SARCASTIC COMMENT DETECTION

A project dissertation submitted to the Bharathidasan University
in partial fulfillment of the requirements
for the award of the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

Submitted by

D.NICHOLAS AROCKIARAJ
Register Number: 185214141

Under the guidance of
Dr. K.RAJKUMAR, M.Sc., M.Phil., Ph.D.,
Associate Professor

PG DEPARTMENT OF COMPUTER SCIENCE (SHIFT-I)
BISHOP HEBER COLLEGE (AUTONOMOUS)
(Ranked 4th at National Level by MHRD through NIRF-2017)
(Nationally Reaccredited at the 'A' Grade by NAAC with the CGPA of 3.58 out of 4)
(Recognized by UGC as "College with Potential for Excellence")
(Affiliated to Bharathidasan University)

TIRUCHIRAPPALLI-620 017

APRIL – 2020

DECLARATION

I hereby declare that the project work presented is originally done by me under the guidance of **Dr. K. Rajkumar, M.Sc., M.Phil., Ph.D., PG Department of Computer Science (Shift-I), Bishop Heber College (Autonomous), Tiruchirappalli-620 017**, and has not been included in any other thesis/project submitted for any other degree.

Name of the Candidate	**:**	**D.NICHOLAS AROCKIARAJ**
Register Number	**:**	**185214141**
Batch	**:**	**2018-2020**

Signature of the Candidate

Dr. K. Rajkumar, M.Sc., M.Phil., Ph.D.,
Associate Professor and Head,
PG Department of Computer Science (Shift-I),
Bishop Heber College (Autonomous),
Tiruchirappalli – 620017.

Date:

CERTIFICATE

This is to certify that the project work titled **"SARCASTIC COMMENT DETECTION"** is a bonafide record of the project work done by **D.NICHOLAS AROCKIARAJ, Register Number: 185214141** in partial fulfillment of the requirements for the award of the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE** during the period **2018- 2020.**

Place: **Signature of the Guide**

PG DEPARTMENT OF COMPUTER SCIENCE (SHIFT-I),
BISHOP HEBER COLLEGE (AUTONOMOUS),
(Ranked 4th at National Level by MHRD through NIRF-2017)
(Nationally Reaccredited at the 'A' Grade by NAAC with the CGPA of 3.58 out of 4)
(Recognized by UGC as "College with Potential for Excellence")
(Affiliated to Bharathidasan University)

TIRUCHIRAPPALLI - 620 017

Date:

Course Title: Project **Course Code: P18CS4PJ**

CERTIFICATE

The Viva-Voce examination for the candidate **D.NICHOLAS AROCKIARAJ Register Number: 185214141** was held on _____.

Signature of the HOD **Signature of the Guide**

Examiners:

1.

2.

ACKNOWLEDGEMENT

I am grateful to GOD Almighty, who showered his blessings on me throughout this project and who has been an ineffable source of strength and inspiration in completing the project.

I am extremely thankful and indebted to **Dr. D. PAUL DHAYABARAN M.Sc., M.Phil., PGDCA., Ph.D., Principal**, Bishop Heber College, Tiruchirappalli, for providing me with the facilities and permission to carry out the project

I am very proud and thankful to **Dr. K. RAJKUMAR, M.Sc., M.Phil., Ph.D., Associate professor** and **Head, PG Department of Computer Science (Shift-I),** Bishop Heber College, Tiruchirappalli. for him encouragement, valuable guidance, suggestion and support to do this project.

I would like to thanks all my staff members of PG Department of Computer Science (Shift-I**)**,. Who put their effort to shape me. I will be failing in my duty if I don't thank my friends who stood with me each and every activity as great wall by extending their excellent support.

Finally. I thank each and every one who has made a contribution towards the successful completion of my project

ABSTRACT

Recent years have seen a huge growth in the use of social media platforms by the people to voice their opinion about a variety of topics, which may lead to comments being ambiguous. For example, sometimes some comments might be misunderstood and may lead to conflict between the reader and the writer. A similar kind of pattern has been seen when it comes to Newspaper headlines or any comment or post on social media. We come across a lot of writing materials throughout the day and while some of them are very straightforward, some can be sarcastic. This may seem offensive to certain groups of readers and result in misunderstanding the subtle nature of humor added to the headlines. Therefore, there is a need to separate the headlines or comments or any writing in general, based on their nature as sarcastic or non-sarcastic. This Project work helps to address this problem by predicting Comments to be sarcastic or not. We applied different Machine learning models and performed a comparative analysis between them. our best performing model achieves a testing accuracy of 68%

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW:

Reddit Comments Sarcasm Detection is a challenging problem, which lies in the domain of Natural Language Understanding. The project involves using the publicly available "Sarcasm on Reddit" dataset to build a model that is able to classify a sarcastic comment on social media platform Reddit. This problem is unique in its nature as it involves differentiating between sentiment and sarcasm based on not just the comments but other features like subreddit(topic), parent comment and score against each comment.

To solve this problem a Machine Learning pipeline is built and the data is passed through it. It includes the following modules:

1. Data Collection
2. Data Preprocessing
3. Model Building
4. Model Evaluation

The end product will be a trained model which would be able to generalize what it has learned from training data and distinguish between sarcastic and non-sarcastic comments on Reddit with good accuracy.

1.2. BACKGROUND:

This project sarcastic comment detection has been based on the machine learning model and the platform used in this model is python. This coding accuracy is testing in jupyter notebook. Both machine learning and natural language processing techniques will play a massive role in this project. This section will provide a brief and general overview of some relevant areas of ML and NLP in regards to this project.

