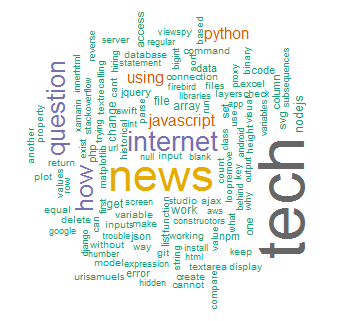
12/10/2016

**Predictive Analytics: Analysis of Stack Overflow Data**

A report to analyse why coders do what they do on Stack Overflow and predict their actions. 

**(generated using R word cloud)**

Analysis Done by:

- Pratik Mrinal

Contents

[1. Executive Summary 2](#_Toc469319410)

[2. Introduction 2](#_Toc469319411)

[2.1 Problem Statement 2](#_Toc469319412)

[2.2 Dataset Details 3](#_Toc469319413)

[2.3 Methods Employed 4](#_Toc469319414)

[3. Literature Review 4](#_Toc469319415)

[4. Process, Implementation, and Result 12](#_Toc469319416)

[4.1 Classify data into spam/non-spam 13](#_Toc469319417)

[4.1.1 Approach 13](#_Toc469319418)

[4.1.2 Implementation 13](#_Toc469319419)

[4.1.3 Result 14](#_Toc469319420)

[4.2 Evaluate quality of answer posts to predict best possible answer 15](#_Toc469319421)

[4.2.1 Approach 15](#_Toc469319422)

[4.2.2 Implementation 16](#_Toc469319423)

[4.2.3 Result 17](#_Toc469319424)

[4.3 Predict comment sentiment using NLP and based on Post features 18](#_Toc469319425)

[4.3.1 Approach 18](#_Toc469319426)

[4.3.2 Implementation 18](#_Toc469319427)

[4.3.3 Result 19](#_Toc469319428)

[5. Conclusion 19](#_Toc469319429)

[6. Individual Roles and Responsibilities 19](#_Toc469319430)

[7. References 20](#_Toc469319431)

[8. Appendix 21](#_Toc469319432)

[Appendix A: Variables 21](#_Toc469319433)

[A1. Variables for Question 1 - Spam Classifier 21](#_Toc469319434)

[A2. Variables for Question 2 – Evaluate best answer 21](#_Toc469319435)

[A3. Variables for Question 3 – Sentiment Analysis using NLP on Post comments 22](#_Toc469319436)

[Appendix B: SQL Queries 23](#_Toc469319437)

[B1. SQL Query for Question 1 - Spam Classifier 23](#_Toc469319438)

[B2. SQL Query for Question 2 – Evaluate best answer 25](#_Toc469319439)

[B3. SQL Query for Question 3 – Sentiment Analysis using NLP on Post comments 27](#_Toc469319440)

[Appendix C : R Code 28](#_Toc469319441)

[C1. Classify data into spam/non-spam (link to complete output) 28](#_Toc469319442)

[C2. Evaluate best possible answer – (link to complete output) 41](#_Toc469319443)

[C3. Predict comment sentiment using NLP 56](#_Toc469319444)

[Appendix D: Terminology and Abbreviations 65](#_Toc469319445)

# 1. Executive Summary

This project involved building classification models to predict characteristics of posts and comments on Stack Overflow, an online platform for discussing programming and software development questions. The primary aim was to identify predictors and construct models which could automate spam detection, suggest the best answer of a question, and gather insights about the posts based on comments.

With over 6.4 million users and approximately 227,000 questions asked per month and answered almost equal number of times, it is imperative to provide best, un-biased suggested answer based on the history of posts and the users involved. Initially, data available from Stack Overflow (on Stack Exchange, see Appendix D) was gathered and analysed. The team tested the viability of the issues and found strong correlations between the components to give interesting results. Proof of concepts were conducted on all the three scenarios and upon getting encouraging results, an approach was designed to handle each issue separately. The prototypes evolved as the study on data progressed and eventually led to a full set of predictors, a lot of them were added while a few were removed. Once the full set of predictors were available, various classification models were applied based on the issue at hand. The report mentions specific technique under each issue. Since there were multiple models which were selected to handle each issue, the best model was chosen based on several factors including the least mis-classification rate.

The results obtained from the study substantiated the motive behind the project. The classification models gave rich results with a capability of predicting best possible answer to a post with almost 95% accuracy. By using Natural Language Processing (NLP), sentiments on the comments were analysed to understand the characteristics of the post on which those comments were posted. As such, the performance of multiple classification models was assessed in the due course of the project.

# 2. Introduction

## 2.1 Problem Statement

Stack Overflow is a collaborative questioning and answering site designed for developers to find answers. As this online community is growing with new users joining every day, it's important to manage the content to avoid irrelevant posts like spams and thus ensure quality of the content. Also, since one question can have multiple answers and only one answer can be marked as the best answer, it would be beneficial to automatically predict and suggest the best answer among the multiple answer. Another aspect of Stack Overflow is the comments which users make on the posts (questions or answer).

The dataset from Stack Overflow can be organized into 5 categories - Posts, Comments, Users, Votes and Badges. The project in addition aims at classify post (question/answer) into spam / non-spam and further evaluate the quality of the answer posts to predict likelihood of it being marked as the best answer. Each post has an option to upvote/downvote. While upvote increases the reputation of the owner by +5, downvote has a negative effect on reputation of both owner (-2) and the user downvoting (-1).

|  |  |
| --- | --- |
| **Users accounts** | **6.4 Million** (1.1 Million answerers and 1.7 Million askers, respectively) |
| **Questions** | **13 Million** (56.76% received best answer, 11.31% no-response questions) |
| **Answers** | **20 Million** (33.93% accepted as best answer) |
| **Votes** | **61 Million** (89.84% Positive votes, average 2.2 upvotes per post, average 0.25 downvotes per post.) |
| **Comments** | **53 Million** (57.25% on answers, 42.75% on questions) |
| **Reputation** | Average reputation points (excluding users who have 1 reputation): 317  Average reputation points (containing users who have 1 reputation): 125 |
| **Tags** | 47,000 types of tags |
| **Attributes** | 188 attributes and 26 tables |

Comments often come in the form of clarifications, suggestions, argument, or contradiction. It is often noted that the community members use comments to express the discontent about the post instead of using the downvote. The reason behind this is that downvotes cause a reduction in reputation points. Hence the users tend to provide a negative feedback as a response to a post instead of downvoting it. We would like to apply Natural Language Processing (NLP) on the comments of each of these posts to understand the correlation between downvoting vs. negative comments and predict the likelihood of posts resulting into negative comments.

## 2.2 Dataset Details

The dataset has been curated from Stack Exchange site available at the link <http://data.stackexchange.com/stackoverflow/queries>. It is available for sharing and reproduction licensed under Creative Commons by ShareAlike 3.0.

Stack Exchange provides a data dump of all user-contributed content on the Stack Overflow. The dataset available and used in this project was last updated on December 1st 2016. The data was fetched via queries run on Stack Exchange Data Explorer tool. This tool facilitates querying from the huge repository and download the data as .csv file.

The data is ***more than 1TB*** and has ***188 columns*** distributed among ***26 tables***. These tables have data about posts, comments, tags, votes, users and reviews. The whole Stack Overflow model is based on reliability of the posts and comments. This makes it essential for the users to know which comment should be relied upon and which should not. The number of votes on a post determines this reliability. Also, Stack Overflow thrives on the community collaboration and gamification lies at the core of community collaboration. The badges, points on upvotes and privileges thus are essential. This data is also available in the dataset.

For this analysis, however, we have used only part of the whole data. We used joins over relevant tables and fetched the required columns while carefully managing the constraint relationships among the necessary columns. The following table summarizes the tables and their respective columns used in this analysis.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table** | **Column** |  | **Table** | **Column** |
| Comments | Id | PostHistory | PostHistoryTypeId |
| PostId | PostId |
| Score | PostHistoryTypes | Name |
| Text | PostTypes | Name |
| UserId | ReviewTasks | ReviewTaskId |
| Post | Id | ReviewTaskTypeId |
| PostTypeId | ReviewTaskResults | RejectionReasonId |
| score | ReviewRejectionReasons | ReviewRejectionFlag |
| Body | ReviewTaskResultTypes | Name |
| LastEditDate | Id |
| CreationDate | Description |
| CommentCount | Name |
| ViewCount | Description |
| FavoriteCount | ReviewTaskStates | Name |
| AnswerCount | Description |
| Title | User | Reputation |
| Tags | Location |
| AcceptedAnswerId | WebsiteUrl |
| PostFeedback | VoteTypeId | AboutMe |
| Votes | Id | Views |
| VoteTypeId | UpVotes |
| PostId | DownVotes |
| VoteTypes | Id |  |  |

## 2.3 Methods Employed

Stack Overflow is a state-of-art web application which is highly scalable and responsive. Being extracted from the from the backup of the application data, The Stack Overflow data is quite stable and has very less missing/ corrupt/ data. Yet, for our analysis we diligently followed the principles of ETL to make our data clean. Followed by the this, we employed the following techniques to do our analysis and the application of these methods will be discussed in details later:

|  |  |
| --- | --- |
| **In-class methods** | **New Methods** |
| * LDA | * SVM |
| * QDA | * NLP |
| * KNN | * C5.0 |
| * Logistic Regression |  |
| * CART |  |
| * LASSO |  |
| * Ridge Regression |  |
| * Dimensionality Reduction |  |
| * Random Forest |  |

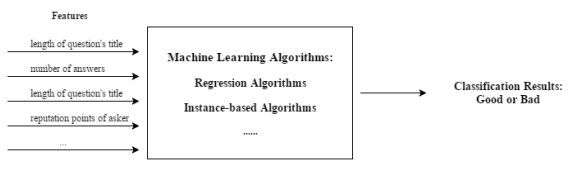
# 3. Literature Review

**Article 1**

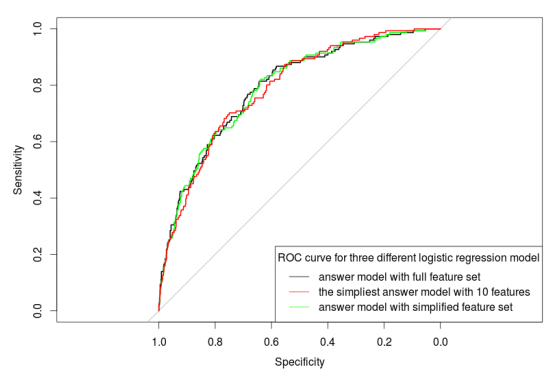
**Complete Reference**:

1. [Daoying Qiu (2015) Evaluation and Prediction of Content Quality in Stack Overflow with Logistic Regression](http://jultika.oulu.fi/files/nbnfioulu-201510282094.pdf)
2. [Ahmad, Rizwan, Dmitriy Kunitskiy, and Aleksander Maricq. "Stack Authority: Predicting Stack Overflow Post Helpfulness Using User Social Authoritativeness."](http://rizwanahmad.net/assets/stack-authority.pdf)

**Objective:** The paper is addressing the problems that users face in getting quality answers from CQA (Collaborative Questioning and Answering) sites. It is important to address this because there are large-scale of questions and answers on CQA sites, hence it is critical to establish automatic evaluation for content quality. User reputation and voting are not strong enough to evaluate content quality because there are many other influential parameters that can define content quality.

**Approach:** The authors have tried to address this problem by building a model to measure and evaluate the content quality of the posts in CQA sites. They have used a multi-methodological approach to perform research on the data. Previous researches have used one of the following methods: factoid, definition, list, complex interactive and target. But in this paper the author has used the features approach of questions and answers to evaluate the content quality. They created a model based on the features of questions and answers like length, number of comments, views, up votes, reputation, profile summary, views, etc.

The model’s coefficients, p-value, multiple R2 and adjusted R2 values are reported and parameters with significance level greater than 0.05 are considered for further analysis. They also took a survey from three people currently working in Information technology field to evaluate the question and answer quality and added this as a human question quality prediction parameter. The result was converted into a binary digit based on the below formula where AE represents average result of an evaluator to a question and Ci denotes evaluator’s rating based on one criterion and n denotes number of criteria.

**Results:** ROC curves technique was implemented to measure the performance of answers model with full feature set, simplified feature set and only 10 features. It was found that all the 3 models are similar. In addition, 10-fold cross validation was performed to check the accuracy which can be verified by the given table.

**Summary:**The major takeaway is content quality in CQA sites can be evaluated based on features of the content and classification method like logistic regression can be utilized to build the model for prediction. The major features utilized were those of the content and the owner. Other features like comments and badges have not been utilized in this paper.

**Article 2**

**Complete Reference:** Ahmad, Rizwan, Dmitriy Kunitskiy, and Aleksander Maricq. "Stack Authority: Predicting Stack Overflow Post Helpfulness Using User Social Authoritativeness.”

**Objective:**

Question-answer website like Stack Overflow and Quora provide a platform where users can ask questions and community members answer them. However, since these questions are open to the community the answers generally suffer from noises. These noises can be in form of span, slangs, inappropriate use of language or simply unhelpful posts. The authors attempt to create a computer model which would classify a post to be helpful or not based on certain calculation. They also attempt to see the effect of adding social information to the post as a feature to check the effect on the model and the prediction it provides. The primary feature that the paper discussed was the ratio of up votes to the total vote a particular paper received on the post. After applying various regression functions, the authors could build two accurate predictors, a linear ridge regression model and a decision tree regression model. Both these out-performed the trivial base line predictor. However, their attempt to add the social factor failed and did not reveal any significant difference on what was predicted by earlier models.

**Approach:**

The authors could mine data from the huge Stack Overflow data corpus available publicly. After careful ETL, the data was refined and readied for analysis. They considered five most common tags: Java, JavaScript, C#, PHP and Android. A total of 26.5 million posts with 9.9 million question and 16.5 million answers were analyzed. To successfully add social context, a URL tracking mechanism was adopted to identify the users and their activity and assign points to the post based on the user’s posting of questions and the answers. A baseline predictive model was developed initially which was on line with the average user post and average “helpful” posts. Regressions features were analyzed from among the post features like length of post, comments, tags, code, hyperlinks and user features like badges, reputation, views etc. Then beyond this Linear ridge regression and Decision Tree Regression models were developed.

**Results:**

It was noted that classification accuracy, MSE and R-squared for the Linear ridge and Decision tree were 0.610, 0.217, and 0.069 and 0.675, 0.192 and 0.179 respectively. Overall, the classification was 80% accurate. Most significant feature was number of views for the post followed by the Stack Overflow reputation of the user. Others were owner profile views and badges. No significant effect was found in terms of social features.

**Summary:**

In summary, the paper highlighted various features of the post and the user and provide a kick start to our project in terms of a focus are. Given that the number of features can be vast, we now understand the factors on which a post can be answered. This can be expanded as per pour needs for the questions that we try to resolve in the project.

**Article 3**

**Complete Reference:** Correa, Denzil, and Ashish Sureka. "Fit or unfit: analysis and prediction of 'closed questions' on stack overflow." *Proceedings of the first ACM conference on Online social networks*. ACM, 2013.

**Objective:**The research paper aims at in-depth analysis of closed questions on Stack Overflow detailing their structure, content and trends, to utilize the analysis into predictive modelling for the question closure probability for the identified feature sets from the huge pile of questions present on the website.

The paper is the first ever known focused research on the ‘closed’ questions section on the Stack Overflow. With a proper analysis into the ever neglected yet most vast aspect of information distribution on the Stack Overflow website, this research aims at developing the predictive models to clearly mark the ‘to be closed’ questions and give the users a proper feedback about their content style to restrict the size of irrelevant content on this information management portfolio.

**2. Approach:** The authors analyzed the models based on 19 features sets available. Aside from the self-explanatory feature sets, other important features considered were Badge Score, Post Score, Accepted Answer Score and #Popular Tags. Based on the exhaustive features list, a series of regression methods was used to design the best predictive model. SGBT modeling was finally chosen as it represented the strongest output and hence it was used to form a strong classifier out of series of weak models each handling a separate entity model. A 70-30% ratio was taken for training and testing data. Based on the confusion matrix analysis, three metrics of F1 score, Accuracy and AUC was used to finally predict the effect of feature set obtained by forward selection method.

**3. Results:** Following were the major results obtained from the research paper:

1. Stack Overflows URLs, Code length, age of account and post score accounts for strongest contribution to prediction of post closures. On the contrary, features like comments score, badge score has a minimal effect on the prediction measure.
2. Feature set analysis represents that new users with a lack of knowledge for stack overflow question formatting process, have highest rate of closure probability.

|  |  |
| --- | --- |
|  |  |

**4. Summary:**The major takeaway from the paper was implementation of SGBT predictive modeling process using standard parameters and sub-sample sizes. It demonstrates the process of extracting the weak predictive models parameters without the actual requirement for creating the tree, thereby avoiding the overfitting of the model to build a strong classifier.

**Article 4**

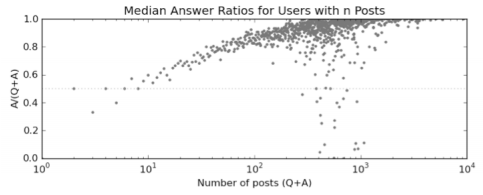
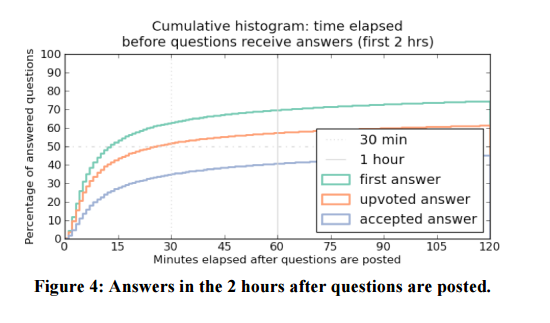
**Complete Reference:** Mamykina, Lena, et al. "Design lessons from the fastest q&a site in the west." *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2011.

**Objective:** The paper analyzes the user behavior of technical Q&A website Stack Overflow(SO) and tries to find the attributes for such high success rates. The researcher wanted to understand the features which contribute to attract users to SO and make it a favorable site to ask questions and get correct answers quickly. They aimed to emulate these features to other websites, however, limited their study to understanding the features of SO.

**Approach:**

The authors conducted the research in two phases. First, they conducted quantitative analysis from the data they gathered from the Stack Overflow website. They gathered different metrics like rate of questions being asked, the time it takes for first answer, time it took for first accepted answer, follow up questions (or comment on questions), activity on each question etc. They applied various statistics formula to calculate mean and median times to analyze the user behavior towards a particular event. The second approach was to follow up on the website designers and the users. This was the part of the qualitative approach. The authors selected and contacted the users based on their activity. The interviewed these users over skype, phone or in person for about 1 hour asking them about their interaction patterns on SO, motivation, comparison to other Q&A websites etc.

