PREDICTING CLOSED VERSUS OPEN QUESTIONS USING MACHINE LEARNING FOR IMPROVING COMMUNITY QUESTION ANSWERING WEBSITES

by

Pradeep Kumar Makkena

Submitted in Partial Fulfilment of Requirements

For the Degree of

Master of Computing and Information Systems

YOUNGSTOWN STATE UNIVERSITY

December 2017

Pradeep Kumar Makkena

I hereby release this thesis to the public. I understand that this will be made available from Ohio LINK ETD Centre and the Maag Library Circulation Desk for public access. I also authorize the university or other individuals to make copies of this thesis as needed for scholarly research.

Signature:

Pradeep Kumar Makkena, Student Date

Approvals:

Dr. Alina Lazar, Thesis Advisor Date

Dr. Yong Zhang, Committee Member Date

Dr. Feng Yu, Committee Member Date

Dr. Salvatore A. Sanders, Associate Dean of Graduate Studies Date

ABSTRACT

Community question answer (CQA) websites add great value to the information available on the Web and they have been gaining popularity in the past few years. Popular CQA websites have millions of users asking thousands of questions every day. To maintain the quality of content, site moderators monitor and close the questions which do not follow the community guidelines. Stack Overflow is a very popular CQA website for programmers with more than 8 million users. Everyday thousands of questions are posted in Stack Overflow and some of these questions do not follow the community guidelines and they will be closed by the moderators. Manual moderation of questions is a tedious task because of the sheer volume of the questions. The main purpose of this thesis is to build a machine learning classifier to predict whether or not a question will be closed. Using various post features, comment features and user features from Stack Overflow data, machine learning models were created using various classification algorithms. Apache Spark machine learning library was used to train and test these models and we found that model trained with random forest algorithm gave the best results with an accuracy of 77.60%. We found that features derived from the body of the post contribute more to the accuracy of the model whereas features derived from the users table contribute less.

Acknowledgements

Firstly, I would like to express my true thankfulness to my advisor, Dr. Alina Lazar. I feel very honoured to work with her and I am grateful for her support, patience and knowledge shared to complete my research. I appreciate her for giving me an opportunity to work with her and I could not have imagined having a better advisor for my Master's.

I would like to thank my committee members Dr. Yong Zhang, Dr. Feng Yu for taking time out of their schedule to share their Knowledge and Insights.

A special thanks to my family and friends for the support they have given to me during my graduate career. I would also thank the Department of Computer Science and Information Systems and the College of Graduate Studies for the financial assistance during my graduate studies.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

The Internet has changed the way people search and share information. If we want to find information online, we usually ask search engines what we need with key words and the search engines provide us relevant webpages. We can browse through these pages and find what we need. However, sometimes we cannot find exact answer and need to spend lot of time browsing through all the pages. Community question answer websites (CQA) provide efficient way to get the answers we want and saves lot of time. CQA websites provide a platform to exchange and share the knowledge. You ask a question in community question answer platform and very soon you will get the answer or opinion of the people who know about that question. More often they will solve your problem and even if they do not solve your problem you are going to provide lot of information about what you are looking for, which in turn saves lot of time. We can also share our knowledge and answer questions, to contribute something back to the community.

The idea of receiving a direct response to a question sounds very appealing, but CQA websites also involve risk regarding the quality of the information provided. There are large number of question answering websites, it is important for question answer website to maintain and provide high-quality content to distinguish itself from other websites. If the website has high quality content more users are going to use it, which in turn increases the website popularity. The importance of high-quality content in community question and answer websites has been recognized and investigated in several studies. CQA sites need to moderate and remove the low-quality content in order to maintain the quality of content. Popular CQA websites have millions of users asking thousands of questions every day, so it

will be difficult and tedious task for the moderators to go through each and every post and remove the posts with low quality.

1.1 Motivation

Stack Overflow is a popular community question answer website for programmers and millions of programmers use it every day. Stack Overflow has more than 8 million registered users asking thousands of questions every day. Stack Overflow moderators can close the questions for various reasons to maintain the quality of content. Moderators can close the questions for following reasons off topic, not constructive, duplicate, too localized, not a real question. Any user who has enough site reputation can flag the questions in one of these categories. The primary responsibility of moderators is go through these flagged posts and take required action. As of August 2017, Stack Overflow has 24 moderators and it is a tedious work for them to moderate such a large website. Stack Overflow has more than 15 million posts and it is a very difficult task for the site moderators to maintain the quality of content and close the posts which do not follow the community guide lines.