1.3. OBJECTIVES:

The main goals of this project are:

- To build a model that would different between sarcastic vs. non-sarcastic comments on Reddit.
- Achieve high recall score.
- Conduct an analysis and provide a brief comparison of different approaches used.

1.4 PURPOSE, SCOPE AND APPLICABILITY:

- **PURPOSE:**

The purpose of Sarcasm comment detection, which is both positively funny and negatively nasty, plays an important part in human social interaction. But sometimes it is difficult to detect whether someone is making fun of us with some irony. So to make it easy we built something which helps you in detecting sarcastic text.

- **SCOPE**

Build a sarcasm detection model that would detect sarcasm in commentary on Reddit utilizing several features that are available with each comment.

The following steps will be taken to accomplish this task:

- Build and test the aforementioned ML pipeline using a subset of the actual 1.3 million Reddit Comments.
- Train models on training set taken from entire dataset and establish baseline accuracy.
- Obtain good performance scores in terms of precision, recall and accuracy on validation/test data.
- Comparing results using performance metrics like precision, recall and accuracy of different models trained with different feature sets.

2.1. PROBLEM DEFINITION

Our task is to train a model by using machine learning algorithm which can predict if a statement is sarcastic or not.

CHAPTER 2

DATA COLLECTION

2. SARCASTIC COMMENTS REDDIT DATASET:

2.1 Context

This dataset contains 1.3 million Sarcastic comments from the Internet commentary website Reddit. The dataset was generated by scraping comments from Reddit (not by me :)) containing the \s (sarcasm) tag. This tag is often used by Redditors to indicate that their comment is in jest and not meant to be taken seriously, and is generally a reliable indicator of sarcastic comment content.

2.2 Content

Data has balanced and imbalanced (i.e true distribution) versions. (True ratio is about 1:100). The corpus has 1.3 million sarcastic statements, along with what they responded to as well as many non-sarcastic comments from the same source.

Labelled comments are in the train-balanced-sarcasm.csv file.

COLUMNS:

- ***Label***: Sarcastic or not
- ***Comment***: Reply to a Parent Reddit comment
- ***author*** :Person who commented
- ***subreddit****:* Commented under which subreddit
- ***score***: Number of upvotes -(minus) Number of downvotes.
- ***Ups***: Number of upvotes
- ***Downs***: Number of downvotes
- ***date:*** Commented date
- ***created_utc***: Commented time in the UTC Timezone
- ***parent_comment***: The Parent Reddit comment to which sarcastic replies are made

CHAPTER 3

DATA PREPROCESSING

3. DATA PREPROCESSING:

Data preprocessing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we preprocess our data before feeding it into our model.

3.1. Tokenization

Is the process of segmenting running text into sentences and words. In essence, it's the task of cutting a text into pieces called *tokens*, and at the same time throwing away certain characters, such as punctuation. Following our example, the result of tokenization would be:

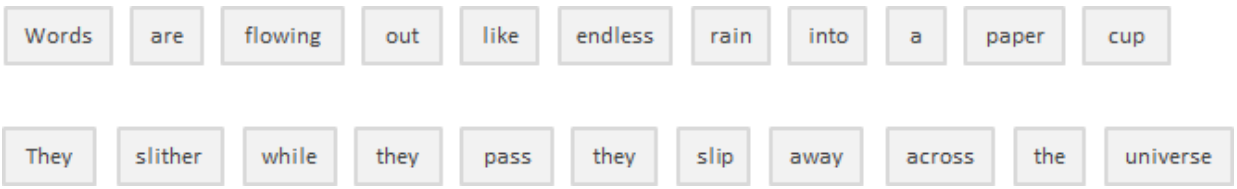

Figure 3.1 Tokenization

Pretty simple, right? Well, although it may seem quite basic in this case and also in languages like English that separate words by a blank space (called segmented languages) not all languages behave the same, and if you think about it, blank spaces alone are not sufficient enough even for English to perform proper tokenizations. Splitting on blank spaces may break up what should be considered as one token, as in the case of certain names (e.g. San Francisco or New York) or borrowed foreign phrases (e.g. laissez faire).

Tokenization can remove punctuation too, easing the path to a proper word segmentation but also triggering possible complications. In the case of periods that follow abbreviation (e.g. dr.), the period following that abbreviation should be considered as part of the same token and not be removed.

The tokenization process can be particularly problematic when dealing with biomedical text domains which contain lots of hyphens, parentheses, and other punctuation marks.

3.2. Stop Words Removal:

Includes getting rid of common language articles, pronouns and prepositions such as “and”, “the” or “to” in English. In this process some very common words that appear to provide little or no value to the NLP objective are filtered and excluded from the text to be processed, hence removing widespread and frequent terms that are not informative about the corresponding text.

Stop words can be safely ignored by carrying out a lookup in a pre-defined list of keywords, freeing up database space and improving processing time.

There is no universal list of stop words. These can be pre-selected or built from scratch. A potential approach is to begin by adopting pre-defined stop words and add words to the list later on. Nevertheless it seems that the general trend over the past time has been to go from the use of large standard stop word lists to the use of no lists at all.

The thing is stop words removal can wipe out relevant information and modify the context in a given sentence. For example, if we are performing a sentiment analysis we might throw our algorithm off track if we remove a stop word like “not”. Under these conditions, you might select a minimal stop word list and add additional terms depending on your specific objective.