**Results:**

The qualitative study revealed that the founders the aim to create a productive environment where searching information is easier. These rewards came in terms of reputation, badges, privileges, and bragging. This was instrumental in understanding what community members needed. Rapidly improving the design of the front end as well as the backend also made Stack Overflow more preferred over other tech blogs and Q&A forums. Another astonishing finding was that the most activity on a question happened in the first hour of the question being posted (Fig. 1). A high success rate of SO was attributed to the finding that a question was answered 92.6% and more than 70% questions were answered multiple times. Other result was that the frequent user of SO were mostly the users who would answer and had high answer ratio.

**Fig 1: Answers in the 2 hours after questions are posted. Fig 2: Generally Frequent users post answers.**

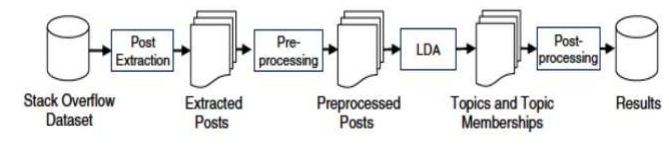
**Summary:** In summary, this paper highlighted various reasons which were responsible for the overall success of Stack Overflow. Moreover, this paper was instrumental in understanding various metrics which will assist us in doing our analysis. We understood the user pattern from the data analysis carried out by the authors. The qualitative analysis highlighted the motivation behind this website. It also highlighted the reasons behind various changes being carried out on the website and the relative implications these changes caused to improve the human interaction with various elements present on the website.

**Article 5**

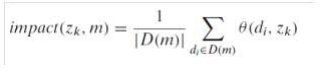
**Complete reference**: [Barua, Anton, Stephen W. Thomas, and Ahmed E. Hassan. "What are developers talking about? an analysis of topics and trends in stack overflow." Empirical Software Engineering 19.3 (2014): 619-654.](http://www.cs.ucsb.edu/~sivabalan/cs290n/CS290N_Project_Report.pdf)

**Objective:**The paper is addressing the cross-cutting areas of concerns for developers that span across multiple domains for different topics. Addressing this problem is important to find if one topic triggers answers in another i.e. if some topics are related to other in terms of questions and answers. This information can be helpful to business analysts to figure out the most used feature of a given software and product developers to assess relative popularity of their products. The objective of the paper is to find the trending topics and the rising interest in certain software or programming language.

**Approach:** The following diagram shows the various phases of data processing.

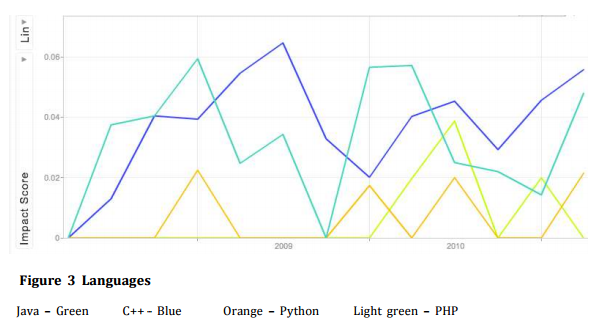
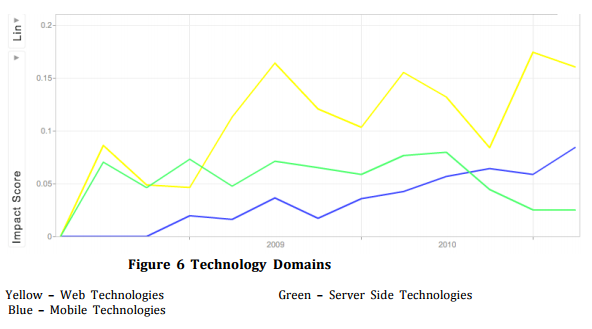


The Stack Overflow dataset is organized into five XML documents badges.xml, comments.xml, posts.xml, users.xml and votes.xml. Out of these the dataset used is posts (questions and answers with tags). The entire dataset was divided into 12 partitions covering 3 years with 3 months per partition. The impact of a topic was calculated using the below formula:



There are 4 different comparisons that were applied- Programming Languages, Web Technologies, Application development, General Trend. Latent Dirichlet allocation (LDA) was applied for classifying the above comparisons. LDA is one of the most common topic model used and is a generative model that allows sets of observations to be explained by unobserved groups that explain why some parts of data are similar.

**Results:** The graphs below show the trend of languages and technologies over a 3-year time frame and with 3-month gap between each period.

During this trend analysis there are several peaks as some topics were trending at some particular point of time. The reasons behind this sudden popularity in certain technologies were discovered and were found associated with real time events.

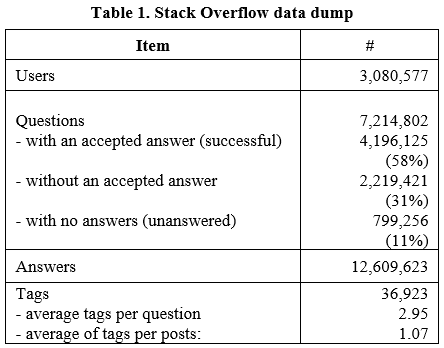
**Summary:**The above analysis is performed on Stack overflow, a popular Q&A website used by millions of active users. LDA was applied to discover topics from textual content and various metrics were applied to quantify the topics and changes over time. The analysis can help contemporary developers, stack overflow team to better understand the content generated by users and in moderation of website.

**Article 6**

**Complete reference:** Novielli, Nicole, Fabio Calefato, and Filippo Lanubile. "Towards discovering the role of emotions in stack overflow." Proceedings of the 6th International Workshop on Social Software Engineering. ACM, 2014.

**Objective:**  The research paper tries to analyze through an empirical study to investigate the role of emotions and lexicons used to design a question and their effectiveness on online Question & Answer portals. The paper further investigations are being carried out to analyze the natural language conversation which will help understand user interactions on such portals. Even a very important question might not attract any response because of ineffective usage of words.

The key objective of this paper tries to address the concerns related to understanding user’s perception in responding to a question on the Q&A portal apart from analyzing the time to get response based on the usage of lexicons. The research implies that better understanding of user sentiments might lead to future enhancements in enhancing user engagement, experience and online tools with embedded emotional intelligence systems.

**Approach:** To understand the relation between usage of emotion and user response to it, the team of researchers used various factors that are collected by the Stack Overflow whenever a user posts or responds (to) a question. Factors like date and time of post, code snippet, length of post, topic it related to; responder’s details like Q&A score, answers accepted by user, answers provided by user, etc. were used to design a regression model which further predicted a binary result called Success which records if an accepted answer for similar question was provided or not. Per the team of researchers, the key contribution of this research is to support their claim that emotion recognition is a growing and interesting field of software engineering and specifically social computing.

**Results:** When the research paper was published, the analyzing process for data was still on-going and hence no concrete evidence has been documented which can be considered the result. But, during the research, authors found that a strong correlation exists between the emotions expressed in the question like how-to, urgency, help, etc. and other factors such as the coding languages for the project, day and time of the week, asker’s social status in the Q&A community, etc.

**Summary:** This research paper provided me with basic understanding of how logistic regression can be applied over set of predictors to understand the success factor of the question to receive the best answer. This paper also helped in understanding what are some critical predictors in the Stack Overflow data that can help design a much efficient and effective model.

**Article 7**

**Literature:** Analysis of Titles from the Questions of the Stack Overflow Community Using Natural Language Processing (NLP) Techniques

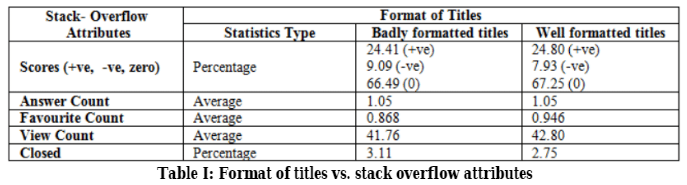
**Objective:**The research paper aims at evaluating the extent stack overflow posts’ title conforms to the guidelines of the Stack Overflow community. Further, it leverages this analysis in sentimental analysis process to check the effect of title format and conformity of rules for a post towards its overall popularity, predicted using the up and down votes for the post. The questions asked in the forum needs to follow a set of rules to help the programmers in their moments of crisis. Many questions which carry considerable potential for the most complex question tend to lose out in the huge pile of the content available on the site. The paper also aims at translating the importance of the tone of the post towards its popularity and meaningfulness. The research implied that format of a question is not a strong contributing factor to the popularity of the question. In other words, a well formatted post may or may not have high upvotes. On the other hand, a question with a positive tone is more likely to get upvotes than a subsequent question with a negative tone. The research showed that it’s the quality of the tone that matters to get higher upvotes for a post, rather than its structure which may or may not get upvotes.

**2. Approach: Summarize your understanding of how authors address this problem (use equations, flow charts if necessary)? If possible, identify the key contributions they bring.**

Response: The authors have broken the research into two parts of titular format checker and sentimental analysis. The extracted data is checked against the pre-defined grouped titles and if found similarity are grouped with the class they are matched with. Similarly, to gather a collection of positive, negative and neutral sentiments, the Bayes classifier is trained against a different data set of movie review which was used to gather positive, negative and neutral sentiment keywords. These classifications were then utilized with the testing data set of Stack Overflow posts to check for the sentimental tone of the posts. The following Naïve Bayes theorem was used in the analysis:

**P (label | features) = P(label) \* P (features | label) / P(features)**

**3. Results:** Some of the major findings from the research demonstrated that approx. 31% of the titles are ill formatted in the community. However, 24% of these titles gathered positive scores despite their ill format. Further, the research showed that there exists other unthought factors like ‘Answercount’, ‘Viewcount’ and ‘Favoritecount’ that have a huge impact on the popularity of posted questions.

The outcomes of the research can be summarized below:

1. Sentiment plays a major role in the popularity and hence directly accounts for the number of upvotes received.
2. High Structure + Positive Sentiment = Highest upvotes
3. Low Structure + Positive Sentiment = Fair upvotes.
4. High Structure + Negative Sentiment = Lowest upvotes
5. Low Structure + Negative Sentiment = Fair upvotes

**4. Summary:**Break and solve model to distinctively handle the sentimental analysis and general classification separately. Training of the Bayes classifier using a different dataset of sentimental data and re analysis of the testing data set based on the classifier gives optimum classification. Exhaustive criteria inclusion to build the model, as many a times most appealing criteria comes out to be a minimal contributor to the model as was the content structure in the given case.

**Article 8**

**Complete reference:** Novielli, Nicole, Fabio Calefato, and Filippo Lanubile. "The challenges of sentiment detection in the social programmer ecosystem." Proceedings of the 7th International Workshop on Social Software Engineering. ACM, 2015.

**1. Objective:** The aim of this paper is to understand the probability of an answer to be accepted with the usage of expressive affective states on Stack Overflow. The paper also aims to raise and prove that analyzing just the polarity of the questions or answers isn’t an efficient parameter for detecting programmer sentiments. Various researches are being carried out to understand what incites social programmer to seek help or help others with programming issues. To address this problem, the research team analysis top 100 questions along with the asker’s comments with highest positive and negative polarity and the best answer to the question. Various state-of-art tools are then used to understand polarity of each lexicon, regression models for analyzing textual occurrences, etc. The resultant dataset helps in supporting the claim made by the research team.

The authors found that analyzing words and their polarity isn’t sufficient to justify the positivity or negativity of the question but analysis of whole text structure or huge group of lexicon together makes sense for efficient sentiment analysis. A word as an individual might reflect the negative sentiments but when used in a statement, it might reflect complete opposite of what was predicted from initial research.

**2. Approach:** The team of researchers first annotated sentiment score in context of the polarity of the statement and its strength. Having done so, they measured the sentiment load in varied threads i.e. questions, answers and comments, to choose cases with highest sentiment score. The team then analyzed the polarity and sentiment strength of the text as an overall lexicon by using the sentiment analysis tool named SentiStrength. To understand the concept properly, they then analyzed top 100 questions from Stack Overflow dump data along with the following threads on the questions like the comments by the asker with highest positive and negative sentiment score. They then use these findings from positive and negative analysis to prove that just analyzing the polarity cannot be an efficient tool to gauge the sentiments of the asker or answerer. Similar thing has been shown in the adjacent figure.

**3. Results:** The research claims that emotions represent variety of moods and opinions and their resultant dataset confirms this claim that such analysis is more complex a phenomenon than just analyzing the polarity of the lexicon. From the research done by the team, it was clear that the sentiment load cannot be predicted accurately even with the usage of state-of-art sentiment analysis tools. This is because, such tools just consider the polarity of the lexicons to predict the user sentiment. The research tries to set a foundation stone in favor to adopt more efficient and affective state models.

**4. Summary:** In context of findings based on the affective analysis of text, the researcher shows that the negativity is reflected from the questions posted by the user as it reflects the frustration about the technology or the problem they are not able to solve. Similarly, the negativity in the answers are associated with the empathy of the responder towards the problems faced by the asker and not directly to the asker. This lays a strong foundation and case to justify the research done by this team to reflect that affective analysis of various parameters and lexicons is essential to understand user sentiments rather than just the polarity test.

# 4. Process, Implementation, and Result

This section deals with the issues that were noticed and believed can be resolved using the predictive analysis techniques. Each issue is introduced, its approach along with the data gathering strategy is elaborated and the methods used for the analysis are discussed. Finally, each issue ends with the results of the study that was carried out and various models used are compared. The flowchart below describes briefly about the whole transition of the project from extracting the data to various models implemented to answer issues on Stack Overflow.

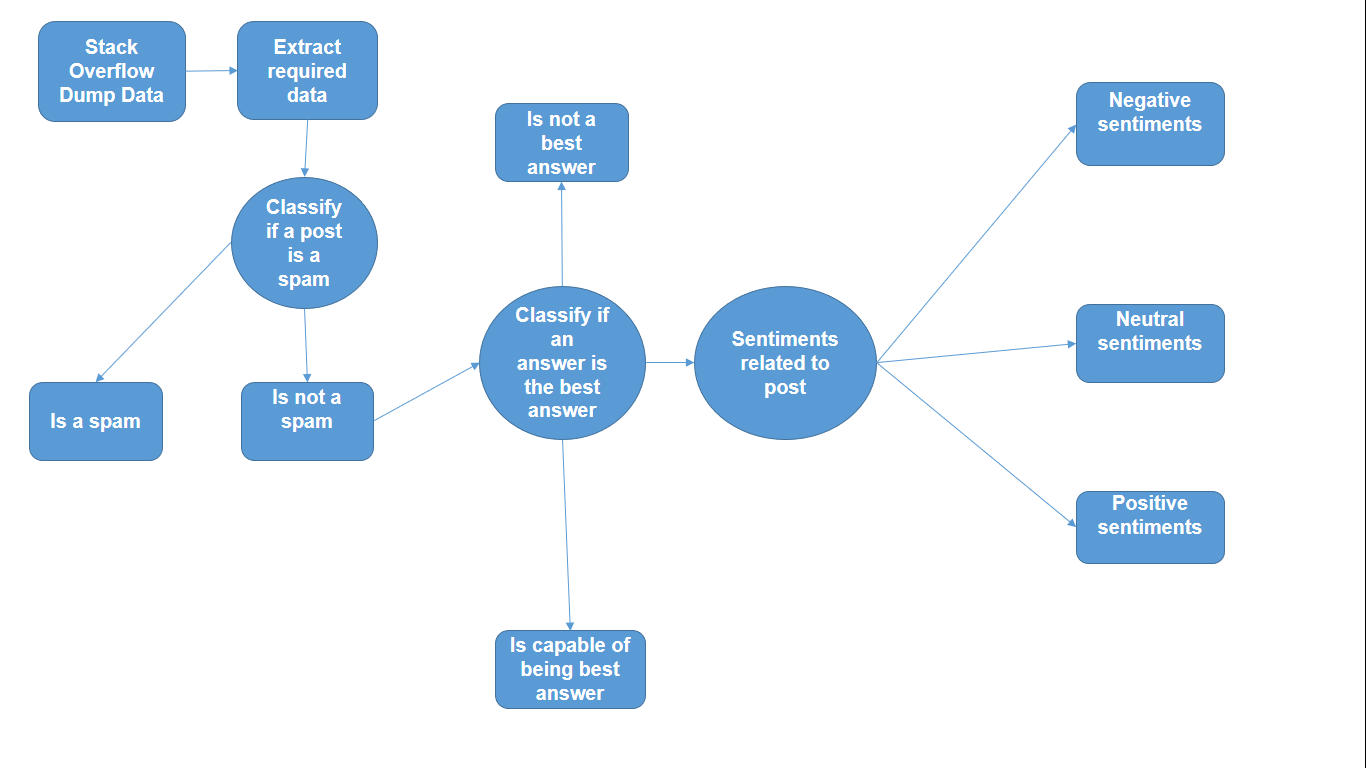


Figure: Flow diagram giving a broad overview of the entire process followed

## 4.1 Classify data into spam/non-spam

### 4.1.1 Approach

The approach for the spammer classification can be detailed by the following points:

1. Data Extraction from the stack overflow in accordance to the most important tables that contains the spam alerts for the records. SQL queries were used to extract the required data from the Stack Overflow workbench.
2. The data was then analyzed using classification models that includes logistic regression, LDA, QDA, KNN, CART, Random forest and SVM.
3. Regularization and K-fold Cross Validation method were implemented to avoid over fitting of the model and to check the effectiveness of the obtained model over the different subsets of data.
4. A new technique called C5.0 was implemented to train the model on the previously classified sample dataset utilizing the concept of information entropy, recurring the partitioned data trees on the highest gained attribute and adding next best nodes as the children nodes.
5. Mutual analysis for the misclassification rate between the models gives the model with the best prediction rate and hence the spam modeler for the Stack Overflow data.