We approached this problem by building a machine learning classifier which can automatically predict the closed questions based on historical data. Stack Exchange data dump is made available to public and it contains data about the posts, users, comments, votes and tags in Stack Overflow. This dataset is very big and popular and lot researchers use it for data mining and machine learning research. The goal of this thesis is to build binary machine learning classifier for Stack Overflow site which can predict whether or not a question will be closed. We compute various features from Stack Overflow datasets including text, post, user, and comments related features. These features are then used to build a machine learning classifier which can predict the closed questions on stack overflow. We have included the user features also because user who repeatedly asks good questions is less likely to ask a

question which will be closed. Comments features were used because user's comments contribute to the question being closed. The Apache Spark Machine Learning Library (MLlib) was used to train and test the classifier. Apache Spark is a popular distributed cloud computing platform that can handle large amounts of data very efficiently.

1.2 Organization

This thesis is organized as follows. The next chapter presents the related work and Chapter 3 explains dataset preparation and feature preparation from the dataset. Chapter 4 describes the machine learning algorithms used to build the classifiers. Chapter 5 presents the experiments and the results. Finally, Chapter 6 concludes the thesis.

CHAPTER 2
RELATED WORK

The popularity of question and answer websites is increasing gradually, and very large amounts of information is available in these websites. Therefore, the analysis of community question and answer websites has become the subject for various studies. Stack Overflow datasets are very large, and they are available to public, so lot of researchers used these datasets for their research. In this chapter we will briefly explain the work related to improving quality of question and answer websites, and predicting closed questions on Stack Overflow.

Bauchuain Li et al analysed the factors of question quality and found that the interaction between askers and topics results in the differences of question quality. Based on this finding they proposed a mutual reinforcement based label propagation algorithm to predict quality of the question. They tried to predict the quality of question before anyone answered the question. They used various features related to the content of the question and features related to the user who asked the question. They found that by including user related features the accuracy improved substantially. They concluded that the proposed algorithm gives better results than the other popular machine learning algorithms like logistic regression, gradient boosted tree.

Lezina et al built a classifier for Stack Overflow website to predict if a question will be closed or not along with the reason the question was closed. They used the dataset and baseline model provided for the Kaggle's "Predict Closed Questions on Stack Overflow" competition. Many different machine learning algorithms, like random forest and support vector machines and online gradient descent from the open source machine learning library Vowpal Wabbit to build the classifier. Various features related to the post and features related

to user who posted the question were included. They have selected the most important features by constructing large number of trees for randomly selected subset of features. They have concluded that text features contribute much more to the result than user features and the model using online gradient descent algorithm from Vowpal Wabbit provides the best results.

Correa et al analysed Stack Overflow datasets and built a predictive model to identify closed question at the time of post creation. They found that despite being marked as closed, subjective questions contain high information value and are very popular with users. They also found the decrease in community participation to mark a closed question which led to increase in moderator's time to identify low quality questions. They also found that closed questions marked with off topic and duplicate labels are more prone to reputation gaming. Using only post and user related features they have built a classifier which can predict closed questions with an accuracy of 73%. Using feature analysis, they have found that Stack Overflow urls and code snippet length are the most differentiating features to predict a closed question on Stack Overflow.

Duijn et al analysed the Stack Overflow dataset and they have tried to improve the classification of Stack Overflow questions using analysis of code samples, by providing an insight into most relevant code metrics for determining the quality of a question. They have found that most important features related to the code quality are length of the code, number of white spaces in the code and number of formatting errors in the code. They found that random forest algorithm gives best results to detect low quality questions at the moment they are posted.

Balatdzhieva et al conducted a survey on question quality in community question answer websites. They divided the features which have influence on question quality in to two

groups. The first group contains question related features like tags, body length, code snippet and the second group include asker related features like user reputation. They found that number of answer received, question score are the good measures of question quality. They found that the influence of question length is mixed on question quality on the other hand, the presence of an example has a positive effect on question score and number of answers.