3.3 Stemming:

Refers to the process of slicing the end or the beginning of words with the intention of removing affixes (lexical additions to the root of the word).

Affixes that are attached at the beginning of the word are called prefixes (e.g. "astro" in the word "astrobiology") and the ones attached at the end of the word are called suffixes (e.g. "ful" in the word "helpful").

The problem is that affixes can create or expand new forms of the same word (called *inflectional* affixes), or even create new words themselves (called *derivational* affixes). In English, prefixes are always derivational (the affix creates a new word as in the example of the prefix "eco" in the word "ecosystem"), but suffixes can be derivational (the affix creates a new word as in the example of the suffix "ist" in the word "guitarist") or inflectional (the affix creates a new form of word as in the example of the suffix "er" in the word "faster").

Ok, so how can we tell the difference and chop the right bit?

Figure 3.3 Stemming

A possible approach is to consider a list of common affixes and rules (Python and R languages have different libraries containing affixes and methods) and perform stemming based on them, but of course this approach presents limitations. Since stemmers use algorithmics approaches, the result of the stemming process may not be an actual word or even change the word (and sentence) meaning. To offset this effect you can edit those predefined methods by adding or removing affixes and rules, but you must consider that you might be improving the performance in one area while producing a degradation in another one. Always look at the whole picture and test your model's performance.

So if stemming has serious limitations, why do we use it? First of all, it can be used to correct spelling errors from the tokens. **Stemmers are simple to use and run very fast** (they perform simple operations on a string), and if speed and performance are

important in the NLP model, then stemming is certainly the way to go. Remember, we use it with the objective of improving our performance, not as a grammar exercise.

3.4 Lemmatization:

Has the objective of reducing a word to its base form and grouping together different forms of the same word. For example, verbs in past tense are changed into present (e.g. "went" is changed to "go") and synonyms are unified (e.g. "best" is changed to "good"), hence standardizing words with similar meaning to their root. Although it seems closely related to the stemming process, lemmatization uses a different approach to reach the root forms of words.

Lemmatization resolves words to their dictionary form (known as lemma*) for which it requires detailed dictionaries in which the algorithm can look into and link words to their corresponding lemmas.*

For example, the words "*running", "runs"* and *"ran"* are all forms of the word "*run"*, so "*run"* is the lemma of all the previous words.

Figure 3.4 Lemmatization

Lemmatization also takes into consideration the context of the word in order to **solve other problems like disambiguation**, which means it can discriminate between identical words that have different meanings depending on the specific context. Think about words like "bat" (which can correspond to the animal or to the metal/wooden club used in baseball) or "bank" (corresponding to the financial institution or to the land alongside a body of water). By providing a part-of-speech parameter to a word (whether it is a noun, a verb, and so on) it's possible to define a role for that word in the sentence and remove disambiguation.

As you might already pictured, lemmatization is a much more resource-intensive task than performing a stemming process. At the same time, since it requires more knowledge about the language structure than a stemming approach, it **demands more computational power** than setting up or adapting a stemming algorithm.

CHAPTER 4

MODEL BUILDING

4. Model Building

4.1 Machine Learning:

Machine learning field is a subfield from the broad field of artificial intelligence, this aims to make machines able to learn like a human. Learning here means understood, observe and represent information about some statistical phenomenon. Machine Learning is automatically learning to make predictions on current data based on past history. It is divided into Supervised and Unsupervised Learning. Predicting a continuous quantitative Output value is referred to Regression Problem. Predicting a non-numerical, Qualitative value or categorical Output value is Classification. Observing only Input Variables and No Output variables and grouping those input variables depending on their characteristics called Clustering. Input variables are referred as Predictors, Independent or Features. Output variables are referred as Response or Dependent variable.

Classification:

Classification is technique to categorize our data into a desired and distinct number of classes where we can assign label to each class.

Applications of Classification are: speech recognition, handwriting recognition, biometric identification, document classification etc.

Classifier can be:

Binary classifiers: Classification with only 2 distinct classes or with 2 possible outcomes

example: Male and Female

example: classification of spam email and non spam email

example: classification of author of book

example: positive and negative sentiment

Multi-Class classifiers: Classification with more than two distinct classes.

example: classification of types of soil

example: classification of types of crops

example: classification of mood/feelings in songs/music

4.2 Machine Learning Algorithms:

4.2.1 Decision Tree:

Given a data of attributes together with its classes, a decision tree produces a sequence of rules that can be used to classify the data.

Description: Decision Tree, as it name says, makes decision with tree-like model. It splits the sample into two or more homogeneous sets (leaves) based on the most significant differentiators in your input variables. To choose a differentiator (predictor), the algorithm considers all features and does a binary split on them (for categorical data, split by cat; for continuous, pick a cut-off threshold). It will then choose the one with the least cost (i.e. highest accuracy), and repeats recursively, until it successfully splits the data in all leaves (or reaches the maximum depth).