### 4.1.2 Implementation

The implementation of the spam classifier consists mainly of four steps:

a.       **Data Extraction:** Stack Overflow provides workbench to extract their processed activity data and related parameters from their backend. A major analysis into the website architecture clearly reveals the huge effort put into the content management portfolio owing to the large number of users working on the website with day in and day out. This calls for a need to classify the data into useful and spammer contents. The backend provides the option for analysis into the spammer content and hence, based on few factors the incoming data traffic can be easily classified to avoid the unintended content. Major tables identified for the related data to the spam content were:

|  |  |
| --- | --- |
| * ·         ReviewTasks * ·         ReviewTaskStates * ·         ReviewTaskResults * ·         ReviewRejectionReasons * ·         ReviewTaskResultTypes * ·         Posts * ·         Users | * ·         PostTypes * ·         PostFeedback * ·         VoteTypes * ·         Votes * ·         PostHistory * ·         PostHistoryTypes |

The complex queries created for the data extraction can be referenced in the Appendix section. Because Stack Overflow provides the access to 50,000 records per cycle, the required data were extracted into

b.   **Data Cleaning:** The extracted data was cleaned for the description columns and irrelevant columns for the spam identification to obtain a data frame consisting of only of factors that are potential contributors to the spam identification process.

c.       **Data Analysis & Modeling:** From the final list of 16 sample features, series of classification models were implemented to check the best model depicting the spammer classifier. Following major factors were identified to be the major contributors in spam detection.

|  |  |
| --- | --- |
| * PostTypeId * PostScore * post\_length * owner\_reputation * owner\_profile\_summary * owner\_upvotes | * editDurationAfterCreation * q\_num\_tags * AnswerCount * CommentCount * post\_views |

Algorithms implemented for checking the best model for the classifier were:

c.1 Logistic Regression:

c.2 LDA:

c.3 QDA:

c.4 KNN:

c.5 CART:

c.6 Random Forest:

c.7 SVM:

Please refer to the Appendix to check for the exact list of steps for the execution.

d.       Evaluation:

The models generated were evaluated based on accuracy and misclassification rate using confusion matrix. Correspondingly, ROC area under the curve were also utilized to further validate the strength of classification. Following table shows the results obtained:

|  |  |  |
| --- | --- | --- |
| **Model** | **Misclassification rate** | **Accuracy Rate (1-Misclassification Rate)** |
| Logistic Regression | 0.3077023 | 0.70 |
| LDA | 0.3075291 | 0.70 |
| QDA | 0.3506516 | 0.65 |
| KNN | **0.01506082** | 0.985 |
| CART | 0.2757068 | 0.73 |
| Random Forest | 0.01524007 | 0.985 |
| SVM | 0.28654 | 0.32 |
| C5.0 | 0.02281 | 0.972 |

*PS: Complete output from all the methods can be found in* [*GitHub*](https://github.com/aloksatpathy/EDAStackOverflow)*.*

### 4.1.3 Result

Hence, as is clear from the above table, Random Forest and KNN modelling gives the least misclassification rate of 0.015. However, KNN has a very slight advantage on the Random Forest and hence comes out to be most economic model of the lot.  This also falls in very accordance to the most industry standards spam classifiers which uses KNN as the best predictor, also illustrated through the literature reviews. The industry should use the KNN model for this classification using the above mentioned critical parameters that came out to be strongest contributor to the variable IsSpam. On top of providing better accuracy rate for classification rate, KNN model also provides easy interpretation of the model and hence easily manages the bias-variance trade-off thereby providing the optimum model to be used.

## 4.2 Evaluate quality of answer posts to predict best possible answer

On stack overflow, a user (also called Original Poster or OP) asks questions and other users or the OP, at times, answer the question. The quality of the answer technically depends on two things a) the number of up/ down votes by users on the answer and b) the acceptance of the answer by OP. Generally, OP verifies whether the answer is the solution to the question asked. There can be only one accepted answer to the question. Ever since the inception of Stack Overflow, there have been 12,319,722 questions posted and they have been answered 19,494,601 times. This data does not include deleted or closed questions. Of 12.3 million questions, only 6,717,441 (i.e. 54.5%) questions have an accepted answer.

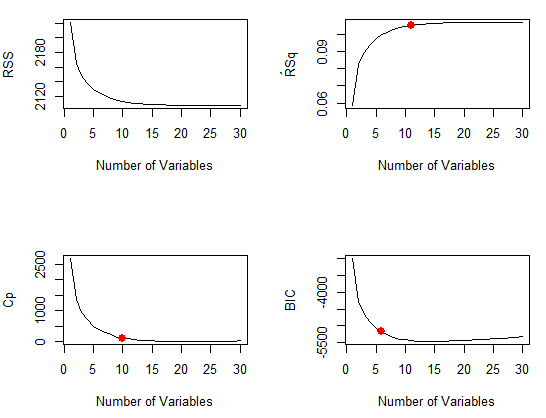
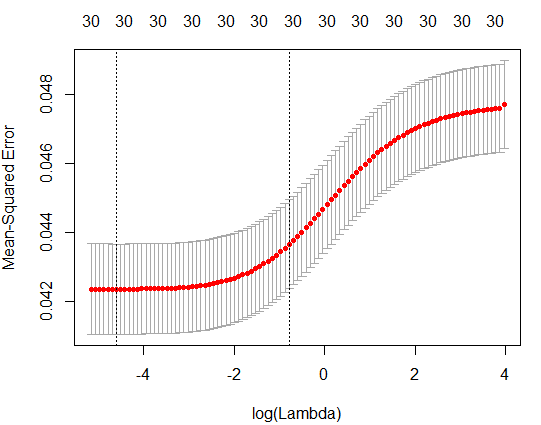
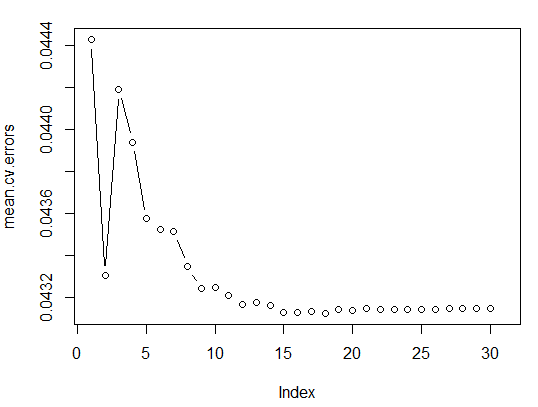
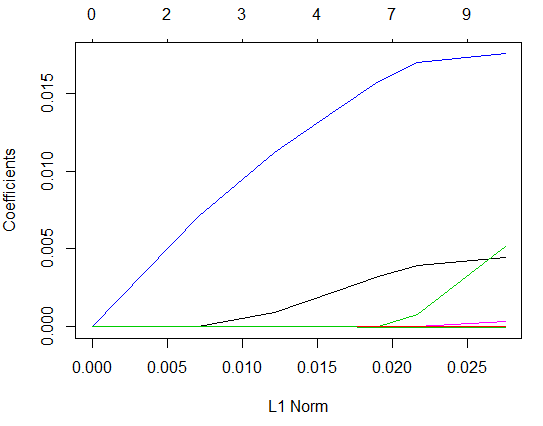
However, as questions are generally answered within 20 minutes, accepted answer is generally the first “working” solution and may or may not be the best solution to the question. It has been noticed thatappx. 30%of questions having an accepted answer, however have another answer/s scoring better than the accepted one. Our attempt to evaluate the quality of the answer and predict the best possible answer can help in 2 major instances:

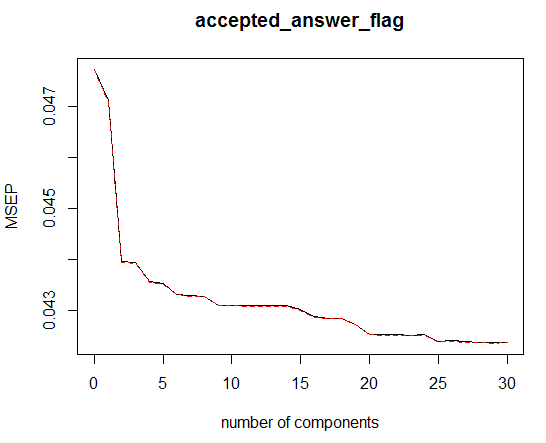
1. Saves time of users who browse for answers to a question without an accepted answer.
2. User can find the best answer to a question which has an accepted answer but may have better answers.

### 4.2.1 Approach

The data was extracted from Posts, Users and Votes table by performing transformations using SQL query as given in the Appendix. A total of 31 features were derived from these tables to prepare a model that can be used to predict the likelihood of it being marked as the best answer. A total of 50,000 answers were extracted dated from January 1, 2016 to December 1, 2016. To increase the quality of the data only questions with more than 4 answers were selected.

Ridge regression, Lasso and Dimensionality Reduction (PCA) methods were used to find the parameters most contributing towards generating classification model for prediction. The classification models include Logistic Regression, LDA, QDA, KNN, CART, Random Forest and SVM.



### 4.2.2 Implementation

A total of 31 features were considered for the analysis. Here we want to see other than the Op’s bias what other factors could lead to an answer getting accepted. After doing this analysis we will be able to predict which answer is highly likely to get accepted if none of the answers to a question have been accepted yet. We apply a series of techniques and compare the results to finally arrive at the model which gives the best result.

Data Extraction: The data was extracted using [SQL Query](https://github.com/aloksatpathy/EDAStackOverflow) run against Stack overflow data dumped in [Stack Exchange](https://data.stackexchange.com/stackoverflow/query/new) repository. Given below is a sample column mapping used to extract the data:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Required Column** | **Actual Table** | **Actual Column** | **Transformation/Filter** | **Comments** |
| a\_score | Post | score | integer given | how many points did the answer gain |
| a\_body\_length | Post | body | length | length of the answer’s content |
| a\_body\_has\_code | Post | body | contains <code> tag | does the answer contain code or not |
| a\_has\_edited | Post | LastEditDate, CreationDate | flag if LastEditDate (non-blank) and CreationDate are different | has the answer been edited or not |
| a\_num\_comment | Post | CommentCount | integer given | number of comments the question receive |
| a\_owner\_reputation | User | Reputation | by joining OwnerUserId with Id User table | reputation points of the answer’s owner |
| a\_owner\_profile\_ summary | User | three columns (Location+WebsiteUrl+AboutMe) | sum of non-blank flags of columns | combination value of “Location”, “WebsiteUrl”, and “AboutMe” from answer’s owner |

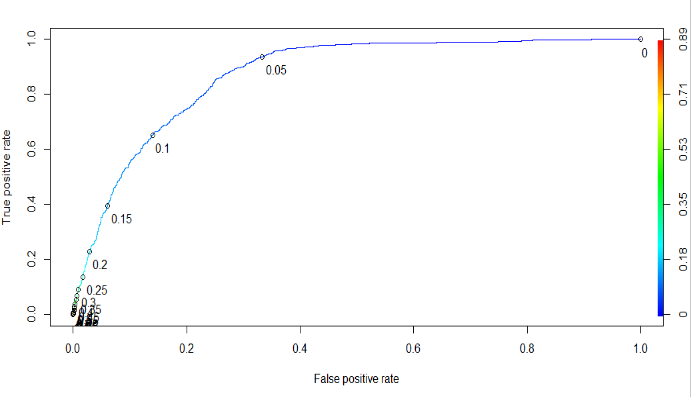
Figure: Sample parameters and the transformation / filter applied to extract from Stack overflow tables

Classification Method:

With 30 predictors to predict “Accepted Answer” probability, Logistic regression was found to be the best model for analysis. On successive elimination of insignificant data, the following model was selected for analysis:

glm (accepted\_answer\_flag ~ a\_body\_length + a\_body\_has\_code + a\_DaysOld + a\_has\_edited + a\_num\_comment + a\_owner\_reputation + a\_owner\_profile\_summary + a\_owner\_downvotes + q\_DaysOld + q\_title\_length + q\_num\_answers + q\_num\_comment + q\_owner\_profile\_summary, data=Q2Train, family="binomial")

Of these 13 predictors, 12 were highly significant (\*\*\*). The definition of the predictors is provided in Appendix A. The code to arrive at the most significant predictor is provided in Appendix C. Cross validation was performed using Leave-one-out cross validation technique and K-fold cross validation. The average mean squared error obtained was 4.16%

The confusion matrix thus obtained with the model is: False Negative rate: 592

A model with lesser False negative rate was instigated further. That means it should predict most acceptable answer. The mis-classification rate obtained from this model is: 0.05296

The ROCR curve thus obtained for this model is shown in the below figure.

### 4.2.3 Result

Out of all the classification techniques we find Logistic regression to be the best classification technique based on its low misclassification rate and ROC curve showing higher area under curve and hence better classification model in predicting the likelihood marking an answer as the best. The following table summarizes the findings obtained after carrying out the analysis.

|  |  |  |
| --- | --- | --- |
| Parameter | Misclassification Rate | ROC Curve |
| Logistic Regression | 0.052 | https://lh6.googleusercontent.com/h_5Awpv17Mv8VvDbdS0Ug-pd-4AR9eNpae6ejmJMVrEDbZJGIN2R94L8jzUOo7cRMN8wLkGLKNjJEfsHoI0AEwjhdk4bDlrWJgEswY5tQaPwibKSfRHSt3egQqqXDe5Wr3SsbHRY |
| LDA | 0.063 | https://lh6.googleusercontent.com/DvKtEvkQ7VROkCbSmcGxXav0ghF-HxOO82Syzk4xeugkestzK9ZxHKw0_q6ZfA05dubZzx5aRMiG-vRwbkF_gUTg7fLX6YihRxQnem6gg6zC5jsasCRGdDmaTehuWEgasJD6Fdw0 |
| QDA | 0.083 | https://lh5.googleusercontent.com/lCa-V6USpyJhqoyY7CIPS51Vb73q0RiQJi-wH9e0SkXnbcZQpU0cwj3jTjCvKKfCwhQi146ZY5gFhkG6qQ2CwKcztayWoohWyleuq-ja8mdqXoN7TUMmN0EFQfGEpd2VzrRzNFrK |
| KNN | 0.052 | https://lh4.googleusercontent.com/5iVIZ794-sm-kZ6JGP_VdLnz44xyfk3dRSfuWjqEGM3pg0A7ebaxZEs7MKoan3688eRVQL7vxtY1HLavENJEmr-e7eJlp7NRcdtBKP-ec6jYzgk33VZwRTXlgqJeCC8QrEn2nXL6 |
| CART | 0.052 | https://lh6.googleusercontent.com/lFrjKx08ZJryDjdo4YI-93oBK3Q2BMzIj98hsujJ758NAxu5g2S-_S4lSBxUrW4Lx9QbQ-ZPZuFBlab-hSjYDbD6AS05AVB345foJOe2YhtIUvAGLgsNtIn5zuZrgFyolhhI2Jm7 |
| Random Forest | 0.88 (no significance) |

*\*Complete output from all the methods can be found in* [*GitHub*](https://github.com/aloksatpathy/EDAStackOverflow)*.*

## 4.3 Predict comment sentiment using NLP and based on Post features

### 4.3.1 Approach

The data was taken from Posts, Comments and Votes table. The questions which were posted on or after January 1, 2016 till December 1, 2016, have 3 or more answers, and have C++, Python, Java or JavaScript as tags have been considered here. Like the previous issue, this gave us a focused and recent dataset to minimize excessive variability. A total of 50,000 questions and answers were considered for this analysis.

Stack overflow data filtered based on date, tags, number of comments and number of answers on the questions. Deleted questions are not included.

The above figure explains the filters that we applied on the corpus to arrive at the target dataset we used for this analysis. These filters were chosen based on the density of the activity on the posts. While almost 50% of the posts had no comments on them, the number of comments decreased drastically with highest number of comments on a post being 78 and there was only 1 such post. In all 170,188 posts were considered for this study. Users posted 545,413 comments on such posts and all were taken into consideration for the analysis.

### 4.3.2 Implementation

A total 32 fields were queried and 50,000 records were extracted that consisted comments on posts with answers count > 4 and at least 1 comment. The text field was run through Natural language processing algorithm to analyse positive, neutral or negative sentiment and find correlation between other features of the comment, the post it belongs to, the features of the owner of the post and the comment and its effect on number of upvotes and downvotes of a post. A [list of words](https://github.com/fnielsen/afinn/blob/master/afinn/data/AFINN-111.txt) was downloaded with sentiment scores applied to each of them. This was used to train and predict and analyse the sentiment of each comment.

model=glm(SentimentScore~CommentLength+comment\_owner\_downvotes+num\_tags+PostAnswerCount+IsAcceptedAnswer+PostCommentCount,data=Q3Train)

### 4.3.3 Result

|  |  |
| --- | --- |
| Sentiment | Number of comments |
| Positive | 20842 |
| Neutral | 21688 |
| Negative | 7470 |

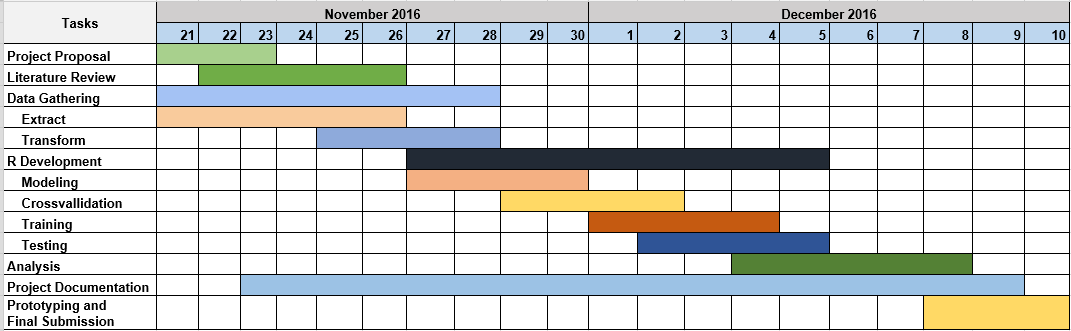
As we can see from the above model, the Sentiment of the comments doesn’t depend on the Post upvotes and Post downvotes. This goes ahead and proves that there is no correlation between positive/negative comments and upvotes/downvotes of posts. This is evident because whenever a person upvotes a comment the owner gets +5 points, however on downvoting the owner of the post gets -2 and the person downvoting gets -1 reputation points. Hence, people usually tend to leave their agreement/disagreement in the form of comments instead of upvoting/downvoting. Also, the accuracy of the model after considering all possible features related to the comment’s sentiment came to only 0.44, hence the comment sentiments can’t be predicted based on Posts features and cannot be used to predict upvotes/downvotes of posts.

# 5. Conclusion

The major take away from the above analyses can be summarized as below:

* + 1. With huge traffic and regular updates, content management for Stack Overflow is an essential requirement where a best fit Spam Classifier would be dramatically helpful in maintaining the useful content on the website. Of all the major classifiers, KNN model gives the best fit for the spam classifier. This is in accordance to the real world actual implementation technologies used in Google and Yahoo mail spammer.
    2. Shrinkage methods of Ridge Regression and LASSO attempts at reducing the critical factors in the model. However, the latter is a better shrinkage method as it attempts to nullify the coefficients for the factors in the ultimate classifier model.
    3. Similarly, logistic regression gives the best modeling for the answer quality evaluation. Hence, with all the classifier models in place, different classifiers may give the optimum model based on the data set and context of the model application.
    4. Bayes Classifier provides the optimum classification model for the sentimental analysis with model cross trained on multiple sample sets of data through reanalysis and cross training.
    5. C5.0 algorithm is the latest development for classification analysis that depicts a large decision tree as a fine-tuned set of objective rules, cross countering the overfitting effect and any associated pruning error.