In this thesis, we have used features derived from the text, code parts of the question. We calculated various text features, TF-IDF features from the text part of the question. We have included the user related features and comment related features. We have trained our dataset using different machine learning algorithms like logistic regression, random forests and support vector machines. We have used Apache Spark machine learning library to train our models using these algorithms.

CHAPTER 3
DATASET PREPARATION AND FEATURE EXTRACTION

The dataset we have used for this thesis is downloaded from Stack Exchange data dump. Stack Exchange is a network of question and answer websites consists of 133 question and answer communities including stack overflow. All user content contributed to the Stack Exchange network is cc-by-sa 3.0 licensed, intended to be shared and used by the public. Stack Exchange data dump is an anonymized dump of all user contributed content on the Stack Exchange network

3.1 Dataset Preparation

All the data about posts, users, tags and all the posts related activities of stack overflow website are stored in different database tables such as Posts, Tags, Comments, Badges, PostHistory, PostLinks, Users, Votes. We have downloaded three tables, first table is posts.xml, users.xml and comments.xml. The Posts table contains data about the all the posts in Stack Overflow. The second table is users.xml contains data about all the active users of stack overflow website. Finally, the comments.xml contains data about all the user comments. All the tables are in xml format. The Posts.xml, Users.xml and Comments.xml files were loaded in a local MySQL database using the built-in MySQL xml reader. Table 1 gives brief description about all the fields in the users table and table 2 explains all the fields in posts table.

Posts.xml file is very big with a size of 54GB and it contains data about more than 35 million posts. Due to the limitations of computing resources we selected to analyse a dataset containing data about 100,000 posts. Stack Overflow Stack Overflow has 636232 closed question out of 14.45 million total questions, which means 4.4%. We want our dataset to be

balanced so we filtered the posts table so that it contains 70,000 open posts and 30,000 closed posts. We also made sure that all these posts are questions. Because the posts table contains both questions and answers as individual records, the answers were filtered out. Finally, we performed inner join operation on posts, users and comments table to merge the three tables in one table.

The final table has data about 100,000 questions including the data about the user who asked the question and comments for that question. The final dataset has all the fields from the users, posts tables and one field from the comments table with text from all the comments for that question.

Table 1. User table fields and their Description

Data Field Name	Type	Description
Id	Int	UserId of the person
Reputation	Int	Reputation of the user in the stack the website
CreationDate	Int	Date and time when the user first signed up
DisplayName	Text	Display name chosen by the user
LastAccessDate	Datetime	Date and time when user last opened the website
websiteURL	Text	Website url filled by the user
Location	Text	Location filled by the user
Age	Int	Age filled by the user
AboutMe	Text	Brief description about the user filled by the user
Views	Int	Number of views received by the user profile
UpVotes	Int	Total number of upvotes received by the user
DownVotes	Int	Total number of downvoted received by the user
AccountId	Int	Stack Exchange network profile id of the user

Table 2. Posts table fields and their description

Data Field Name	Type	Description
Id	Int	Unique Id of the post
PostTypeId	Int	1 if the post is a question else 2
ParentId	Int	If the post is an answer it will give user Id of the author who posted that answer
AcceptedAnswerId	Int	postId of the accepted answer for question
CreationDate	Datetime	Date and time, when the post created
Score	Int	Score for the given post.

ViewCount	Int	Total number of views for the post
Body	Text	Content of the post
OwnerUserId	Int	UserId of the author of the post
OwnerDisplayName	Text	Display name of the author of the post
LastEditorUserId	Int	User Id of the person last edited.
LastEditorDisplayName	Text	Display name of the last edited person
LastEditDate	Datetime	Data and time when the post last edited
LastActivityDate	Datetime	Date and time when there is last activity on a post
CommunityOwnedDate	Datetime	If the post is owned by the community, then date and time when that happened
Title	Text	Title of the post given by the author
Tags	Text	Tags given for the post by the author
AnswerCount	Int	Number of answers for the given post
CommentCount	Int	Number of comments for the post
FavoriteCount	Int	Number of people marked the post as favourite
ClosedDate	Datetime	If the post is closed, then it will give date and time when the post is closed.