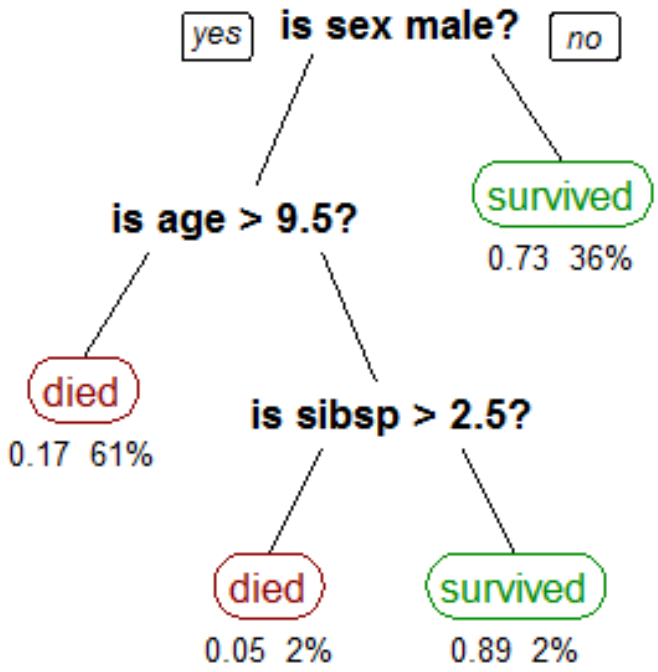

Figure 4.1 Tree like representation of data in Decision tree

Advantages: Decision Tree is simple to understand and visualize, requires little data preparation, and can handle both numerical and categorical data.

Disadvantages: Decision tree can create complex trees that do not generalize well, and decision trees can be unstable because small variations in the data might result in a completely different tree being generated.

4.2.2 Random Forest

Random forest is an ensemble model that grows multiple tree and classify objects based on the "votes" of all the trees. i.e. An object is assigned to a class that has most votes from all the trees. By doing so, the problem with high bias (overfitting) could be alleviated.(— from Kaggle).

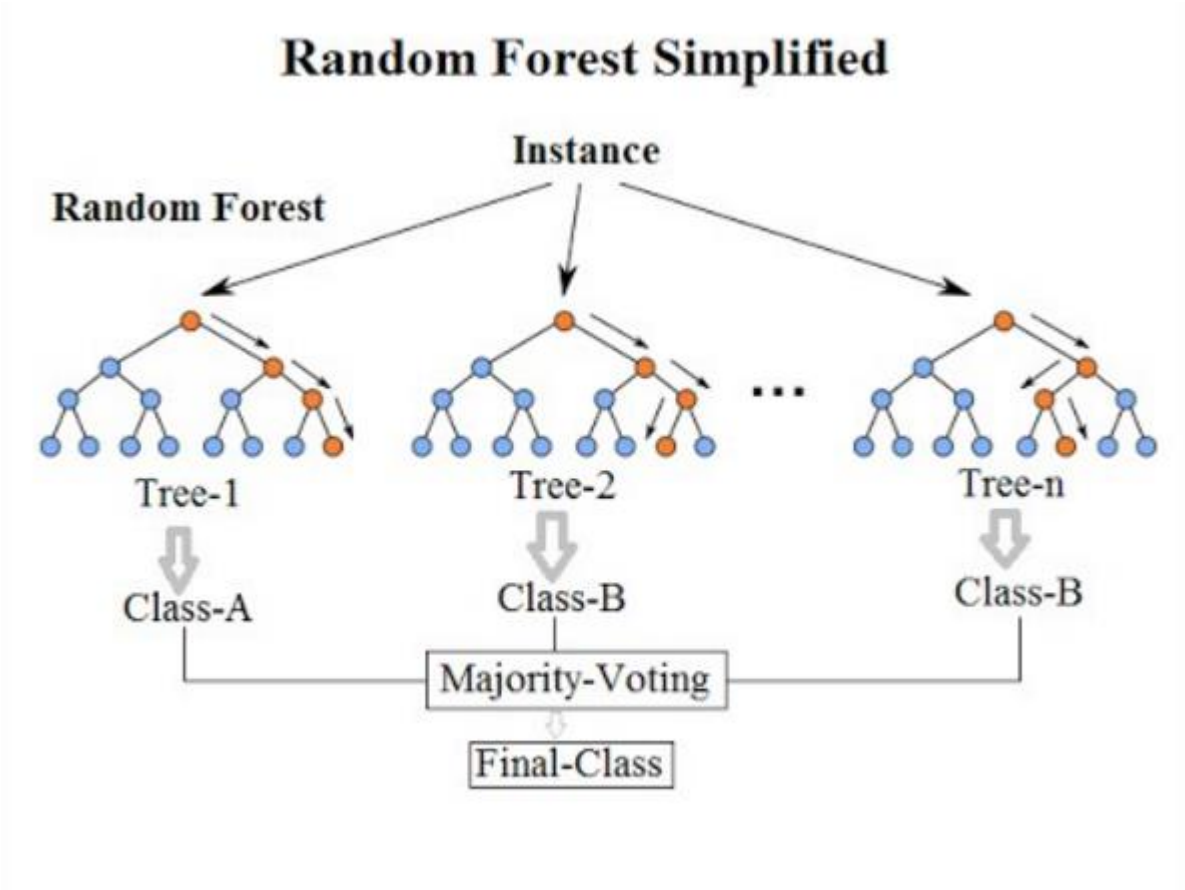

Figure 4.2. Random forest

Random forest classifier is a meta-estimator that fits a number of decision trees on various sub-samples of datasets and uses average to improve the predictive accuracy of the model and controls over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement.

Pros of RF:

- It could handle large data set with high dimensionality, output **Importance of Variable**, useful to explore the data
- Could handle missing data while maintaining accuracy

Cons of RF:

- Could be a black box, users have little control on what the model does

Advantages: Reduction in over-fitting and random forest classifier is more accurate than decision trees in most cases.

