CLICKBAIT DETECTION

B.Tech. Major Project Report

BY

N RITVIKA REDDY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

RAIPUR- CG (INDIA)

MAY, 2016

CLICKBAIT DETECTION

A Major Project Report

submitted in partial fulfillment of the

requirements for the award of the degree

of

Bachelor of Technology

in

COMPUTER SC. & ENGINEERING

BY

N RITVIKA REDDY

DEPARTMENT OF COMPUTER SC. & ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

RAIPUR , CG (INDIA)

MAY, 2016

DECLARATION

I hereby declare that the project entitled “**Clickbait Detection**” submitted in partial fulfilment of the requirements for the award of the degree of B.Tech. in Computer Science & Engineering to the Department of Computer Science & Engineering, NIT Raipur, is our original work and the project has not formed the basis for the award of any other degree associate-ship, fellowship or any other similar titles.

Signature of the Student:	Place:
N Ritvika Reddy	Date:
R.No. 12115045	

CERTIFICATE

I hereby certify that the work which is being presented in the B.Tech. Major Project Report entitled **“Clickbait Detection”,** in partial fulfillment of the requirements for the award of the **Bachelor of Technology in Computer Sc. & Engineering** and submitted to the Department of Computer Sc. & Engineering of National Institute of Technology Raipur is an authentic record of my own work carried out during a period from Jan 2016 to May 2016 under the supervision of **Mr. Dilip Singh Sisodia, Assistant Professor, CSE Department**.

The matter presented in this thesis has not been submitted by me for the award of any other degree elsewhere.

Signature of Candidate

N RITVIKA REDDY

R.No. 12115045

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date: ***Signature of Supervisor(s)***

Mr. Dilip Singh Sisodia

Assistant Professor

Head

Computer Sc. & Engineering Department

National Institute of Technology Raipur CG

ACKNOWLEDGEMENT

I am deeply thankful to my project supervisor, Mr.Dilip Singh Sisodia for helping me throughout the course in accomplishing this project. His enthusiastic guidance and support enabled me in achieving the objectives of the project.

Signature of Candidate

N RITVIKA REDDY

R.No. 12115045

ABSTRACT

The emergence of the Web has given us many services. One of these is the access to worldwide happenings in the form of online news. The online publishing of news has led to the cropping up of “clickbaits”. Clickbaits are articles on the internet with content whose main purpose is to attract attention and encourage visitors to click on a link to a particular web page [1]. They exaggerate the content of the page keeping in mind the end goal to acquire income by the clicks received. Such article headlines lead to spread of false news and some kind of system is required to distinguish an authentic news headline from a clickbait. We identify 19 features of the headlines which are to be used for classifying a given headline into the clickbait class or non-clickbait class. A model to identify whether a given phrase is a clickbait or not can be developed by combining natural language processing techniques with the classifiers of machine learning. We use 3 simple learners and 3 ensemble learners and compare their performance based on different traiing sets used. We conclude that the C4.5 decision tree classifier detects clickbaits better than the other classifiers with an accuracy of 0.9161, a total precision of 0.92, recall 0f 0.92 and f-measure of 0.91.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

CHAPTER - 1
INTRODUCTION TO CLICKBAITS

Chapter 1 - Introduction To Clickbaits

1.1 Introduction

In this chapter, we give an introduction about clickbaits in Section 1.2. The motivation of the project is presented in Section 1.3 and the objective in Section 1.4.

1.2 Clickbaits

The availability of news on the internet can be considered both a boon and a bane. As a boon, it provides us with the opportunity to access worldwide news just at the click of a button or the press of a key. But, as a bane the availability of news on the internet has created a frenzy among various websites to earn more income. Since websites are paid according to the number of clicks they receive, many websites exaggerate and sensationalise headlines of various articles to generate more clicks. These headlines are known as Clickbaits. Such articles do not deliver on their headline's promise and in fact waste the time of the reader. Clickbait is sometimes considered as a new type of content marketing.

As given in the Merriam Webster dictionary, a clickbait is usually a headline designed to make readers want to click on hyperlinks especially when the links lead to content of dubious value or interest [2]. In any case, clickbaits are not so much spam or fake pages. They can be authentic pages conveying low quality substance with deluding titles.

Clickbaits are the kind of headlines which intentionally withhold information from the readers. They make individuals considerably more inquisitive by giving them something they know a tad bit about, yet not all that much. This leads to a curiosity gap in the minds of the readers. This has been explained by George Loewenstein as the information gap theory of curiosity. He defines curiosity as the intrinsic human behaviour that is activated when people feel there is a gap between what they know and what they need to know [3]. It is stated in [3] that theory views curiosity as emerging at the point when consideration gets to be centered around a gap in one's information. Such information gaps produce the feeling of deprivation called curiosity. The inquisitive individual is inspired to get the missing data to diminish or eliminate the curiosity. Clickbaits create this gap in the mind of the readers by using words which have a tendency to lure the reader into opening the link. Some organizations purportedly depend for the most part on clickbait for their traffic.

Most clickbaits fail on their promise of delivering an intriguing story on the web page it points to thus, disappointing the reader. Words commanding authority are generally used in clickbaits to assure the reader that what the article has to see is genuine and not made-up [4]. Since the headlines consist of words used in such a manner to create an enormous enticing impact on the readers, natural language processing techniques can help in extracting the features of those clickbaits and then we can use classifiers to categorise any given phrase.

The thing with clickbaits is that most people are frequently mindful of this manipulation, but then powerless to oppose it. It is stated in [5] that this has a considerable measure to do with emotion and the part it plays in our daily decision-making processes. Emotional arousal, or the level of physical reaction you have to a feeling, is a key ingredient in clicking behaviours. Another reason stated in [5] as to why clickbaits work is the anticipation of pleasure in humans. Reading a clickbait headline, which promises some reward of cute pictures, or anything of that sort, itself creates a sort of pleasure in humans which makes them go on and click on the link. The headline itself is said to give pleasure even before the web page opens.

Clickbait ordinarily has a few of the accompanying qualities as stated in [6]:

- An eye-catching and convincing feature
- Effectively skimmable
- Funny or memorable images or videos
- Humorous tone or offers strongly to a particular emotion
- Intended to encourage social sharing

Natural language processing tasks like parts-of-speech tagging and sentiment analysis can be used to extract the features of the given data. Modern NLP algorithms are based on machine learning, especially statistical machine learning. The paradigm of machine learning is different from that of most prior attempts at language processing. Prior implementations of language-processing tasks typically involved the direct hand coding of large sets of rules. The machine-learning paradigm calls instead for using general learning algorithms (often, although not always, grounded in statistical inference) to automatically learn such rules through the analysis of large corpora of typical real-world examples. A corpus (plural, "corpora") is a set of documents (or sometimes, individual sentences) that have been hand-annotated with the correct values to be learned.

Many different classes of machine learning algorithms have been applied to NLP tasks. These algorithms take as input a large set of "features" that are generated from the input data. Some of the earliest-used algorithms, such as decision trees, produced systems of hard if-then rules similar to the systems of hand-written rules that were then common. Increasingly, however, research has focused on statistical models, which make soft, probabilistic decisions based on attaching real-valued weights to each input feature. Such models have the advantage that they can express the relative certainty of many different possible answers rather than only one, producing more reliable results when such a model is included as a component of a larger system.

Classification techniques can also be divided into a two categories: Supervised vs. unsupervised and non-adaptive vs. adaptive/reinforcement techniques. Supervised approach is when we have pre-labeled data samples available and we use them to train our classifier. Training the classifier means to use the pre-labeled to extract features that best model the patterns and differences between each of the individual classes, and then classifying an unlabeled data sample according to whichever pattern best describes it. Unsupervised classification is when we do not have any labeled data for training. In addition to this adaptive classification techniques deal with feedback from the environment.

1.3 Motivation

Clickbaits can be considered as a tool used by websites to boost the amount of clicks they receive. This trick does not benefit the user in any good manner. There are no reported advantages of clickbaits. Clickbaits serve as a reading material with not much useful information for users browsing the web in their leisure time. Clickbaits may sometimes serve as marketing tools for few companies. Websites implement clickbaits to augment their revenues. Clickbaits have been so common that the line between actual news and clickbaits is becoming very thin and vague.

Thus, clickbait detection is useful in saving the time of people on the internet and to prevent the spread of false or inconsistent news on the web.

1.4 Objective

In this project, we aim at collecting a sufficiently large amount of data consisting of a combination of clickbaits and news headlines (collected from authentic sources) and then randomly dividing the entire data into different sets which are used as inputs to various machine learning algorithms. These machine learning algorithms work on some selected features of the data which are extracted using natural language processing techniques.

We also aim at comparing the performances of the machine learning algorithms given the selected features.

CHAPTER - 2

LITERARY REVIEW

Chapter 2 - Literary Review

2.1 Introduction

In this chapter, we review the literature related to our project. The limitations of the past works are described in Section 2.2 and all the related works are summarised in Section 2.3.

2.2 Limitations Of Previous Works

Not much research has been done in the field of detecting clickbaits. Potthast et al proposed a model to detect clickbaits by employing three algorithms – the naïve Bayes, logistic regression and random forest [5]. They use features extracted from the tweet headline, the website it links to and the metadata of the tweet. Biyani et al presented a machine learning model to detect clickbaits [7]. They analysed properties of clickbait and non-clickbait articles by extracting features from the headline of the clickbait as well as its body or content and the url of the webpage. Their gradient boosted decision tree model achieved a F-1 score of 74.9% in predicting clickbaits.

2.3 Related Works

Many companies use clickbait to attract more users to their websites to increase traffic. The links to these web pages are shared and posted publicly in many social networking sites. Their prosperity on those networks recently made Facebook make a move against clickbait as announced in [8]. In an initial survey conducted by the authors of [8] about what type of content users preferred to see, 80% of the time individuals favored features that helped them choose if they wanted to peruse the full article before navigating away. However, not much publication is available about Facebook’s clickbait filtering techniques.

Another publication which can be considered is [9]. Vijgen studies articles that consist of lists of things and such lists are called as “listicles”. The homogenous structure of the titles of these listicles is similar to that of the clickbait headlines. It is observed in [9] that all the titles contain a cardinal number and that 85% of all titles begin with this cardinal number. It is also seen that the numbers between 5 and 25 are more popular and also that an odd number is preferred to an even number. He observed that the most prominent nouns in the titles consist of “things”, “reasons”, “signs”, and “ways”. The most seen adjectives are observed to be “best”, “worst”, “awesome”, “great”, “amazing”, “delightful” and “beautiful”. These nouns and adjectives are used to convey power and melodrama. These observations can be taken into consideration when detecting clickbaits.

Blom et al study phorocity in headlines as a way to stimulate interest [4]. They analysed 2000 random headlines collected from a Danish news website and distinguished two forms of forward-references – discourse deixis (references at discourse level) and cataphora (discourses at phrase level) expressed by demonstrative pronouns, personal pronouns, adverbs and definite articles but have not proposed any approach for their detection.

Reis et al analysed 69,907 news headlines from majors news publication websites [10]. They performed sentiment analysis of the headline and discovered that the sentiment of the headline is unequivocally identified with the prevalence of the news.

Potthast et al gathered and manually annotated a corpus of clickbaits from Twitter. Their corpus is publicly available and consists of 3000 tweets [5]. They collected the tweets from the top 20 news publishers on Twitter, examined what amount of their total content was clickbait and concluded that all of the news distributers utilize clickbaits on a normal premise. Their proposed detection model is based on features extracted from the clickbait headline, the website it is linked to and the meta information associated with it. Out of the total 215 features they use, 203 features are extracted from the clickbait headline, 8 features from the webpage the clickbait points to and 4 features from the metadata of the tweets. They used a 2:1 training-testing ratio of samples in the dataset and balanced the training data before training by oversampling clickbaits. They employed the Naïve Bayes algorithm, Logistic Regression algorithm and Random Forest algorithm using their default parameters. They measured precision and recall for the clickbait class, and ROC-AUC (area under the curve AUC of the receiver operating characteristic ROC) to evaluate and compare the algorithms used. Their model achieved an ROC-AUC of 0.74 with random forest, 0.72 with logistic regression and 0.69 with naïve Bayes for all features combined. The precision scores remained almost the same for all three classifiers while the recall scores varied from 0.66 with naïve Bayes to 0.73 with random forest.

They conclude that the features of the clickbait headline outperformed all features combined based on precision, recall and ROC-AUC with naïve Bayes and random forest.

Biyani et al collected 1349 clickbait and 2724 non-clickbait pages by writing their own parser and scraper and randomly split the data into a training set containing 3000 samples and a testing set containing 1073 samples [7]. They employed gradient boosted decision trees by selecting parameters as 500 trees, 20 leaf nodes, 0.1 shrinkage and 0.5 sampling rate to perform the classification. They used 5-fold cross validation. Biyani et al suggest that clickbaits are different from spam and fake pages and hence features like blacklists of urls, host and IPs and link-structure cannot be employed in detecting clickbaits. They also categorize clickbaits into 8 types – exaggeration, teasing, inflammatory, formatting, graphic, bait-and-switch, ambiguous and wrong. They suggest that in addition to content features, features such as informality of a web page, its url and similarity between its title and body are solid markers of it being a clickbait. They use informality as one of the features. Since, clickbaits are simply low quality web pages, the language used tend be very informal when compared with the language of professionally written authentic news articles. The performance of their model was evaluated based on precision, recall and F-1 score. They observed that their model has a 0.755 precision and 0.760 recall on the test set. They also observed a 0.712 precision and a 0.548 recall for the clickbait class while a 0.752 precision and a 0.842 recall for the non-clickbait class. They justify the better performance for the non-clickbait class by attributing it to the presence of more number of non-clickbait samples in the entire data. They also conclude that informality and forward-reference features lead to the best performance while the performance of all the features combined is better than performances of individual features.

CHAPTER - 3
PROCESS FLOW AND
DATA COLLECTION

Chapter 3 - Process Flow And Data Collection

3.1 Introduction

In this chapter we enumerate the steps followed in the detection of clickbaits in Section 3.2. The method of collecting data is described in Section 3.3. In Section 3.4, the data sets used in the project are defined.