3.2 Feature Extraction

Feature extraction is the process of getting derived features from the dataset which gives us insight in to the dataset and helps us to build a good predictive model. We need to select the features in the dataset that are most useful or most relevant for predicting closed questions on Stack Overflow. Feature selection helps us to create a predictive model with good accuracy. Feature selection removes irrelevant and redundant attributes from the dataset that do not contribute to the accuracy of the model. Fewer attributes are desirable because it reduces the complexity of the model, and if the model is simple it is going to be easy to explain and understand.

From the dataset we have selected the most relevant features leaving out the attributes which are not helpful for predicting closed question. We have also derived many features using the data from original dataset which will give good amount of information and helps us to build a model with good accuracy. Table 3 gives a brief description about the derived features and table 4 lists the features selected from the original dataset.

The goal is to predict closed questions on Stack Overflow and we derive a lot of useful features from the content of the post. Most of the questions in Stack Overflow have code associated with them. So, first we parsed the body of the post and divided it in to text and code components. The text components contain all the sentences from the post and the code components contain all the code blocks. From the text component we derived features like length, first sentences, last sentence, number of question marks and punctuation marks. From the code component we derived the features like number of code blocks, length, and first code block.

Table 3. Features derived from the dataset

Feature Name	Description
FirstCodeLen	Length of the first code block. If there is no code block, then 0
FirstSenLen	Length of the first sentence
LastCodeLen	Last code block length. If there is one or no code block, then 0
LastSenLen	Length of the last sentence
CodeTextRatio	Ratio of length of code to length of text
FirstCodeSenRatio	Ratio of length of first sentence to first code block
LastCodeSenRatio	Ratio of length of last sentence to last code block
NoOfTags	Total number of tags given to the post by the author
TextLength	Length of the text component of the post
CodeLength	Length of the code component of the post
IDFfeatures	Text features obtained using tf-idf feature extractor
TitleLength	Length of the title
NoOfI	Number of occurrences of the word “I”
NoOfYou	Number of occurrences of the word “you”
NoOfQuestionMark	Number of occurrences of the “?”
NoOfExclamationMark	Number of occurrences of the word “!”
NoOfHelp	Number of occurrences of the word “help”
NoOfPls	Number of occurrences of the word “please”
NoOfInternet	Number of occurrences of the word “internet”
NoOfFind	Number of occurrences of the word “find”
NoOfInterWords	Number of interrogative words like when, why, how, what
CodeSente	Number of lines of code
TextSente	Number of sentences in the text component of the post
AgeFilled	If the user filled their age, then 1 otherwise 0
AboutMeFilled	If the user filled about me section, then 1 otherwise 0
LocationFilled	If the user filled their location, then 1 otherwise 0
NoOfDuplicate	Number of occurrences of word “duplicate” in comments
NoOfThanks	Number of occurrences of word “thanks” in comments
SOUrl	Number of occurrences of word “stack overflow” in comments
NoOfAns	Number of occurrences of word “answer” in comments
NoOfIntWords	Number of occurrences of interrogative words in comments
NoOfSorry	Number of occurrences of word “sorry” in comments

NoOfRecc	Number of occurrences of word “recommend” in comments
Comments_len	Length of the all the comments combined

Table 4. Features taken from the dataset

Feature Name	Description
Score	Score given for the post
Views	Number of profile views received by the author of the post
ViewCount	Number of views received by the post
AnswerCount	Number of answers received by the post
FavoriteCount	Number of people marked this post as their favourite
CommeentCount	Number of comments received by the posts
Reputation	Reputation of the author in the stack overflow
UpVotes	Total Number of upvotes received by the author of the post
DownVotes	Total number of downvotes received by the author of the post

CHAPTER 4
MACHINE LEARNING ALGORITHMS

This chapter briefly explains the machine learning algorithms and the tools we have used run these machine learning algorithms on our dataset. We have used logistic regression, random forest, support vector machine algorithms to train our model and Apache Spark Machine Learning Library (MLlib) to run these algorithms.

4.1 Logistic Regression

Logistic regression is a form of a regression analysis and it can be used for binary classification problems in machine learning. Binary logistic regression is applied when the predicted variable can only have two possible outcomes like closed or open, positive or negative.

The problem we are dealing with is predicting whether a question is closed or not. The outcome only has two values, so we have used binary logistic regression. In this thesis we have used 1 for closed question and 0 for open question. This means if the outcome is 1 then the model predicted it as a closed question and if the outcome is 0 then the model predicted it as a closed question.