# 6. Timelines for Analysis:



# 7. References

1. Evaluation and Prediction of Content Quality in Stack Overflow with Logistic Regression, Daoying Qiu, December 16, 2015, Finland
2. Stack Authority: Predicting Stack Overflow Post Helpfulness Using User Social Authoritativeness, Ahmed, Kunistkiy and Maricq, November 2015, CA, USA
3. Fit or Unfit : Analysis and Prediction of ‘Closed Questions’ on Stack Overflow, Correa and Sureka, July 2013, Delhi, India
4. Design Lessons from the Fastest Q&A Site in the West, Mamykina, Manoim, Mittal, Hripcsak and Hartmann, ACM, 2011
5. What developers are talking about? An Analysis of Stack Overflow Data,
6. Towards discovering the role of emotions in stack overflow, Nicole Novielli, Fabio Calefato, Filippo Lanubile, Bari, Italy, May 2014
7. Analysis of Titles from the Questions of the Stack Overflow Community Using Natural Language Processing (NLP) Techniques, Tapan Kumar Hazra, Aryak Sengupta, Anirban Ghosh, August 2015
8. The Challenges of Sentiment Detection in the Social Programmer Ecosystem, Nicole Novielli, Fabio Calefato, Filippo Lanubile, Bari, Italy, May 2014
9. "Understanding the Output of C5.0 Classification Model Using the CARET Package." R - Understanding the Output of C5.0 Classification Model Using the CARET Package - Cross Validated. N.p., n.d. Web. 10 Dec. 2016.
10. "Fitting a Neural Network in R; Neuralnet Package." R-bloggers. N.p., 2015. Web. 10 Dec. 2016.
11. Stack Exchange Website: https://stackexchange.com/about

# 8. Appendix

## Appendix A: Variables

### A1. Variables for Question 1 - Spam Classifier

|  |  |
| --- | --- |
| **Predictor** | **Description** |
| PostTypeId | question or answer |
| PostScore | net of upvote or downvote |
| post\_length | number of characters including spaces in a post |
| owner\_reputation | reputation points of the posts’s owner |
| owner\_profile\_summary | Owner’s profile summary |
| owner\_views | Number of views on owner’s profile |
| owner\_upvotes | Number of upvotes owner has received |
| owner\_downvotes | Number of down votes owner has received |
| editDurationAfterCreation | Time taken between creation and editing of the post |
| q\_num\_tags | Number of tags on the post |
| AnswerCount | Number of answers on the question type post |
| CommentCount | Number of comments on the posts |
| has\_code | Binary, post has code? |
| post\_views | Number of views the post has received |
| UserId | Userid who posted the post(OP) |

### A2. Variables for Question 2 – Evaluate best answer

|  |  |
| --- | --- |
| **Predictor** | **Description** |
| a\_score | how many points did the answer gain |
| a\_body\_length | length of the answer’s content |
| a\_body\_has\_code | does the answer contain code or not |
| a\_has\_edited | has the answer been edited or not |
| a\_num\_comment | number of comments the answer receive |
| a\_owner\_reputation | reputation points of the answer’s owner |
| a\_owner\_profile\_summary | combination value of “Location”, “WebsiteUrl”, and “AboutMe” from answer’s owner |
| a\_owner\_views | number of homepage views of the answer’s owner |
| a\_owner\_upvotes | number of upvotes the answer’s owner |
| a\_owner\_downvotes | number of downvotes the answer’s owner |
| q\_score | how many points did the answer gain |
| q\_num\_views | how many people have read the question |
| q\_body\_length | length of the question’s body |
| q\_has\_edited | has the question been edited or not |
| q\_title\_length | length of question’s title |
| q\_num\_tags | how many tags did the question receive |
| q\_num\_answers | number of answers received for a given question |
| q\_num\_comment | number of comments for a question |
| q\_num\_favorite | how many users have marked the question as favorite |
| q\_owner\_reputation | reputation points of the question’s owner |
| q\_owner\_profile\_summary | combination value of “Location”, “WebsiteUrl”, and “AboutMe” from question’s owner |
| q\_owner\_views | number of homepage views of the question’s owner |
| q\_owner\_upvotes | number of upvotes the question’s owner |
| q\_owner\_downvotes | number of downvotes the question’s owner |
| AcceptedAnswerId | only present if posttypeid=1 i.e. question type |
| a\_DaysOld | Answers |
| q\_DaysOld | Questions |
| a\_num\_views | how many people have read the answer |
| a\_PostUpVote | how many points did the answer gain |
| q\_PostDownVote | length of the answer’s content |

### A3. Variables for Question 3 – Sentiment Analysis using NLP on Post comments

|  |  |
| --- | --- |
| **Predictors** | **Description** |
| CommentLength | Number of characters in the comments |
| comment\_owner\_downvotes | Number of downvotes comment user has received in the whole life |
| num\_tags | Number of tags in the question post |
| PostAnswerCount | Number of aswers on the question post |
| IsAcceptedAnswer | Binary, is any answer marked as Accepted by OP |
| PostCommentCount | Nummber of comments on the post(question/ answer) |

## Appendix B: SQL Queries

### B1. SQL Query for Question 1 - Spam Classifier

select \* from

(

select

distinct

P.Id PostId,

case

when V.VoteTypeId=2 then 'Y'

else 'N'

end as UpVoted,

case

when V.VoteTypeId=3 then 'Y'

else 'N'

end as DownVoted,

P.PostTypeId PostTypeId,

P.Score PostScore,

len(P.Body) post\_length,

U.Reputation owner\_reputation,

CASE WHEN LTRIM(RTRIM(U.WebsiteURL)) = '' OR U.WebsiteURL IS NULL THEN 0 ELSE 1 END +

CASE WHEN LTRIM(RTRIM(U.Location)) = '' OR U.Location IS NULL THEN 0 ELSE 1 END +

CASE WHEN LTRIM(RTRIM(U.AboutMe)) = '' OR U.AboutMe IS NULL THEN 0 ELSE 1 END

as owner\_profile\_summary,

U.Views owner\_views,

U.Upvotes owner\_upvotes,

U.Downvotes owner\_downvotes,

U.LastAccessDate owner\_lastactivity\_days,

DATEDIFF(mi, P.CreationDate, P.LastEditDate) as editDurationAfterCreation,

DATEDIFF(mi, P.CreationDate, P.LastActivityDate) as activityDurationAfterCrea,

P.Title,

LEN(P.Tags)-LEN(REPLACE(P.Tags, '<', '')) as q\_num\_tags,

P.AnswerCount,

P.CommentCount,

CASE WHEN CHARINDEX('<code>', P.Body)>0

THEN 1

ELSE 0 END as has\_code,

P.ViewCount post\_views,

U.Id UserId,

R.Id ReviewTaskId, R.ReviewTaskTypeId ReviewTaskTypeId,

RR.RejectionReasonId RejectionReasonId, RRT.Name ReviewTaskResultTypes,

RRT.Description, RTS.Name ReviewTaskState, RTS.Description ReviewTaskStateDescription,

PT.Name PostTypeName, PHT.Name PostHistoryTypeName, VT.Name VoteTypeName,

RRR.Id ReviewRejectionFlag, RRR.Name ReviewRejectionReason, RRR.Description ReviewRejectionReasonDescription

from ReviewTasks R, ReviewTaskStates RTS, ReviewTaskResults RR,

ReviewRejectionReasons RRR, ReviewTaskResultTypes RRT,

Posts P, Users U, PostTypes PT, PostFeedback PF, VoteTypes VT, Votes V, PostHistory PH, PostHistoryTypes PHT

where R.Id=RR.ReviewTaskId

and R.ReviewTaskStateId = RTS.Id

and RR.RejectionReasonId = RRR.Id

and RR.ReviewTaskResultTypeId = RRT.Id

and R.PostId=P.Id

and P.OwnerUserId=U.Id

and V.PostId=P.Id

and P.PostTypeId=PT.Id

and PF.PostId=P.Id

and PF.VoteTypeId=VT.Id

and PH.PostHistoryTypeId= PHT.Id

and P.Id=PH.PostId

and upper(RRR.Name) LIKE '%SPAM%'

and RTS.Name='Completed'

and RRT.Name in ('Reject','Delete','Recommend Deletion')

and RR.RejectionReasonId is not null

UNION

select SELECTEDNONREJECTED.\*,RRR.Id ReviewRejectionFlag, RRR.Name ReviewRejectionReason, RRR.Description ReviewRejectionReasonDescription from

(select top 45000 \* from

(select

distinct

P.Id PostId,

case

when V.VoteTypeId=12 then 1

else 0

end as IsSpam,

case

when V.VoteTypeId=2 then 'Y'

else 'N'

end as UpVoted,

case

when V.VoteTypeId=3 then 'Y'

else 'N'

end as DownVoted,

P.PostTypeId PostTypeId,

P.Score PostScore,

len(P.Body) post\_length,

U.Reputation owner\_reputation,

CASE WHEN LTRIM(RTRIM(U.WebsiteURL)) = '' OR U.WebsiteURL IS NULL THEN 0 ELSE 1 END +

CASE WHEN LTRIM(RTRIM(U.Location)) = '' OR U.Location IS NULL THEN 0 ELSE 1 END +

CASE WHEN LTRIM(RTRIM(U.AboutMe)) = '' OR U.AboutMe IS NULL THEN 0 ELSE 1 END

as owner\_profile\_summary,

U.Views owner\_views,

U.Upvotes owner\_upvotes,

U.Downvotes owner\_downvotes,

U.LastAccessDate owner\_lastactivity\_days,

DATEDIFF(mi, P.CreationDate, P.LastEditDate) as editDurationAfterCreation,

DATEDIFF(mi, P.CreationDate, P.LastActivityDate) as activityDurationAfterCrea,

P.Title,

LEN(P.Tags)-LEN(REPLACE(P.Tags, '<', '')) as q\_num\_tags,

P.AnswerCount,

P.CommentCount,

CASE WHEN CHARINDEX('<code>', P.Body)>0

THEN 1

ELSE 0 END as has\_code,

P.ViewCount post\_views,

U.Id UserId,

R.Id ReviewTaskId, R.ReviewTaskTypeId ReviewTaskTypeId,

RR.RejectionReasonId RejectionReasonId, RRT.Name ReviewTaskResultTypes,

RRT.Description, RTS.Name ReviewTaskState, RTS.Description ReviewTaskStateDescription,

PT.Name PostTypeName, PHT.Name PostHistoryTypeName, VT.Name VoteTypeName

from ReviewTasks R, ReviewTaskStates RTS, ReviewTaskResults RR, ReviewTaskResultTypes RRT,

Posts P, Users U, PostTypes PT, PostFeedback PF, Votes V, VoteTypes VT, PostHistory PH, PostHistoryTypes PHT

where R.Id=RR.ReviewTaskId

and R.ReviewTaskStateId = RTS.Id

--and RR.RejectionReasonId = RRR.Id

and RR.ReviewTaskResultTypeId = RRT.Id

and R.PostId=P.Id

and P.OwnerUserId=U.Id

and V.PostId=P.Id

and P.PostTypeId=PT.Id

and PF.PostId=P.Id

and PF.VoteTypeId=VT.Id

and PH.PostHistoryTypeId= PHT.Id

and P.Id=PH.PostId

and RTS.Name='Completed'

and RRT.Name not in ('Reject','Delete','Recommend Deletion')

and RR.RejectionReasonId is null

) NOTREJECTED

) SELECTEDNONREJECTED

LEFT OUTER JOIN

ReviewRejectionReasons RRR

ON SELECTEDNONREJECTED.RejectionReasonId = RRR.Id

)

DATA

order by PostId

--OFFSET 50000 ROWS

### B2. SQL Query for Question 2 – Evaluate best answer

CREATE TABLE #votesUp(PostId int, VoteTypeIdCnt bigint)

INSERT INTO #votesUp SELECT PostId, Count(\*) as VoteTypeIdCnt FROM Votes WHERE VoteTypeId=2 GROUP BY PostId

CREATE TABLE #votesDown(PostId int, VoteTypeIdCnt bigint)

INSERT INTO #votesDown SELECT PostId, Count(\*) as VoteTypeIdCnt FROM Votes WHERE VoteTypeId=3 GROUP BY PostId

CREATE TABLE #votesSpam(PostId int, VoteTypeIdCnt bigint)

INSERT INTO #votesSpam SELECT PostId, Count(\*) as VoteTypeIdCnt FROM Votes WHERE VoteTypeId=12 GROUP BY PostId

select top 500

P.PostTypeId as postTypeId,

CASE WHEN P.AcceptedAnswerId IS NOT NULL

THEN 1

ELSE 0 END as IsAcceptedAnswer,

P.Score as postScore,

LEN(P.Body) as post\_length,

P.CreationDate,

P.LastEditDate,

P.LastActivityDate,

DATEDIFF(mi, P.CreationDate, P.LastEditDate) as editDurationAfterCreation,

DATEDIFF(mi, P.CreationDate, P.LastActivityDate) as activityDurationAfterCreation,

LEN(P.Title) as title\_length,

LEN(P.Tags)-LEN(REPLACE(P.Tags, '<', '')) as num\_tags,

P.CommentCount as num\_comment,

P.FavoriteCount as num\_favorite,

CASE WHEN CHARINDEX('<code>', P.Body)>0

THEN 1

ELSE 0 END as has\_code,

P.ViewCount as post\_views,

U.Reputation as owner\_reputation,

U.WebsiteURL as owner\_WebsiteURL,

U.Location as owner\_Location,

U.AboutMe as owner\_AboutMe,

CASE WHEN LTRIM(RTRIM(U.WebsiteURL)) = '' OR U.WebsiteURL IS NULL THEN 0 ELSE 1 END +

CASE WHEN LTRIM(RTRIM(U.Location)) = '' OR U.Location IS NULL THEN 0 ELSE 1 END +

CASE WHEN LTRIM(RTRIM(U.AboutMe)) = '' OR U.AboutMe IS NULL THEN 0 ELSE 1 END as owner\_profile\_summary,

U.Views as owner\_views,

U.UpVotes as owner\_upvotes,

U.DownVotes as owner\_downvotes,

DATEDIFF(d, U.LastAccessDate, GetDate()) as owner\_lastactivity\_days,

VU.VoteTypeIdCnt as upVotes,

VD.VoteTypeIdCnt as downVotes,

VS.VoteTypeIdCnt as isSpam

from PostsWithDeleted as P inner join

Users as U on P.OwnerUserId=U.Id left outer join

#votesUp as VU on VU.PostId=P.Id left outer join

#votesDown as VD on VD.PostId=P.Id left outer join

#votesSpam as VS on VS.PostId=P.Id

where (P.PostTypeId=1 OR P.PostTypeId=2)

and VS.VoteTypeIdCnt>0

### B3. SQL Query for Question 3 – Sentiment Analysis using NLP on Post comments

CREATE TABLE #votesUp(PostId int, VoteTypeIdCnt bigint)

INSERT INTO #votesUp SELECT PostId, Count(\*) as VoteTypeIdCnt FROM Votes WHERE VoteTypeId=2 GROUP BY PostId

CREATE TABLE #votesDown(PostId int, VoteTypeIdCnt bigint)

INSERT INTO #votesDown SELECT PostId, Count(\*) as VoteTypeIdCnt FROM Votes WHERE VoteTypeId=3 GROUP BY PostId

select

C.Id as CommentId,

C.Score as CommentScore,

C.Text as CommentText,

DATEDIFF(d, C.CreationDate, GetDate()) as CommentCrDays,

LEN(C.Text) as CommentLength,

C.CreationDate as CommentCrDt,

C.UserId as CommentUserId,

CU.Reputation as comment\_owner\_reputation,

CASE WHEN LTRIM(RTRIM(CU.WebsiteURL)) = '' OR CU.WebsiteURL IS NULL THEN 0 ELSE 1 END +

CASE WHEN LTRIM(RTRIM(CU.Location)) = '' OR CU.Location IS NULL THEN 0 ELSE 1 END +

CASE WHEN LTRIM(RTRIM(CU.AboutMe)) = '' OR CU.AboutMe IS NULL THEN 0 ELSE 1 END as comment\_owner\_profile\_summary,

CU.Views as comment\_owner\_views,

CU.UpVotes as comment\_owner\_upvotes,

CU.DownVotes as comment\_owner\_downvotes,

DATEDIFF(d, CU.LastAccessDate, GetDate()) as comment\_owner\_lastactivity\_days,

P.Id as PostId,

DATEDIFF(d, P.CreationDate, P.LastEditDate) as editDurationAfterCreation,

DATEDIFF(d, P.CreationDate, P.LastActivityDate) as activityDurationAfterCreation,

LEN(P.Title) as title\_length,

LEN(P.Tags)-LEN(REPLACE(P.Tags, '<', '')) as num\_tags,

P.AnswerCount as PostAnswerCount,

P.FavoriteCount as num\_favorite,

CASE WHEN CHARINDEX('<code>', P.Body)>0

THEN 1

ELSE 0 END as hascode,

P.ViewCount as post\_views,

P.PostTypeId as postTypeId,

P.Id,

P.AcceptedAnswerId,

CASE WHEN QP.AcceptedAnswerId=P.Id THEN 1 ELSE 0 END as IsAcceptedAnswer,

P.Score as postScore,

LEN(P.Body) as post\_length,

P.CommentCount as PostCommentCount,

P.CreationDate as PostCrDt,

VU.VoteTypeIdCnt as PostUpVotes,

VD.VoteTypeIdCnt as PostDownVotes

from

Comments as C inner join

Posts as P on C.PostId=P.Id inner join

Users as CU on CU.Id=C.UserId inner join

Users as PU on PU.Id=P.OwnerUserId left outer join

Posts as QP on P.ParentId=QP.Id left outer join

#votesUp as VU on VU.PostId=P.Id left outer join

#votesDown as VD on VD.PostId=P.Id

where

(P.Tags like '%<python>%' or P.Tags like '%<java>%' or P.Tags like '%<javascript>%' or P.Tags like '%<c++>%'

or QP.Tags like '%<python>%' or QP.Tags like '%<java>%' or QP.Tags like '%<javascript>%' or QP.Tags like '%<c++>%')

and P.CreationDate > '2016-01-01 00:00:00' and P.commentCount>0 and

((P.PostTypeId=1 and P.AnswerCount>3) or (P.PostTypeId=2 and QP.AnswerCount>3))

and P.Id>35000000

## Appendix C : R Code

### C1. Classify data into spam/non-spam ([*link to complete output*](https://github.com/aloksatpathy/EDAStackOverflow))

#reading and transforming the data

df1= read.csv(file="QueryResults1.csv",header=TRUE)

df2<- read.csv(file="QueryResults2.csv",header=TRUE)

StackOverflow<- rbind(df1,df2)

#str(StackOverflow)

#names(StackOverflow)

nrow(StackOverflow)

#replacing NA with 1001

StackOverflow [is.na(StackOverflow)] <- 1001

#removing column that contain text or are unrelated to this question

StackOverflowdata<- StackOverflow[,-c(1, 2,3,12,14,15,22,23,25,26,27,28,29,30,31,32,33,34)]

