment. This however, requires that word-level sentiment classification be performed on each sentence prior to sentence-level classification. One of the main challenges of document-level sentiment analysis and classification is that not every part of the document is equally informative in inferring the sentiment of the whole document. However identifying the useful sentences automatically is itself a difficult learning problem. Some researchers have proposed automatic approaches to extracting useful sentences. Others have considered all sentences in the document in conjunction with algorithms that were able to produce better experimental results.

## Factors that affect sentiment analysis and classification

To classify a document as either positive or negative according to the overall sentiment expressed by the author is seen as a more challenging problem than strict text classification. For example, opinions can often be expressed in a more complex manner, making it difficult to be identified by any of the words in the sentence or document when considered in isolation. By just looking at the words in a review, one may not be able to accurately classify the sentiments which have been expressed. Consider this review:

“This film should be brilliant. It sounds like a great plot, the actors are first grade, and the supporting cast is good as well, and Stallone is attempting to deliver a good performance, however, it can't hold up"

The presence of words such as “brilliant”, “great”, “good” suggest a positive sentiment so one might think it easy to identify the sentiment of a review by a set of keywords. However, Pang and Lee found out from their experimental results that the accuracy of classification generated from the use of human generated list of keywords was lower as compared to that of keywords generated using machine learning techniques.

The above review has an anaphor (e.g., the phrase “it can’t hold up”). It is not clear what “it” in this phrase refers to, whether the movie or Stallone’s performance. This makes it difficult to determine the overall polarity of the review. Many free format reviews make use of many anaphora, abbreviations, and lack of capitals, poor spelling, poor punctuation, and poor grammar.

The main factors that affect how opinions are analyzed and classified include the domain of the datasets, the size of the datasets to be analyzed, the format of the datasets (labelled or unlabelled), and quality of the dataset. The accuracy of sentiment classification can be influenced by the domain of the items to which it is applied [1]. One reason is that the same phrase can indicate different sentiment in different domains: for example, “go read the book” most likely indicates positive sentiment for book reviews, but negative sentiment for movie reviews. A review such as “it is so easy to predict the next action….” is a negative sentiment for a movie review such as “it is so easy to predict the next action….” is a negative sentiment for a movie 10 plot but a positive sentiment for a political review. Difference in vocabularies across different domains also adds to the difficulty when applying classifiers trained on labelled data in one domain to test data in another.

Depending on the size of datasets to be used, either manual or automatic, or both approaches, could be used. However, it is always best to use a combination of both approaches as it has been realized from a number of experiments that even the worst results obtained from using both approaches are superior to the best of manual approaches and some automatic approaches. The availability of labelled data also improves the time taken to classify opinions. Without it, many researchers have had to use the linguistic/semantic approach of building lexicons, which is very time consuming and yet does not yield better performance. Lexicons are also specific to a particular language and domain so making the sentiment analysis and classification task even more difficult.

The quality of the dataset is also bound to affect sentiment classification performance. Due to the fact that there is no quality control measure in place to check reviews, anyone can write anything on the web. These result in many low quality reviews and review spam. Bing et al. studied opinion spam and discovered that online reviews are normally made up of spam messages which consist of untruthful or fake reviews, irrelevant reviews, and reviews which are not actual reviews, but rather statements or questions. Effective pre-processing of the data is needed to reduce the level of spam in these reviews. However, this is a research area which has not received much attention

# SYSTEM ANALYSIS

## SYSTEM ANALYSIS

* 1. **EXISTING SYSTEM**

The Existing System of Review classification is done manually by reading every review and labeling it either positive or negative. The existing system is incapable of processing huge amount of data within specified amount of time. Also, this type of data is limited for processing structured data and accuracy of existing technologies is incomparable when compared to modern technologies.

## PROBLEM STATEMENT

In the existing system, the time taken for classifying the review is more as it done by manually .It becomes nearly impossible to generate result for huge amount of data present in today’s world.

## PROPOSED SYSTEM

The problem of classifying the Movie Review into respective categories can be solved or simplified by making use of the text classification approach of supervised machine learning algorithms. This project focuses on building a machine learning model which can be used to automate the process of classifying the Movie Review into two categories either positive or negative. We use models like Random Forest, Decision Tree, MLP, Logistic Regression, etc…**.**These above mentioned models are to be trained using the IMDB data taken from the official IMDB website.

The model giving the highest accuracy will be considere d as the best suited model for this project.

## FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

1. **ECONOMICAL FEASIBILITY**
2. **TECHNICAL FEASIBILITY**
3. **SOCIAL FEASIBILITY**

## ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

## TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

## SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently.

## (i) Technical feasibility

In the technical feasibility study, one has to test whether the proposed system can be developed using existing technology or not. It is planned to implement the proposed system in Android. The project entitled is technically feasible because of the following reasons.

1. All necessary technology exists to develop the system.
2. The existing system is so flexible that it can be developed further.

## (ii) Economic feasibility

As a part of this, the costs and benefits associated with the proposed systems are to be compared. The project is economically feasible only if tangible and intangible benefits outweigh the cost. We can say the proposed system is feasible based on the following grounds.

1. The cost of developing the full system is reasonable.
2. The cost of hardware and software for the application is less.

## Operational feasibility

The project is operationally feasible because there is sufficient support from the project management and the users of the proposed system. Proposed system

definitely does not harm and will not produce the bad results and no problem will arise after implementation of the system.

## Software Requirement Specification

* + 1. **Software Requirements**

Operating System : Windows, Linux or UNIX Software : Anaconda

IDE : Jupyter notebook

Editor : Text Editor

Toolkit : NLTK Code Language : Python 3.6

* + 1. **Hardware Requirements** Processor : Intel i3, i5, i7 and above Hard Disk : 100 GB