Logistic regression is robust, popular and widely used in binary classification problems. Logistic regression uses a statistical function named logistic function. Logistic function is S-shaped curve which will take real value numbers as an input and constraints the output probability between 0 and 1. A typical representation of logistical regression is as shown in the below picture

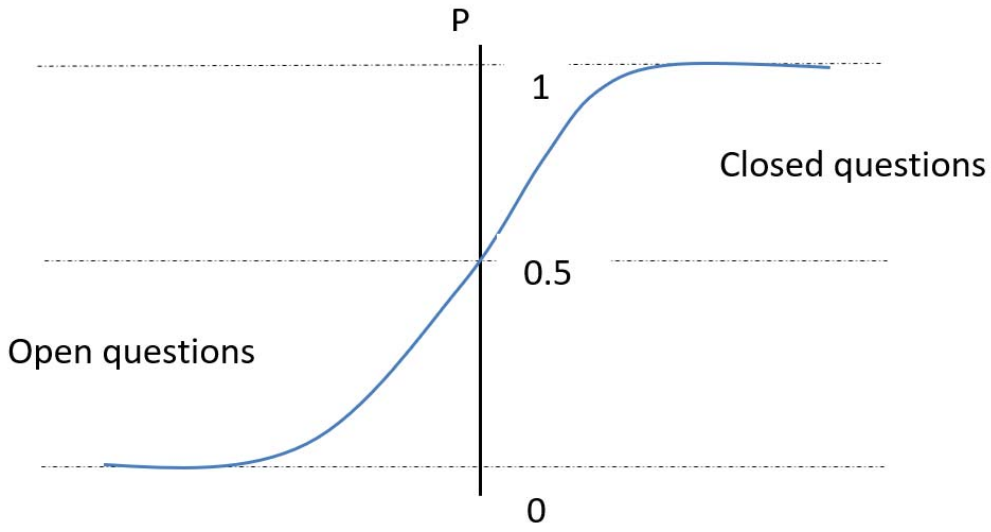

Fig:1 Logistic Regression Classification

From the figure we can see that if the probability P for a post is greater than 0.5 it is predicted as closed question. If the probability is less than 0.5 then it is being predicted as a closed question. The probability P is calculated by below equation.

$$P = e^{(b0 + b1*x)} / (1 + e^{(b0 + b1*x)}) \qquad 4.1$$

In the above equation 4.1 P is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the input feature. Every input feature has its coefficient and as the number of features increases the complexity of the model also increases. When we train the model using training dataset, all these coefficients are calculated to fit the training data. When the model is finished the training, intercept and all the coefficients are stored, and they will represent the model. Whenever a test case is given, the model will calculate the probability using the intercept and coefficients. If the calculated probability is greater than 0.5 then it will predict the post as a closed question.

4.2 Random Forest

Random forest algorithm is a supervised classification algorithm and it can be used for binary classification problems. Random forest creates many classification trees by taking random samples from the input training data. In random forest algorithm the accuracy of the model depends upon the number classification trees in the model. If the number of trees increase the accuracy of the model also increases. To reduce the error rate in random forest algorithm there should less correlation between the trees in the forest and the strength of each individual tree should be higher.

In random forests algorithm If the total number of features we are using are *N*, then from those total *N* features, *n* features are selected randomly, and a classification tree is built using these *n* features. During the whole training process *n* remains constant. Increasing the value of *n* will increase both correlation and strength of the tree and decreasing the value of *n* will reduce both correlation and strength. So, we should choose an optimal value for the *n*. Random forest and Decision tree algorithms are very similar except that in random forest algorithm the process of selecting the root node and splitting the feature nodes is random.

The advantage with random forest algorithm is it can handle the missing values and if there are sufficient number of trees are available then random forest algorithm does not over fit the model. Random forests algorithm can handle thousands of features and it can handle large amounts of data effectively.

Random forests algorithm begins with selecting *n* features out of total *N* features. After that using the features it has selected it will choose a root node using best split approach. After selecting root node, the daughter nodes will be calculated using same best split approach. Then we repeat the above stages to create other trees in random forests.