#str(StackOverflowdata)

nrow(StackOverflowdata)

#to check uniqueness of data

lapply(StackOverflowdata["RejectionReasonId"], unique)

StackOverflowdata$IsSpam= ifelse(StackOverflowdata$RejectionReasonId==101,1,0)

#str(StackOverflowdata)

#names(StackOverflowdata)

StackOverflowdata<- StackOverflowdata[,-c(16)]

lapply(StackOverflowdata["IsSpam"], unique)

#dividing data for training and testing purpose

smp\_size<- floor(0.75\*nrow(StackOverflowdata))

set.seed(123)

train\_ind<- sample(seq\_len(nrow(StackOverflowdata)), size=smp\_size)

training\_data<- StackOverflowdata[train\_ind,]

testing\_data<- StackOverflowdata[-train\_ind,]

testing\_y<- testing\_data$IsSpam

nrow(training\_data)

nrow(testing\_data)

#str(training\_data)

#names(training\_data)

#GLM

logistics\_model<-glm(IsSpam~.,data=training\_data,family="binomial")

#to discover better model

#step(logistics\_model, direction = "forward")

logistics\_model = glm(IsSpam ~ PostTypeId + PostScore + post\_length +

owner\_reputation + owner\_profile\_summary + owner\_views +

owner\_upvotes + owner\_downvotes + editDurationAfterCreation +

q\_num\_tags + AnswerCount + CommentCount + has\_code +post\_views +

UserId, family = "binomial", data = training\_data)

summary(logistics\_model)

#performing trial modelling

#step(logistics\_model, direction = "backward")

#final regression model

logistics\_model = glm(formula = IsSpam ~ PostTypeId + PostScore + post\_length +

owner\_reputation + owner\_profile\_summary +

owner\_upvotes+ editDurationAfterCreation +

q\_num\_tags + AnswerCount + CommentCount+ post\_views, family = "binomial", data = training\_data)

summary(logistics\_model)

#Bayesian information criterion

BIC(logistics\_model)

logistics\_probs<-predict(logistics\_model,training\_data,type="response")

head(logistics\_probs)

logistics\_pred\_y=rep(0,length(testing\_y))

logistics\_pred\_y[logistics\_probs>0.55]=1

training\_y=training\_data$IsSpam

table(logistics\_pred\_y,training\_y)

mean(logistics\_pred\_y!=training\_y,na.rm=TRUE)

logistics\_probs<- predict(logistics\_model,testing\_data, type="response")

head(logistics\_probs)

logistics\_pred\_y=rep(0,length(testing\_y))

logistics\_pred\_y[logistics\_probs>0.55]=1

table(logistics\_pred\_y,testing\_y)

mean(logistics\_pred\_y!=testing\_y,na.rm=TRUE)

#kfold for regression

library(boot)

MSE\_10\_Fold\_CV=cv.glm(training\_data,logistics\_model,K=10)$delta[1]

MSE\_10\_Fold\_CV

MSE\_10\_Fold\_CV=NULL

for(i in 1:10){

model=glm(IsSpam~poly(PostTypeId + PostScore + post\_length +

owner\_reputation + owner\_profile\_summary +

owner\_upvotes+ editDurationAfterCreation +

q\_num\_tags + AnswerCount + CommentCount+ post\_views),data=training\_data)

MSE\_10\_Fold\_CV[i]=cv.glm(training\_data,model,K=10)$delta[1]

}

#summary(model)

MSE\_10\_Fold\_CV

#ROC logistic regression

#install.packagesll.packages("ROCR")

library(ROCR)

ROCRpred=prediction(logistics\_probs,testing\_y)

ROCRperf=performance(ROCRpred,"tpr","fpr")

#plot(ROCRperf)

#plot(ROCRperf,colorize=TRUE)

#plot(ROCRperf,colorize=TRUE,print.cutoffs.at=seq(0,1,0.05),text.adj=c(-0.2,1.7))

as.numeric(performance(ROCRpred,"auc")@y.values)

#Linear Discriminant Analysis

library(MASS)

lda.model<- lda(IsSpam~PostTypeId + PostScore + post\_length +

    owner\_reputation + owner\_profile\_summary +

    owner\_upvotes+ editDurationAfterCreation +

    q\_num\_tags + AnswerCount + CommentCount+ post\_views,data=training\_data)

#lda.model

#summary(lda.model)

lda\_pred<- predict(lda.model,training\_data)

lda\_class<- lda\_pred$class

table(lda\_class,training\_data$IsSpam)

mean(lda\_class!=training\_data$IsSpam)

lda\_pred<- predict(lda.model,testing\_data)

lda\_class<- lda\_pred$class

table(lda\_class,testing\_data$IsSpam)

mean(lda\_class!=testing\_data$IsSpam)

#Cross Validation LDA

library(rpart)

library(ipred)

ip.lda <- function(object, newdata) predict(object, newdata = newdata)$class

errorest(factor(training\_data$IsSpam)~ training\_data$PostTypeId + training\_data$PostScore + training\_data$post\_length +

    training\_data$owner\_reputation + training\_data$owner\_profile\_summary +

    training\_data$owner\_upvotes+ training\_data$editDurationAfterCreation +

    training\_data$q\_num\_tags + training\_data$AnswerCount + training\_data$CommentCount+ training\_data$post\_views, data=training\_data, model=lda, estimator="cv",est.para=control.errorest(k=10), predict=ip.lda)$err

#ROC LDA

S=lda\_pred$posterior[,2]

roc.curve=function(s,print=FALSE){

 Ps=(S>s)\*1

 FP=sum((Ps==1)\*(testing\_data$IsSpam==0))/sum(testing\_data$IsSpam==0)

TP=sum((Ps==1)\*(testing\_data$IsSpam==1))/sum(testing\_data$IsSpam==1)

 if(print==TRUE){

     print(table(Observed=testing\_data$IsSpam,Predicted=Ps))

  }

 vect=c(FP,TP)

 names(vect)=c("FPR","TPR")

 return(vect)

}

threshold=0.53

roc.curve(threshold,print=TRUE)

ROC.curve=Vectorize(roc.curve)

M.ROC=ROC.curve(seq(0,1,by=.01))

plot(M.ROC[1,],M.ROC[2,],col="blue",lwd=2,type="l",main="ROC for LDA")

abline(0,1)

#Quadratic Discriminant Analysis

qda.model<- qda(IsSpam~PostTypeId + PostScore + post\_length +

    owner\_reputation + owner\_profile\_summary +

    owner\_upvotes+ editDurationAfterCreation +

    q\_num\_tags + AnswerCount + CommentCount+ post\_views,data=training\_data)

#qda.model

qda\_pred=predict(qda.model,training\_data)

qda\_class=qda\_pred$class

table(qda\_class,training\_data$IsSpam)

mean(qda\_class!=training\_data$IsSpam)

qda\_pred1=predict(qda.model,testing\_data)

qda\_class\_n=qda\_pred1$class

table(qda\_class\_n,testing\_data$IsSpam)

mean(qda\_class\_n!=testing\_data$IsSpam)

#Cross Validation QDA

ip.qda <- function(object, newdata) predict(object, newdata = newdata)$class

errorest(factor(training\_data$IsSpam)~ training\_data$PostTypeId + training\_data$PostScore + training\_data$post\_length +

    training\_data$owner\_reputation + training\_data$owner\_profile\_summary +

    training\_data$owner\_upvotes+ training\_data$editDurationAfterCreation +

    training\_data$q\_num\_tags + training\_data$AnswerCount + training\_data$CommentCount+ training\_data$post\_views, data=training\_data, model=qda, estimator="cv",est.para=control.errorest(k=10), predict=ip.qda)$err

#ROC QDA

qda.S=qda\_pred1$posterior[,2]

roc.curve=function(s,print=FALSE){

 Ps=(qda.S>s)\*1

 FP=sum((Ps==1)\*(testing\_data$IsSpam==0))/sum(testing\_data$IsSpam==0)

 TP=sum((Ps==1)\*(testing\_data$IsSpam==1))/sum(testing\_data$IsSpam==1)

 if(print==TRUE){

    print(table(Observed=testing\_data$IsSpam,Predicted=Ps))

    }

 vect=c(FP,TP)

 names(vect)=c("FPR","TPR")

 return(vect)

 }

threshold=0.85

roc.curve(threshold,print=TRUE)

ROC.curve=Vectorize(roc.curve)

M.ROC=ROC.curve(seq(0,1,by=.01))

plot(M.ROC[1,],M.ROC[2,],col="blue",lwd=2,type="l",main="ROC for LDA")

abline(0,1)

#k nearest neighbours

library(class)

#install.packagesll.packages("cars")

#library(cars)

train.x<- cbind(training\_data$PostTypeId + training\_data$PostScore + training\_data$post\_length + training\_data$owner\_reputation + training\_data$owner\_profile\_summary + training\_data$owner\_upvotes + training\_data$editDurationAfterCreation + training\_data$q\_num\_tags + training\_data$AnswerCount + training\_data$CommentCount + training\_data$post\_views)

test.x<- cbind(testing\_data$PostTypeId + testing\_data$PostScore + testing\_data$post\_length + testing\_data$owner\_reputation + testing\_data$owner\_profile\_summary + testing\_data$owner\_upvotes + testing\_data$editDurationAfterCreation + testing\_data$q\_num\_tags + testing\_data$AnswerCount + testing\_data$CommentCount + testing\_data$post\_views)

train1.x=train.x[!duplicated(train.x),drop=FALSE]

test1.x=test.x[!duplicated(test.x), drop=FALSE]

tt<- training\_data$IsSpam[duplicated(train.x)=='FALSE']

head(tt)

length(tt)

knn.pred<- knn(data.frame(train1.x),data.frame(test1.x),tt,k=1)

tt1<- testing\_data$IsSpam[duplicated(test.x)=='FALSE']

length(tt1)

table(knn.pred,tt1)

mean(knn.pred!=tt1)

knn.pred<- knn(data.frame(train1.x),data.frame(test1.x),tt,k=2)

table(knn.pred,tt1)

mean(knn.pred!=tt1)

#Classification and Regression Trees

#CART Modeling:

#install.packagesll.packages("tree")

library(tree)

tree.training\_data=tree(as.factor(IsSpam)~PostTypeId + PostScore + post\_length +

    owner\_reputation + owner\_profile\_summary +

    owner\_upvotes+ editDurationAfterCreation +

    q\_num\_tags + AnswerCount + CommentCount+ post\_views,training\_data)

text(tree.training\_data,pretty=0)

summary(tree.training\_data)

plot(tree.training\_data)

text(tree.training\_data,pretty=0)

lf<- seq(1,nrow(training\_data))

tree.training\_data=tree(as.factor(IsSpam)~PostTypeId + PostScore + post\_length +

    owner\_reputation + owner\_profile\_summary +

    owner\_upvotes+ editDurationAfterCreation +

    q\_num\_tags + AnswerCount + CommentCount+ post\_views,training\_data,subset=lf)

tree.pred=predict(tree.training\_data,testing\_data,type="class")

table(tree.pred,testing\_y)

mean(tree.pred!=testing\_data$IsSpam)

#Cross Validation and Pruning for the Classification Tree:

cv.training\_data=cv.tree(tree.training\_data,FUN=prune.misclass)

names(cv.training\_data)

#cv.training\_data

par(mfrow=c(1,2))

plot(cv.training\_data$size,cv.training\_data$dev,type="b")

plot(cv.training\_data$k,cv.training\_data$dev,type="b")

par(mfrow=c(1,1))

prune.training\_data=prune.misclass(tree.training\_data,best=5)

plot(prune.training\_data)

text(prune.training\_data,pretty=0)

tree.pred=predict(prune.training\_data,testing\_data,type="class")

table(tree.pred,testing\_y)

mean(tree.pred!=testing\_data$IsSpam)

#ROC for CART:

tree.pred=predict(tree.training\_data,testing\_data,type="vector",prob=TRUE)

#tree.pred

tree.S=tree.pred[,2]

roc.curve=function(s,print=FALSE){

Ps=(tree.S>s)\*1

FP=sum((Ps==1)\*(testing\_data$IsSpam==0))/sum(testing\_data$IsSpam==0)

TP=sum((Ps==1)\*(testing\_data$IsSpam==1))/sum(testing\_data$IsSpam==1)

if(print==TRUE){

 print(table(Observed=testing\_data$IsSpam,Predicted=Ps))

}

vect=c(FP,TP)

names(vect)=c("FPR","TPR")

return(vect)

}

threshold=0.55

roc.curve(threshold,print=TRUE)

ROC.curve=Vectorize(roc.curve)

M.ROC=ROC.curve(seq(0,1,by=.01))

plot(M.ROC[1,],M.ROC[2,],col="blue",lwd=2,type="l",main="ROC for CART")

abline(0,1)

#Random Forest Modeling:

#install.packagesll.packages("randomForest")

library(randomForest)

bag.training\_data=randomForest(as.factor(IsSpam)~PostTypeId + PostScore + post\_length +

    owner\_reputation + owner\_profile\_summary +

    owner\_upvotes+ editDurationAfterCreation +

    q\_num\_tags + AnswerCount + CommentCount+ post\_views,data=training\_data,subset=lf,importance=TRUE)

#bag.training\_data

xyz = predict(bag.training\_data, newdata = testing\_data)

table(testing\_y, xyz)

mean(xyz!=testing\_y)

#C5.0

#install.packagesll.packages("C50")

library(C50)

c50\_model <- C5.0(training\_data[-16], as.factor(training\_data$IsSpam))

c50\_model

#summary(c50\_model)

#testing C50

c50\_pred <- predict(c50\_model, testing\_data)

#install.packagesll.packages("gmodels")

library(gmodels)

CrossTable(testing\_data$IsSpam, c50\_pred,

prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,

dnn = c('actual default', 'predicted default'))

#improving C50 with adaptive boosting

c50\_boost10 <- C5.0(training\_data[-16], as.factor(training\_data$IsSpam),

trials = 10)

c50\_boost10

c50\_pred10 <- predict(c50\_boost10, testing\_data)

CrossTable(testing\_data$IsSpam, c50\_pred10,

prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,

dnn = c('actual default', 'predicted default'))

#Dimensional Reduction

#install.packages("ISLR")

library(ISLR)

#fix(StackOverflowdata)

#names(StackOverflowdata)

dim(StackOverflowdata)

sum(is.na(StackOverflowdata$IsSpam))

StackOverflowdata=na.omit(StackOverflowdata)

dim(StackOverflowdata)

sum(is.na(StackOverflowdata))

#install.packages("pls")

library(pls)

set.seed(2)

pcr.fit=pcr(IsSpam~., data=StackOverflowdata,scale=TRUE,validation="CV")

summary(pcr.fit)

validationplot(pcr.fit,val.type="MSEP")

set.seed(1)

pcr.fit=pcr(IsSpam~., data=training\_data,scale=TRUE, validation="CV")

validationplot(pcr.fit,val.type="MSEP")

pcr.pred=predict(pcr.fit,testing\_data,ncomp=7)

mean((pcr.pred-testing\_y)^2)

pcr.fit=pcr(IsSpam~.,data=testing\_data, scale=TRUE,ncomp=7)

mean((pcr.pred-testing\_y)^2)

summary(pcr.fit)

# Partial Least Squares

set.seed(1)

pls.fit=plsr(IsSpam~., data=StackOverflowdata,scale=TRUE, validation="CV")

summary(pls.fit)

validationplot(pls.fit,val.type="MSEP")

pls.pred=predict(pls.fit,testing\_data,ncomp=2)

mean((pls.pred-testing\_y)^2)

pls.fit=plsr(IsSpam~., data=StackOverflowdata,scale=TRUE,ncomp=2)

summary(pls.fit)

# Subset Selection Methods

# Best Subset Selection

#install.packages("ISLR")

library(ISLR)

#sum(is.na(StackOverflow$IsSpam))

lapply(StackOverflowdata["IsSpam"], unique)

#Confirms NO NA data

#install.packages("leaps")

library(leaps)

regfit.full=regsubsets(IsSpam~.,StackOverflowdata)

summary(regfit.full)

regfit.full=regsubsets(IsSpam~.,data=StackOverflowdata,nvmax=19)

reg.summary=summary(regfit.full)

names(reg.summary)

reg.summary$rsq

par(mfrow=c(2,2))

plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")

plot(reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted

RSq",type="l")

which.max(reg.summary$adjr2)

points(11,reg.summary$adjr2[11], col="red",cex=2,pch=20)

plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp",type='l')

which.min(reg.summary$cp)

points(10,reg.summary$cp[10],col="red",cex=2,pch=20)

which.min(reg.summary$bic)

plot(reg.summary$bic,xlab="Number of Variables",ylab="BIC",type='l')

points(6,reg.summary$bic[6],col="red",cex=2,pch=20)

coef(regfit.full,6)

# Forward and Backward Stepwise Selection

regfit.fwd=regsubsets(IsSpam~.,data=StackOverflowdata,nvmax=19,method="forward")

summary(regfit.fwd)

regfit.bwd=regsubsets(IsSpam~.,data=StackOverflowdata,nvmax=19,method="backward"

)

summary(regfit.bwd)

coef(regfit.full,7)

coef(regfit.fwd,7)

coef(regfit.bwd,7)

# Choosing Among Models

set.seed(1)

train=sample(c(TRUE,FALSE), nrow(StackOverflowdata),rep=TRUE)

test=(!train)

regfit.best=regsubsets(IsSpam~.,data=StackOverflowdata[train,],nvmax=19)

test.mat=model.matrix(IsSpam~.,data=StackOverflowdata[test,])

val.errors=rep(NA,19)

for(i in 1:11){

coefi=coef(regfit.best,id=i)

pred=test.mat[,names(coefi)]%\*%coefi

val.errors[i]=mean((StackOverflowdata$IsSpam[test]-pred)^2)

}

val.errors

which.min(val.errors)

coef(regfit.best,10)

predict.regsubsets=function(object,newdata,id,...){

form=as.formula(object$call[[2]])

mat=model.matrix(form,newdata)

coefi=coef(object,id=id)

xvars=names(coefi)

mat[,xvars]%\*%coefi

}

regfit.best=regsubsets(IsSpam~.,data=StackOverflowdata,nvmax=19)

coef(regfit.best,10)

k=10

set.seed(1)

folds=sample(1:k,nrow(StackOverflowdata),replace=TRUE)

cv.errors=matrix(NA,k,11, dimnames=list(NULL, paste(1:11)))

for(j in 1:k){

best.fit=regsubsets(IsSpam~.,data=StackOverflowdata[folds!=j,],nvmax=11)

for(i in 1:11){

pred=predict(best.fit,StackOverflowdata[folds==j,],id=i)

cv.errors[j,i]=mean( (StackOverflowdata$IsSpam[folds==j]-pred)^2)