RAM : 4 GB or above

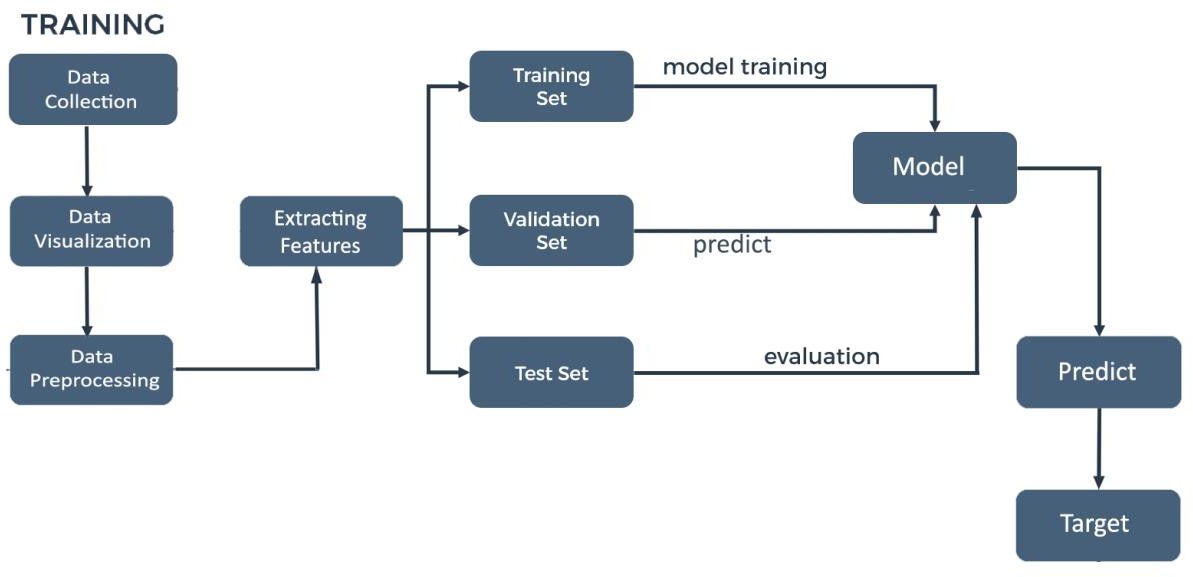
# SYSTEM DESIGN

## SYSTEM DESIGN

* 1. **SYSTEM ARCHITECTURE**

Here, we provide a brief overview of the process of sentiment analysis and classification. A more detailed description is provided in Chapter 5. A diagram summarizing the steps required for text classification using sentiment is shown in Figure 1.1 the pre-processing step involves extracting the reviews or documents from a source dataset. Terms in each review are cleaned in several ways. The terms in each document, together with their sentiment scores, are stored as feature vectors to be used as input to a text classifier.

The next step involves using a text classifier to classify the selected features as either positive or negative. The stored feature vectors become input to the text classifier. If necessary, feature selection can be applied to reduce the number of features. In order to arrive at the best results, a series of iterative steps, known as cross-validation can be used to estimate how accurately a predictive model will perform in practice.



**Figure 4.1: System Architecture**

## Data Collection

The dataset used for this task was collected from Large Movie Review Dataset which was used by the AI department of Stanford University for the associated publication. The dataset contains 50,000 training examples collected from IMDb where each review is labelled with the rating of the movie on scale of 1-10. A more detailed description is provided in Chapter 5.

## Data Visualisation

Data visualization is the graphic representation of data. It involves producing images that communicate relationships among the represented data to viewers of the images. This communication is achieved through the use of a systematic mapping between graphic marks and data values in the creation of the visualization. This mapping establishes how data values will be represented visually, determining how and to what extent a property of a graphic mark, such as size or colour, will change to reflect change in the value of a datum.

## Data Pre-processing

Data pre-processing is an important step in the data mining process. The phrase "garbage in, garbage out" is particularly applicable to data mining and machine learning projects. Data-gathering methods are often loosely controlled, resulting in out-of-range values (e.g., Income: −100), impossible data combinations (e.g., Sex: Male, Pregnant: Yes), missing values, etc. Analysing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running an analysis often, data pre- processing is the most important phase of a machine learning project, especially in computational biology.

## Extracting Features

In text processing, words of the text represent discrete, categorical features. How do we encode such data in a way which is ready to be used by the algorithms? The mapping from textual data to real valued vectors is called feature extraction. Some simplest techniques to numerically represent text is Bag of Words and TF-IDF.

## Data Splitting

**Training Dataset**: The sample of data used to fit the model.

**Test Dataset**: The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.

**Validation Dataset**: The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.

## Model Building

The overall task in this project is for classification of reviews as favourable or unfavourable. Therefore, for this classification task we explored multiple

classification models on above feature representations. We used the models ranging from the simple Logistic Regression to the state of-art SVM Classifier. We also used other classification models like MLP Classifier, Decision Tree and Random Forest Classifier. Apart from these, we also trained the above feature representations on Naïve Bayes’ Classifier as this is primarily used in case of text mining in combination with Bag of Words and TF-IDF Modelling. We also trained a model based on k- Nearest Neighbours to match the similarity between the reviews and classify them accordingly.

## Predicting Target

The target variable of a dataset is the feature of a dataset about which you want to gain a deeper understanding. A supervised machine learning algorithm uses historical data to learn patterns and uncover relationships between other features of your dataset and the target. The target variable will vary depending on the business goal and available data

# SYSTEM IMPLEMENTATION

## SYSTEM IMPLEMENTATION

* 1. **Dataset**

The dataset used for this task was collected from Large Movie Review Dataset which was used by the AI department of Stanford University for the associated publication . The dataset contains 50,000 training examples collected from IMDb where each review is labelled with the rating of the movie on scale of 1-10. 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 was above 5, we deduced that the person liked the movie otherwise he did not. Initially the dataset was divided into two subsets containing 25,000 examples each for training and testing. We found this division to be sub-optimal as the number of training examples was very small and leading to under-fitting. We then tried to redistribute the examples as 40,000 for training and 10,000 for testing. While this produced better models, it also led to over-fitting on training examples and worse performance on the test set. Finally, we decided to use Cross Validation in which the complete dataset is divided into multiple folds with different samples for training and validation each time and the final performance statistic of the classifier is averaged over all results. This improved the accuracy of our models across the boards. A typical review text looks like this:

I'm a fan of TV movies in general and this was one of the good ones. The cast performances throughout were pretty solid and there were twists I didn't see coming before each commercial. To me it was kind of like Medium meets CSI. <br/><br>Did anyone else think that in certain lights, the daughter looked like a young Nicole Kidman? Are they related in any way? I'd definitely watch it agin or rent it if it ever comes to video. . <br/><br>Dedee was great. Haven't seen in her in a lot of things and she did her job very convincingly. If you're into to TV mystery movies, check this one out if you have a chance.

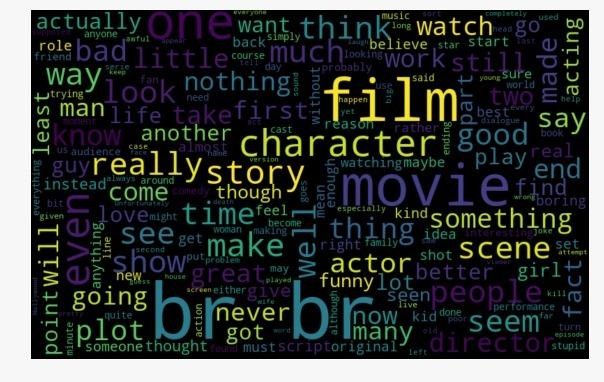
As seen above, one necessary pre-processing step prior to feature extraction was removal of HTML tags like “<br>”. We used simple regular expressions matching to remove these HTML tags from the text. Another important step was to make the text case-insensitive as that would help us count the word occurrences across all reviews and prune unimportant words. We also removed all the punctuation marks like ‘!’, ‘?’, etc. As they do not provide any substantial information and are used by different people with varying connotations. This was achieved using standard python libraries for text and string manipulation. We also removed stop words from the text for some of our feature extraction tasks, which is described in greater detail in later sections. One important point to note is that we did not use stemming of words as some information is lost while stemming a word to its root form.

## Data Pre-processing

Before performing any algorithm, we have to make sure to clean up the data, in order to make it easier to process. Also, by finding and removing noise words in advance, we can increase greatly the accuracy of our algorithms.

## Cleaning the data

The IMDb reviews contains html tags which do not serve any purpose for detecting sentiment, we also decided to remove punctuation whatsoever, even if this means that we get rid of emoticons (there are very few of them anyway), but it makes it easier for us to handle. Finally we lowercase everything. We also apply Porter stemming algorithm, which helps us replace every word with its root, and so words like cats and cat, or running and run, become the same. This has been shown to improve classification accuracy in sentiment analysis tasks.