After the taring finishes the trees are saved for future use. When a test case is given to random forests model to perform the prediction we need to pass all the test features through the rules created by every tree. These trees will give different results depending upon the possible outcome and the result is the one which is predicted by most number of tress.

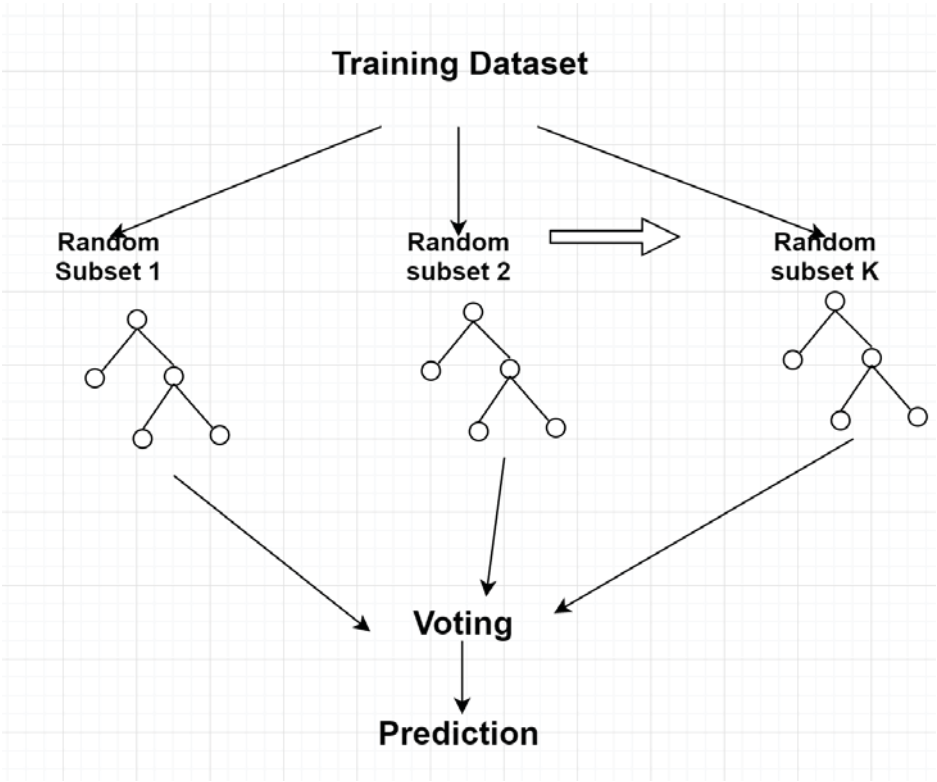

Fig 2: Random Forests Simplified

4.3 Support Vector Machines

Support vector machines (SVM) are supervised classification algorithm and it can be used for binary classification problems. The binary classification problems have two possible outcomes only and SVM model build a hyper plane which can separate the two possible outcomes. SVM converts all the training examples into mapped points and find the best hyper plane which separates the two categories with largest margin.