}

}

mean.cv.errors=apply(cv.errors,2,mean)

mean.cv.errors

par(mfrow=c(1,1))

plot(mean.cv.errors,type='b')

reg.best=regsubsets(IsSpam~.,data=StackOverflowdata, nvmax=11)

coef(reg.best,11)

# Ridge Regression and LASSO

#install.packages("ISLR")

library(ISLR)

fix(Hitters)

names(Hitters)

dim(Hitters)

sum(is.na(Hitters$Salary))

Hitters=na.omit(Hitters)

dim(Hitters)

sum(is.na(Hitters))

#install.packages("glmnet")

library(glmnet)

#the package invokes inputs and outputs separately unlike lm and glm

x=model.matrix(Salary~.,Hitters)[,-1]

y=Hitters$Salary

# set vector of lambda values to study range from 10^10 to 0.01, total length=100

grid=10^seq(10,-2,length=100)

ridge.mod=glmnet(x,y,alpha=0,lambda=grid)

dim(coef(ridge.mod))

# let us look at a few results here

#first lambda=50

ridge.mod$lambda[50]

coef(ridge.mod)[,50]

sqrt(sum(coef(ridge.mod)[-1,50]^2))

#next, lambda=60

ridge.mod$lambda[60]

coef(ridge.mod)[,60]

sqrt(sum(coef(ridge.mod)[-1,60]^2))

#prediction of the coefficients for lambda=50 (play with this)

predict(ridge.mod,s=500,type="coefficients")[1:20,]

#prepare for training and validation set testing

set.seed(1)

train=sample(1:nrow(x), nrow(x)/2)

test=(-train)

y.test=y[test]

ridge.mod=glmnet(x[train,],y[train],alpha=0,lambda=grid, thresh=1e-12)

ridge.pred=predict(ridge.mod,s=4,newx=x[test,])

#evaluate and compare test MSE and the spread of y.test

mean((ridge.pred-y.test)^2)

mean((mean(y[train])-y.test)^2)

#test wth two other lambdas

ridge.pred=predict(ridge.mod,s=1e10,newx=x[test,])

mean((ridge.pred-y.test)^2)

ridge.pred=predict(ridge.mod,s=0,newx=x[test,],exact=T)

mean((ridge.pred-y.test)^2)

# compare with lm

# The following two are the same

lm(y~x, subset=train)

predict(ridge.mod,s=0,exact=T,type="coefficients")[1:20,]

#Cross validation to get the best lambda

set.seed(1)

cv.out=cv.glmnet(x[train,],y[train],alpha=0)

plot(cv.out)

bestlam=cv.out$lambda.min

bestlam

#now predict with the best lambda

ridge.pred=predict(ridge.mod,s=bestlam,newx=x[test,])

mean((ridge.pred-y.test)^2)

out=glmnet(x,y,alpha=0)

predict(out,type="coefficients",s=bestlam)[1:20,]

# Lasso

#only difference in model building is to use aloha=1

lasso.mod=glmnet(x[train,],y[train],alpha=1,lambda=grid)

plot(lasso.mod)

# use CV to get best lambda

set.seed(1)

cv.out=cv.glmnet(x[train,],y[train],alpha=1)

plot(cv.out)

bestlam=cv.out$lambda.min

#use best lambda for prediction

lasso.pred=predict(lasso.mod,s=bestlam,newx=x[test,])

mean((lasso.pred-y.test)^2)

out=glmnet(x,y,alpha=1,lambda=grid)

lasso.coef=predict(out,type="coefficients",s=bestlam)[1:20,]