**Figure 5.1: Word Cloud before data cleaning**



**Figure 5.2: Word Cloud after data cleaning**

## Stemming and lemmatization

For grammatical reasons, documents are going to use different forms of a word, such as *organize*, *organizes*, and *organizing*. Additionally, there are families of derivationally related words with similar meanings, such as *democracy*, *democratic*, and *democratization*. In many situations, it seems as if it would be useful for a search for one of these words to return documents that contain another word in the set.

The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. For instance:

$\Rightarrow$am,are,is be

$\Rightarrow$car, cars, car's, cars' car

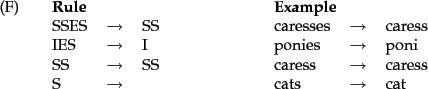
The result of this mapping of text will be something like:

$\Rightarrow$The boy's cars are differentcolors the boy car be differ color

However, the two words differ in their flavor. *Stemming* usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes. *Lemmatization* usually refers to doing things properly with the use of a

vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the *lemma*. If confronted with the token *saw*, stemming might return just *s*, whereas lemmatization would attempt to return either *see* or *saw* depending on whether the use of the token was as a verb or a noun. The two may also differ in that stemming most commonly collapses derivationally related words, whereas lemmatization commonly only collapses the different inflectional forms of a lemma. Linguistic processing for stemming or lemmatization is often done by an additional plug-in component to the indexing process, and a number of such components exist, both commercial and open-source.

The most common algorithm for stemming English, and one that has repeatedly been shown to be empirically very effective, is *Porter's algorithm* [4]. The entire algorithm is too long and intricate to present here, but we will indicate its general nature. Porter's algorithm consists of 5 phases of word reductions, applied sequentially. Within each phase there are various conventions to select rules, such as selecting the rule from each rule group that applies to the longest suffix. In the first phase, this convention is used with the following rule group:



Many of the later rules use a concept of the *measure* of a word, which loosely checks the number of syllables to see whether a word is long enough that it is reasonable to regard the matching portion of a rule as a suffix rather than as part of the stem of a word. For example, the rule:

$\rightarrow$($m>1$) EMENT

would map *replacement* to *replac*, but not *cement* to *c*.

## Tokenization

Given a character sequence and a defined document unit, tokenization is the task of chopping it up into pieces, called *tokens*, perhaps at the same time throwing away certain characters, such as punctuation. Here is an example of tokenization:

Input: Friends, Romans, Countrymen, lend me your ears;

Output:



These tokens are often loosely referred to as terms or words, but it is sometimes important to make a type/token distinction. A *token* is an instance of a

sequence of characters in some particular document that are grouped together as a useful semantic unit for processing. A *type* is the class of all tokens containing the same character sequence. A *term* is a (perhaps normalized) type that is included in the IR system's dictionary. The set of index terms could be entirely distinct from the tokens, for instance, they could be semantic identifiers in taxonomy, but in practice in modern IR systems they are strongly related to the tokens in the document.

## Normalization

*Token normalization* is the process of canonicalizing tokens so that matches occur despite superficial differences in the character sequences of the tokens. The most standard way to normalize is to implicitly create *equivalence classes*, which are normally named after one member of the set. For instance, if the tokens *anti- discriminatory* and *antidiscriminatory* are both mapped onto the term antidiscriminatory, in both the document text and queries, then searches for one term will retrieve documents that contain either.

The advantage of just using mapping rules that remove characters like hyphens is that the equivalence classing to be done is implicit, rather than being fully calculated in advance: the terms that happen to become identical as the result of these rules are the equivalence classes. It is only easy to write rules of this sort that remove characters. Since the equivalence classes are implicit, it is not obvious when you might want to add characters. For instance, it would be hard to know to turn *antidiscriminatory* into *anti-discriminatory*.

|  |
| --- |
| \begin{figure}\begin{tabular}{ll} \textbf{Query term} & \textbf{Terms in documen... ... Windows, windows, window \\ window & window, windows \end{tabular}\end{figure} |
| **Figure 5.4: An example of how asymmetric expansion of query terms can usefully model users' expectations.** |

An alternative to creating equivalence classes is to maintain relations between unnormalized tokens. This method can be extended to hand-constructed lists of synonyms such as *car* and *automobile*. These term relationships can be achieved in two ways. The usual way is to index unnormalized tokens and to maintain a query expansion list of multiple vocabulary entries to consider for a certain query term. A query term is then effectively a disjunction of several postings lists. The alternative is to perform the expansion during index construction. When the document contains automobile, we index it under car as well (and, usually, also vice-versa). Use of either of these methods is considerably less efficient than equivalence classing, as there are

more postings to store and merge. The first method adds a query expansion dictionary and requires more processing at query time, while the second method requires more space for storing postings. Traditionally, expanding the space required for the postings lists was seen as more disadvantageous, but with modern storage costs, the increased flexibility that comes from distinct postings lists is appealing.

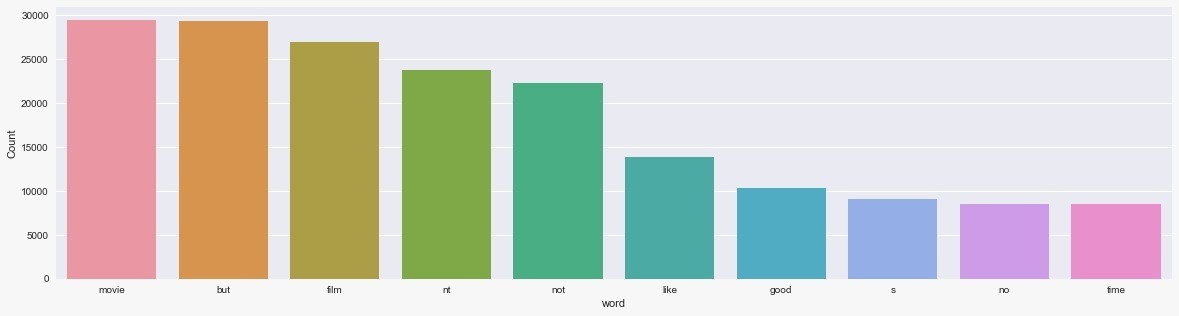
These approaches are more flexible than equivalence classes because the expansion lists can overlap while not being identical. This means there can be an asymmetry in expansion. An example of how such an asymmetry can be exploited is shown in Figure 4 : if the user enters windows, we wish to allow matches with the capitalized *Windows* operating system, but this is not plausible if the user enters window, even though it is plausible for this query to also match lowercase *windows*.