3.2 Steps Involved in the Project

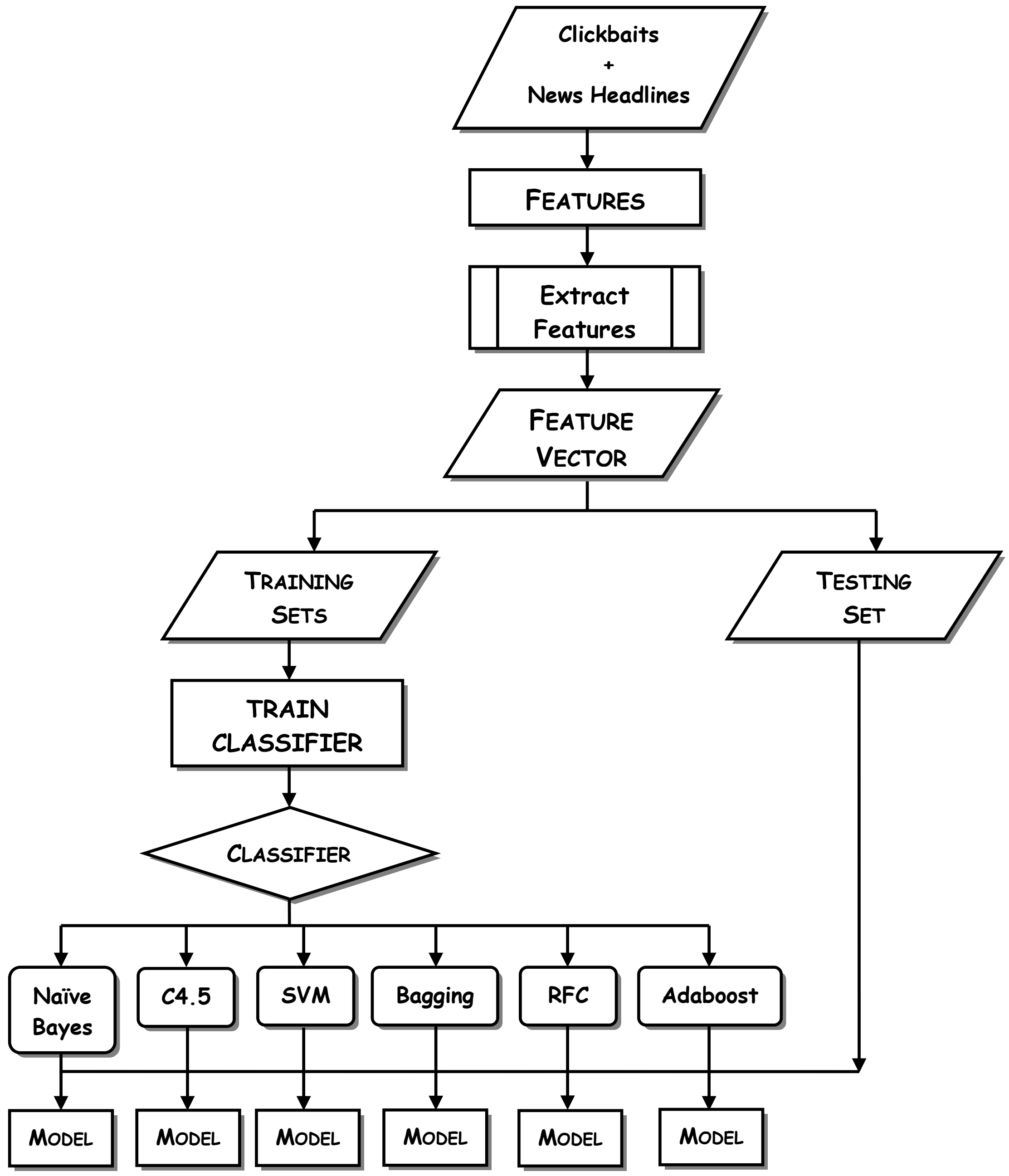

Figure 1: Process Flow Diagram

The process followed in this project is shown in Figure 1. The foremost thing we do is collect the clickbait and authentic headline samples. Then, we decide the various features based on which we perform the classification. These features are then extracted by specific codes. Each headline (clickbait or authentic news) is represented by its feature vector. All the feature vectors of the training samples are passed as input to the classifier along with the class label they belong to and thus the classifiers undergo learning. After the learning of the classifiers is completed, the feature vectors of the testing samples are sent as input to the models and are labeled by the classifiers.

The features which are used are described in Chapter 4 and the classifiers implemented in Chapter 5.

3.3 Collecting Data

The clickbaits used in this project were downloaded from [11]. We have used the titles of articles published on Buzzfeed - a popular social news and entertainment website. The entire data consists of around 60,000 clickbaits published during 2014.

The authentic news headlines were collected from Reuters, Associated Press, and The New York Times. They were retrieved using the Article Search API of the New York Times API available for Python. These headlines were published from January, 2013 to March, 2016. We have collected around 40,000 news headlines.

3.4 Pre-processing of Data

3.4.1 TRAINING DATA

We randomly divide the entire data collected into different data sets. Table 1 shows the composition of all the training data sets.

The first 3 data sets are balanced i.e. they contain equal samples of clickbaits and news headlines while the last 2 data sets contain unbalanced data. We use these 5 data sets as training inputs to the classifiers, one at a time.

Table 1: Composition of Training Data Sets

Name	Clickbaits	Headlines	Total
DS1	5,000	5,000	10,000
DS2	10,000	10,000	20,000
DS3	15,000	15,000	30,000
DS4	5,000	10,000	15,000
DS5	10,000	5,000	15,000

3.4.2 Testing Data

We use the 5 training sets themselves as test sets for the classifiers and thus perform self-validation of the data.

CHAPTER - 4

FEATURE SELECTION AND EXTRACTION

Chapter 4 - Feature Selection And Extraction

4.1 Introduction

In this chapter, the features which are selected are described in Section 4.2 and the manner of their extraction from the headline is summarised in Section 4.3.

4.2 Selecting Features

Feature selection is a process where you automatically select those features in your data that contribute most to the prediction variable or output in which you are interested. Feature selection can be utilized to recognize and eliminate unneeded, irrelevant and redundant attributes from data that do not add to the accuracy of the classifier or may at times tend to diminish the accuracy of the model.

Feature selection is used to improvement of models to make them less demanding to translate by analysts, shorter training times, and enhanced speculation by lessening overfitting.

We have considered 19 attributes to be selected as features to train the classifiers. All these features are extracted only from the title of the article and we do not consider the content of the article. In [7], the authors consider both the attributes of the title of the article as well the first few lines of the body of the article.

The features we have considered are as follows:

1. Acronyms:

   An acronym is an abbreviation which is generally used in content published on the internet. Users are more pulled in towards brief titles and using acronyms is one way of ensuring that the title still grabs the attention of the reader despite the fact that it is short.

   In the following examples, DIY stands for “Do It Yourself” while OMG stands for “Oh My God”.

   E.g.:

   “25 Awesome DIY Ideas For Bookshelves”

   “He Couldn't Finish The National Anthem...And That's When The Crowd Came

In. OMG. I'm Crying”

2. Adv_adj:

   The titles need to be presented in such a manner that they instantly capture the eye of the reader. Adjectives and adverbs are words which describe a noun in a manner which appeals to the readers. The presence of such words stir emotions in the reader, thus making him open the link.

   E.g.:

   “Celebrities Riding Invisible Bikes Is Weirdly Hilarious”

   “11 CREEPY Things Kids Said To Their Parents About Their Imaginary Friends”

3. Baity_words:

   One thing that can be observed very easily is that most of the clickbaits use some phrases or words all the time and hence we can make use of such a list of words to detect a clickbait [7] . The commonly used baity words are – “click”, “happens”, “next”, “believe”, “reasons”, “ways”, “things”.

   E.g.:

   “The 12 Craziest Conspiracy Theories That American Voters Believe”

   “Someone Dressed This Dog Up As A Giant Mutant Spider...And What Happens Next Is Hilarious “

4. Caps_words:

   The use of capitalised letters in any headline makes it stand out in the entire web page and captures the eye of the reader. Clickbaits use capitalised words to be regarded by the reader so that the readers opens it.

   E.g.:

   “ A TATTOO BAN on Capitol Hill”

   “CAN'T BE UNSEEN: 30 Art History Snapchats That Are So Inappropriate, But SO Funny”

5. Cont_nums:

   It is stated in [12] that numbers tend to stand out when a person is looking through an unending stream of headlines – especially odd numbers. The numbers likewise measure story length and indicate the amount of time a user needs to peruse the story.

   E.g.:

   “22 New Ways To Use Your Printer”

   “Sweet Recycling Game Made By An 11-Year-Old”

6. Cont_the:

   ‘The’ is a determiner which is generally used to draw emphasis to the accompanying noun. Authentic news headlines refrain from using ‘the’ in most of their headlines while clickbaits use it a lot.

   E.g.:

   “The 24 Nerdiest Things That Have Ever Happened”

   “The Best Ever Song About Cats”

7. Demonstratives:

   Demonstrative pronouns are the same pronouns used for demonstrative adjectives – “this”, “that”, “these” and “those”. The difference is in the sentence structure. The demonstative pronoun takes the place of the noun phrase.

   E.g.:

   “These 13 Insanely Clever Paint Jobs Will Have You Doing Mind Flips”

   “This Mischievous Squirrel Stole A GoPro...Can You Guess What Happens Next?”

8. Exclamations:

   Exclamation marks are sometimes used in clickbait headlines to display high emotions and to fall under the radar of the reader.

   E.g.:

   “These Paint Splatters Are More Genius Than You'd Ever Expect. The Owl Is

FLAWLESS!”

“Check Out All This Cool Vintage Redskins Gear!!!”

9. Neg_words:

   Words conveying negative sentiment are also used to stir up the feelings of the readers and lure them into opening the link.

   E.g.:

   “Do Heat Fans Get A Bad Rap, Or Are They Even Worse Than We Think?”

   “New Things Can Be Scary, Even For Corgis”

10. Pos_words:

    Words conveying positive sentiment are also used in clickbait headlines.

    E.g.:

    “People Who Think Classic Linkin Park Is Awesome”

    “Nine Inch Nails Have A New Song And Its Amazing”

11. Quoted_words:

    Single quotations or double quotations are used in clickbait headlines more than in authentic news headlines to draw attention to its content.

    E.g.:

    “Patrick Stewarts Appearance On "The Arsenio Hall Show" Is Pure Awesome”

    “Pepsi Made "Trendy" Clothes In The 80s”

12. Question_marks:

    Question marks are used to give more emphasis to the question being asked in the clickbait headline.

    E.g.:

    “Do Mitch And Cam Actually Hate Each Other?”

    “Do Games Shape Our Dreams?”

13. Starts_adv:

    Headlines that start with an adverb are more successful in drawing attention to themselves. Clickbaits usually start with adverbs, as it also creates an instant curiosity in the reader.

    E.g.:

    “Beautiful Golden Retriever Puppies”

    “Exclusive: Shakeup At NSA After BuzzFeed News Reports On Potential Conflict Of Interest”

14. Start_num:

    Headlines that start with a number also fall under the eye of the reader as such headlines usually point to lists and lists are something which lure the readers most often.

    E.g.:

    “9 Coffee Swirl Art Masterpieces”

    “7 Mind-Bending Abduction Stories”

15. Swear_words:

    Clickbaits also contain swear words or bad words which are unprofessional in nature and are never used in authentic headlines. We can consider the presence or number of swear words present in a headline to detect if it is a clickbait or not. We use the list of bad words available in [13].

    E.g.:

    “People Are Shipping The Weirdest Sh*t And Its Kind Of Awesome”

    “Disney Put Out An EDM Version Of Let It Go And Its Pretty Damn Good”

16. Third_pronouns:

    Third pronouns like “he”, “she”, “his”, “her”, “they”, “it”, and “them” are also used regularly in clickbait headlines.

    E.g.:

    “This Woman Has Never Won The Lottery, But She Was Still Able To Ruin Her

Own Life”

“We Could All Use More Dancing Shetland Ponies In Our Lives”

17. Words_5w1h:

    A study is cited in [14] which suggests that provoking interest in headlines all comes down to making inquiries that reference the reader. They tested three kinds of headlines – a simple declarative headline, a headline posing a question and self-referencing headlines using the second person. The results showed that headlines containing questions received more clicks.

    E.g.:

    “What About Larry King?”

    “How Many Hours Of Sleep Should You Get?”

18. Words_cont_repeated_chars:

    Some clickbaits use words containing repeated characters as a way of displaying a greater amount of emotion in it. Authentic news headlines do not contain such words.

19. Words_title:

    The clickbait headlines tend to be larger than authentic news headlines but smaller than regular sentences. The number of words in the headline (or title) can be a feature used to distinguish between clickbaits and news headlines.

The features of the phrase to be classified are summarised in Table 2.

Table 2: Features to be Extracted

S.NO	FEATURE	TYPE
1	Acronyms	Numeric
2	Adv_adj	Numeric
3	Baity_words	Numeric
4	Caps_words	Numeric
5	Cont_nums	Numeric
6	Cont_the	Numeric

7	Demonstratives	Numeric
8	Exclamations	Numeric
9	Neg_words	Numeric
10	Pos_words	Numeric
11	Quoted_words	Numeric
12	Question_words	Numeric
13	Start_adv	Boolean
14	Starts_num	Boolean
15	Swear_words	Numeric
16	Third_pronouns	Numeric
17	Words_5w1h	Numeric
18	Words_cont_repeated_chars	Numeric
19	Words_title	Numeric

4.3 Extracting Features

All the features that have been selected are extracted from each headline using codes written in Python. The manner in which they were extracted are as follows:

1. Acronyms:

   Since acronyms are usually capitalised words and their length is usually less than or equal to 3 characters, we use this logic to identify acronyms.

2. Adv_adj:

   To count the number of adjectives or adverbs present in a headline, we perform parts-of-speech tagging of the entire headline first and then check for the tags – ‘JJ’, ‘JJR’, and ‘JJS’ for adjectives and ‘RB’, ‘RBR’, and ‘RBS’ for adverbs.