Disadvantages: Slow real time prediction, difficult to implement, and complex algorithm.

4.2.3 Adaptive Boosting(AdaBoost):

AdaBoost is a boosting ensemble model and works especially well with the decision tree. Boosting model's key is learning from the previous mistakes, e.g. misclassification data points.

AdaBoost learns from the mistakes by increasing the weight of misclassified data points.

Let's illustrate **how AdaBoost adapts**.

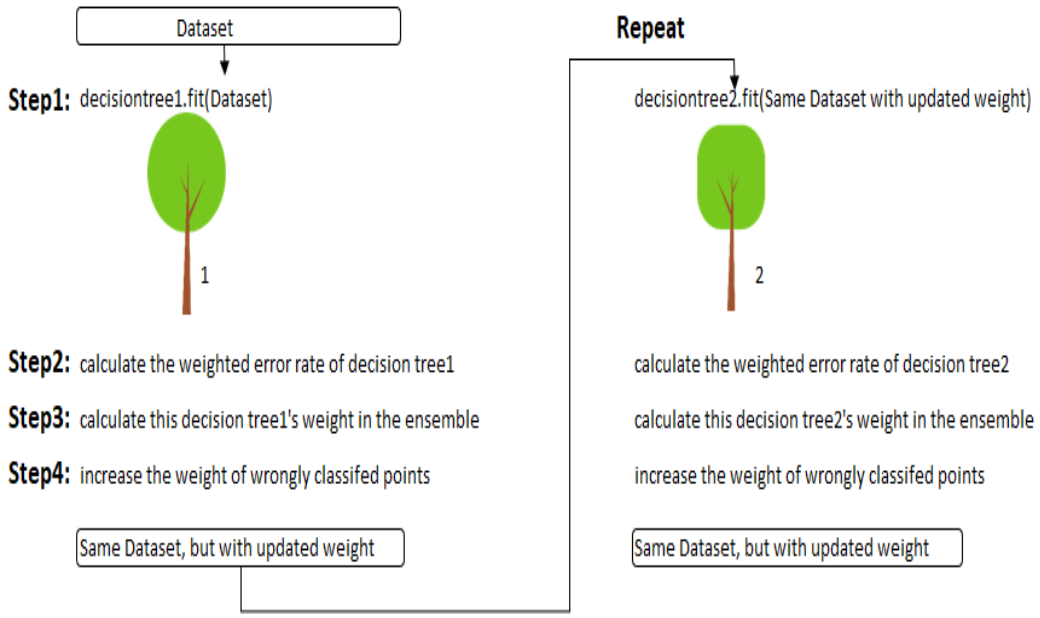

Figure 4.3 Adaboost

Step 0: **Initialize the weights** of data points. if the training set has 100 data points, then each point's initial weight should be 1/100 = 0.01.

Step 1: **Train** a decision tree

Step 2: **Calculate the weighted error rate (e)** of the decision tree. **The weighted error rate (e)** is just how many wrong predictions out of total and you treat the wrong predictions differently based on its data point's weight. **The higher the weight**, **the more the corresponding error will be weighted** during the calculation of the (e).

Step 3: **Calculate this decision tree's weight** in the ensemble

the weight of this tree = learning rate * log((1 — e) / e)

- the higher weighted error rate of a tree, □, the less decision power the tree will be given during the later voting
- the lower weighted error rate of a tree, □, the higher decision power the tree will be given during the later voting

Step 4: **Update weights** of wrongly classified points

the weight of each data point =

- if the model got this data point correct, the weight stays the same
- if the model got this data point wrong, the new weight of this point = old weight * np.exp(weight of this tree)

Note: The higher the weight of the tree (more accurate this tree performs), the more boost (importance) the misclassified data point by this tree will get. The weights of the data points are normalized after all the misclassified points are updated.

Step 5: **Repeat** Step 1(until the number of trees we set to train is reached)

Step 6: **Make the final prediction**

The AdaBoost makes a new prediction by adding up the weight (of each tree) multiply the prediction (of each tree). Obviously, the tree with higher weight will have more power of influence the final decision.

4.2.4 eXtreme Gradient Boosting(XGBoost):

XGBoost algorithm was developed as a research project at the University of Washington. Tianqi Chen and Carlos Guestrin presented their paper at SIGKDD Conference in 2016 and caught the Machine Learning world by fire. Since its introduction, this algorithm has not only been credited with winning numerous Kaggle competitions but also for being the driving force under the hood for several cutting-edge industry applications.

Figure 4.4 XGBoost

Algorithmic Enhancements:

1. **Regularization**: It penalizes more complex models through both LASSO (L1) and Ridge (L2) regularization to prevent overfitting.

2. **Sparsity Awareness**: XGBoost naturally admits sparse features for inputs by automatically 'learning' best missing value depending on training loss and handles different types of sparsity patterns in the data more efficiently.

3. **Weighted Quantile Sketch:** XGBoost employs the distributed weighted Quantile Sketch algorithm to effectively find the optimal split points among weighted datasets.

4. **Cross-validation**: The algorithm comes with built-in cross-validation method at each iteration, taking away the need to explicitly program this search and to specify the exact number of boosting iterations required in a single run.