## Stop Words

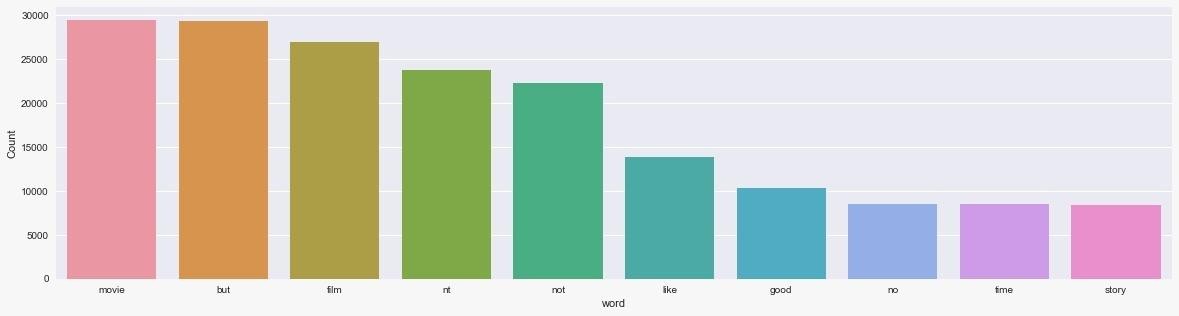
\begin{figure}\begin{tabular}{llllllllll} a & an & and & are & as & at & be & by... ... & on & that & the \\ to & was & were & will & with \end{tabular} \end{figure}

Sometimes, some extremely common words which would appear to be of little value in helping select documents matching a user need are excluded from the vocabulary entirely. These words are called *stop words* . The general strategy for determining a stop list is to sort the terms by *collection frequency* (the total number of times each term appears in the document collection), and then to take the most frequent terms, often hand-filtered for their semantic content relative to the domain of the documents being indexed, as a *stop list* , the members of which are then discarded during indexing. An example of a stop list is shown in Figure above . Using a stop list significantly reduces the number of postings that a system has to store. And a lot of the time not indexing stop words does little harm: keyword searches with terms like the and by don't seem very useful. However, this is not true for phrase searches. The phrase query ``President of the United States'', which contains two stop words, is more precise than President AND ``United States''. The meaning of flights to London is likely to be lost if the word to is stopped out. A search for Vannevar Bush's article As we may think will be difficult if the first three words are stopped out, and the system searches simply for documents containing the word think. Some special query types are disproportionately affected. Some song titles and well known pieces of verse consist entirely of words that are commonly on stop lists (To be or not to be, Let It Be, I don't want to be, ...). The general trend in IR systems over time has been from standard use of quite large stop lists (200-300 terms) to very small stop lists (7-12 terms) to no stop list whatsoever. Web search engines generally do not use stop lists. Some of the design of modern IR systems has focused precisely on how we can

exploit the statistics of language so as to be able to cope with common words in better ways.



**Figure 5.3: Before removing stop words**



**Figure 5.4 After removing stop words**

## Data Visualisation

Data visualization is the graphic representation of data. It involves producing images that communicate relationships among the represented data to viewers of the images. This communication is achieved through the use of a systematic mapping between graphic marks and data values in the creation of the visualization. This mapping establishes how data values will be represented visually, determining how and to what extent a property of a graphic mark, such as size or colour, will change to reflect change in the value of a datum.

**Tag Clouds**

The simplest and most common form of text visualization is tag cloud. They depict tags arranged in space varied in size, colour, and position based on tag frequency, categorization, or significance.



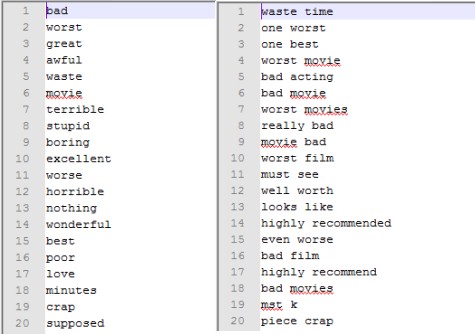
**Figure 5.5: Word Cloud**

## Extracting Features

In text processing, words of the text represent discrete, categorical features. How do we encode such data in a way which is ready to be used by the algorithms? The mapping from textual data to real valued vectors is called feature extraction. Some simplest techniques to numerically represent text is TF-IDF and Bag of Words.

## TF-IDF

Tf-Idf stands for term frequency, inverse document frequency. And it is a very useful technique used mainly in information retrieval in order to rank how important a keyword is to a given document in a corpus1 . Returning to the word movie. This made us think that we still had garbage, and we had the idea that maybe a word can be very important to the whole set of reviews, but not to any single review by itself, and thus we needed a way of computing which words were important to which



**Figure 5.6: Tf-Idf**

reviews. After searching a bit we came with the concept of tf-idf, which solves precisely our problem. We found that this was actually a common approach for sentiment analysis(although we couldn’t find a paper were this was first proposed).

It works as follows, we want to compute the function *w(w, d)* for every word w document d pair, this will give how important is the word for indexing the document. We have the two auxiliary functions *tf(w, d),* which counts how important is the word for the document.

*tf(w, d)* = 1 + log(Number of occurrences of w in d)

This intuitively gives how important is that word in the document, but it doesn’t take into account the possibility of a word belonging to every document (which will render it useless for indexing), and thus we need a balance term *idf(w, D),* which is defined over the whole corpus D, and is given by



The more documents w is in, the lower its idf score, which is what we want.

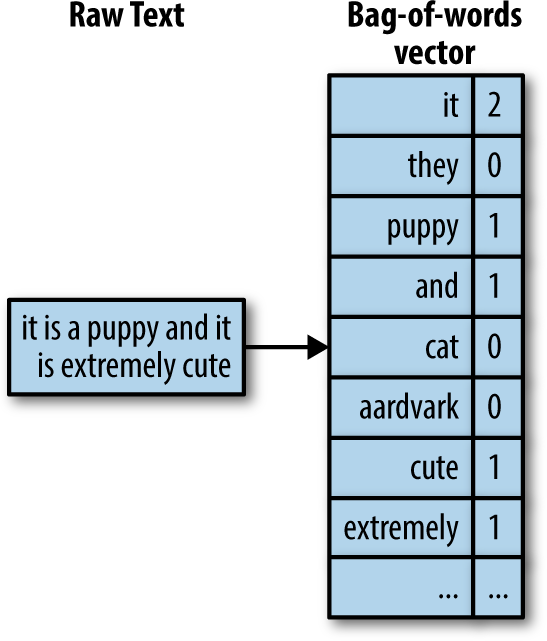
So now, we get that

*f(w, D)* = tf(w, d) ∗ idf(w, D)

And thus a word is important if it is frequent in the document, but if it is not to frequent among all other documents.

## Bag of Words

The bag-of-words model is a representation of text used in natural language processing. This model represents a text as the set (bag) of its words and does not make use of the grammar or other linguistic structures that the words and sentences have. Let’s assume that after pre-processing of the data, we have N distinct words in the entire document. Also, suppose that the document contains R reviews. Each review is represented by a N-dimensional feature vector. The entries of these vectors correspond to the N words in the document and the feature values reflect the frequency of their corresponding words in the review. Alternatively, we can construct a binary feature vector where 1 indicates the presence of the corresponding word in the review and zero represents otherwise. However, we need to do ”feature selection” in order to reduce the dimension of the data points, since the text document is very large and contains too many words. After pre-processing and cleaning the data, there are N = 78, 767 words in the document. These words are sorted based on their importance which is measured using the mutual information criteria introduced in the previous sections and the top d = 5000 words are selected for constructing the feature vectors. Of course, larger number of the words would result in better accuracy results; however, we are limited by the memory size and the run-time of the algorithms.



**Figure 5.7: Bag of Words**

**Classification**

In order to classify the reviews into positive and negative classes based on the bag of words model, we classify the feature vectors using three different supervised classification methods, namely, support vector machine, logistic regression, and random forests.