3. Baity_words:

   We check each word of the headline to see if it is present in our list of baity-words i.e if it is any of the following words- ‘click’, ‘happens’, ‘next’, ‘believe’, ‘reasons’, ‘ways’, ‘things’ and ‘exlusive’.

4. Caps_words:

   We use the isupper() function of python to check if a word is capitalised or not.

5. Cont_nums:

   We use the isdigit() function of python to check if the word under consideration is a number or not.

6. Cont_the:

   We check if the word is ‘the’ or not by simple comparison.

7. Demonstratives:

   We check the word to see if it is a ‘this’, ‘that’, ‘these’, or ‘those’

8. Exclamations:

   The word is compared with an exclamation mark ‘!’.

9. Neg_words:

   We use the sentiment polarity attribute present in the TextBlob package for Python to find out the sentiment of the word in the headline. If the value of the sentiment of the word is a negative value (less than zero), then the word is a negative word.

10. Pos_words:

    If the value of the sentiment of the word is a positive value (greater than zero), then the word is a positive word.

11. Quoted_words:

    We use regular expressions to check if the word is surrounded by either ‘single quotes’ or “double quotes”.

12. Question_marks:

    The word is compared with a question mark ‘?’.

13. Starts_adv:

    From the parts-of-speech tags generated, we check if the tag of the first word of the headline is ‘RB’, ‘RBR’, or ‘RBS’ to see if the headline starts with an adverb or not.

14. Start_num:

    We use the isdigit() function of Python on the first word of the headline to check if the headline starts with a number or not.

15. Swear_words:

    We check of the word is anyone of the words present in the Bad-Words list [16].

16. Third_pronouns:

    We check if the word is any one of the following – ‘he’, ‘she’, ‘it’, ‘they’, ‘his’, ‘her’, or ‘them’.

17. Words_5w1h:

    We check if the word is any one of – ‘what’, ‘why’, ‘who’, ‘which’, ‘when’ or ‘how’.

18. Words_cont_repeated_chars:

    We check if any character or characters of the word are repeated more than 2 times.

19. Words_title:

    The total number of the words in the headline are counted.

CHAPTER - 5

MACHINE LEARNING ALGORTIHMS IMPLEMENTED

Chapter 5 – Machine Learning Algorithms Implemented

5.1 Introduction

In this project, we have used 3 simple learners and 3 ensemble learners available in the Scikit-Learn package [15] for Python. Figure 2 shows the classifiers which are used.

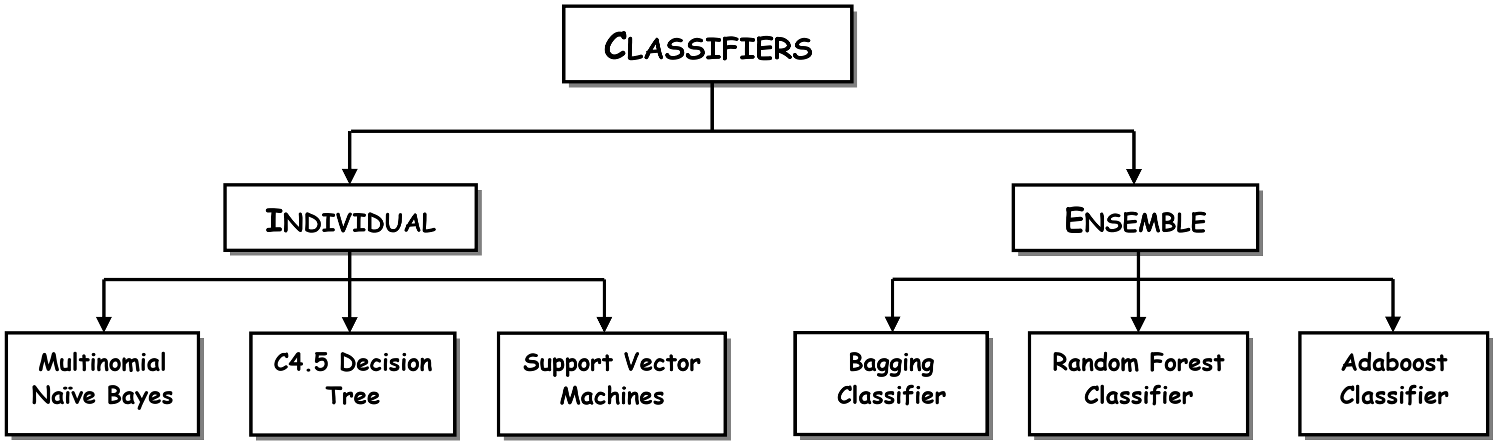

Figure 2 : Classifiers Used

5.2 Individual Learners

The purpose of supervised learning is to classify patterns (also known as instances) into a set of categories which are also referred to as classes or labels. Commonly, the classification is based on a classification models (classifiers) that are induced from an exemplary set of preclassified patterns. Alternatively, the classification utilizes knowledge that is supplied by an expert in the application domain. In a typical supervised learning setting, a set of instances, also referred to as a training set is given. The labels of the instances in the training set are known and the goal is to construct a model in order to label new instances. An algorithm which constructs the model is called inducer and an instance of an inducer for a specific training set is called a classifier.

The individual (simple) learners or classifiers that we employ in this project are the multinomial naive Bayes classifier, the C4.5 decision tree classifier and the support vector machine classifier.

5.2.1 MULTINOMIAL NAIVE BAYES CLASSIFIER

Multinomial Naive Bayes is a particular variant of Naive Bayes that is composed more for text documents. Whereas simple Naive Bayes would model a document as the presence and absence of particular words, Multinomial Naive Bayes explicitly models the word counts and adjusts the underlying calculations to deal with in.

The multinomial model catches word frequency data in records [16]. The manner in which the multinomial naive Bayes computes class probabilities as given in [17] is mentioned in Eq 5.2.1a :

$$\boldsymbol{P(c|t_i) = \frac{P(c)P(t_i|c)}{P(t_i)}, \qquad c \in C} \qquad [5.2.1a]$$

Where

$\boldsymbol{C}$ denotes the set of classes and

test document $\boldsymbol{t_i}$ is assigned to class $\boldsymbol{c}$.

The second term of the numerator is calculated as shown in Eq 5.2.1b:

$$\boldsymbol{P(t_i|c) = \left(\sum_n f_{ni}\right)! \prod_n \frac{P(w_n|c)^{f_{ni}}}{f_{ni}!}} \qquad [5.2.1b]$$

Where

$\boldsymbol{f_{ni}}$ is the count of word $\boldsymbol{n}$ in our test document $\boldsymbol{t_i}$ and

$\boldsymbol{P(w_n|c)}$ is the probability of word $\boldsymbol{n}$ given class $\boldsymbol{c}$ (defined in Eq 5.2.1c)

$$\boldsymbol{\widehat{P}(w_n|c) = \frac{1+F_{nc}}{N+\sum_{x=1}^{N} F_{xc}}} \qquad [5.2.1c]$$

Where

$\boldsymbol{F_{xc}}$ is the count of word $\boldsymbol{x}$ in all the training documents belonging to class $\boldsymbol{c}$

The probability of occurrence of term document $\boldsymbol{t_i}$ is given in Eq 5.2.1d.

$$\boldsymbol{P(t_i) = \sum_{k=1}^{|C|} P(k)P(t_i|k)} \qquad [5.2.1d]$$

The computationally expensive terms in Eq 5.2.1b can be deleted because neither depends on the class, and hence it becomes Eq 5.2.1e .

$$\boldsymbol{P(t_i|c) = \alpha \prod_n P(w_n|c)^{f_{ni}}} \qquad [5.2.1e]$$

Where $\boldsymbol{\alpha}$ is a constant.

5.2.2 DECISION TREE CLASSIFIER

Decision tree learning uses a decision tree as a predictive model which maps observations about an item to conclusions about the item's target value. It is one of the predictive modelling approaches used in statistics, data mining and machine learning. Tree models where the target variable can take a finite set of values are called classification trees. In these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees.

C4.5 is an algorithm used to generate a decision tree developed by Ross Quinlan. It is an extension of Quinlan's earlier ID3 algorithm. The decision trees generated by C4.5 can be used for classification, and for this reason, C4.5 is often referred to as a statistical classifier.

As mentioned in [18], C4.5 algorithm has many features like:

- Speed - C4.5 is significantly faster than ID3 (it is faster in several orders of magnitude)
- Memory - C4.5 is more memory efficient than ID3
- Size of decision Trees – C4.5 gets smaller decision trees.
- Ruleset - C4.5 can give ruleset as an output for complex decision tree.
- Missing values – C4.5 algorithm can respond on missing values by ‘?’
- Overfitting problem - C4.5 solves overfitting problem through Reduce error pruning technique.

At each node of the tree, C4.5 chooses the attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. The splitting criterion is the normalized information gain (difference in entropy). The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then recurs on the smaller sublists.

This algorithm has a few base cases.

- All the samples in the list belong to the same class. When this happens, it simply creates a leaf node for the decision tree saying to choose that class.
- None of the features provide any information gain. In this case, C4.5 creates a decision node higher up the tree using the expected value of the class.
- Instance of previously-unseen class encountered. Again, C4.5 creates a decision node higher up the tree using the expected value.

The required values in C4.5 algorithm are calculated as follows [19]. The information of database is calculated as shown in Eq 5.2.2a.

$$\boldsymbol{info(T)} = -\sum_{\boldsymbol{j=1}}^{\boldsymbol{NClass}} \left(\frac{\boldsymbol{freq(C_j,T)}}{\boldsymbol{|T|}}\right) \times \mathbf{log_2}\left(\frac{\boldsymbol{freq(C_j,T)}}{\boldsymbol{|T|}}\right) \qquad [5.2.2a]$$

Where

$\boldsymbol{T}$ is a set of cases.

The gain for an attribute $\boldsymbol{a}$ is calculated as shown in Eq 5.2.2b:

$$\boldsymbol{gain} = \boldsymbol{info(T)} - \sum_{\boldsymbol{i=1}}^{\boldsymbol{s}} \frac{\boldsymbol{|T_i|}}{\boldsymbol{|T|}} \times \boldsymbol{info(T_i)} \qquad [5.2.2b]$$

The split info for any attrtibute for a set of test cases is described in Eq 5.2.2c:

$$\boldsymbol{Split(T)} = -\sum_{\boldsymbol{i=1}}^{\boldsymbol{s}} \frac{\boldsymbol{|T_i|}}{\boldsymbol{|T|}} \times \mathbf{log_2}\left(\frac{\boldsymbol{|T_i|}}{\boldsymbol{|T|}}\right) \qquad [5.2.2c]$$

The attribute with the highest gain ratio is selected as the next node for partitioning. The gain ratio is calculated as shown in Eq 5.2.2d.

$$\boldsymbol{gain\,ratio(T)} = \frac{\boldsymbol{gain}}{\boldsymbol{Split(T)}} \qquad [5.2.2d]$$

The algorithm for the C4.5 decision tree classifier is shown below.

Algorithm: C4.5 Decision Tree Classifier

Input: **an attribute-valued dataset *D***
Output: classified instances

```
1.  Tree = {}
2.  if D is “pure” OR other stopping criteria met then
3.      terminate
4.  end if
5.  for all attribute $a \in D$ do
6.      compute information gain ratio if we split on $a$
7.  end for
8.  $a_{best}$ = Attribute with maximum information gain ratio
9.  Tree = Create a decision node that tests $a_{best}$ in the root
10. $D_v$ = Induced sub-datasets from D based on $a_{best}$
11. for all $D_v$ do
12.     $Tree_v$ = C4.5($D_v$)
13.     Attach $Tree_v$ to the corresponding branch of Tree
14. end for
15. retrun Tree
```

5.2.3 Support Vector Machines

In machine learning, support vector machines (SVMs) are supervised learning models with related learning algorithms that examine information and perceive designs, utilized for grouping and regression analysis. Given an arrangement of training samples, each set apart to belong to one of two classifications, a SVM training algorithm manufactures a model that allots new illustrations into one class or the other, making it a non-probabilistic binary linear classifier as shown in Figure 3.

Support Vector Machines (SVM) as of late got to be a standout amongst the most famous classification methods. They have been utilized as a part of a wide assortment of uses such as text classification [20], facial expression recognition [21], gene analysis [22] and many others.

An SVM classifies data by transforming the data into higher dimension using a kernel function and then finding the best hyperplane that separates the patterns of one class from those of the other class. The best hyperplane for an SVM refers to the one with the maximum margin between the classes. Margin means the maximal width of the slab parallel to the hyperplane that has no interior patterns. The support vectors are the data points that are closest to the separating hyperplane; these points are on the boundary of the slab.

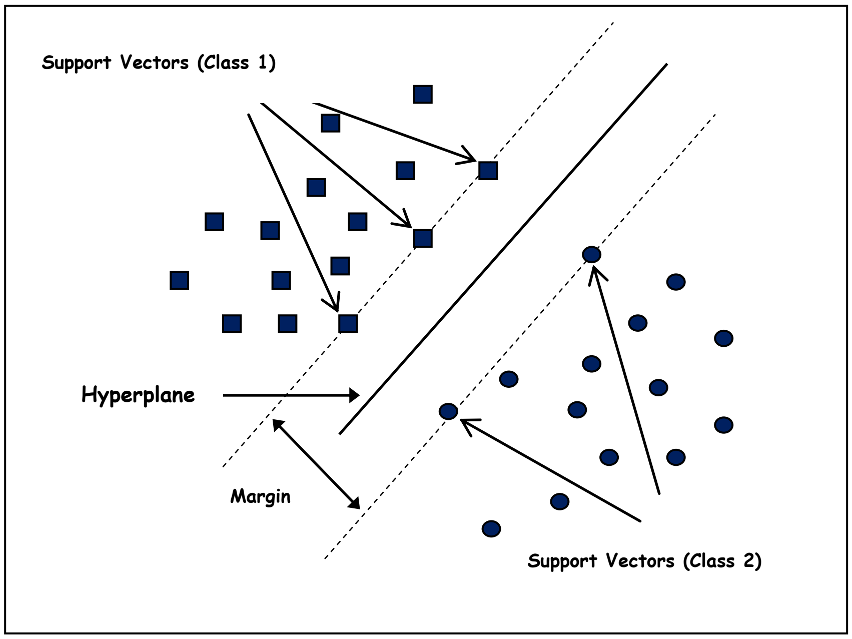

Figure 3 : Support Vector Machine Classifier

A SVM model is a representation of the illustrations as points in space, mapped so that the samples of the different classes are isolated by a reasonable gap that is as wide as could be allowed. New samples are then mapped into that same space and anticipated to have a place with a class in view of which side of the gap they fall on. Support vector machines (SVMs) are an arrangement of regulated learning techniques utilized for order, relapse and anomalies identification. Support Vector Machines are based on the Structural Risk Minimization principle from statistical learning theory formulated by Vpnik [23][24][25].