lasso.coef

lasso.coef[lasso.coef!=0]

```

# Support Vector Classifier

set.seed(1)

install.packages("e1071")

librarIsSpam(e1071)

svmfit=svm(IsSpam~., data=training\_data, kernel="linear", cost=10,scale=FALSE)

plot(svmfit, training\_data)

#which ones are support vectors

svmfit$index

summary(svmfit)

svmfit=svm(IsSpam~., data=training\_data, kernel="linear", cost=0.1,scale=FALSE)

plot(svmfit, training\_data)

svmfit$index

#cross validation

set.seed(1)

tune.out=tune(svm,IsSpam~.,data=training\_data,kernel="linear",ranges=list(cost=c(0.001, 0.01, 0.1, 1,5,10,100)))

summary(tune.out)

bestmod=tune.out$best.model

summary(bestmod)

ypred=predict(bestmod,testing\_data)

table(predict=ypred, truth=testing\_y)

svmfit=svm(IsSpam~., data=training\_data, kernel="linear", cost=.01,scale=FALSE)

ypred=predict(svmfit,testing\_data)

table(predict=ypred, truth=testing\_y)

plot(x, col=(y+5)/2, pch=19)

dat=data.frame(x=x,y=as.factor(IsSpam))

svmfit=svm(IsSpam~., data=dat, kernel="linear", cost=1e5)

summarIsSpam(svmfit)

plot(svmfit, dat)

svmfit=svm(IsSpam~., data=dat, kernel="linear", cost=1)

summarIsSpam(svmfit)

plot(svmfit,dat)

set.seed(1)

svmfit=svm(IsSpam~., data=training\_data, kernel="radial", gamma=1, cost=1)

plot(svmfit, training\_data)

summary(svmfit)

svmfit=svm(IsSpam~., data=training\_data, kernel="radial",gamma=1,cost=1e5)

plot(svmfit,training\_data)

set.seed(1)

tune.out=tune(svm, IsSpam~., data=training\_data, kernel="radial", ranges=list(cost=c(0.1,1,10,100,1000),gamma=c(0.5,1,2,3,4)))

summary(tune.out)

table(true=testing\_data, pred=predict(tune.out$best.model,newx=testing\_data))

#ROC

install.packages("ROCR")

library(ROCR)

rocplot=function(pred, truth, ...){

predob = prediction(pred, truth)

perf = performance(predob, "tpr", "fpr")

plot(perf,...)}

svmfit.opt=svm(IsSpam~., data=training\_data, kernel="radial",gamma=2, cost=1,decision.values=T)

fitted=attributes(predict(svmfit.opt,training\_data,decision.values=TRUE))$decision.values

par(mfrow=c(1,2))

rocplot(fitted,training\_data,main="Training Data")

svmfit.flex=svm(IsSpam~., data=training\_data, kernel="radial",gamma=50, cost=1, decision.values=T)

fitted=attributes(predict(svmfit.flex,training\_data,decision.values=T))$decision.values

rocplot(fitted,training\_data,add=T,col="red")

fitted=attributes(predict(svmfit.opt,testing\_y,decision.values=T))$decision.values

rocplot(fitted,testing\_y,main="Test Data")

fitted=attributes(predict(svmfit.flex,testing\_data,decision.values=T))$decision.values

rocplot(fitted,testing\_y,add=T,col="red")

### C2. Evaluate best possible answer – ([*link to complete output*](https://github.com/aloksatpathy/EDAStackOverflow))

getwd()

setwd("C:/Users/Alok Satpathy/Desktop/Fall 2016/EDA/Final Project")

Q2csv=read.csv("Question2Dataset.csv")

attach(Q2csv)

#Selecting columns that would be used for preparing model

Q2df=data.frame(a\_score, a\_body\_length, a\_body\_has\_code, a\_DaysOld, a\_has\_edited, a\_num\_comment, a\_owner\_reputation, a\_owner\_profile\_summary, a\_owner\_views, a\_owner\_upvotes, a\_owner\_downvotes, q\_score, q\_num\_views, q\_body\_length, q\_body\_has\_code, q\_DaysOld, q\_has\_edited, q\_title\_length, q\_num\_tags, q\_num\_answers, q\_num\_comment, q\_owner\_reputation, q\_owner\_profile\_summary, q\_owner\_views, q\_owner\_upvotes, q\_owner\_downvotes, accepted\_answer\_flag, a\_votes\_up, a\_votes\_down, q\_votes\_up, q\_votes\_down)

str(Q2df)

#Replacing NA with 0

Q2df[is.na(Q2df)]=0

str(Q2df)

# Subset Selection Methods

# Best Subset Selection

lapply(Q2df["accepted\_answer\_flag"], unique)

#install.packages("leaps")

library(leaps)

regfit.full=regsubsets(accepted\_answer\_flag~.,Q2df)

summary(regfit.full)

regfit.full=regsubsets(accepted\_answer\_flag~.,data=Q2df,nvmax=32)

reg.summary=summary(regfit.full)

names(reg.summary)

reg.summary$rsq

par(mfrow=c(2,2))

plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")

plot(reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted

RSq",type="l")

which.max(reg.summary$adjr2)

points(11,reg.summary$adjr2[11], col="red",cex=2,pch=20)

plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp",type='l')

which.min(reg.summary$cp)

points(10,reg.summary$cp[10],col="red",cex=2,pch=20)

which.min(reg.summary$bic)

plot(reg.summary$bic,xlab="Number of Variables",ylab="BIC",type='l')

points(6,reg.summary$bic[6],col="red",cex=2,pch=20)

coef(regfit.full,6)

# Forward and Backward Stepwise Selection

regfit.fwd=regsubsets(accepted\_answer\_flag~.,data=Q2df,nvmax=32,method="forward")

summary(regfit.fwd)

regfit.bwd=regsubsets(accepted\_answer\_flag~.,data=Q2df,nvmax=32,method="backward"

)

summary(regfit.bwd)

coef(regfit.full,7)

coef(regfit.fwd,7)

coef(regfit.bwd,7)

# Choosing Among Models

set.seed(1)

train=sample(c(TRUE,FALSE), nrow(Q2df),rep=TRUE)

test=(!train)

regfit.best=regsubsets(accepted\_answer\_flag~.,data=Q2df[train,],nvmax=32)

test.mat=model.matrix(accepted\_answer\_flag~.,data=Q2df[test,])

val.errors=rep(NA,31)

for(i in 1:11){

coefi=coef(regfit.best,id=i)

pred=test.mat[,names(coefi)]%\*%coefi

val.errors[i]=mean((Q2df$accepted\_answer\_flag[test]-pred)^2)

}

val.errors

which.min(val.errors)

coef(regfit.best,30)

predict.regsubsets=function(object,newdata,id,...){

form=as.formula(object$call[[2]])

mat=model.matrix(form,newdata)

coefi=coef(object,id=id)

xvars=names(coefi)

mat[,xvars]%\*%coefi

}

regfit.best=regsubsets(accepted\_answer\_flag~.,data=Q2df,nvmax=32)

coef(regfit.best,30)

k=30

set.seed(1)

folds=sample(1:k,nrow(Q2df),replace=TRUE)

cv.errors=matrix(NA,k,31, dimnames=list(NULL, paste(1:31)))

for(j in 1:k){

best.fit=regsubsets(accepted\_answer\_flag~.,data=Q2df[folds!=j,],nvmax=30)

for(i in 1:30){

pred=predict(best.fit,Q2df[folds==j,],id=i)

cv.errors[j,i]=mean( (Q2df$accepted\_answer\_flag[folds==j]-pred)^2)

}

}

mean.cv.errors=apply(cv.errors,2,mean)

mean.cv.errors

par(mfrow=c(1,1))

plot(mean.cv.errors,type='b')

reg.best=regsubsets(accepted\_answer\_flag~.,data=Q2df, nvmax=30)

coef(reg.best,3)

# Ridge Regression and LASSO

fix(Q2df)

names(Q2df)

dim(Q2df)

sum(is.na(Q2df$accepted\_answer\_flag))

Q2df=na.omit(Q2df)

dim(Q2df)

sum(is.na(Q2df))

install.packages("glmnet")

library(glmnet)

#The package invokes inputs and outputs separately unlike lm and glm

x=model.matrix(accepted\_answer\_flag~.,Q2df)[,-1]

y=Q2df$accepted\_answer\_flag

#Set vector of lambda values to study range from 10^10 to 0.01, total length=100

grid=10^seq(10,-2,length=100)

ridge.mod=glmnet(x,y,alpha=0,lambda=grid)

dim(coef(ridge.mod))

#Let us look at a few results here

#first lambda=50

ridge.mod$lambda[50]

coef(ridge.mod)[,50]

sqrt(sum(coef(ridge.mod)[-1,50]^2))

#next, lambda=60

ridge.mod$lambda[60]

coef(ridge.mod)[,60]

sqrt(sum(coef(ridge.mod)[-1,60]^2))

#prediction of the coefficients for lambda=50

predict(ridge.mod,s=500,type="coefficients")[1:31,]

#prepare for training and validation set testing

set.seed(1)

train=sample(1:nrow(x), nrow(x)/2)

test=(-train)

y.test=y[test]

ridge.mod=glmnet(x[train,],y[train],alpha=0,lambda=grid, thresh=1e-12)

ridge.pred=predict(ridge.mod,s=4,newx=x[test,])

#evaluate and compare test MSE and the spread of y.test

mean((ridge.pred-y.test)^2)

mean((mean(y[train])-y.test)^2)

#test wth two other lambdas

ridge.pred=predict(ridge.mod,s=1e10,newx=x[test,])

mean((ridge.pred-y.test)^2)

ridge.pred=predict(ridge.mod,s=0,newx=x[test,],exact=T)

mean((ridge.pred-y.test)^2)

#compare with lm

# The following two are the same

lm(y~x, subset=train)

predict(ridge.mod,s=0,exact=T,type="coefficients")[1:31,]

#Cross validation to get the best lambda

set.seed(1)

cv.out=cv.glmnet(x[train,],y[train],alpha=0)

plot(cv.out)

bestlam=cv.out$lambda.min

bestlam

#now predict with the best lambda

ridge.pred=predict(ridge.mod,s=bestlam,newx=x[test,])

mean((ridge.pred-y.test)^2)

out=glmnet(x,y,alpha=0)

predict(out,type="coefficients",s=bestlam)[1:31,]

# Lasso

#only difference in model building is to use alpha=1

lasso.mod=glmnet(x[train,],y[train],alpha=1,lambda=grid)

plot(lasso.mod)

# use CV to get best lambda

set.seed(1)

cv.out=cv.glmnet(x[train,],y[train],alpha=1)

plot(cv.out)

bestlam=cv.out$lambda.min

#use best lambda for prediction

lasso.pred=predict(lasso.mod,s=bestlam,newx=x[test,])

mean((lasso.pred-y.test)^2)

out=glmnet(x,y,alpha=1,lambda=grid)

lasso.coef=predict(out,type="coefficients",s=bestlam)[1:31,]

lasso.coef

lasso.coef[lasso.coef!=0]

#Dimensionality Reduction PCA:

install.packages("pls")

library(pls)

set.seed(2)

pcr.fit=pcr(accepted\_answer\_flag~., data=Q2df,scale=TRUE,validation="CV")

summary(pcr.fit)

validationplot(pcr.fit,val.type="MSEP")

set.seed(1)

pcr.fit=pcr(accepted\_answer\_flag~., data=Q2df,subset=train,scale=TRUE, validation="CV")

validationplot(pcr.fit,val.type="MSEP")

pcr.pred=predict(pcr.fit,x[test,],ncomp=7)

mean((pcr.pred-y.test)^2)

pcr.fit=pcr(y~x,scale=TRUE,ncomp=7)

summary(pcr.fit)

# Partial Least Squares

set.seed(1)

pls.fit=plsr(accepted\_answer\_flag~., data=Q2df,subset=train,scale=TRUE, validation="CV")

summary(pls.fit)

validationplot(pls.fit,val.type="MSEP")

pls.pred=predict(pls.fit,x[test,],ncomp=2)

mean((pls.pred-y.test)^2)

pls.fit=plsr(accepted\_answer\_flag~., data=Q2df,scale=TRUE,ncomp=2)

summary(pls.fit)

#Dividing dataset 75% for training and 25% for testing

Q2smp\_size=floor(0.75\*nrow(Q2df))

set.seed(123)

Q2train\_ind=sample(seq\_len(nrow(Q2df)), size = Q2smp\_size)

Q2Train=Q2df[Q2train\_ind, ]

Q2Test=Q2df[-Q2train\_ind, ]

#Initial logistic regression model with all parameters

Q2lm=glm(accepted\_answer\_flag~.,data=Q2Train,family="binomial")

summary(Q2lm)

#Checking for significant parameters with forward step

Q2modelstepforward=step(Q2lm, direction="forward")

summary(Q2modelstepforward)

#Checking for significant parameters with backward step

Q2modelstepbackward=step(Q2lm, direction="backward")

summary(Q2modelstepbackward)

#Removing insignificant parameters and remodelling

Q2lm2=glm(accepted\_answer\_flag~a\_body\_length+a\_body\_has\_code+a\_DaysOld+a\_has\_edited+a\_num\_comment+a\_owner\_reputation+a\_owner\_profile\_summary+a\_owner\_downvotes+q\_num\_views+q\_DaysOld+q\_title\_length+q\_num\_answers+q\_num\_comment+q\_owner\_profile\_summary+q\_owner\_downvotes,data=Q2Train,family="binomial")

summary(Q2lm2)

#Again removing insignificant parameters and remodelling

Q2lm3=glm(accepted\_answer\_flag~a\_body\_length+a\_body\_has\_code+a\_DaysOld+a\_has\_edited+a\_num\_comment+a\_owner\_reputation+a\_owner\_profile\_summary+a\_owner\_downvotes+q\_DaysOld+q\_title\_length+q\_num\_answers+q\_num\_comment+q\_owner\_profile\_summary,data=Q2Train,family="binomial")

summary(Q2lm3)

BIC(Q2lm3)

#Predicting based on the logistic model built

logistic\_probs=predict(Q2lm3, Q2Train, type="response")

head(logistic\_probs)

testing\_y=Q2Test$accepted\_answer\_flag

logistic\_pred\_y=rep(0,length(testing\_y))

logistic\_pred\_y[logistic\_probs>0.5]=1

training\_y=Q2Train$accepted\_answer\_flag

table(logistic\_pred\_y,training\_y)

mean(logistic\_pred\_y!=training\_y,na.rm=TRUE)

logistic\_probs=predict(Q2lm3, Q2Test, type="response")

head(logistic\_probs)

logistic\_pred\_y=rep(0,length(testing\_y))

logistic\_pred\_y[logistic\_probs>0.5]=1

table(logistic\_pred\_y,testing\_y)

mean(logistic\_pred\_y!=testing\_y,na.rm=TRUE)

#Cross Validation for logistic regression

#K-fold

library(boot)

MSE\_10\_Fold\_CV=cv.glm(Q2Train,Q2lm3,K=10)$delta[1]

MSE\_10\_Fold\_CV

MSE\_10\_Fold\_CV=NULL

for(i in 1:10){

model=glm(accepted\_answer\_flag~a\_body\_length+a\_body\_has\_code+a\_DaysOld+a\_has\_edited+a\_num\_comment+a\_owner\_reputation+a\_owner\_profile\_summary+a\_owner\_downvotes+q\_DaysOld+q\_title\_length+q\_num\_answers+q\_num\_comment+q\_owner\_profile\_summary,data=Q2Train)

MSE\_10\_Fold\_CV[i]=cv.glm(Q2Train,model,K=10)$delta[1]

}

MSE\_10\_Fold\_CV

#ROC of Logistic Regression

#install.packages("ROCR")

library(ROCR)

ROCRpred<- prediction(logistic\_probs,testing\_y)

ROCRperf<- performance(ROCRpred,"tpr","fpr")

plot(ROCRperf)

plot(ROCRperf,colorize=TRUE)

plot(ROCRperf, colorize = TRUE, print.cutoffs.at = seq(0,1,0.05), text.adj = c(-0.2,1.7))

as.numeric(performance(ROCRpred,"auc")@y.values)

#install.packages("lda")

library(lda)

library(MASS)

lda.model = lda(accepted\_answer\_flag~a\_body\_length+a\_body\_has\_code+a\_DaysOld+a\_has\_edited+a\_num\_comment+a\_owner\_reputation+a\_owner\_profile\_summary+a\_owner\_downvotes+q\_DaysOld+q\_title\_length+q\_num\_answers+q\_num\_comment+q\_owner\_profile\_summary, data=Q2Train, family = binomial)

lda.model

#predicting the LDA model with training data

lda\_pred=predict(lda.model,Q2Train)

lda\_class=lda\_pred$class

#CONFUSION MATRIX with LDA Training data

table(lda\_class,Q2Train$accepted\_answer\_flag)

#Misclassification Rate with LDA training data

mean(lda\_class!=Q2Train$accepted\_answer\_flag)

#predicting the LDA model with test data

lda\_prediction=predict(lda.model,Q2Test)

lda\_class\_n=lda\_prediction$class

#CONFUSION MATRIX with LDA Testing data

table(lda\_class\_n,Q2Test$accepted\_answer\_flag)

#Misclassification Rate with LDA testing data

mean(lda\_class\_n!=Q2Test$accepted\_answer\_flag)

#CROSS VALIDATION of LDA Model

library(rpart)

library(ipred)

ip.lda <- function(object, newdata) predict(object, newdata = newdata)$class

errorest(factor(Q2Train$accepted\_answer\_flag)~ Q2Train$a\_body\_length+Q2Train$a\_body\_has\_code+Q2Train$a\_DaysOld+Q2Train$a\_has\_edited+Q2Train$a\_num\_comment+Q2Train$a\_owner\_reputation+Q2Train$a\_owner\_profile\_summary+Q2Train$a\_owner\_downvotes+Q2Train$q\_DaysOld+Q2Train$q\_title\_length+Q2Train$q\_num\_answers+Q2Train$q\_num\_comment+Q2Train$q\_owner\_profile\_summary, data=Q2Train, model=lda, estimator="cv",est.para=control.errorest(k=10), predict=ip.lda)$err

#ROC for LDA

S = lda\_prediction$posterior[,2]

roc.curve=function(s,print=FALSE){

Ps=(S>s)\*1

FP=sum((Ps==1)\*(Q2Test$accepted\_answer\_flag == 0))/sum(Q2Test$accepted\_answer\_flag == 0)

TP=sum((Ps==1)\*(Q2Test$accepted\_answer\_flag == 1))/sum(Q2Test$accepted\_answer\_flag == 1)

if(print==TRUE){

print(table(Observed=Q2Test$accepted\_answer\_flag,Predicted=Ps))

}

vect=c(FP,TP)

names(vect)=c("FPR","TPR")

return(vect)

}

threshold = 0.5

roc.curve(threshold,print=TRUE)

ROC.curve=Vectorize(roc.curve)

M.ROC=ROC.curve(seq(0,1,by=.01))

plot(M.ROC[1,],M.ROC[2,],col="blue",lwd=2,type="l", main = "ROC for LDA")

abline(0,1)

qda\_model = qda(accepted\_answer\_flag~a\_body\_length+a\_body\_has\_code+a\_DaysOld+a\_has\_edited+a\_num\_comment+a\_owner\_reputation+a\_owner\_profile\_summary+a\_owner\_downvotes+q\_DaysOld+q\_title\_length+q\_num\_answers+q\_num\_comment+q\_owner\_profile\_summary, data=Q2Train, family=binomial)

qda\_model

#predicting the model with training data

qda\_pred=predict(qda\_model, Q2Train)

qda\_class=qda\_pred$class

#CONFUSION MATRIX for QDA training data

table(qda\_class,Q2Train$accepted\_answer\_flag)

#Misclassification Rate for QDA training data

mean(qda\_class!=Q2Train$accepted\_answer\_flag)

#predicting the model with test data for QDA

qda\_pred1=predict(qda\_model,Q2Test)

qda\_class\_n=qda\_pred1$class

#CONFUSION MATRIX for QDA test data

table(qda\_class\_n,Q2Test$accepted\_answer\_flag)

#Misclassification Rate for QDA test data

mean(qda\_class\_n!=Q2Test$accepted\_answer\_flag)

#CROSS VALIDATION for QDA

ip.qda <- function(object, newdata) predict(object, newdata = newdata)$class

errorest(factor(Q2Train$accepted\_answer\_flag)~ Q2Train$a\_body\_length+Q2Train$a\_body\_has\_code+Q2Train$a\_DaysOld+Q2Train$a\_has\_edited+Q2Train$a\_num\_comment+Q2Train$a\_owner\_reputation+Q2Train$a\_owner\_profile\_summary+Q2Train$a\_owner\_downvotes+Q2Train$q\_DaysOld+Q2Train$q\_title\_length+Q2Train$q\_num\_answers+Q2Train$q\_num\_comment+Q2Train$q\_owner\_profile\_summary, data=Q2Train, model=qda, estimator="cv",est.para=control.errorest(k=10), predict=ip.qda)$err

#ROC for QDA

qda.S = qda\_pred1$posterior[,2]

roc.curve=function(s,print=FALSE){

Ps=(qda.S>s)\*1

FP=sum((Ps==1)\*(Q2Test$accepted\_answer\_flag == 0))/sum(Q2Test$accepted\_answer\_flag == 0)

TP=sum((Ps==1)\*(Q2Test$accepted\_answer\_flag == 1))/sum(Q2Test$accepted\_answer\_flag == 1)

if(print==TRUE){

print(table(Observed= Q2Test$accepted\_answer\_flag, Predicted=Ps))

}

vect=c(FP,TP)

names(vect)=c("FPR","TPR")

return(vect)

}

threshold = 0.5

roc.curve(threshold,print=TRUE)

ROC.curve=Vectorize(roc.curve)

M.ROC=ROC.curve(seq(0,1,by=.01))

plot(M.ROC[1,],M.ROC[2,],col="blue",lwd=2,type="l", main = "ROC for QDA")

abline(0,1)

library(class)

train.x<-cbind(Q2Train$a\_body\_length+Q2Train$a\_body\_has\_code+Q2Train$a\_DaysOld+Q2Train$a\_has\_edited+Q2Train$a\_num\_comment+Q2Train$a\_owner\_reputation+Q2Train$a\_owner\_profile\_summary+Q2Train$a\_owner\_downvotes+Q2Train$q\_DaysOld+Q2Train$q\_title\_length+Q2Train$q\_num\_answers+Q2Train$q\_num\_comment+Q2Train$q\_owner\_profile\_summary)

test.x<- cbind(Q2Test$a\_body\_length+Q2Test$a\_body\_has\_code+Q2Test$a\_DaysOld+Q2Test$a\_has\_edited+Q2Test$a\_num\_comment+Q2Test$a\_owner\_reputation+Q2Test$a\_owner\_profile\_summary+Q2Test$a\_owner\_downvotes+Q2Test$q\_DaysOld+Q2Test$q\_title\_length+Q2Test$q\_num\_answers+Q2Test$q\_num\_comment+Q2Test$q\_owner\_profile\_summary)

set.seed(123)

#predicting the model with testing data for KNN with k=1

knn.pred<- knn(data.frame(train.x),data.frame(test.x),Q2Train$accepted\_answer\_flag,k=1,prob=TRUE)

#CONFUSION matrix for KNN

table(knn.pred,Q2Test$accepted\_answer\_flag)

#misclassification rate for KNN

mean(knn.pred!=Q2Test$accepted\_answer\_flag)

#Trying to predict KNN with different k values to get the lowest misclassification rate

#predicting the KNN model with k=2

knn.pred<- knn(data.frame(train.x),data.frame(test.x),Q2Train$accepted\_answer\_flag,k=2,prob=TRUE)

table(knn.pred,Q2Test$accepted\_answer\_flag)

mean(knn.pred!=Q2Test$accepted\_answer\_flag)

#predicting the KNN model with k=3

knn.pred<- knn(data.frame(train.x),data.frame(test.x),Q2Train$accepted\_answer\_flag,k=3,prob=TRUE)

table(knn.pred,Q2Test$accepted\_answer\_flag)

mean(knn.pred!=Q2Test$accepted\_answer\_flag)

#predicting the KNN model with k=4

knn.pred<- knn(data.frame(train.x),data.frame(test.x),Q2Train$accepted\_answer\_flag,k=4,prob=TRUE)

table(knn.pred,Q2Test$accepted\_answer\_flag)

mean(knn.pred!=Q2Test$accepted\_answer\_flag)

#predicting the KNN model with k=5

knn.pred<- knn(data.frame(train.x),data.frame(test.x),Q2Train$accepted\_answer\_flag,k=5,prob=TRUE)

table(knn.pred,Q2Test$accepted\_answer\_flag)

mean(knn.pred!=Q2Test$accepted\_answer\_flag)

#predicting the KNN model with k=6

knn.pred<- knn(data.frame(train.x),data.frame(test.x),Q2Train$accepted\_answer\_flag,k=6,prob=TRUE)

table(knn.pred,Q2Test$accepted\_answer\_flag)

mean(knn.pred!=Q2Test$accepted\_answer\_flag)

#predicting the KNN model with k=7

knn.pred<- knn(data.frame(train.x),data.frame(test.x),Q2Train$accepted\_answer\_flag,k=7,prob=TRUE)

table(knn.pred,Q2Test$accepted\_answer\_flag)

mean(knn.pred!=Q2Test$accepted\_answer\_flag)

#predicting the KNN model with k=8

knn.pred<- knn(data.frame(train.x),data.frame(test.x),Q2Train$accepted\_answer\_flag,k=8,prob=TRUE)

table(knn.pred,Q2Test$accepted\_answer\_flag)

mean(knn.pred!=Q2Test$accepted\_answer\_flag)

#predicting the KNN model with k=9

knn.pred<- knn(data.frame(train.x),data.frame(test.x),Q2Train$accepted\_answer\_flag,k=9,prob=TRUE)

table(knn.pred,Q2Test$accepted\_answer\_flag)

mean(knn.pred!=Q2Test$accepted\_answer\_flag)

#predicting the KNN model with k=10

knn.pred<- knn(data.frame(train.x),data.frame(test.x),Q2Train$accepted\_answer\_flag,k=10,prob=TRUE)

table(knn.pred,Q2Test$accepted\_answer\_flag)

mean(knn.pred!=Q2Test$accepted\_answer\_flag)

#predicting the KNN model with k=11

knn.pred<- knn(data.frame(train.x),data.frame(test.x),Q2Train$accepted\_answer\_flag,k=11,prob=TRUE)

table(knn.pred,Q2Test$accepted\_answer\_flag)

mean(knn.pred!=Q2Test$accepted\_answer\_flag)

#predicting the KNN model with k=12