## Data Splitting

After cleaning and pre-processing of the reviews, we can now split our data into 3 sets train, validation, and test — in the ratio of 70%, 15% and 15%, respectively. We will use the train set for training our model, and validation and test sets to make predictions and evaluate our model.

**Training Dataset**: The sample of data used to fit the model.

**Test Dataset**: The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.

**Validation Dataset**: The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.

## Model Building

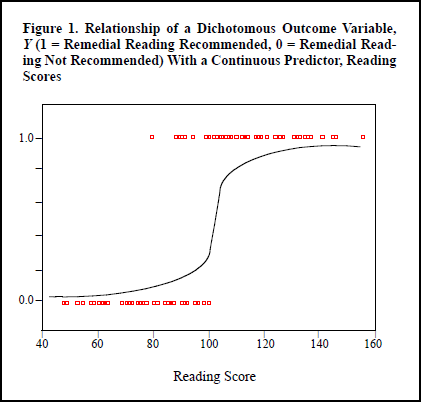
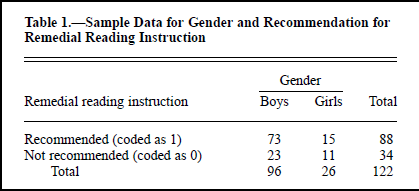
The overall task in this project is for classification of reviews as favourable or unfavourable. Therefore, for this classification task we explored multiple classification models on above feature representations. We used the models ranging from the simple Logistic Regression to the state of-art SVM Classifier. We also used other classification models like MLP Classifier, Decision Tree and Random Forest Classifier. Apart from these, we also trained the above feature representations on Naïve Bayes’ Classifier as this is primarily used in case of text mining in combination with Bag of Words and TF-IDF Modelling. We also trained a model based on k- Nearest Neighbours to match the similarity between the reviews and classify them accordingly

## Logistic Regression

The central mathematical concept that underlies logistic regression [5] is the logit—the natural logarithm of an odds ratio. The simplest example of a logit derives from a 2 × 2 contingency table. Consider an instance in which the distribution of a dichotomous outcome variable (a child from an inner city school who is recommended for remedial reading classes) is paired with a dichotomous predictor variable (gender). Example data are included in Table 1. A test of independence using chi-square could be applied. The results yield χ2(1) = 3.43. Alternatively, one might

prefer to assessa boy’s odds of being recommended for remedial reading instruction relative to a girl’s odds. The result is an odds ratio of 2.33, which suggests that boys are 2.33 times more likely, than not, to be recommended for remedial reading classes compared with girls. The odds ratio is derived from two odds (73/23 for boys and 15/11 for girls); its natural logarithm [i.e., ln(2.33)] is a logit, which equals 0.85. The value of 0.85 would be the regression coefficient of the gender predictor if logistic regression were used to model the two outcomes of a remedial recommendation as it relates to gender.

Generally, logistic regression is well suited for describing and testing hypotheses about relationships between a categorical outcome variable and one or more categorical or continuous predictor variables. In the simplest case of linear regression for one continuous predictor X (a child’s reading score on a standardized test) and one dichotomous outcome variable Y (the child being recommended for remedial reading classes), the plot of such data results in two parallel lines, each corresponding to a value of the dichotomous outcome (Figure 1). Because the two parallel lines are difficult to be described with an ordinary least squares regression equation due to the dichotomy of outcomes, one may instead create categories for the predictor and compute the mean of the outcome variable for the respective categories. The resultant plot of categories’ means will appear linear in the middle, much like what one would expect to see on an ordinary scatter plot,



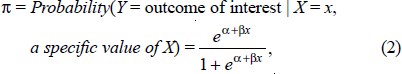
**Figure 5.8: Relationship of a Dichotomous Outcome Variable, *Y* (1 = Remedial Reading Recommended, 0 = Remedial Reading Not Recommended) With a Continuous Predictor, Reading Scores**

but curved at the ends (Figure 1, the S-shaped curve). Such a shape, often referred to as sigmoidal or S-shaped, is difficult to describe with a linear equation for two reasons. First, the extremes do not follow a linear trend. Second, the errors are neither normally distributed nor constant across the entire range of data (Peng, Manz, & Keck, 2001). Logistic regression solves these problems by applying the logit transformation to the dependent variable. In essence, the logistic model predicts the logit of Y from X. As stated earlier, the logit is the natural logarithm (ln) of odds of Y, and odds are ratios of probabilities (π) of Y happening (i.e., a student is recommended for remedial reading instruction) to probabilities (1 – π) of Y not happening (i.e., a student is not recommended for remedial reading instruction). Although logistic regression can accommodate categorical outcomes that are polytomous, in this article we focus on dichotomous outcomes only. The illustration presented in this article can be extended easily to polytomous variables with ordered (i.e., ordinal-scaled) or unordered (i.e., nominal-scaled) outcomes.

The simple logistic model has the form



For the data in Table 1, the regression coefficient (β) is the logit (0.85) previously explained. Taking the antilog of Equation 1 on both sides, one derives an equation to predict the probability of the occurrence of the outcome of interest as follows:



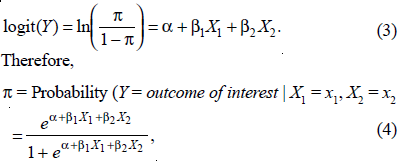
where π is the probability of the outcome of interest or “event,” such as a child’s referral for remedial reading classes, α is the Y intercept, β is the regression coefficient, and e = 2.71828 is the base of the system of natural logarithms.

X can be categorical or continuous, but Y is always categorical. According to Equation 1, the relationship between logit (Y) and X is linear. Yet, according to Equation 2, the relationship between the probability of Y and X is nonlinear.

For this reason, the natural log transformation of the odds in Equation 1 is necessary to make the relationship between a categorical outcome variable and its predictor(s) linear. The value of the coefficient β determines the direction of the relationship between X and the logit of Y. When β is greater than zero, larger (or smaller) X values are associated with larger (or smaller) logits of Y. Conversely, if β is less than zero, larger (or smaller) X values are associated with smaller (or larger)

logits of Y. Within the framework of inferential statistics, the null hypothesis states that β equals zero, or there is no linear relationship in the population. Rejecting such a null hypothesis implies that a linear relationship exists between X and the logit of Y. If a predictor is binary, as in the Table 1 example, then the odds ratio is equal to e, the natural logarithm base, raised to the exponent of the slope β (eβ).

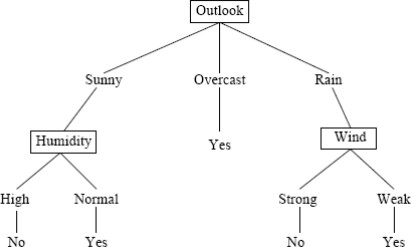
Extending the logic of the simple logistic regression to multiple predictors (say X1 = reading score and X2 = gender), one can construct a complex logistic regression for Y (recommendation for remedial reading programs) as follows



where π is once again the probability of the event, α is the Y intercept, βs are regression coefficients, and Xs are a set of predictors. α and βs are typically estimated by the maximum likelihood (ML) method, which is preferred over the weighted least squares approach by several authors, such as Haberman (1978) and Schlesselman (1982). The ML method is designed to maximize the likelihood of reproducing the data given the parameter estimates. Data are entered into the analysis as 0 or 1 coding for the dichotomous outcome, continuous values for continuous predictors, and dummy codings (e.g., 0 or 1) for categorical predictors. The null hypothesis underlying the overall model states that all βs equal zero. A rejection of this null hypothesis implies that at least one β does not equal zero in the population, which means that the logistic regression equation predicts the probability of the outcome better than the mean of the dependent variable Y. The interpretation of results is rendered using the odds ratio for both categorical and continuous predictors.