Given some training data $\boldsymbol{D}$, a set of $\boldsymbol{n}$ points of the form shown in Eq 5.2.3a

$$\boldsymbol{D} = \{(\boldsymbol{x_i}, \boldsymbol{y_i}) | \boldsymbol{x_i} \in \mathfrak{R}^{\boldsymbol{p}}, \boldsymbol{y_i} \in \{-\mathbf{1}, \mathbf{1}\}\}_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{n}} \qquad [5.2.3a]$$

where the $\boldsymbol{y_i}$ is either 1 or −1, indicating $\boldsymbol{x_i}$ the class to which the point belongs.

Each $\boldsymbol{x_i}$ is a $\boldsymbol{p}$ -dimensional real vector. We want to find the maximum-margin hyperplane that divides the points having $\boldsymbol{y_i = 1}$ from those having $\boldsymbol{y_i = -1}$. Any hyperplane can be written as the set of points $\boldsymbol{x}$ satisfying Eq 5.2.3b

$$\boldsymbol{w \cdot x - b = 0} \qquad [5.2.3b]$$

where $\cdot$ denotes the dot product and $\boldsymbol{w}$ the (not necessarily normalized) normal vector to the hyperplane. The parameter $\frac{\boldsymbol{b}}{\|\boldsymbol{w}\|}$ determines the offset of the hyperplane from the origin along the normal vector $\boldsymbol{w}$.

If the training data are linearly separable, we can select two hyperplanes in a way that they separate the data and there are no points between them, and then try to maximize their distance. The region bounded by them is called "the margin". These hyperplanes can be described by the Eq 5.2.3c and Eq 5.2.3d

$$\boldsymbol{w \cdot x - b = 1} \qquad [5.2.3c]$$

And

$$\boldsymbol{w \cdot x - b = -1} \qquad [5.2.3d]$$

Geometrically, the distance between these two hyperplanes is $\frac{\boldsymbol{2}}{\|\boldsymbol{w}\|}$, so to maximize the distance between the planes we want to minimize $\|\boldsymbol{w}\|$. As we also have to prevent data points from falling into the margin, we add the following constraint: for each $\boldsymbol{i}$ either for Eq 5.2.3e or Eq 5.2.3f

$$\boldsymbol{w \cdot x_i - b \geq 1\, for\; x_i\;\; of\; the\; first\; class} \qquad [5.2.3e]$$

Or

$$\boldsymbol{w \cdot x_i - b \leq -1\, for\; x_i\;\; of\; the\; second\; class} \qquad [5.2.3f]$$

This can be rewritten as Eq 5.2.3g:

$$\boldsymbol{y_i(w \cdot x_i - b) \geq 1, for\; all\; 1 \leq i \leq n}. \qquad [5.2.3g]$$

We can put this together to get the optimization problem as shown in Eq 5.2.3h:

Minimize (in $\boldsymbol{w}, \boldsymbol{b}$) $\|\boldsymbol{w}\|$

subject to $\left(\textit{for any } \boldsymbol{i} = 1,\ldots,\boldsymbol{n}\right)$

$$\boldsymbol{y_i}(\boldsymbol{w} \cdot \boldsymbol{x_i} - \boldsymbol{b}) \geq \mathbf{1}. \qquad [5.2.3h]$$

The algorithm employed in the linear kernel of the support vector machine classifier is given below [26].

Algorithm: Support Vector Machine Classifier – Linear Kernel

Input: **a linearly separable set *S*, learning rate** $\eta \in \Re^{+}$

Output: classified instances

```
1.  $w_0 = 0; b_0 = 0; k = 0;$
2.  $R = \max_{1 \le i \le l} \|x_i\|$
3.  While at least one mistake is made in the for loop do
4.      for $i = 1, \ldots, l$ do
5.          if $y_i(\langle w_k, x_i \rangle + b_k) \le 0$ then
6.              $w_{k+1} = w_k + \eta y_i x_i$
7.              $b_{k+1} = w_k + \eta y_i R^2$ (updating bias)
8.              $k = k + 1$
9.          end if
10.     end for
11. end while
12. return $w_k, b_k$, where $k$ is the number of mistakes
```

5.3 Ensemble Learners

Ensemble classifier refers to a group of individual classifiers that are cooperatively trained on data set in a supervised classification problem [27]. A standout amongst the most active fields of research in supervised learning has been to study techniques for building good ensemble classifiers. The principle revelation is that ensembles are regularly much more exact than the individual classifiers that make them up [28].

It is stated in [29] that the idea of ensemble methodology is to build a predictive model by integrating multiple models. It is well-known that ensemble methods can be used for improving prediction performance. The main idea behind the ensemble methodology is to weigh several individual classifiers, and combine them in order to obtain a classifier that outperforms every one of them. In fact, human being tends to seek several opinions before making any important decision.We weigh the individual opinions, and combine them to reach our final decision [30].

A typical ensemble method for classification tasks contains the following building blocks:

1. Training set—A labeled dataset used for ensemble training. The training set can be described in a variety of languages. Most frequently, the instances are described as attribute- value vectors. We use the notation A to denote the set of input attributes containing n attributes: $A = \{a_1, ..., a_i, ..., a_n\}$ and y to represent the class variable or the target attribute.
2. Base Inducer—The inducer is an induction algorithm that obtains a training set and forms a classifier that represents the generalized relationship between the input attributes and the target attribute. Let I represent an inducer. We use the notation $M = I(S)$for representing a classifier M which was induced by inducer I on a training set S.
3. Diversity Generator—This component is responsible for generating the diverse classifiers.
4. Combiner—The combiner is responsible for combining the classifications of the various classifiers.

Base classifiers refer to individual classifiers used to construct the ensemble classifiers. Neural network, support vector machine, and k-NN classifiers are some of the commonly used base classifiers. Diversity can be achieved by manipulating the training parameters of the base classifiers in an ensemble.

5.3.1 BAGGING CLASSIFIER

Bagging is a “bootstrap” ensemble method that creates individuals for its ensemble by training each classifier on a random redistribution of the training set. Each classifier’s training set is generated by randomly drawing, with replacement, N examples - where N is the size of the original training set; many of the original examples may be repeated in the resulting training set while others may be left out. Each individual classifier in the ensemble is generated with a different random sampling of the training set.

Breiman showed that Bagging is effective on “unstable” learning algorithms where small changes in the training set result in large changes in predictions [31]. He claimed that neural networks and decision trees are examples of unstable learning algorithms.

In bagging [31], the training subsets are randomly drawn (with replacement) from the training set. Homogeneous base classifiers are trained on the subsets. The class chosen by most base classifiers is the considered to be the final verdict of the ensemble classifier.

The most well-known independent method is bagging (bootstrap aggregating). The method aims to increase accuracy by creating an improved composite classifier, *I**, by amalgamating the various outputs of learned classifiers into a single prediction. Its working is depicted in Figure 4.

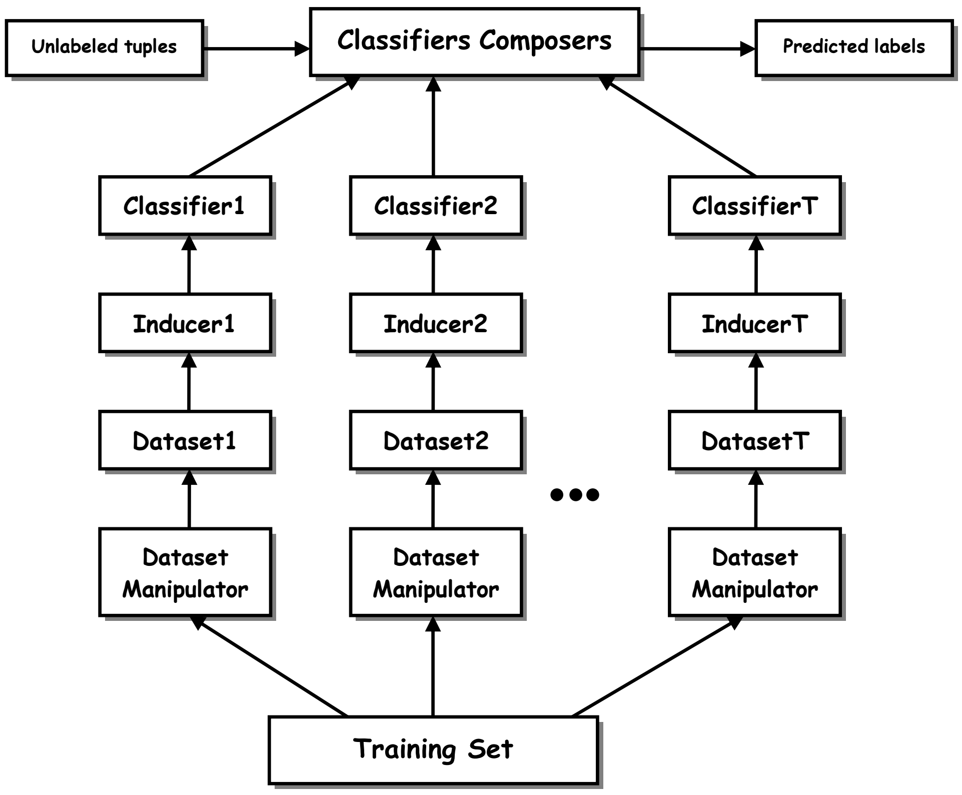

Figure 4 : Bagging - an Independent Ensemble Learner

The bagging ensemble is used with a decision tree as a base estimator. The algorithm for the classifier is shown below:

Algorithm: Bagging Classifier

Output: classified instances

```
Training Phase:
    1. Initialize the parameters
        • $D = \emptyset$.
        • $L$ , the number of classifiers to train.
    2. For $k = 1,\ldots,L$
        • Take a bootstrap sample $S_k$ from $\mathbf{Z}$.
        • Build a classifier $D_k$ using $S_k$ as the training set.
        • Add the classifier to the current ensemble, $D = D \cup D_k$.
    3. Return $D$.
Classification Phase:
    4. Run $D_1,\ldots, D_L$ on the input $\mathbf{x}$.
5. The class with the maximum number of votes is chosen as the label for $\mathbf{x}$.
```

Bagging, like boosting, is a technique that improves the accuracy of a classifier by generating a composite model that combines multiple classifiers all of which are derived from the same inducer. Both methods follow a voting approach, which is implemented differently, in order to combine the outputs of the different classifiers. In boosting, as opposed to bagging, each classifier is influenced by the performance of those that were built prior to its construction. Specifically, the new classifier pays more attention to classification errors that were done by the previously built classifiers where the amount of attention is determined according to their performance. In bagging, each instance is chosen with equal probability, while in boosting, instances are chosen with a probability that is proportional to their weight. Furthermore, as mentioned above, bagging requires an unstable learner as the base inducer, while in boosting inducer instability in not required, only that the error rate of every classifier be kept below 0.5.

5.3.2 Random Forest Classifier

A Random Forest ensemble (also known as random subspace) uses a large number of individual, unpruned decision trees [32]. Breiman et al proposed random forests, which add an additional layer of randomness to bagging. In addition to constructing each tree using a different bootstrap sample of the data, random forests change how the classification or regression trees are constructed. In standard trees, each node is split using the best split among all variables. In a random forest, each node is split using the best among a subset of predictors randomly chosen at that node. This somewhat counterintuitive strategy turns out to perform very well compared to many other classifiers, including discriminant analysis, support vector machines and neural networks, and is robust against overfitting. In addition, it is very user-friendly in the sense that it has only two parameters (the number of variables in the random subset at each node and the number of trees in the forest), and is usually not very sensitive to their values [33].

Random forests is a notion of the general technique of random decision forests that are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

The algorithm for the random forest classifier is as follows:

Algorithm: Random Forest Classifier

Input: IDT(a decision tree inducer), S(the training set), μ(the subsample size), and N(the number of attributes used in each node)
Output: classified instances

```
1. $t$ = 1
2. repeat
    • $S_t$ = sample $\boldsymbol{\mu}$ instances from S with replacement.
    • Build classifier $M_t$ using IDT(N) on $S_t$ .
    • $t$++.
3. until $\boldsymbol{t} > \boldsymbol{T}$.
```

In Breiman's algorithm, each tree in the forest is trained using a bootstrapped sample of the training data. Bootstrapping means that each tree is trained on a slightly different version of the data, which helps ensure the trees are uncorrelated. Each tree is built recursively, starting from the root. At each node a randomized search procedure is applied to determine how the node should be split. When the best split is found, the data in the node are partitioned according to the chosen split, and the procedure is applied recursively to split the left and right children. Splitting continues until no acceptable split can be found or a maximum depth is reached.

The input parameter *N* represents the number of input variables that will be used to determine the decision at a node of the tree. This number should be much less than the number of attributes in the training set. Note that Bagging can be thought of as a special case of Random Forests obtained when *N* is set to the number of attributes in the original training set.