CHAPTER 5

MODEL EVALUATION

5. Evaluation Metrics:

5.1 Comparison Table of Evaluation Metrics :

Algorithms	Training accuracy	Test accuracy	Precision	Recall	F1
Decision tree	62	62	61	62	58
Random forest	64	63	70	63	54
Adaboost	64	64	67	64	57
XGBoost	71	68	68	67	65

Table 5.1

5.2 Comparison of ROC Curve:

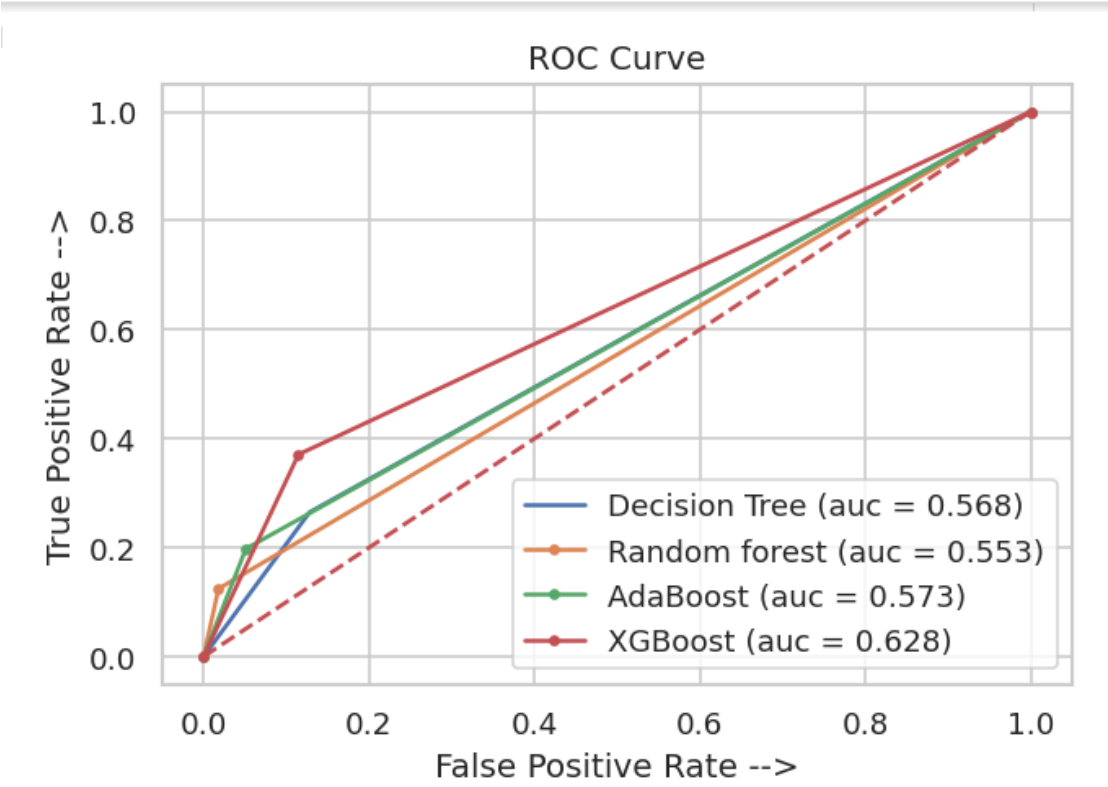

Figure 5.1 ROC curve

CHAPTER 6

CONCLUSION

5.1. Conclusion:

From these project it looks like XGBoost with Count Vectorization is working well and has achieved a test accuracy of 68% which is far better than other models. This model is capable of detecting sarcasm in any Reddit Comment with 68% confidence which is fairly good.

5.2. Future Scope of the Project

In future large sarcasm datasets can be used. Also, cross domain experiment needs to be done to explore how model performs in such situation. Sarcasm is a never-ending problem in obtaining true opinion. Hence data needs to be collected from various different sources and make use of word embedding as well as manual features whenever applicable. For manual feature the dictionaries can be improved by adding more words and also polarity score of the emoticons can included as in this research only the presence of the emoticons is taken into consideration. Along with paragraph vectors, deep learning techniques can be used in future.

5.3. Reflection of Learning:

The technological that I have learnt during the development of project is about the machine learning algorithms and preprocessing techniques, different classifiers etc. and learn about the natural language processing

Reference

1. Bo Pang and Lillian Lee. “Opinion mining and sentiment analysis”. In: Foundations and trends in information retrieval 2.1-2 (2008), pp. 1–135.

2. Kevin Patrick Murphy ,“Machine Learning: a Probabilistic Perspective” ,edition 2, 22 October 2012

3. Steven Bird ,”Natural Language Processing with Python”, O'Reilly Media; 1 edition (12 June 2009)

4. John D. Kelleher,”Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies”, The MIT Press; 1 edition (31 July 2015)

5. Trevor Hastie, Robert Tibshirani, and Jerome Friedman “The Elements of Statistical Learning: Data Mining, Inference, and Prediction” Edition 2 May, 2012

6. Bharti, S., Vachha, B., Pradhan, R., Babu, K. and Jena, S. (2016). Sarcastic sentiment detection in tweets streamed in real time: a big data approach, Digital Communications and Networks 2(3): 108 – 121. Advances in Big Data. URL: http://www.sciencedirect.com/science/article/pii/S235286481630027X

7. Bouazizi, M. and Otsuki Ohtsuki, T. (2016). A pattern-based approach for sarcasm detection on twitter, IEEE Access 4: 5477–5488.