## Decision Tree Classifier

A decision tree[6] is a tree whose internal nodes are tests and whose leaf nodes are categories. Each internal node test one attribute and each branch from a node selects one value for the attribute. The attribute used to make the decision is not defined. So we can use the attribute which gives maximum information. And the leaf node predicts a category or class. The decision trees are not limited to Boolean functions, but they can be extended for general categorically values functions.



**Figure 5.9: Decision Tree**

In the above example the given instances can be divided based on the values it take for the attribute “outlook”. The instances are split based on attributes and the one which gives the maximum information is selected as the decision for that node. Hence in the above example we could say that selecting “Outlook” at the root node gives maximum information at that level. And the edges represent the values the attributes can take and the instances are divided accordingly to each child nodes. The tree can be a mary tree depending upon the values that the attributes can take. The attribute selection is based on a heuristic approach that the particular attribute will give the best split at a particular level. But this approach has been successful over the past.

## Random Forest Classifier

A Random Forest[7] is a classifier consisting of collection of tree-structured classifiers where independent random vectors are distributed identically and each tree cast a unit vote for the most popular class at input x. A random vector is generated which is independent of the past random vectors with same distribution and a tree is generated by using the training test [Brieman (2001)]. For random forests, an upper bound is derived to obtain the generalization error in terms of two parameters that are given below:

* + - * The accuracy of individual classifiers
      * The dependency between the individual classifiers

The generalization of error for random forest includes two segments. These segments are defined below:

* + - * The strength of the individual classifiers in the forest.
      * The correlation between them in terms of raw margin function

In order to improve accuracy of random forest, the correlation should be minimized while retaining their strength. Brieman (2001) studied that forests consist of randomly

selected inputs or combination of inputs at each node to grow tree. The class of procedures has desirable characteristics that are listed below:

* + - * Accuracy is good and sometimes better.
      * Relatively robust to outliers and noise.
      * Faster than Bagging and Boosting.
      * Simple and can easily be parallelized.

Brieman has proposed a randomization approach that works better with bagging or random space method. To generate each tree of random forest, following steps are followed that are described below:

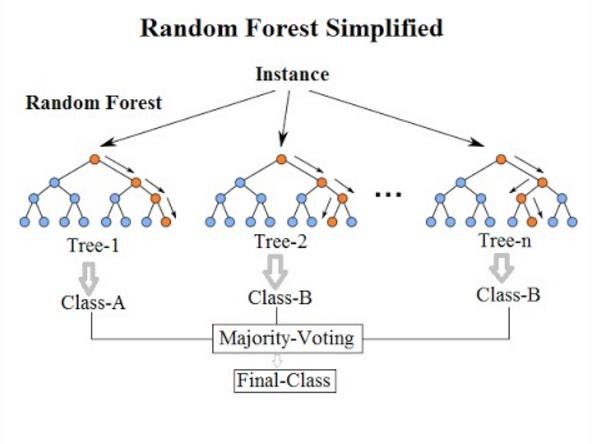
* + - * Training dataset consist of N number of records.
      * Samplings of N number of records are performed randomly but with replacement.
      * This sample of dataset is named as bootstrap sample.
      * If this training set would consist of M number of input variables, m<<M number of inputs are selected randomly out of M and the best split on these m attributes is used to split the node.
      * The value of m will remain constant during forest growing.
      * The tree will be grow to the largest possible level.

There are two reasons for using Bagging approach. They are given below: [Brieman (1994)]

* + - * It seems that the use of bagging along with the random features generates more accurate results.
      * Bagging can be used to provide on-going estimation of generalization error as well as the estimation of strength and correlation.

When the forest is trained to classify a new instance, this whole process is executed across all the trees included in the forest. Each tree of the forest has to provide a vote which is recorded as a classification for the new instance. The votes of all trees are combined together and the votes are counted, the class having maximum number of votes are declared as classification of new instance. This process is known as Forest RI process. The description of process for building forest is given below:

* + - * When bootstrap sample is built by sampling the data with replacement of each tree, then one-third of the instances are left out.
      * The left out instances are known as OOB (Out of Bag) data.
      * Each tree of the forest has its own OOB data which is used for the error estimation of individual trees known as OOB error estimation.
      * Random forest also consist of in-built facilities for calculating variable importance and proximities.
      * The proximities are used for removing and replacing missing values and outliers.



**Figure 5.10: Random Forest**

## Multinomial Naïve Bayes Classifier

In the MNB classifier [8] each document is viewed as a collection of words and the order of words is considered irrelevant. The probability of a class value c given a test document d is computed as

P(c|d) = P(c) Q w∈d P(w|c) nwd P(d) , (1)

where nwd is the number of times word w occurs in document d, P(w|c) is the probability of observing word w given class c, P(c) is the prior probability of class c, and P(d) is a constant that makes the probabilities for the different classes sum to one. P(c) is estimated by the proportion of training documents pertaining to class c and P(w|c) is estimated as

P(w|c) = 1 + P d∈Dc nwd k + P w0 P d∈Dc nw0d , (2)

where Dc is the collection of all training documents in class c, and k is the size of the vocabulary (i.e. the number of distinct words in all training documents). The additional one in the numerator is the so-called Laplace correction, and corresponds to initializing each word count to one instead of zero. It requires the addition of k in the denominator to obtain a probability distribution that sums to one. This kind of

correction is necessary because of the zero-frequency problem: a single word in test document d that does not occur in any training document pertaining to a particular category c will otherwise render P(c|d) zero.

## MLP Classifier

A multilayer perceptron (MLP) [9] is a class of feedforward artificial neural network (ANN). An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

If a multilayer perceptron has a linear activation function in all neurons, that is, a linear function that maps the weighted inputs to the output of each neuron, then linear algebra shows that any number of layers can be reduced to a two-layer input- output model. In MLPs some neurons use a nonlinear activation function that was developed to model the frequency of action potentials, or firing, of biological neurons.

The two historically common activation functions are both sigmoids, and are described by





In recent developments of deep learning the rectifier linear unit (ReLU) is more frequently used as one of the possible ways to overcome the numerical problems related to the sigmoid.The first is a hyperbolic tangent that ranges from -1 to 1, while the other is the logistic function, which is similar in shape but ranges from 0 to 1. Here yi is the output of the i th node (neuron) and vi is the weighted sum of the input connections. Alternative activation functions have been proposed, including the rectifier and soft plus functions. More specialized activation functions include radial basis functions (used in radial basis networks, another class of supervised neural network models).