5.3.3 ADABOOST CLASSIFIER

AdaBoost (Adaptive Boosting) is a popular ensemble algorithm that improves the simple boosting algorithm via an iterative process. The main idea behind this algorithm is to give more focus to patterns that are harder to classify. The amount of focus is quantified by a weight that is assigned to every pattern in the training set. Initially, the same weight is assigned to all the patterns. In each iteration the weights of all misclassified instances are increased while the weights of correctly classified instances are decreased. As a consequence, the weak learner is forced to focus on the difficult instances of the training set by performing additional iterations and creating more classifiers. Furthermore, a weight is assigned to every individual classifier. This weight measures the overall accuracy of the classifier and is a function of the total weight of the correctly classified patterns. Thus, higher weights are given to more accurate classifiers. These weights are used for the classification of new patterns. This iterative procedure provides a series of classifiers that complement one another.

Boosting creates data subsets for base classifier training by re-sampling the training patterns, however, by providing the most informative training pattern for each consecutive classifier [34]. Each of the training patterns is assigned a weight that determines how well the instance was classified in the previous iteration. Boosting encompasses a family of methods. The focus of these methods is to produce a series of classifiers. The training set used for each member of the series is chosen based on the performance of the earlier classifier(s) in the series. In Boosting, examples that are incorrectly predicted by previous classifiers in the series are chosen more often than examples that were correctly predicted [35]. Thus Boosting attempts to produce new classifiers that are better able to predict examples for which the current ensemble's performance is poor.

Ada-Boosting can use the approach of (a) selecting a set of examples based on the probabilities of the examples, or (b) simply using all of the examples and weight the error of each example by the probability for that example (i.e., examples with higher probabilities have more effect on the error). This latter approach has the clear advantage that each example is incorporated (at least in part) in the training set. Furthermore, Friedman et al. have demonstrated that this form of Ada-Boosting can be viewed as a form of additive modeling for optimizing a logistic loss function [36].

Ada-Boosting combines the classifiers $C_1, \ldots, C_k$ using weighted voting where C_k has weight $\log(B_k)$. These weights allow Ada-Boosting to discount the predictions of classifiers that are not very accurate on the overall problem. Friedman et al have also suggested an alternative mechanism that fits together the predictions of the classifiers as an additive model using a maximum likelihood criterion [36].

Adaboost is a dependent ensemble learner as shown in Figure 5. In a dependent framework the output of a classifier is used in the construction of the next classifier. Thus it is possible to take advantage of knowledge generated in previous iterations to guide the learning in the next iterations. Alternatively each classifier is built independently and their outputs are combined in some fashion.

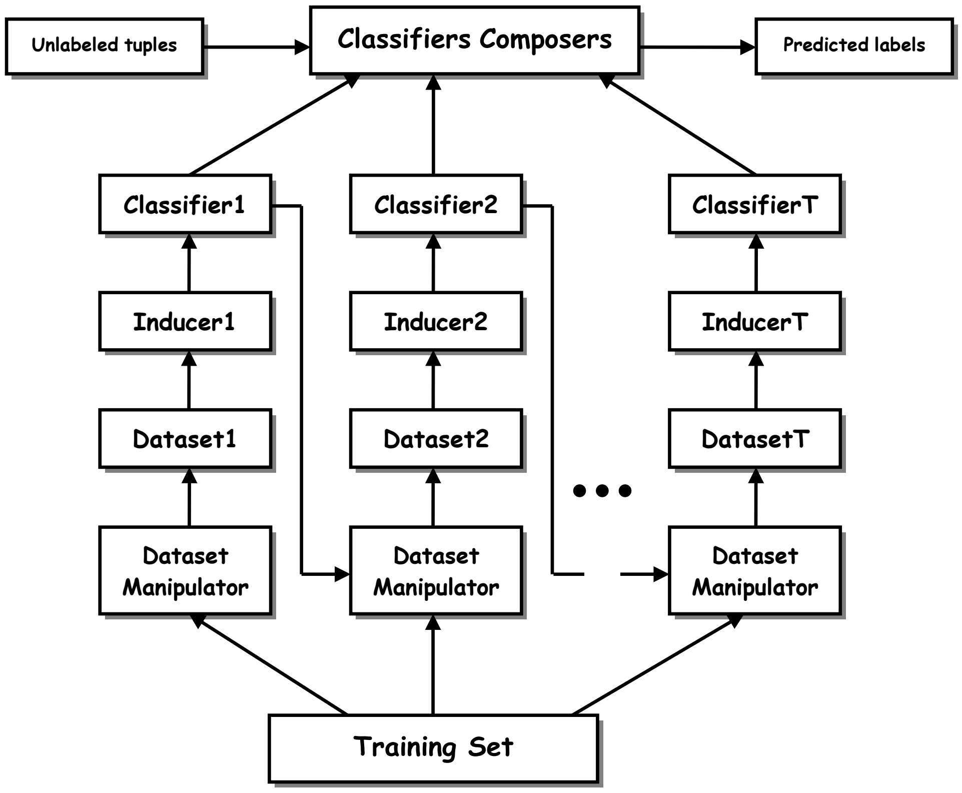

Figure 5 : Adaboost - a Dependent Ensemble Learner

The algorithm for the Adaboost classifier is shown below.

Algorithm: Adaboost Classifier

Output: classified instances

Training Phase:

1. Initialize the parameters
   - Set the weights $\boldsymbol{w}^1 = [w_1, \dots, w_N]$, $w_j^1 \in [0,1], \sum_{j=1}^{N} w_j^1 = 1$. (Usually $w_j^1 = \frac{1}{N}$).
   - Initialize the ensemble $= \emptyset$.
   - Pick *L*, the number of classifiers to train.
2. For *k* = 1,….,*L*
   - Take a sample S_k from **Z** using distribution $\boldsymbol{w}^k$.
   - Build a classifier D_k using S_k as the training set.
   - Calculate the weighted ensemble error at step k by

     $$\epsilon_k = \sum_{j=1}^{N} w_j^k l_k^j ,$$

     ($l_k^j = 1$ if D_k misclassifies $\boldsymbol{z}_j$, $l_k^j = 0$ otherwise.)
   - If $\epsilon_k = 0$ or $\epsilon_k \geq 0.5$, ignore D_k, reinitialize the weights w_j^k to $\frac{1}{N}$ and continue.

- Else, calculate

$$\beta_k = \frac{\epsilon_k}{1-\epsilon_k}, \text{ where } \epsilon_k \in (0,0.5),$$

- Update the individual weights

$$w_j^{k+1} = \frac{w_j^k \beta_k^{(1-l_k^j)}}{\sum_{i=1}^N w_i^k \beta_k^{(1-l_k^i)}}, j = 1, \dots, N$$

3. Return D and $\beta_1, \dots, \beta_L$.

Classification Phase:

4. Calculate the support for class ω_i by

$$\mu_i(\boldsymbol{x}) = \sum_{D_k(\boldsymbol{x})=\omega_i} \ln\left(\frac{1}{\beta_k}\right)$$

5. The class with the maximum support is chosen as the label for **x**.

Adaboost seems to improve the performance accuracy for two main reasons:

1. It generates a final classifier whose misclassification rate can be reduced by combining many classifiers whose misclassification rate may be high.

2. It produces a combined classifier whose variance is significantly lower than the variances produced by the weak base learner.

However, Adaboost sometimes fails to improve the performance of the base inducer. According to [37], the main reason for Adaboost's failure is overfitting. The objective of boosting is to construct a composite classifier that performs well on the data by iteratively improving the classification accuracy. Nevertheless, a large number of iterations may result in an over complex composite classifier, which is significantly less accurate than a single classifier. One possible way to avoid overfitting is to keep the number of iterations as small as possible.

CHAPTER - 6
EXPERIMENTAL RESULTS AND DISCUSSIONS

Chapter 6 - Experimental Results And Discussions

6.1 Introduction

Section 6.2 enumerates the metrics which we use for evaluating the classifiers implemented. We show the results of the classifications in Section 6.3. In Section 6.4 the discussions are presented.

6.2 Classifier Evaluation Metrics

When we utilize a classifier model for an issue, we quite often need to take a gander at the exactness of that model as the quantity of right forecasts from all expectations made. This is the classifier accuracy. When we have to choose whether it is a sufficient model to take care of the issue, accuracy is by all account not the only metric for assessing the viability of a classifier. Two other valuable measurements are precision and recall. These two measurements can give much more prominent knowledge into the execution attributes of a classifier. We can also use the confusion matrix to analyze the performances of the classifiers.

A false positive error, or in short false positive, commonly called a 'false alarm', is a result that indicates a given condition has been fulfilled; when it actually has not been fulfilled i.e. erroneously a positive effect has been assumed.

A false negative error, or in short false negative, is where a test result indicates that a condition failed; while it actually was successful i.e. erroneously no effect has been assumed.

True positives are relevant items that we correctly identified as relevant. True negatives are irrelevant items that we correctly identified as irrelevant.

A confusion matrix $\boldsymbol{C}$ is such that $\boldsymbol{C_{i,j}}$ is equal to the number of observations known to be in group $\boldsymbol{i}$ but predicted to be in group $\boldsymbol{j}$. A confusion matrix is used to describe the performance of the classifier.

Table 3: Confusion Matrix

		Predicted Class	
		Yes	No
Actual Class	Yes	TP	FN
	No	FP	TN

Accuracy is how close a measured value is to the actual (true) value. It is the proportion of instances whose class the classifier can correctly predict. It can be calculated as shown in Eq 6.2a.

$$\boldsymbol{Accuracy = \frac{T_p + T_n}{Total\ number\ of\ samples}} \qquad [6.2a]$$

where

$\boldsymbol{T_p}$ is the number of true positives

$\boldsymbol{F_n}$ is the number of false positives

Precision measures the exactness of a classifier. A higher precision means less false positives, while a lower precision means more false positives. This is often at odds with recall, as an easy way to improve precision is to decrease recall. Precision is defined as the number of true positives over the number of true positives plus the number of false positives. Its formula is shown in Eq 6.2b.

$$\boldsymbol{P = \frac{T_p}{T_p + F_p}} \qquad [6.2b]$$

where

$\boldsymbol{T_p}$ is the number of true positives

$\boldsymbol{F_p}$ is the number of false positives

Recall measures the completeness, or sensitivity, of a classifier. Higher recall means less false negatives, while lower recall means more false negatives. Improving recall can often decrease precision because it gets increasingly harder to be precise as the sample space increases. Recall is defined as the number of true positives over the number of true positives plus the number of false negatives. Its formula is shown in Eq 6.2c.

$$\boldsymbol{R = \frac{T_p}{T_p + F_n}} \qquad [6.2c]$$

where

$\boldsymbol{T_p}$ is the number of true positives.

$\boldsymbol{F_n}$ is the number of false negatives.

Precision and recall can be combined to produce a single metric known as F-measure, which is the weighted harmonic mean of precision and recall. Its equation is shown in Eq 6.2d.

$$\boldsymbol{F1 = 2\frac{P \times R}{P+R}} \qquad [6.2d]$$

where

$\boldsymbol{P}$ is the precision.

$\boldsymbol{R}$ is the recall.

We compare the performance of the classifiers that we used based on their precision, recall, F-measure, accuracy and confusion matrices.

6.3 Experimental Results

We train each of the classifiers using the 5 data sets individually. First, we test all the classifiers on each training set one at a time and then we test it on the test set. The tables below summarize the results of all the experiments performed.

We use the C4.5 algorithm in the decision tree classifier. SVM classifier is implemented using the linear kernel. The default parameters for the ensemble learners are used. For the bagging classifier, the base estimator is the decision tree classifier and the number of base estimators used is 10. The random forest classifier uses 10 estimators i.e the number of trees in the forest. For the Adaboost classifier also, the base estimator used is the decision tree classifier and the number of estimators i.e the maximum number of estimators at which boosting is terminated is 50.

Table 4 : Class-wise Performance Metrics - Multinomial Naive Bayes Classifier

Dataset	Clickbait			Not Clickbait		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure
DS1	0.80	0.71	0.75	0.74	0.82	0.78
DS2	0.79	0.69	0.74	0.73	0.82	0.77
DS3	0.79	0.70	0.74	0.73	0.82	0.77

DS4	0.81	0.59	0.68	0.82	0.93	0.87
DS5	0.78	0.82	0.80	0.61	0.55	0.57

From Table 4, we can observe that the multinomial naïve Bayes classifier achieved a 0.81 precision for DS4, 0.82 recall for DS5 and 0.80 f-measure for DS5 for the clickbait class. For the non-clickbait class, it achieved a 0.82 precision, 0.93 recall and 0.87 f-measure all for DS5.

Table 5 : Class-wise Performance Metrics - Decision Tree Classifier

DATASET	CLICKBAIT			NOT CLICKBAIT		
	PRECISION	RECALL	F-MEASURE	PRECISION	RECALL	F-MEASURE
DS1	0.92	0.87	0.90	0.88	0.93	0.90
DS2	0.92	0.84	0.88	0.85	0.93	0.89
DS3	0.92	0.83	0.87	0.84	0.93	0.88
DS4	0.93	0.81	0.87	0.91	0.97	0.94
DS5	0.92	0.89	0.91	0.80	0.85	0.82

From Table 5, we can observe that the decision tree classifier achieved a 0.93 precision for DS4, 0.89 recall for DS5 and 0.91 f-measure for DS5 for the clickbait class. For the non-clickbait class, it achieved a 0.91 precision, 0.97 recall and 0.94 f-measure for DS5.

Table 6 : Class-wise Performance Metrics - Support Vector Machines

DATASET	CLICKBAIT			NOT CLICKBAIT		
	PRECISION	RECALL	F-MEASURE	PRECISION	RECALL	F-MEASURE
DS1	0.83	0.74	0.78	0.76	0.85	0.80
DS2	0.83	0.72	0.77	0.75	0.85	0.80
DS3	0.81	0.74	0.78	0.76	0.83	0.80
DS4	0.85	0.55	0.67	0.81	0.95	0.87
DS5	0.88	0.78	0.83	0.64	0.78	0.71

From Table 6, we can observe that the support vector machine classifier achieved a 0.88 precision, 0.78 recall and 0.83 f-measure all for DS5 for the clickbait class. For the non-clickbait class, it achieved a 0.81 precision, 0.95 recall and 0.87 f-measure for DS4.