APPENDICES

A.SAMPLE CODE:

```
# import libraries
import pandas as pd
import numpy as np
import nltk
import xgboost
from scipy.sparse import hstack
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="whitegrid")
import re
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
import warnings
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.metrics import precision_recall_fscore_support
warnings.filterwarnings('ignore')
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV

# load dataset
df=pd.read_csv('C:/Users/nick1\Desktop/project/dataset/train-balanced-sarcasm.csv')
```

```
subset_ratio = 0.05
subset_df = df[:int(df.shape[0] * subset_ratio)]
print('Dimensions of subset:', subset_df.shape)

# check if there are any missing values in data
print('Missing values in data:', subset_df.isnull().values.any())
data.isnull().sum()
data = subset_df.dropna()
print('No. of rows before removing missing vals:', subset_df.shape[0])
print('No. of after removing missing vals:', subset_df.shape[0])
print('No. of rows removed:', subset_df.shape[0] - data.shape[0])

# check the distribution of sarcastic vs. non-sarcastic comments
print('No. of Non-sarcastic comments(0):', data[data['label'] == 0].shape[0])
print('No. of Sarcastic comments(1):', data[data['label'] == 1].shape[0])

#visualize distribution of sarcastic vs. non-sarcastic comments
ax = sns.countplot(x="label", data=data)

# basic text preprocessing of data
cols = ['comment', 'subreddit', 'parent_comment']
#stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()
stopwords = stopwords.words('english')

for c in cols:
    # remove special characters
    data[c] = data[c].map(lambda x: re.sub(r'\W', ' ', x))

    # replace multiple spaces with single space
    data[c] = data[c].map(lambda x: re.sub(r'\s+', ' ', x, flags=re.I))

    # remove all single characters
    data[c] = data[c].map(lambda x:re.sub(r'\s+[a-zA-Z]\s+', ' ', x))
```

```
    # covert text to lower case
    data[c] = data[c].str.lower()

    # tokenize text
    data[c] = data[c].str.split()

    # apply stemming
    data[c] = data[c].map(lambda x: ' '.join([lemmatizer.lemmatize(w) for w in x if w
not in stopwords]))

# generate CountVectorizer for comment column
tfidf_comment = CountVectorizer(ngram_range=(1, 2), max_features=50000)
comment_features_tfidf = tfidf_comment.fit_transform(data['comment'])
comment_features_tfidf.shape

# generate CountVectorizer for subreddit column
tfidf_subreddit = CountVectorizer(ngram_range=(1, 2), max_features=50000)
subreddit_features_tfidf = tfidf_subreddit.fit_transform(data['subreddit'])
subreddit_features_tfidf.shape

# generate CountVectorizer for parent_comment column
tfidf_parent_comment = CountVectorizer(ngram_range=(1, 2), max_features=50000)
parent_comment_features_tfidf =
tfidf_parent_comment.fit_transform(data['parent_comment'])
parent_comment_features_tfidf.shape

# standardize continous variables
scaler = StandardScaler()
score = scaler.fit_transform(data[['score', 'ups', 'downs']])
```

```
# stack up all the sparse features matrices horrizontally
y_tfidf = data['label']
X_tfidf = hstack([comment_features_tfidf, subreddit_features_tfidf,
parent_comment_features_tfidf, score])
print(X_tfidf.shape, y_tfidf.shape)

# train test split
X_train_tfidf, X_test_tfidf, y_train_tfidf, y_test_tfidf = train_test_split(X_tfidf,
y_tfidf, test_size=0.2, random_state=0)
print("Dimensions of train set with TF-IDF Features", X_train_tfidf.shape)
print("Dimensions of validation set with TF-IDF Features", X_test_tfidf.shape)

# train logitic DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier
dtree=DecisionTreeClassifier(criterion='gini',max_depth=5)
dtree.fit(X_train_tfidf, y_train_tfidf)

# print accuracy on train and test sets
print("Model Accuracy on Training Set:", round(accuracy_score(y_train_tfidf,
dtree.predict(X_train_tfidf)), 3))
print("Model Accuracy on Test Set:", round(accuracy_score(y_test_tfidf,
dtree.predict(X_test_tfidf)), 3))

# make predictions on test set and print performance metrics
print("Classification Report on Test Set:\n")
print(classification_report(y_test_tfidf, dtree.predict(X_test_tfidf)))
print(confusion_matrix(y_test_tfidf, dtree.predict(X_test_tfidf)))

# train RandomForestClassifier
rfc_tfidf = RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                    criterion='gini', max_depth=100, max_features='auto',
                    max_leaf_nodes=None, max_samples=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=10, min_samples_split=10,
```

```
                    min_weight_fraction_leaf=0.0, n_estimators=1600,
                    n_jobs=-1, oob_score=False, random_state=None,
                    verbose=0, warm_start=False)
rfc_tfidf.fit(X_train_tfidf, y_train_tfidf)


# print accuracy on train and test sets
print("Model Accuracy on Training Set:", round(accuracy_score(y_train_tfidf,
rfc_tfidf.predict(X_train_tfidf)), 3))
print("Model Accuracy on Test Set:", round(accuracy_score(y_test_tfidf,
rfc_tfidf.predict(X_test_tfidf)), 3))

# make predictions on test set and print performance metrics
print("Classification Report on Test Set:\n")
print(classification_report(y_test_tfidf, rfc_tfidf.predict(X_test_tfidf)))
print(confusion_matrix(y_test_tfidf, rfc_tfidf.predict(X_test_tfidf)))

# train AdaBoostClassifier
from sklearn.ensemble import AdaBoostClassifier

clf = AdaBoostClassifier(DecisionTreeClassifier(max_depth=1), n_estimators=500,
learning_rate=0.05, random_state=100)
clf.fit(X_train_tfidf, y_train_tfidf)

# print accuracy on train and test sets
print("Model Accuracy on Training Set:", round(accuracy_score(y_train_tfidf,
clf.predict(X_train_tfidf)), 3))
print("Model Accuracy on Test Set:", round(accuracy_score(y_test_tfidf,
clf.predict(X_test_tfidf)), 3))

# make predictions on test set and print performance metrics
print("Classification Report on Test Set:\n")
print(classification_report(y_test_tfidf, clf.predict(X_test_tfidf)))
print(confusion_matrix(y_test_tfidf, clf.predict(X_test_tfidf)))
```

```
# train XGBoostClassifier
xgclassifier=xgboost.XGBClassifier(colsample_bytree=0.7,gamma=0.2,learning_rate
=0.3,max_depth=10,min_child_weight=5)
xgclassifier.fit(X_train_tfidf, y_train_tfidf)

# print accuracy on train and test sets
print("Model Accuracy on Training Set:", round(accuracy_score(y_train_tfidf,
xgclassifier.predict(X_train_tfidf)), 3))
print("Model Accuracy on Test Set:", round(accuracy_score(y_test_tfidf,
xgclassifier.predict(X_test_tfidf)), 3))

# make predictions on test set and print performance metrics
print("Classification Report on Test Set:\n")
print(classification_report(y_test_tfidf, xgclassifier.predict(X_test_tfidf)))
print(confusion_matrix(y_test_tfidf, xgclassifier.predict(X_test_tfidf)))

#plotting ROC curve
from sklearn.metrics import roc_curve, auc

dtree_fpr, dtree_tpr, threshold = roc_curve( y_test_tfidf , dtree.predict(X_test_tfidf))
auc_dtree = auc(dtree_fpr, dtree_tpr)

rfc_fpr, rfc_tpr, threshold = roc_curve( y_test_tfidf , rfc_tfidf.predict(X_test_tfidf))
auc_rfc = auc(rfc_fpr, rfc_tpr)

clf_fpr, clf_tpr, threshold = roc_curve( y_test_tfidf ,  clf.predict(X_test_tfidf))
auc_clf = auc(clf_fpr, clf_tpr)

xg_fpr, xg_tpr, threshold = roc_curve( y_test_tfidf , xgclassifier.predict(X_test_tfidf))
auc_xg = auc(xg_fpr, xg_tpr)

plt.figure( dpi=120)
```

```
plt.plot(dtree_fpr, dtree_tpr,  linestyle='-', label='Decision Tree (auc = %0.3f)' %
auc_dtree)
plt.plot(rfc_fpr, rfc_tpr, marker='.', label='Random forest (auc = %0.3f)' % auc_rfc)
plt.plot(clf_fpr, clf_tpr, marker='.', label='AdaBoost (auc = %0.3f)' % auc_clf)
plt.plot(xg_fpr, xg_tpr, marker='.', label='XGBoost (auc = %0.3f)' % auc_xg)

plt.xlabel('False Positive Rate -->')
plt.ylabel('True Positive Rate -->')

plt.plot([0,1],[0,1],'r--')
plt.title('ROC Curve')
plt.legend(loc="lower right")

plt.show()
```