knn.pred<- knn(data.frame(train.x),data.frame(test.x),Q2Train$accepted\_answer\_flag,k=12,prob=TRUE)

table(knn.pred,Q2Test$accepted\_answer\_flag)

mean(knn.pred!=Q2Test$accepted\_answer\_flag)

#predicting the KNN model with k=13

knn.pred<- knn(data.frame(train.x),data.frame(test.x),Q2Train$accepted\_answer\_flag,k=13,prob=TRUE)

table(knn.pred,Q2Test$accepted\_answer\_flag)

mean(knn.pred!=Q2Test$accepted\_answer\_flag)

#predicting the KNN model with k=14

knn.pred<- knn(data.frame(train.x),data.frame(test.x),Q2Train$accepted\_answer\_flag,k=14,prob=TRUE)

table(knn.pred,Q2Test$accepted\_answer\_flag)

mean(knn.pred!=Q2Test$accepted\_answer\_flag)

#predicting the KNN model with k=15

knn.pred<- knn(data.frame(train.x),data.frame(test.x),Q2Train$accepted\_answer\_flag,k=15,prob=TRUE)

table(knn.pred,Q2Test$accepted\_answer\_flag)

mean(knn.pred!=Q2Test$accepted\_answer\_flag)

#predicting the KNN model with k=17

knn.pred<- knn(data.frame(train.x),data.frame(test.x),Q2Train$accepted\_answer\_flag,k=17,prob=TRUE)

table(knn.pred,Q2Test$accepted\_answer\_flag)

mean(knn.pred!=Q2Test$accepted\_answer\_flag)

#predicting the KNN model with k=18

knn.pred<- knn(data.frame(train.x),data.frame(test.x),Q2Train$accepted\_answer\_flag,k=18,prob=TRUE)

table(knn.pred,Q2Test$accepted\_answer\_flag)

mean(knn.pred!=Q2Test$accepted\_answer\_flag)

#predicting the KNN model with k=19

knn.pred<- knn(data.frame(train.x),data.frame(test.x),Q2Train$accepted\_answer\_flag,k=19,prob=TRUE)

table(knn.pred,Q2Test$accepted\_answer\_flag)

mean(knn.pred!=Q2Test$accepted\_answer\_flag)

#predicting the KNN model with k=20

knn.pred<- knn(data.frame(train.x),data.frame(test.x),Q2Train$accepted\_answer\_flag,k=20,prob=TRUE)

table(knn.pred,Q2Test$accepted\_answer\_flag)

mean(knn.pred!=Q2Test$accepted\_answer\_flag)

#predicting the KNN model with k=21

knn.pred<- knn(data.frame(train.x),data.frame(test.x),Q2Train$accepted\_answer\_flag,k=21,prob=TRUE)

table(knn.pred,Q2Test$accepted\_answer\_flag)

mean(knn.pred!=Q2Test$accepted\_answer\_flag)

#predicting the KNN model with k=22

knn.pred<- knn(data.frame(train.x),data.frame(test.x),Q2Train$accepted\_answer\_flag,k=22,prob=TRUE)

table(knn.pred,Q2Test$accepted\_answer\_flag)

mean(knn.pred!=Q2Test$accepted\_answer\_flag)

#predicting the KNN model with k=100

knn.pred<- knn(data.frame(train.x),data.frame(test.x),Q2Train$accepted\_answer\_flag,k=100,prob=TRUE)

table(knn.pred,Q2Test$accepted\_answer\_flag)

mean(knn.pred!=Q2Test$accepted\_answer\_flag)

#taking the model built using knn=8

#CROSS VALIDATION of KNN

bwpredict.knn <- function(object, newdata) predict.ipredknn(object, newdata, type="class")

errorest(factor(Q2Train$accepted\_answer\_flag) ~ Q2Train$a\_body\_length+Q2Train$a\_body\_has\_code+Q2Train$a\_DaysOld+Q2Train$a\_has\_edited+Q2Train$a\_num\_comment+Q2Train$a\_owner\_reputation+Q2Train$a\_owner\_profile\_summary+Q2Train$a\_owner\_downvotes+Q2Train$q\_DaysOld+Q2Train$q\_title\_length+Q2Train$q\_num\_answers+Q2Train$q\_num\_comment+Q2Train$q\_owner\_profile\_summary, data=Q2Train, model=ipredknn, estimator="cv", est.para=control.errorest(k=10), predict=bwpredict.knn, kk=10)$err

#ROC for KNN

library(ROCR)

prob<- attr(knn.pred,"prob")

prob

prob <- (2\*(ifelse(knn.pred == "0", 1-prob, prob)) - 1)

knn <- prediction(prob, testing\_y)

pred\_knn <- prediction(prob, testing\_y)

pred\_knn <- performance(pred\_knn, "tpr", "fpr")

plot(pred\_knn, avg= "threshold", colorize=T, lwd=3, main="ROC curve for KNN")

as.numeric(performance(ROCRpred,"auc")@y.values)

#install.packages("tree")

library(tree)

tree.Q2Train = tree(accepted\_answer\_flag~a\_body\_length+a\_body\_has\_code+a\_DaysOld+a\_has\_edited+a\_num\_comment+a\_owner\_reputation+a\_owner\_profile\_summary+a\_owner\_downvotes+q\_DaysOld+q\_title\_length+q\_num\_answers+q\_num\_comment+q\_owner\_profile\_summary,Q2Train)

summary(tree.Q2Train)

plot(tree.Q2Train)

text(tree.Q2Train,pretty=0)

#Classification tree

lf=seq(1,nrow(Q2Train))

tree.Q2Train=tree(as.factor(accepted\_answer\_flag)~a\_body\_length+a\_body\_has\_code+a\_DaysOld+a\_has\_edited+a\_num\_comment+a\_owner\_reputation+a\_owner\_profile\_summary+a\_owner\_downvotes+q\_DaysOld+q\_title\_length+q\_num\_answers+q\_num\_comment+q\_owner\_profile\_summary,subset=lf)

summary(tree.Q2Train)

plot(tree.Q2Train)

text(tree.Q2Train,pretty=0)

tree.pred=predict(tree.Q2Train,Q2Test,type="class")

#Confusion matrix for CART

table(tree.pred,testing\_y)

#Misclassification rate for CART

mean(tree.pred!=Q2Test$accepted\_answer\_flag)

#Pruning and Cross Validation of the Classification Tree

cv.Q2Train=cv.tree(tree.Q2Train,FUN=prune.misclass)

names(cv.Q2Train)

cv.Q2Train

par(mfrow=c(1,2))

plot(cv.Q2Train$size, cv.Q2Train$dev, type="b")

plot(cv.Q2Train$k, cv.Q2Train$dev, type="b")

par(mfrow=c(1,1))

prune.Q2Train=prune.misclass(tree.Q2Train, best=9)

plot(prune.Q2Train)

text(prune.Q2Train,pretty=0)

tree.pred=predict(prune.Q2Train, Q2Test, type="class")

table(tree.pred, testing\_y)

mean(tree.pred!=Q2Test$accepted\_answer\_flag)

library(randomForest)

set.seed(123)

bag.Q2Train=randomForest(accepted\_answer\_flag~a\_body\_length+a\_body\_has\_code+a\_DaysOld+a\_has\_edited+a\_num\_comment+a\_owner\_reputation+a\_owner\_profile\_summary+a\_owner\_downvotes+q\_DaysOld+q\_title\_length+q\_num\_answers+q\_num\_comment+q\_owner\_profile\_summary,data=Q2Train, subset=lf, mtry=4, importance=TRUE)

bag.Q2Train

yhat.bag=predict(bag.Q2Train, Q2Test)

plot(yhat.bag, testing\_y)

abline(0,1)

mean((yhat.bag-testing\_y)^2)

mean(yhat.bag!=testing\_y)

#ROC for CART

tree.pred=predict(tree.Q2Train,Q2Test,type="vector",prob=TRUE)

tree.pred

tree.S = tree.pred[,2]

roc.curve=function(s,print=FALSE){

Ps=(tree.S>s)\*1

FP=sum((Ps==1)\*(Q2Train$accepted\_answer\_flag == 0))/sum(Q2Train$accepted\_answer\_flag == 0)

TP=sum((Ps==1)\*(Q2Train$accepted\_answer\_flag == 1))/sum(Q2Train$accepted\_answer\_flag == 1)

if(print==TRUE){

print(table(Observed=Q2Test$accepted\_answer\_flag,Predicted=Ps))

}

vect=c(FP,TP)

names(vect)=c("FPR","TPR")

return(vect)

}

threshold = 0.5

roc.curve(threshold,print=TRUE)

ROC.curve=Vectorize(roc.curve)

M.ROC=ROC.curve(seq(0,1,by=.01))

plot(M.ROC[1,],M.ROC[2,],col="blue",lwd=2,type="l", main = "ROC for CART")

abline(0,1)

### C3. Predict comment sentiment using NLP

library(devtools)

library(sentiment)

library(twitteR)

library(plyr)

library(ggplot2)

library(wordcloud)

library(RColorBrewer)

library(devtools)

library(stringr)

getwd()

setwd("C:/Users/Alok Satpathy/Desktop/Fall 2016/EDA/Final Project")

Q3csv=read.csv("Question3Dataset.csv")

attach(Q3csv)

str(Q3csv)

Q3csv[is.na(Q3csv)]=0

str(Q3csv)

Q3csv["SentimentScore"]=NA

str(Q3csv)

#function to clean data

cleancomments = function(commentstest)

{

commentstest\_cl = gsub("(RT|via)((?:\\b\\W\*@\\w+)+)","",commentstest)

commentstest\_cl = gsub("http[^[:blank:]]+", "", commentstest\_cl)

commentstest\_cl = gsub("@\\w+", "", commentstest\_cl)

commentstest\_cl = gsub("[ \t]{2,}", "", commentstest\_cl)

commentstest\_cl = gsub("^\\s+|\\s+$", "", commentstest\_cl)

commentstest\_cl = gsub("[[:punct:]]", " ", commentstest\_cl)

commentstest\_cl = gsub("[^[:alnum:]]", " ", commentstest\_cl)

commentstest\_cl <- gsub('\\d+', '', commentstest\_cl)

return(commentstest\_cl)

}

#function to calculate number of words in each category within a sentence

sentimentScore <- function(sentences, vNegTerms, negTerms, posTerms, vPosTerms){

final\_scores <- matrix('', 0, 5)

scores <- lapply(sentences, function(sentence, vNegTerms, negTerms, posTerms, vPosTerms){

initial\_sentence <- sentence

#remove unnecessary characters and split up by word

sentence = cleancomments(sentence)

sentence <- tolower(sentence)

wordList <- strsplit(sentence, '\\s+')

words <- unlist(wordList)

#build vector with matches between sentence and each category

vPosMatches <- match(words, vPosTerms)

posMatches <- match(words, posTerms)

vNegMatches <- match(words, vNegTerms)

negMatches <- match(words, negTerms)

#sum up number of words in each category

vPosMatches <- sum(!is.na(vPosMatches))

posMatches <- sum(!is.na(posMatches))

vNegMatches <- sum(!is.na(vNegMatches))

negMatches <- sum(!is.na(negMatches))

score <- c(vNegMatches, negMatches, posMatches, vPosMatches)

#add row to scores table

newrow <- c(initial\_sentence, score)

final\_scores <- rbind(final\_scores, newrow)

return(final\_scores)

}, vNegTerms, negTerms, posTerms, vPosTerms)

return(scores)

}

#load pos,neg statements

afinn\_list <- read.delim(file='AFINN-111.txt', header=FALSE, stringsAsFactors=FALSE)

names(afinn\_list) <- c('word', 'score')

summary(afinn\_list$score)

afinn\_list$word <- tolower(afinn\_list$word)

#categorize words as very negative to very positive and add some movie-specific words

vNegTerms <- afinn\_list$word[afinn\_list$score==-5 | afinn\_list$score==-4]

negTerms <- afinn\_list$word[afinn\_list$score==-2 | afinn\_list$score==-1 | afinn\_list$score==-3]

posTerms <- afinn\_list$word[afinn\_list$score==3 | afinn\_list$score==2 | afinn\_list$score==1]

vPosTerms <- afinn\_list$word[afinn\_list$score==5 | afinn\_list$score==4]

#Loop through each row in dataframe and calculate the sentiment

for(i in 1:nrow(Q3csv))

{

rowSenti=as.data.frame(sentimentScore(Q3csv[i,"CommentText"], vNegTerms, negTerms, posTerms, vPosTerms))

totalSenti=((as.numeric(as.character(rowSenti[,2])))\*(-2)) + ((as.numeric(as.character(rowSenti[,3])))\*(-1)) + ((as.numeric(as.character(rowSenti[,4])))\*(1)) + ((as.numeric(as.character(rowSenti[,5])))\*(2))

#print(totalSenti)

if(totalSenti>0)

{

totalSenti=1

}

else if(totalSenti<0)

{

totalSenti=-1

}

Q3csv[i,"SentimentScore"]=totalSenti

if(i%%1000 == 0)

{

cat(i, " Sentiments Analyzed\n")

}

}

attach(Q3csv)

#Selecting columns that would be used for preparing model

Q3df=data.frame(CommentScore, CommentCrDays, CommentLength, comment\_owner\_reputation, comment\_owner\_profile\_summary, comment\_owner\_views, comment\_owner\_upvotes, comment\_owner\_downvotes, comment\_owner\_lastactivity\_days, editDurationAfterCreation, activityDurationAfterCreation, title\_length, num\_tags, PostAnswerCount, num\_favorite, hascode, post\_views, postTypeId, IsAcceptedAnswer, postScore, post\_length, PostCommentCount, PostUpVotes, PostDownVotes, SentimentScore)

Q3df$SentimentScore <- as.factor(Q3df$SentimentScore

str(Q3df)

counts <- ddply(Q3df, .(Q3df$SentimentScore), nrow)

counts

# Subset Selection Methods

# Best Subset Selection

lapply(Q3df["SentimentScore"], unique)

#install.packages("leaps")

library(leaps)

regfit.full=regsubsets(SentimentScore~.,Q3df)

summary(regfit.full)

regfit.full=regsubsets(SentimentScore~.,data=Q3df,nvmax=26)

reg.summary=summary(regfit.full)

names(reg.summary)

reg.summary$rsq

par(mfrow=c(2,2))

plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")

plot(reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted

RSq",type="l")

which.max(reg.summary$adjr2)

points(11,reg.summary$adjr2[11], col="red",cex=2,pch=20)

plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp",type='l')

which.min(reg.summary$cp)

points(10,reg.summary$cp[10],col="red",cex=2,pch=20)

which.min(reg.summary$bic)

plot(reg.summary$bic,xlab="Number of Variables",ylab="BIC",type='l')

points(6,reg.summary$bic[6],col="red",cex=2,pch=20)

coef(regfit.full,6)

# Forward and Backward Stepwise Selection

regfit.fwd=regsubsets(SentimentScore~.,data=Q3df,nvmax=26,method="forward")

summary(regfit.fwd)

regfit.bwd=regsubsets(SentimentScore~.,data=Q3df,nvmax=26,method="backward"

)

summary(regfit.bwd)

coef(regfit.full,7)

coef(regfit.fwd,7)

coef(regfit.bwd,7)

# Choosing Among Models

set.seed(1)

train=sample(c(TRUE,FALSE), nrow(Q3df),rep=TRUE)

test=(!train)

regfit.best=regsubsets(SentimentScore~.,data=Q3df[train,],nvmax=26)

test.mat=model.matrix(SentimentScore~.,data=Q3df[test,])

val.errors=rep(NA,25)

for(i in 1:11){

coefi=coef(regfit.best,id=i)

pred=test.mat[,names(coefi)]%\*%coefi

val.errors[i]=mean((Q2df$accepted\_answer\_flag[test]-pred)^2)

}

val.errors

which.min(val.errors)

coef(regfit.best,24)

predict.regsubsets=function(object,newdata,id,...){

form=as.formula(object$call[[2]])

mat=model.matrix(form,newdata)

coefi=coef(object,id=id)

xvars=names(coefi)

mat[,xvars]%\*%coefi

}

regfit.best=regsubsets(SentimentScore~.,data=Q3df,nvmax=26)

coef(regfit.best,24)

k=24

set.seed(1)

folds=sample(1:k,nrow(Q3df),replace=TRUE)

cv.errors=matrix(NA,k,25, dimnames=list(NULL, paste(1:25)))

for(j in 1:k){

best.fit=regsubsets(SentimentScore~.,data=Q3df[folds!=j,],nvmax=24)

for(i in 1:24){

pred=predict(best.fit,Q3df[folds==j,],id=i)

cv.errors[j,i]=mean( (Q3df$SentimentScore[folds==j]-pred)^2)

}

}

mean.cv.errors=apply(cv.errors,2,mean)

mean.cv.errors

par(mfrow=c(1,1))

plot(mean.cv.errors,type='b')

reg.best=regsubsets(SentimentScore~.,data=Q3df, nvmax=24)

coef(reg.best,3)

# Ridge Regression and LASSO

fix(Q3df)

names(Q3df)

dim(Q3df)

sum(is.na(Q3df$accepted\_answer\_flag))

Q2df=na.omit(Q3df)

dim(Q3df)

sum(is.na(Q3df))

#install.packages("glmnet")

library(glmnet)

#the package invokes inputs and outputs separately unlike lm and glm

x=model.matrix(SentimentScore~.,Q3df)[,-1]

y=Q3df$SentimentScore

# set vector of lambda values to study range from 10^10 to 0.01, total length=100

grid=10^seq(10,-2,length=100)

ridge.mod=glmnet(x,y,alpha=0,lambda=grid)

dim(coef(ridge.mod))

# let us look at a few results here

#first lambda=50

ridge.mod$lambda[50]

coef(ridge.mod)[,50]

sqrt(sum(coef(ridge.mod)[-1,50]^2))

#next, lambda=60

ridge.mod$lambda[60]

coef(ridge.mod)[,60]

sqrt(sum(coef(ridge.mod)[-1,60]^2))

#prediction of the coefficients for lambda=50 (play with this)

predict(ridge.mod,s=500,type="coefficients")[1:25,]

#prepare for training and validation set testing

set.seed(1)

train=sample(1:nrow(x), nrow(x)/2)

test=(-train)

y.test=y[test]

ridge.mod=glmnet(x[train,],y[train],alpha=0,lambda=grid, thresh=1e-12)

ridge.pred=predict(ridge.mod,s=4,newx=x[test,])

#evaluate and compare test MSE and the spread of y.test

mean((ridge.pred-y.test)^2)

mean((mean(y[train])-y.test)^2)

#test wth two other lambdas

ridge.pred=predict(ridge.mod,s=1e10,newx=x[test,])

mean((ridge.pred-y.test)^2)

ridge.pred=predict(ridge.mod,s=0,newx=x[test,],exact=T)

mean((ridge.pred-y.test)^2)

#compare with lm

# The following two are the same

lm(y~x, subset=train)

predict(ridge.mod,s=0,exact=T,type="coefficients")[1:25,]

#Cross validation to get the best lambda

set.seed(1)

cv.out=cv.glmnet(x[train,],y[train],alpha=0)

plot(cv.out)

bestlam=cv.out$lambda.min

bestlam

#now predict with the best lambda

ridge.pred=predict(ridge.mod,s=bestlam,newx=x[test,])

mean((ridge.pred-y.test)^2)

out=glmnet(x,y,alpha=0)

predict(out,type="coefficients",s=bestlam)[1:25,]

# Lasso

#only difference in model building is to use alpha=1

lasso.mod=glmnet(x[train,],y[train],alpha=1,lambda=grid)

plot(lasso.mod)

# use CV to get best lambda

set.seed(1)

cv.out=cv.glmnet(x[train,],y[train],alpha=1)

plot(cv.out)

bestlam=cv.out$lambda.min

#use best lambda for prediction

lasso.pred=predict(lasso.mod,s=bestlam,newx=x[test,])

mean((lasso.pred-y.test)^2)

out=glmnet(x,y,alpha=1,lambda=grid)

lasso.coef=predict(out,type="coefficients",s=bestlam)[1:25,]

lasso.coef

lasso.coef[lasso.coef!=0]

#Dimensionality Reduction PCA:

#install.packages("pls")

library(pls)

set.seed(2)

pcr.fit=pcr(SentimentScore~., data=Q3df,scale=TRUE,validation="CV")

summary(pcr.fit)

validationplot(pcr.fit,val.type="MSEP")

set.seed(1)

pcr.fit=pcr(SentimentScore~., data=Q3df,subset=train,scale=TRUE, validation="CV")

validationplot(pcr.fit,val.type="MSEP")

pcr.pred=predict(pcr.fit,x[test,],ncomp=7)

mean((pcr.pred-y.test)^2)

pcr.fit=pcr(y~x,scale=TRUE,ncomp=7)

summary(pcr.fit)

# Partial Least Squares

set.seed(1)

pls.fit=plsr(SentimentScore~., data=Q3df,subset=train,scale=TRUE, validation="CV")

summary(pls.fit)

validationplot(pls.fit,val.type="MSEP")

pls.pred=predict(pls.fit,x[test,],ncomp=2)

mean((pls.pred-y.test)^2)

pls.fit=plsr(SentimentScore~., data=Q3df,scale=TRUE,ncomp=2)

summary(pls.fit)

#Dividing dataset 75% for training and 25% for testing

Q3smp\_size=floor(0.75\*nrow(Q3df))

set.seed(123)

Q3train\_ind=sample(seq\_len(nrow(Q3df)), size = Q3smp\_size)

Q3Train=Q3df[Q3train\_ind,]

Q3Test=Q3df[-Q3train\_ind,]

nrow(Q3Train)

nrow(Q3Test)

#Initial logistic regression model with all parameters

attach(Q3Train)

Q3lm=glm(as.factor(SentimentScore)~.,data=Q3Train,family="binomial")

summary(Q3lm)

#Checking for significant parameters with forward step

Q3modelstepforward=step(Q3lm, direction="forward")

summary(Q3modelstepforward)

#Checking for significant parameters with backward step

Q3modelstepbackward=step(Q3lm, direction="backward")

summary(Q3modelstepbackward)

#Removing insignificant parameters and remodelling

Q3lm2=glm(as.factor(SentimentScore)~CommentLength+comment\_owner\_profile\_summary+comment\_owner\_downvotes+num\_tags+ +PostAnswerCount+num\_favorite+IsAcceptedAnswer+PostCommentCount,data=Q3Train,family="binomial")

summary(Q3lm2)

#Again removing insignificant parameters and remodelling

Q3lm3=glm(as.factor(SentimentScore)~CommentLength+comment\_owner\_downvotes+num\_tags+PostAnswerCount+IsAcceptedAnswer+PostCommentCount,data=Q3Train,family="binomial")

summary(Q3lm3)

BIC(Q3lm3)

#Predicting based on the logistic model built

logistic\_probs=predict(Q3lm3, Q3Train, type="response")

head(logistic\_probs)

testing\_y=Q3Test$SentimentScore

logistic\_pred\_y=rep(-1,length(testing\_y))

logistic\_pred\_y[logistic\_probs>0.8]=0

logistic\_pred\_y[logistic\_probs>0.9]=1

training\_y=Q3Train$SentimentScore

table(logistic\_pred\_y,training\_y)

mean(logistic\_pred\_y!=training\_y,na.rm=TRUE)

logistic\_probs=predict(Q3lm3, Q3Test, type="response")

head(logistic\_probs)

logistic\_pred\_y=rep(-1,length(testing\_y))

logistic\_pred\_y[logistic\_probs>0.8]=0

logistic\_pred\_y[logistic\_probs>0.9]=1

table(logistic\_pred\_y,testing\_y)

mean(logistic\_pred\_y!=testing\_y,na.rm=TRUE)

#Cross Validation for logistic regression

#K-fold

library(boot)

MSE\_10\_Fold\_CV=cv.glm(Q3Train,Q3lm3,K=10)$delta[1]

MSE\_10\_Fold\_CV

MSE\_10\_Fold\_CV=NULL

for(i in 1:10){ model=glm(SentimentScore~CommentLength+comment\_owner\_downvotes+num\_tags+PostAnswerCount+IsAcceptedAnswer+PostCommentCount,data=Q3Train)

MSE\_10\_Fold\_CV[i]=cv.glm(Q3Train,model,K=10)$delta[1]

}

MSE\_10\_Fold\_CV

#ROC of Logistic Regression

#install.packages("ROCR")

library(ROCR)

ROCRpred<- prediction(logistic\_probs,testing\_y)

ROCRperf<- performance(ROCRpred,"tpr","fpr")

plot(ROCRperf)

plot(ROCRperf,colorize=TRUE)

plot(ROCRperf, colorize = TRUE, print.cutoffs.at = seq(0,1,0.05), text.adj = c(-0.2,1.7))

as.numeric(performance(ROCRpred,"auc")@y.values)

## Appendix D: Terminology and Abbreviations

|  |  |  |
| --- | --- | --- |
| **Term** | **:** | **Meaning** |
|  |  |  |
| User | : | Community members who have a registered account to use Stack Overflow, existing or deleted. |
| Post | : | Question or answer, both are considered posts in context of SO. |
| OP | : | Original Poster; User who post the question. |
| SO | : | Abbr. of Stack Overflow. |
| Comment | : | Messages, observations etc. left by users on posts. |
| Reputation | : | Points system on SO. Users are awarded or penalised based on the policy of Stack Overflow. (http://stackoverflow.com/help/whats-reputation). |
| Vote | : | An expression of accepting or rejecting a post. Voting up or down has a direct consequence on the reputation. |
| Flag | : | An extreme measure taken by user to report failure of adherence to SO standards of posting question, answers or comments. Flagged items are sent to moderators for review. |
| Privileges | : | Access to various controls as a user. Increases based on reputation. |
| Moderator | : | A privilege awarded to users with 10000 reputation points. |
| Stack Exchange | : | Network of 150+ community based Q&A website |
| Data Explorer | : | Online tool which provides query interface to mine data for the websites which are a part of StackExchange network(http://data.stackexchange.com/) |