## Code

#importing required libraries import numpy as np

import pandas as pd import nltk

import spacy

from tqdm import tqdm import os

import re

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

# load file names

pos\_files = os.listdir("C:/Users/sreya/Desktop/machineLearning/positve/") neg\_files = os.listdir("C:/Users/sreya/Desktop/machineLearning/negative/")

# read and store positive reviews in a list pos = []

for i in pos\_files: pos.append(read\_review("C:/Users/sreya/Desktop/machineLearning/positve/"+i))

# read and store negative reviews in a list neg = []

for i in neg\_files:

neg.append(read\_review("C:/Users/sreya/Desktop/machineLearning/negative/"+i))

#Data cleaning

replace\_1 = re.compile("(\.)|(\;)|(\:)|(\!)|(\')|(\?)|(\,)|(\")|(\()|(\))|(\[)|(\])") replace\_2 = re.compile("(<br\s\*/><br\s\*/>)|(\-)|(\/)")

#function to clean data

def clean\_reviews(reviews): reviews=[replace\_1.sub("",line.lower()) for line in reviews] reviews=[replace\_2.sub("",line.lower())for line in reviews] return reviews

# tokenization using spaCy def tokenization(x):

reviews\_tokens = [] for i in tqdm(x):

i = nlp(i) temp = [] for j in i:

temp.append(j.text) reviews\_tokens.append(temp)

return reviews\_tokens

# function to remove stopwords def strip\_stopwords(reviews):

s = []

for r in tqdm(reviews): s\_2 = []

for token in r:

if nlp.vocab[token].is\_stop == True: continue

else:

s\_2.append(token) s.append(" ".join(s\_2))

return s

#Building word cloud

all\_words = ' '.join([text for text in df['review\_cleaned']]) from wordcloud import WordCloud

wordcloud = WordCloud(width=800, height=500, random\_state=21, max\_font\_size=110).generate(all\_words)

plt.figure(figsize=(10, 7)) plt.imshow(wordcloud, interpolation="bilinear") plt.axis('off')

plt.show()

# build bag-of-words features for train data

from sklearn.feature\_extraction.text import CountVectorizer,TfidfVectorizer

bow = CountVectorizer(binary=False, min\_df=5, max\_df=1.0, ngram\_range=(1,2)) bow\_train = bow.fit\_transform(train['review\_cleaned'])

# build TF-IDF features for train data

tfidf = TfidfVectorizer(use\_idf=True, min\_df=5, max\_df=1.0, ngram\_range=(1,2), sublinear\_tf=True)

tfidf\_train = tfidf.fit\_transform(train['review\_cleaned']) # create features for validation and test set

bow\_val = bow.transform(val['review\_cleaned']) tfidf\_val = tfidf.transform(val['review\_cleaned'])

bow\_test = bow.transform(test['review\_cleaned']) tfidf\_test = tfidf.transform(test['review\_cleaned'])

print('BOW model:> Train features shape:', bow\_train.shape, ' Validation features shape:', bow\_val.shape,

' Test features shape:', bow\_test.shape)

print('TFIDF model:> Train features shape:', tfidf\_train.shape, ' Validation features shape:', tfidf\_val.shape,

' Test features shape:', tfidf\_test.shape) #split data

#After cleaning and pre-processing of the reviews,

#we can now split our data into 3 sets — train, validation, #and test — in the ratio of 70%, 15% and 15%, respectively

# splitting data into train, test, and validation set from sklearn.model\_selection import train\_test\_split

train, temp = train\_test\_split(df, stratify = df['class'], test\_size=0.3, random\_state=42)

test, val = train\_test\_split(temp, stratify = temp['class'], test\_size=0.5, random\_state=42)

# build bag-of-words features for train data

bow = CountVectorizer(binary=False, min\_df=5, max\_df=1.0, ngram\_range=(1,2)) bow\_train = bow.fit\_transform(train['review\_cleaned'])

# build TF-IDF features for train data

tfidf = TfidfVectorizer(use\_idf=True, min\_df=5, max\_df=1.0, ngram\_range=(1,2), sublinear\_tf=True)

tfidf\_train = tfidf.fit\_transform(train['review\_cleaned']) # create features for validation and test set

bow\_val = bow.transform(val['review\_cleaned']) tfidf\_val = tfidf.transform(val['review\_cleaned'])

bow\_test = bow.transform(test['review\_cleaned']) tfidf\_test = tfidf.transform(test['review\_cleaned'])

print('BOW model:> Train features shape:', bow\_train.shape, ' Validation features shape:', bow\_val.shape,

' Test features shape:', bow\_test.shape)

print('TFIDF model:> Train features shape:', tfidf\_train.shape, ' Validation features shape:', tfidf\_val.shape,

' Test features shape:', tfidf\_test.shape)

#Model Building

from sklearn.neural\_network import MLPClassifier mlp=MLPClassifier(activation='relu', alpha=0.0001, batch\_size='auto', beta\_1=0.9,

beta\_2=0.999, early\_stopping=False, epsilon=1e-08, hidden\_layer\_sizes=(100,), learning\_rate='constant', learning\_rate\_init=0.001, max\_iter=200, momentum=0.9, n\_iter\_no\_change=10, nesterovs\_momentum=True, power\_t=0.5, random\_state=None, shuffle=True, solver='adam', tol=0.0001, validation\_fraction=0.1, verbose=False, warm\_start=False)

# train model on bag of words mlp.fit(bow\_train,train['class'])

#make predictions on validation set bow\_val\_preds=mlp.predict(bow\_val) #make predictions on validation set bow\_test\_preds=mlp.predict(bow\_test)

# model evaluation

print("Validation accuracy score: ", accuracy\_score(val['class'], bow\_val\_preds)) print("Test accuracy score: ", accuracy\_score(test['class'], bow\_test\_preds))

mlp.fit(tfidf\_train, train['class']) #make predictions on validation set

tfidf\_val\_preds = mlp.predict(tfidf\_val)

#make predictions on test set tfidf\_test\_preds = mlp.predict(tfidf\_test) # model evaluaton

print("Validation accuracy score: ", accuracy\_score(val['class'], tfidf\_val\_preds)) print("Test accuracy score: ", accuracy\_score(test['class'], tfidf\_test\_preds))

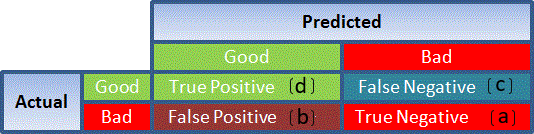
# SYSTEM TESTING

## SYSTEM TESTING

* 1. **Model Evaluation**

The process of model building is not complete without evaluation of model’s performance. Suppose we have the predictions from the model, how can we decide whether the predictions are accurate? We can plot the results and compare them with the actual values, i.e. calculate the distance between the predictions and actual values. Lesser this distance more accurate will be the predictions. Since this is a classification problem, we can evaluate our models using any one of the following evaluation metrics:

* **Accuracy**: Let us understand it using the confusion matrix which is a tabular representation of Actual vs Predicted values. This is how a confusion matrix looks like:



**Figure 6.1 : Accuracy**

* True Positive - Targets which are actually true(Y) and we have predicted them true(Y)
* True Negative - Targets which are actually false(N) and we have predicted them false(N)
* False Positive - Targets which are actually false(N) but we have predicted them true(T)
* False Negative - Targets which are actually true(T) but we have predicted them false(N)

Using these values, we can calculate the accuracy of the model. The accuracy is given by:

https://s3.amazonaws.com/thinkific/file_uploads/118220/images/e17/3a8/ae9/1549257180560.jpg

* **Precision**: It is a measure of correctness achieved in true prediction i.e. of observations labeled as true, how many are actually labeled true.