B.DATASET

	label	comment	author	subreddit	score	ups	downs	date	created_utc	parent_comment
0	0	NC and NH.	Trumpbart	politics	2	-1	-1	2016-10	2016-10-16 23:55:23	Yeah, I get that argument. At this point, I'd ...
1	0	You do know west teams play against west teams...	Shbshb906	nba	-4	-1	-1	2016-11	2016-11-01 00:24:10	The blazers and Mavericks (The wests 5 and 6 s...
2	0	They were underdogs earlier today, but since G...	Creepeth	nfl	3	3	0	2016-09	2016-09-22 21:45:37	They're favored to win.
3	0	This meme isn't funny none of the "new york ni...	icebrotha	BlackPeopleTwitter	-8	-1	-1	2016-10	2016-10-18 21:03:47	deadass don't kill my buzz
4	0	I could use one of those tools.	cush2push	MaddenUltimateTeam	6	-1	-1	2016-12	2016-12-30 17:00:13	Yep can confirm I saw the tool they use for th...
5	0	I don't pay attention to her, but as long as s...	only7inches	AskReddit	0	0	0	2016-09	2016-09-02 10:35:08	do you find ariana grande sexy ?
6	0	Trick or treating in general is just weird...	only7inches	AskReddit	1	-1	-1	2016-10	2016-10-23 21:43:03	What's your weird or unsettling Trick or Treat...
7	0	Blade Mastery+Masamune or GTFO!	P0k3rm4s7	FFBraveExvius	2	-1	-1	2016-10	2016-10-13 21:13:55	Probably Sephiroth. I refuse to taint his grea...
8	0	You don't have to, you have a good build, buy ...	SoupToPots	pcmasterrace	1	-1	-1	2016-10	2016-10-27 19:11:06	What to upgrade? I have $500 to spend (mainly ...
9	0	I would love to see him at lolla.	chihawks	Lollapalooza	2	-1	-1	2016-11	2016-11-21 23:39:12	Probably count Kanye out Since the rest of his...