Table 7 : Class-wise Performance Metrics - Bagging Classifier

Dataset	Clickbait			Not Clickbait		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure
DS1	0.93	0.85	0.89	0.86	0.94	0.90
DS2	0.93	0.82	0.87	0.84	0.94	0.88
DS3	0.93	0.81	0.87	0.83	0.94	0.88
DS4	0.94	0.79	0.85	0.90	0.97	0.94
DS5	0.92	0.88	0.90	0.78	0.86	0.82

From Table 7, we can observe that the bagging classifier achieved a 0.94 precision for DS4, 0.88 recall for DS5 and 0.90 f-measure for DS5 for the clickbait class. For the non-clickbait class, it achieved a 0.90 precision, 0.97 recall and 0.94 f-measure for DS4.

Table 8 : Class-wise Performance Metrics - Random Forest Classifier

Dataset	Clickbait			Not Clickbait		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure
DS1	0.93	0.85	0.89	0.86	0.94	0.90
DS2	0.92	0.82	0.87	0.84	0.93	0.88
DS3	0.93	0.81	0.86	0.83	0.94	0.88
DS4	0.94	0.79	0.86	0.90	0.97	0.94
DS5	0.93	0.88	0.90	0.78	0.86	0.82

From Table 8, we can observe that the random forest classifier achieved a 0.94 precision for DS4, 0.88 recall for DS5 and 0.90 f-measure for DS5 for the clickbait class. For the non-clickbait class, it achieved a 0.90 precision, 0.97 recall and 0.94 f-measure for DS4.

Table 9 : Class-wise Performance Metrics - Adaboost Classifier

Dataset	Clickbait			Not Clickbait		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure
DS1	0.86	0.74	0.79	0.77	0.88	0.82
DS2	0.86	0.72	0.78	0.76	0.88	0.81
DS3	0.85	0.72	0.78	0.76	0.88	0.81

DS4	0.84	0.61	0.71	0.83	0.94	0.88
DS5	0.85	0.84	0.84	0.68	0.71	0.70

From Table 9, we can observe that the bagging classifier achieved a 0.86 precision for DS1 and DS2, 0.84 recall for DS5 and 0.84 f-measure for DS5 for the clickbait class. For the non-clickbait class, it achieved a 0.83 precision, 0.94 recall and 0.88 f-measure for DS4.

We can see that all the classifiers used give a good precision, recall and f-measure for the clickbait class as well as the non-clickbait class with the datasets DS4 and DS5. This shows that the classifiers work better when trained with unbalanced datasets. Out of the simple learners used, the decision tree classifier outperforms the other two. Likewise, out of the ensemble learners used, the bagging classifier and the random forest classifier both give the same performance though better than the Adaboost classifier.

Table 10 : Performance Metrics - Multinomial Naive Bayes Classifier

Datasets	Precision	Recall	F-Measure
DS1	0.77	0.76	0.76
DS2	0.76	0.76	0.76
DS3	0.76	0.76	0.76
DS4	0.82	0.82	0.81
DS5	0.72	0.73	0.73

From Table 10, we can observe that the multinomial naïve bayes classifier achieves a precision of 0.82, recall of 0.82 and f-measure of 0.81 for DS4.

Table 11 : Performance Metrics - Decision Tree Classifier

Datasets	Precision	Recall	F-Measure
DS1	0.90	0.90	0.90
DS2	0.89	0.88	0.88
DS3	0.88	0.88	0.88
DS4	0.92	0.92	0.91
DS5	0.88	0.88	0.88

From Table 11, we can observe that the decision tree classifier achieves a precision of

0.92, recall of 0.92 and f-measure of 0.91 for DS4.

Table 12 : Performance Metrics - Support Vector Machines

Datasets	Precision	Recall	F-Measure
DS1	0.80	0.79	0.79
DS2	0.79	0.78	0.78
DS3	0.79	0.79	0.79
DS4	0.82	0.82	0.81
DS5	0.80	0.78	0.79

From Table 12, we can observe that the support vector machine classifier achieves a precision of 0.82, recall of 0.82 and f-measure of 0.81 for DS4.

Table 13 : Performance Metrics - Bagging Classifier

Datasets	Precision	Recall	F-Measure
DS1	0.90	0.89	0.89
DS2	0.88	0.88	0.88
DS3	0.88	0.87	0.87
DS4	0.91	0.91	0.91
DS5	0.88	0.87	0.87

From Table 13, we can observe that the bagging classifier achieves a precision of 0.91, recall of 0.91 and f-measure of 0.91 for DS4.

Table 14 : Performance Metrics - Random Forest Classifier

Datasets	Precision	Recall	F-Measure
DS1	0.90	0.89	0.89
DS2	0.88	0.88	0.88
DS3	0.88	0.87	0.87
DS4	0.91	0.91	0.91
DS5	0.88	0.87	0.87

From Table 14, we can observe that the random forest classifier achieves a precision of 0.91, recall of 0.91 and f-measure of 0.91 for DS4.

Table 15 : Performance Metrics - Adaboost Classifier

DATASETS	PRECISION	RECALL	F-MEASURE
DS1	0.81	0.81	0.81
DS2	0.81	0.80	0.80
DS3	0.81	0.80	0.80
DS4	0.83	0.83	0.83
DS5	0.80	0.79	0.80

From Table 15, we can observe that the adaboost classifier achieves a precision of 0.83, recall of 0.83 and f-measure of 0.83 for DS4.

Table 16 : Accuracy - Multinomial Naive Bayes Classifier

DATASETS	ACCURACY
DS1	0.7627
DS2	0.7562
DS3	0.7565
DS4	0.8158
DS5	0.7309

From Table 16, we can see that the multinomial naïve bayes classifier gives an accuracy of 0.8158 for DS4.

Table 17 : Accuracy - Decision Tree Classifier

DATASETS	ACCURACY
DS1	0.8989
DS2	0.8823
DS3	0.8765
DS4	0.9161
DS5	0.8784

From Table 17, we can see that the decision tree classifier gives an accuracy of 0.9161 for DS4.

Table 18 : Accuracy - Support Vector Machines

DATASETS	ACCURACY
DS1	0.7924
DS2	0.7846

DS3	0.7861
DS4	0.8178
DS5	0.7826

From Table 18, we can see that the support vector machine classifier gives an accuracy of 0.8178 for DS4.

Table 19 : Accuracy - Bagging Classifier

DATASETS	ACCURACY
DS1	0.8935
DS2	0.8776
DS3	0.8736
DS4	0.9108
DS5	0.8727

From Table 19, we can see that the bagging classifier gives an accuracy of 0.9108 for DS4.

Table 20 : Accuracy - Random Forest Classifier

DATASETS	ACCURACY
DS1	0.8938
DS2	0.8786
DS3	0.8732
DS4	0.9116
DS5	0.8734

From Table 20, we can see that the random forest classifier gives an accuracy of 0.9116 for DS4.

Table 21 : Accuracy - Adaboost Classifier

DATASETS	ACCURACY
DS1	0.8066
DS2	0.7986
DS3	0.7977
DS4	0.8334
DS5	0.7945

From Table 21, we can see that the Adaboost classifier gives an accuracy of 0.8334 for DS4.

Figure 6 shows the accuracies of all the datasets with respect to each classifier used.

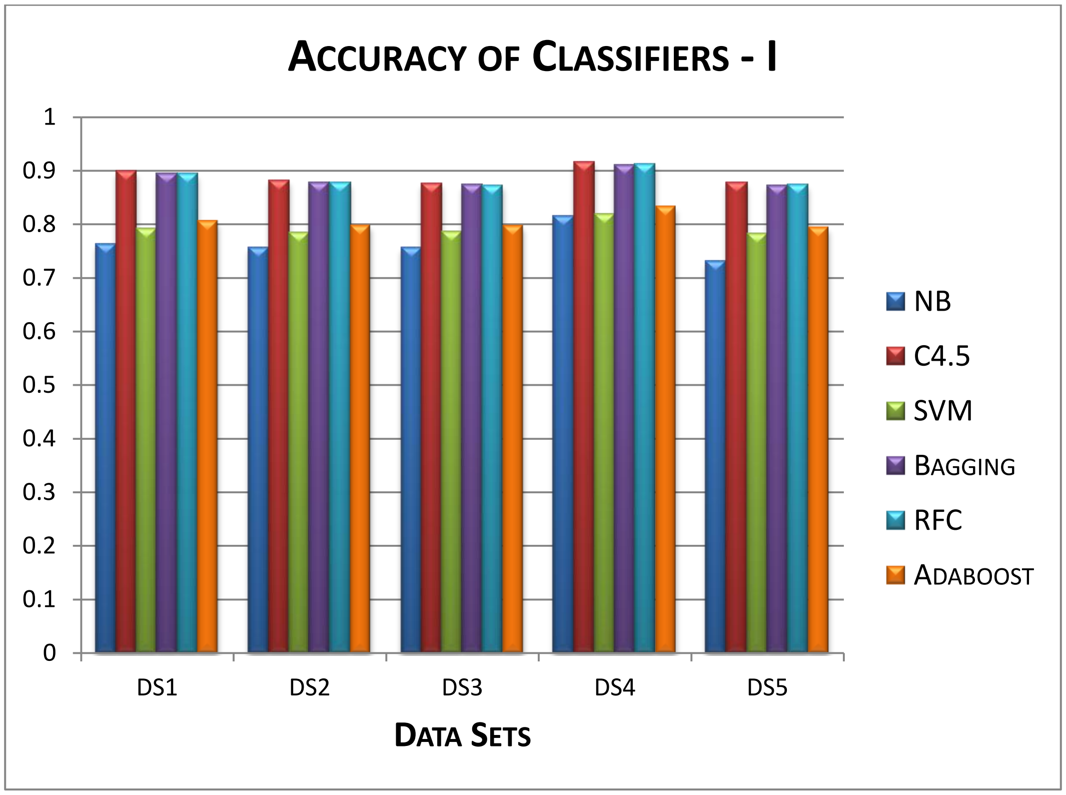

Figure 6 : Accuracy of Classifiers Used Based on Datasets

From Figure 7, we can conclude that the C4.5 decision tree classifier outperforms all the other classifiers for all datasets used. The multinomial naive Bayes classifier gives the lowest accuracies.

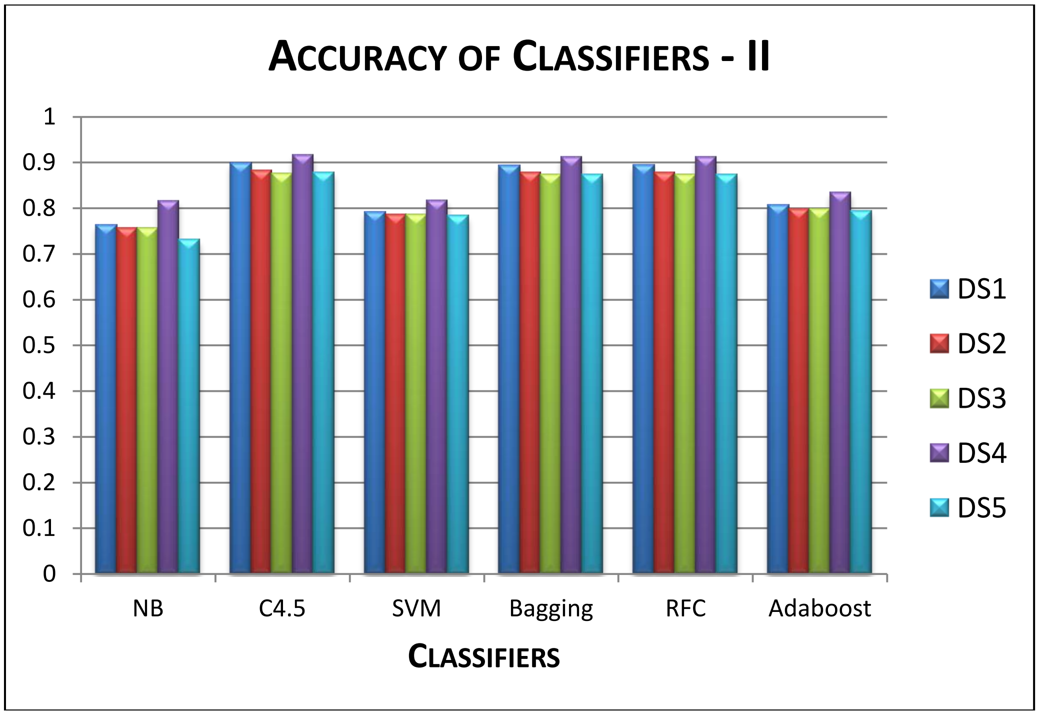

Figure 7 : Accuracy of Classifiers

From Figure 8, we can conclude that the dataset DS4 has highest precision scores for all the classifiers used.

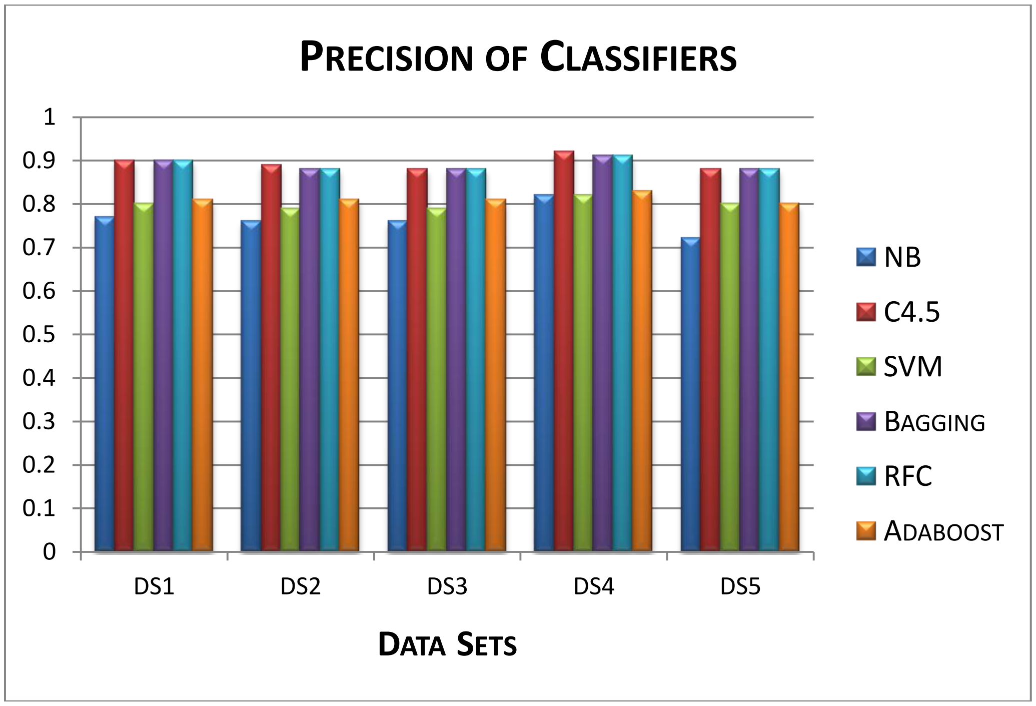

Figure 8 : Precision of Classifiers based on Datasets

The graph depicted in Figure 9 shows that the C4.5 decision tree classifier gives the highest precision for the dataset DS4. It is closely followed by the bagging classifier and the random forest classifier.