Precision = TP / (TP + FP)

* **Recall(Sensitivity)** - It is a measure of actual observations which are predicted correctly i.e. how many observations of true class are labeled correctly. It is also known as ‘Sensitivity’.

Recall = TP / (TP + FN)

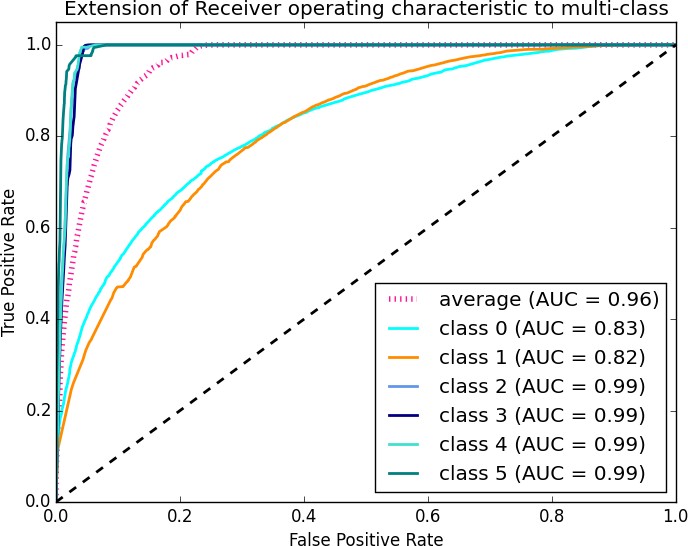
* **Specificity** - It is a measure of how many observations of false class are labeled correctly.

Specificity = TN / (TN + FP)

Specificity and Sensitivity plays a crucial role in deriving ROC curve.

* **ROC curve**
* Receiver Operating Characteristic(ROC) summarizes the model’s performance by evaluating the trade offs between true positive rate (sensitivity) and false positive rate(1- specificity).
* The area under curve (AUC), referred to as index of accuracy(A) or concordance index, is a perfect performance metric for ROC curve. Higher the area under curve, better the prediction power of the model.

This is how a ROC curve looks like:



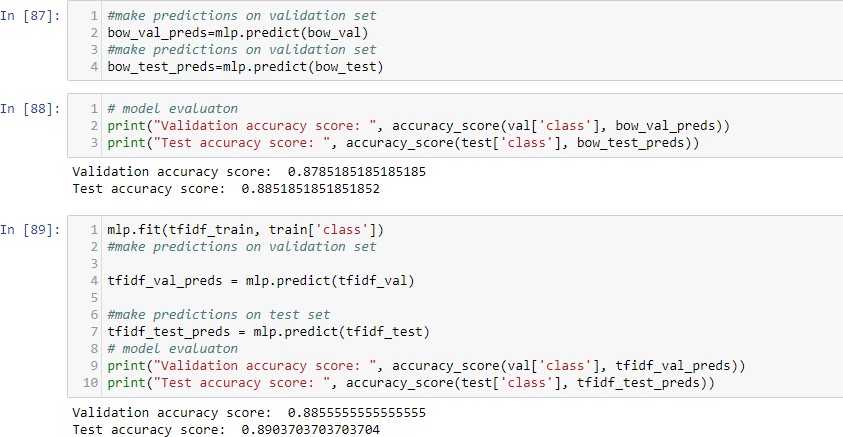
**Figure 6.2 : ROC curve**

* The area of this curve measures the ability of the model to correctly classify true positives and true negatives. We want our model to predict the true classes as true and false classes as false.
* So it can be said that we want the true positive rate to be 1. But we are not concerned with the true positive rate only but the false positive rate too. For example in our problem, we are not only concerned about predicting the Y classes as Y but we also want N classes to be predicted as N.
* We want to increase the area of the curve which will be maximum for class 2,3,4 and 5 in the above example.
* For class 1 when the false positive rate is 0.2, the true positive rate is around

0.6. But for class 2 the true positive rate is 1 at the same false positive rate. So, the AUC for class 2 will be much more as compared to the AUC for class 1. So, the model for class 2 will be better.

* The class 2,3,4 and 5 model will predict more accurately as compared to the class 0 and 1 model as the AUC is more for those classes

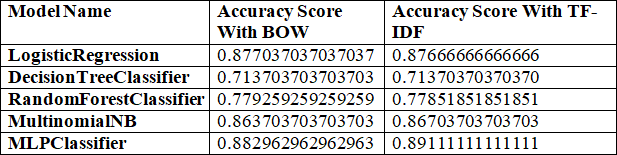
## Predicting Target



**Figure 6.3 : Predicting Target**

# RESULT ANALYSIS

## RESULT ANALYSIS



**Table 7.1 : RESULT ANALYSIS**

This project uses above mentioned models with both BOW and TF-IDF. The accuracies of the models are compared .The model which gets best accuracy is taken.

# CONCLUSION

## CONCLUSION

Review classification using NLTK, Spacy, Scikit-learn is a machine learning model which uses modern Natural language processing techniques. With this project, the people can easily classify the review of the movie without putting much effort

.The modern algorithms in this project plays key role in getting high accuracy for a given data. The MLP classifier gives the High accuracy for this project. Hence we can conclude that MLP classifier is the best fitting model for review classification.

# FUTURE ENHANCEMENTS

## FUTURE ENHANCEMENTS

Text classification is the activity of labeling natural language texts with relevant categories from a predefined set. In layman's terms, text classification is the process of extracting generic tags from unstructured text. These generic tags come from a set of predefined categories

# BIBLIOGRAPHY

## BIBLIOGRAPHY

1. Bing Liu,"Exploring User Opinions in Recommender Systems", Proceeding of the second KDD workshop on Large Scale Recommender Systems and the Netflix Prize Competition", Aug 24, 2008, Las Vegas, Nevada, USA.
2. Dave.D, Lawrence.A, Pennock.D,"Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews", Proceedings of International World Wide Web Conference (WWW‟03), 2003.
3. Turney, P, "Thumbs Up or Thumbs Down? SemanticOrientation Applied to Unsupervised Classification of Reviews", ACL‟02, 2002.

[4 ] <http://people.scs.carleton.ca/~armyunis/projects/KAPI/porter.pdf>

1. An Introduction to Logistic Regression Analysis and Reporting Article *in* The Journal of Educational Research · September 2002
2. [https://www.cs.princeton.edu/courses/archive/spr07/cos424/papers/mitchell-](https://www.cs.princeton.edu/courses/archive/spr07/cos424/papers/mitchell-dectrees.pdf) [dectrees.pdf](https://www.cs.princeton.edu/courses/archive/spr07/cos424/papers/mitchell-dectrees.pdf)
3. An Improved Random Forest Classifier for Text Categorization Baoxun Xu Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen 518055, China
4. <http://www.cs.cmu.edu/~knigam/papers/multinomial-aaaiws98.pdf>
5. Training Multilayer Perceptron Classifiers Based on a Modified Support Vector Method J. A. K. Suykens and J. Vandewalle