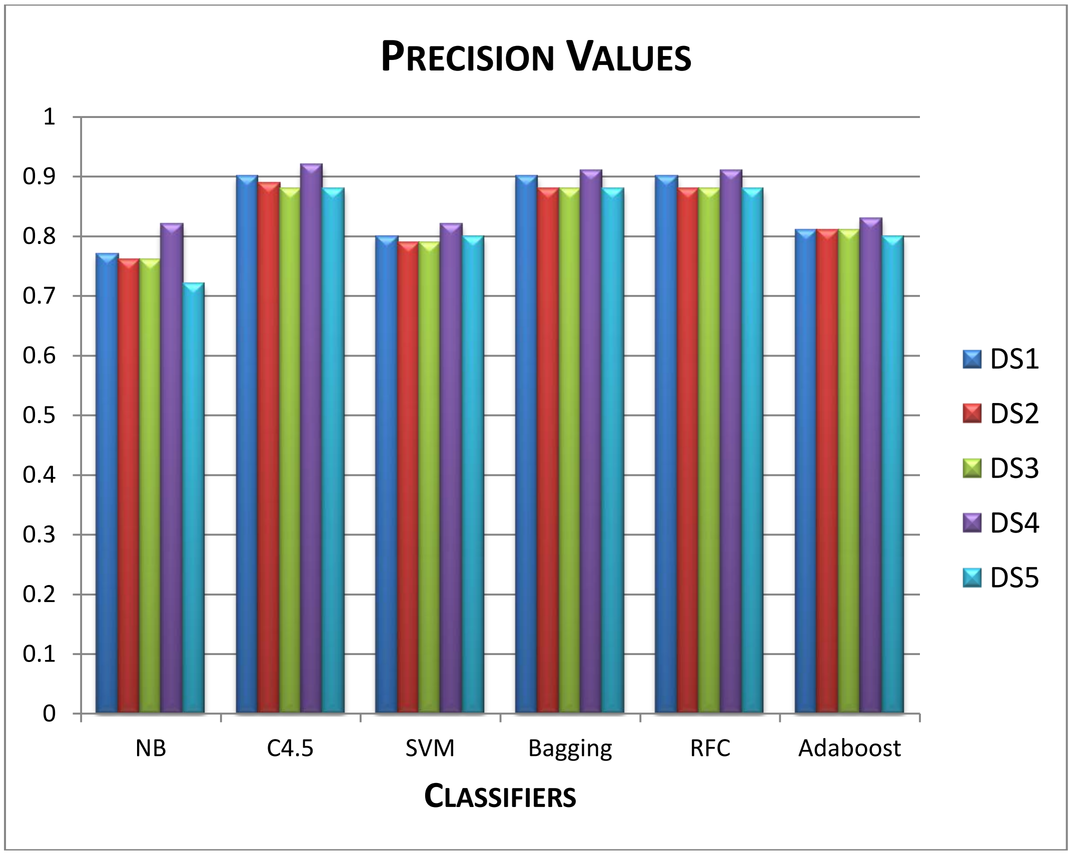

Figure 9 : Precision Values for Classifiers

The graph depicted in Figure 10 shows that even in case of recall values, the dataset DS4 has the highest recall values for all classifiers out of all the datasets used. Just like precision, for the remaining datasets also, the decision tree classifier gives the best performance in terms of recall value and is closely followed by the bagging classifier and the random forest classifier.

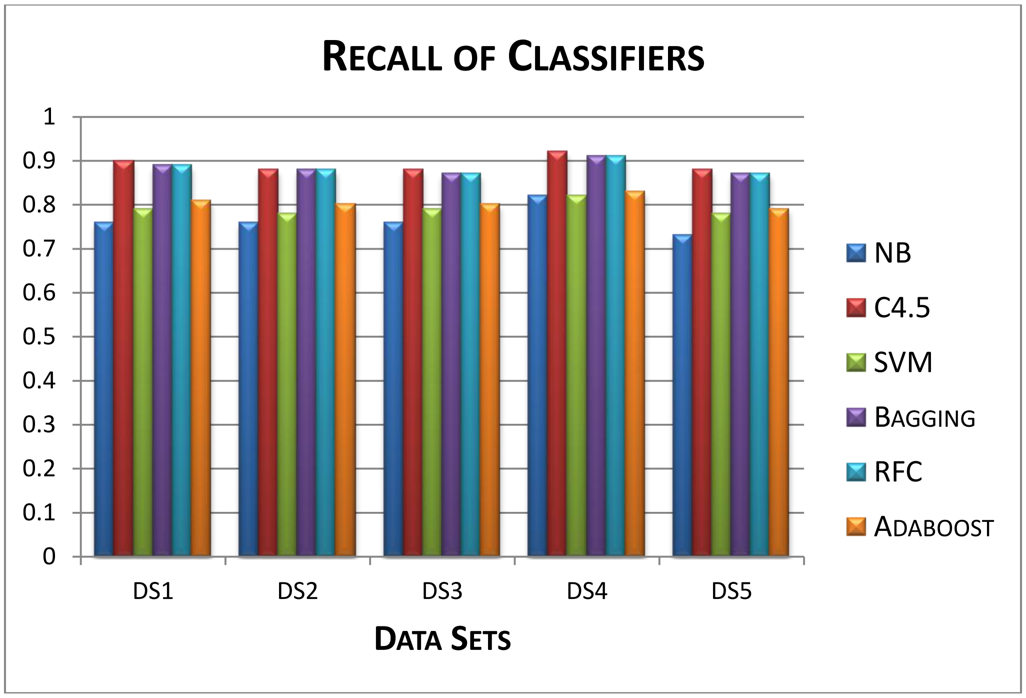

Figure 10 : Recall of Classifiers

The graph depicted in Figure 11 shows that the C4.5 decision tree classifier gives the highest precision for the dataset DS4. It is closely followed by the bagging classifier and the random forest classifier.

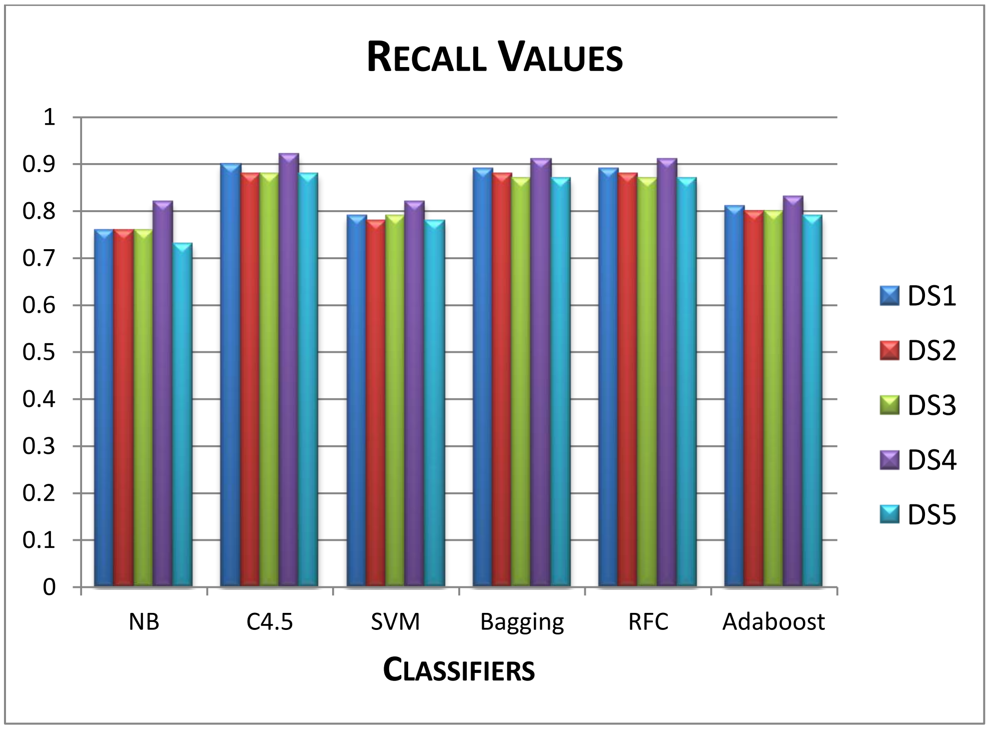

Figure 11 : Recall Values for Classifiers

The same trend that occurred for the precision and recall values repeats for the F-measure of the classifiers for all the datasets as can be seen in Figure 12 and Figure 13.

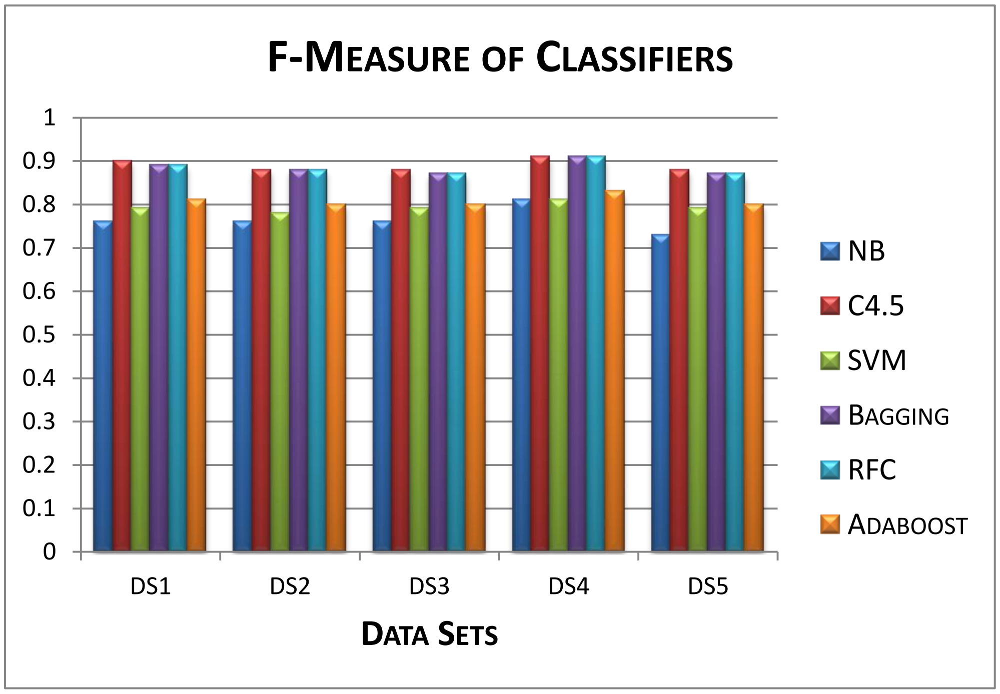

Figure 12 : F-Measure of Classifiers based on Datasets

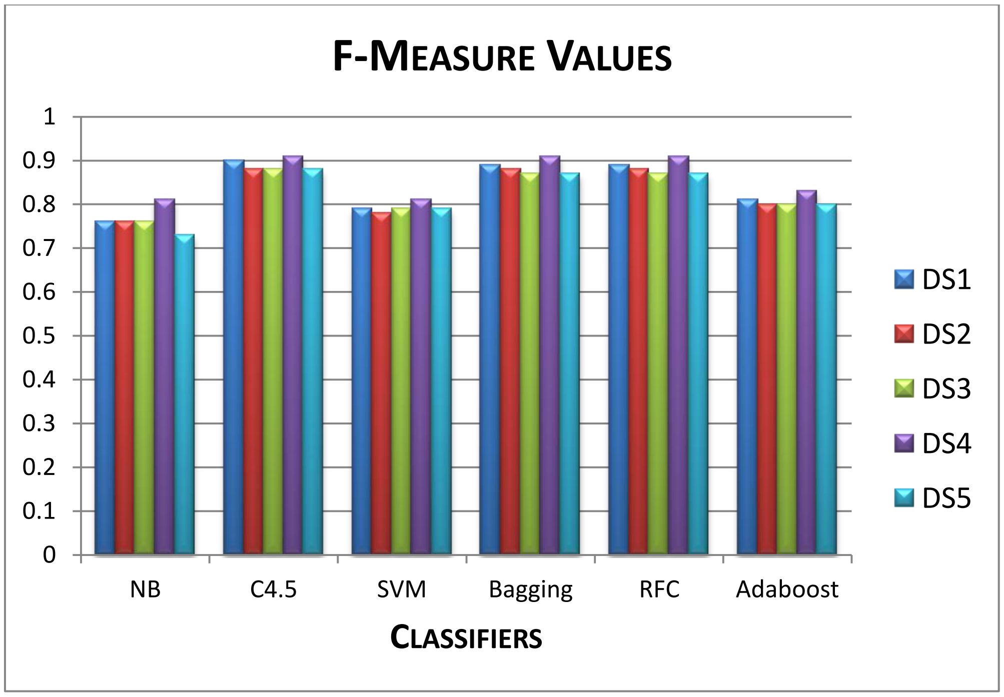

Figure 13 : F-Measure Values for Classifiers

The ROC-AUC scores of all the classifiers are given below.

Table 22 : Roc_Auc Score - Multinomial Naive Bayes Classifier

DATASETS	ROC_AUC_SCORE
DS1	0.7627
DS2	0.7562
DS3	0.7565
DS4	0.8158
DS5	0.7309

Table 22 shows the roc-auc scores of the multinomial naïve bayes classifer corresponding to each dataset.

Table 23 : Roc_Auc Score - Decision Tree Classifier

DATASETS	ROC_AUC_SCORE
DS1	0.8989
DS2	0.8823
DS3	0.8765
DS4	0.9161
DS5	0.8784

Table 23 shows the roc-auc scores of the decision tree classifer corresponding to each dataset.

Table 24 : Roc_Auc Score - Support Vector Machine Classifier

DATASETS	ROC_AUC_SCORE
DS1	0.7924
DS2	0.7846
DS3	0.7861
DS4	0.8178
DS5	0.7826

Table 24 shows the roc-auc scores of the support vector machine classifer corresponding to each dataset.

Table 25 : Roc_Auc Score - Bagging Classifier

Datasets	Roc_Auc_Score
DS1	0.8935
DS2	0.8776
DS3	0.8736
DS4	0.9108
DS5	0.8727

Table 25 shows the roc-auc scores of the bagging classifer corresponding to each dataset.

Table 26 : Roc_Auc Score - Random Forest Classifier

Datasets	Roc_Auc_Score
DS1	0.8938
DS2	0.8786
DS3	0.8732
DS4	0.9116
DS5	0.8734

Table 26 shows the roc-auc scores of the random forest classifer corresponding to each dataset.

Table 27 : Roc_Auc Score - Adaboost Classifier

Datasets	Roc_Auc_Score
DS1	0.8066
DS2	0.7986
DS3	0.7977
DS4	0.8334
DS5	0.7945

Table 27 shows the roc-auc scores of the Adaboost classifer corresponding to each dataset.

The following tables display the confusion matrices for each dataset for each classifier used.

The confusion matrices for each dataset for multinomial naïve bayes classifier are shown in Table 28.

Table 28: Confusion Matrix for Multinomial Naive Bayes Classifier

Datasets		Clickbait	Not Clickbait
DS1	Clickbait	3534	1466
	Not Clickbait	907	4093
DS2	Clickbait	6935	3065
	Not Clickbait	1810	8190
DS3	Clickbait	10428	4572
	Not Clickbait	2732	12268
DS4	Clickbait	2930	2070
	Not Clickbait	692	9308
DS5	Clickbait	8236	1764
	Not Clickbait	2272	2728

The confusion matrices for each dataset for decision tree classifier are shown in Table 29.

Table 29: Confusion Matrix for Decision Tree Classifier

Datasets		Clickbait	Not Clickbait
DS1	Clickbait	4353	647
	Not Clickbait	364	4636
DS2	Clickbait	8378	1622
	Not Clickbait	731	9269
DS3	Clickbait	12381	2619
	Not Clickbait	1084	13916
DS4	Clickbait	4059	941
	Not Clickbait	317	9683
DS5	Clickbait	8914	1086
	Not Clickbait	737	4263

The confusion matrices for each dataset for support vector machine classifier are shown in Table 30.

Table 30: Confusion Matrix for Support Vector Machines

Datasets		Clickbait	Not Clickbait
DS1	Clickbait	3683	1317
	Not Clickbait	759	4241
DS2	Clickbait	7181	2819
	Not Clickbait	1489	8511
DS3	Clickbait	11113	3887
	Not Clickbait	2528	12472
DS4	Clickbait	2772	2228
	Not Clickbait	505	9495
DS5	Clickbait	7828	2172
	Not Clickbait	1088	3912

The confusion matrices for each dataset for bagging classifier are shown in Table 31.

Table 31: Confusion Matrix for Bagging Classifier

Datasets		Clickbait	Not Clickbait
DS1	Clickbait	4232	768
	Not Clickbait	297	4703
DS2	Clickbait	8200	1800
	Not Clickbait	647	9353
DS3	Clickbait	12167	2833
	Not Clickbait	957	14043
DS4	Clickbait	3931	1069
	Not Clickbait	268	9732
DS5	Clickbait	8811	1189
	Not Clickbait	720	4280

The confusion matrices for each dataset for random forest classifier are shown in Table 32.

Table 32: Confusion Matrix for Random Forest Classifier

Datasets		Clickbait	Not Clickbait
DS1	Clickbait	4245	755
	Not Clickbait	307	4693
DS2	Clickbait	8249	1751
	Not Clickbait	677	9323
DS3	Clickbait	12146	2854
	Not Clickbait	949	14051
DS4	Clickbait	3940	1060
	Not Clickbait	265	9735
DS5	Clickbait	8799	1201
	Not Clickbait	697	4303

The confusion matrices for each dataset for Adaboost classifier are shown in Table 33.

Table 33: Confusion Matrix for Adaboost Classifier

Datasets		Clickbait	Not Clickbait
DS1	Clickbait	3684	1316
	Not Clickbait	618	4382
DS2	Clickbait	7187	2813
	Not Clickbait	1214	8786
DS3	Clickbait	10783	4217
	Not Clickbait	1850	13150
DS4	Clickbait	3067	1933
	Not Clickbait	566	9434
DS5	Clickbait	8636	1637
	Not Clickbait	1445	3555

6.4 Discussions

From all the results of the experiments, we can come to some conclusions about the classifiers and type of training data that can be used when detecting clickbaits.

We can see that all the classifiers achieve highest precision, recall and f-measure scores for the dataset DS4. Out of all the classifiers used, the decision tree classifier performs the best followed by both the bagging classifier and the random forest classifier.

We can say that all the classifiers give better accuracy with the DS4 dataset. Out of the simple learners, the decision tree classifier gives the highest accuracy while out of the ensemble learners; the random forest classifier gives the highest accuracy. The decision tree classifier outperforms the other classifiers in terms of accuracy. The dataset DS4 is an unbalanced dataset consisting of 5,000 clickbaits and 10,000 authentic news headlines. The dataset DS5 performs worst of all the datasets.

CHAPTER - 7
CONCLUSION AND FUTURE WORK

Chapter 7 – Conclusion And Future Work

7.1 Introduction

In Section 7.2, we summarise the conclusions of our project and briefly explain the future work which can be done in this field in Section 7.3.

7.2 Conclusion

The main goal of this project was that given any phrase representing a headline, we must be able to identify if it is a clickbait or an authentic news headline. We strived to achieve this goal by using few of the many available machine learning algorithms for classification. We divide the entire data into 5 different data sets, 3 of which are balanced and 2 are unbalanced. 3 balanced data sets with different number of samples of clickbaits and news headlines are used to analyse the performance of the learners. The reason for dividing the data into unbalanced sets is because in the real-world scenario the data available is obviously varying in proportions; sometimes there may be a high prevalence of clickbaits than news headlines.

The features which we have considered are those which can be easily extracted from the title of the headline without much difficulty. These features are very simple and fundamental in the process of identifying a clickbait.

From the results obtained, we can observe that the unbalanced data set DS4 with more headlines than clickbaits gives the best performance for all the classifiers used. This can be attributed to the fact that as of now the occurrences of clickbaits are less compared to those of authentic news headlines on the web.

Contrary to the expectation that ensemble learners tend to perform better than individual learners, the results that we observe conclude that out of both the individual and ensemble learners, the C4.5 decision tree algorithm with an accuracy of 0.9161 is the better choice when utilized to detect clickbaits. Of the ensemble classifiers, the random forest classifier achieves the highest performance with an accuracy of 0.9116.

We used accuracy, precision, recall, f-measure, and roc-auc scores as metrics for comparing the performance of the classifiers. All the metrics show the same trend in the results.

7.3 Future Work

The features that we have used are those which we have extracted from only the title of the headline. Similarly, many more features can be extracted from other attributes of the headline like the content (we can consider either the entire content or a part of it), the URL of the link or the website on which the clickbait headline is posted. The content of the clickbait can be cross-checked with the headline to see if it is a clickbait or not. The strategies of natural language processing can be applied in a more advanced manner to detect clickbaits.

Sometimes, clickbaits are accompanied by pictures also. The presence of images and the type of image can also be considered as additional features.

Many other learners can also be used for classification of headlines. Further experiments can be performed by varying the parameters of the classifiers. The type of base estimators can be varied along with the number of estimators used in the ensemble learner.

REFERENCES:

[1] “Oxford Dictionary - Definition of Clickbait.” [Online]. Available: http://www.oxforddictionaries.com/definition/english/clickbait.

[2] “Merriam Webster Dictionary - definition of clickbait.” [Online]. Available: http://www.merriam-webster.com/dictionary/clickbait.

[3] G. Loewenstein, “The psychology of curiosity: A review and reinterpretation,” 1994.

[4] J. N. Blom and K. R. Hansen, “Click bait: Forward-reference as lure in online news headlines,” *Journal of Pragmatics*, vol. 76, pp. 87–100.

[5] M. Potthast, S. Köpsel, B. Stein, and M. Hagen, “Clickbait Detection,” no. 1.

[6] “You Can’t Not Click: Weighing the Pros & Cons of Clickbait.” [Online]. Available: http://www.wordstream.com/blog/ws/2014/07/15/clickbait.

[7] P. Biyani, K. Tsioutsiouliklis, and J. Blackmer, “‘ 8 Amazing Secrets for Getting More Clicks ’: Detecting Clickbaits in News Streams Using Article Informality.”

[8] “News Feed FYI: Click-baiting.” [Online]. Available: http://newsroom.fb.com/news/2014/08/news-feed-fyi-click-baiting/.

[9] B. Vijgen, “The Listicle: An Exploring Research on an Interesting Shareable New Media Phenomenon,” *Studia Ubb Ephemerides*, pp. 103–122, 2014.

[10] J. Reis, P. Olmo, R. Prates, H. Kwak, and J. An, “Breaking the News : First Impressions Matter on Online News,” *Proceedings of the Ninth International Conference on Weblogs and Social Media, Oxford, UK, May 26-29, 2015*, pp. 357–366, 2015.

[11] “Clickbaits Dataset.” [Online]. Available: https://docs.google.com/spreadsheets/d/1WSx45rT4jZfysmZfzJtjaPO7AxW4XMaJYaCUd5HB2ns/edit#gid=2121187934.

[12] “You’ll Be Outraged at How Easy It Was to Get You to Click on This Headline.” [Online]. Available: http://www.wired.com/2015/12/psychology-of-clickbait/.

[13] “List of Bad-Words.” [Online]. Available: http://www.cs.cmu.edu/?biglou/resources/bad-words.txt.

[14] “Why You Can’t Resist Clicking on This Article: The Clickbait Conundrum.”

[Online]. Available: http://blog.hubspot.com/marketing/clickbait-conundrum-history-psychology-ethics .

[15] Pedregosa, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[16] A. McCallum and K. Nigam, “A Comparison of Event Models for Naive Bayes Text Classification,” *AAAI/ICML-98 Workshop on Learning for Text Categorization*, pp. 41–48, 1998.

[17] A. M. Kibriya, “Multinomial naive bayes for text categorization revisited,” in *Advances in Artificial Intelligence*, 2004, pp. 488–499.

[18] A. S. Galathiya, “Classification with an improved Decision Tree Algorithm,” vol. 46, no. 23, pp. 1–6, 2012.

[19] S. Ruggieri, “Efficient C4. 5 [classification algorithm],” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 14, no. 2, pp. 438–444, 2002.

[20] T. Joachims, “Text Categorization with Support Vector Machines: Learning with Many Relevant Features,” *Machine Learning*, vol. 1398, no. LS-8 Report 23, pp. 137–142, 1998.

[21] P. Michel and R. El Kaliouby, “Real time facial expression recognition in video using support vector machines,” *Proceedings of the 5th international conference on Multimodal interfaces - ICMI ’03*, p. 258, 2003.

[22] X. Wang and J. Tian, “Gene Selection for Cancer Classification using Support Vector Machines,” *Computational and mathematical methods in medicine*, vol. 2012, p. 586246, 2012.

[23] V. Vapnik, *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[24] D. Lindsay, “An introduction to Support Vector Machine implementations in MATLAB,” 2003.

[25] K.-L. Li, “Active learning with simplified SVMs for spam categorization,” in *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*, 2002, pp. 1198–1202.

[26] Fradkin, Dmitriy, and Muchnik, “Support vector machines for classification,”

Discrete methods in epidemiology, vol. 70, pp. 13–20, 2006.

[27] A. Rahman and S. Tasnim, “Ensemble Classifiers and Their Applications : A Review,” *International Journal of Computer Trends and Technology (IJCTT)*, vol. 10, no. 1, pp. 31–35, 2014.

[28] T. G. Dietterich, “Ensem ble Methods in Mac hine Learning,” *Multiple Classifier Systems*, vol. 1857, pp. 1–15, 2000.

[29] L. Rokach, “Ensemble-based classifiers,” *Artificial Intelligence Review*, vol. 33, no. 1–2, pp. 1–39, 2010.

[30] Polikar and Robi, “Ensemble based systems in decision making.,” *Circuits and systems magazine, IEEE*, pp. 21–45, 2006.

[31] L. Breiman, “Bagging Predictors,” *Machine Learning*, vol. 24, no. 421, pp. 123–140, 1996.

[32] L. Breiman, “Random Forests,” *Machine Learning*, vol. 45, no. 5, pp. 1–35, 1999.

[33] a Liaw and M. Wiener, “Classification and Regression by randomForest,” *R news*, vol. 2, no. December, pp. 18–22, 2002.

[34] R. E. Schapire, “The Strength of Weak Learnability,” *Machine Learning*, vol. 5, no. 2, pp. 197–227, 1990.

[35] Y. Freund and R. R. E. Schapire, “Experiments with a New Boosting Algorithm,” *International Conference on Machine Learning*, pp. 148–156, 1996.

[36] J. Friedman, T. Hastie, and R. Tibshirani, “Additive logistic regression: A statistical view of boosting,” *Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[37] J. R. Quinlan, “Bagging, boosting, and C4.5,” *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, vol. 5, no. Quinlan 1993, pp. 725–730, 2006.