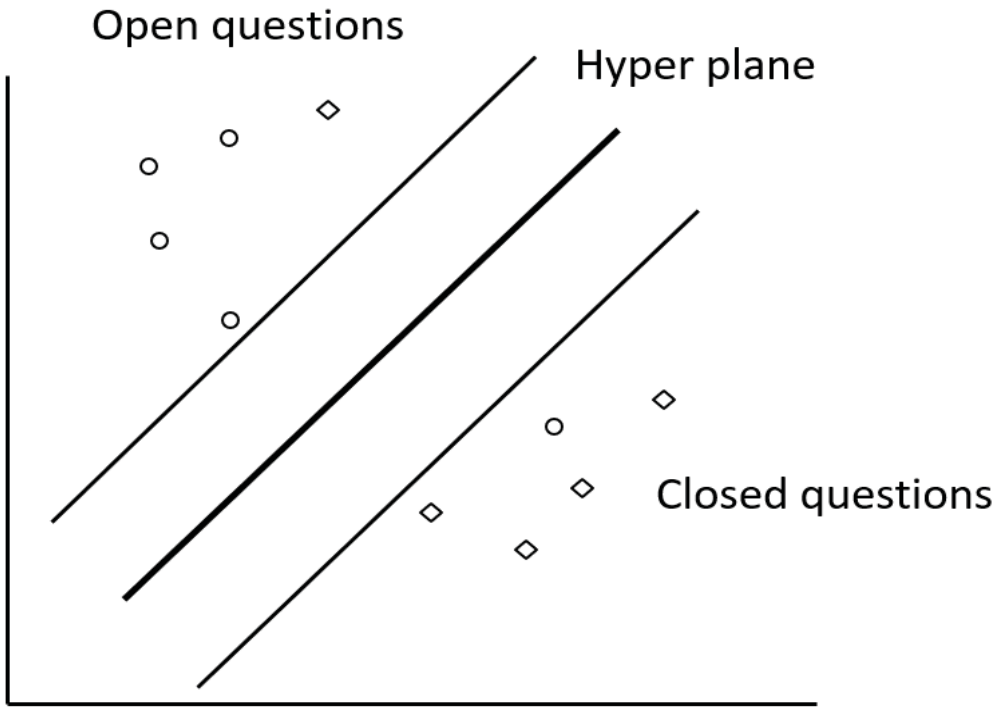

Fig 3: Support Vector Machines Classification

From the figure 3 we can see that there are many hyper planes that can separate the closed questions and open questions. However, just by looking at the picture we can say the that the thick middle line the best hyper plane that separates the two groups with maximum margin. The goal of the SVM algorithm is to find that hyperplane which will fit the dataset with maximum margin. Margin is the defined as the maximum distance between the hyper plane and closest point in the dataset. Maximizing the margin is good because data points near the hyper plane are more likely to be classified wrongs. So, we want as less data points as possible nearer to the hyper plane. If the margin high the accuracy of model improves, and the classifier will make more confident decisions. From this we can say that large margin is important and once we decided about margin value SVM model will try to find the hyper plane that will fit the training set with a maximum margin.

4.4 Apache Spark Machine Learning Library (MLlib)

We have trained our dataset with logistic regression, random forest, support vector machine algorithms using Apache Spark Machine Learning Library. Apache Spark is a powerful open source processing engine built around speed, ease of use, and sophisticated analytics. Apache Spark is the largest open source project in data processing and it is used by enterprises across a wide range of industries. Apache Spark proved to out-perform the previously popular big data platform Hadoop by scale of 10 to 100 times.

Apache Spark has an advanced Directed Acyclic Graph execution engine that supports acyclic data flow and in-memory computing. Apache Spark supports lazy evaluation. Lazy evaluation is the method of analysing the data only when the there is a need. The functions in spark are termed as Actions and Transformations. In Spark because of lazy evaluation, whenever a transformation is called upon, the data is read and stored. No further processing is done until an Action is called upon the data. Because of these two features in-memory computing and Lazy evaluation spark is gaining lot of popularity.

Apache spark powers a stack of libraries including SQL, Data Frames, MLlib for machine learning, GraphX, and Spark Streaming. You can combine these libraries seamlessly in the same application. Apache Spark runs on top of Hadoop, Mesos, standalone, or in the cloud. It can access diverse data sources including HDFS, Cassandra, HBase, and S3. You can use spark interactively from the Scala, Python and R shells.

In this thesis we have used the Apache spark version 2.2.0 in standalone mode and we have used JDBC connector to get the data from MySQL. We have used the Scala interface for this thesis. We used different libraries in Apache Spark like Spark-SQL, Spark MLlib, DataFrames.

CHAPTER 5
EXPERIEMENTS AND RESULTS

The final dataset which is prepared by joining the posts, users, comments table is available in my local MySQL database. Using JDBC connector we have loaded that dataset into standalone Apache Spark environment. We have used several user defined function (UDF), Spark in-built function to compute the derived features. Apache Spark MLlib provides feature extractors, feature transformers and we have used some of them. We have used TF-IDF on the text part of the question to get tf-idf features. TF-IDF which is a feature vectorization method widely used in text mining to reflect the importance of a term to a document in the corpus.

We also used feature transformers like tokenizer to split the text in to words, and normalizer to standardize all the features between 0 to1. We also used label indexer to index the output column which is being predicted. For random forest algorithm we have used string indexer because we need to index the features also. We have used vector assembler to bring together all the features and form a feature vector. Vector assembler takes all the input columns and returns a single vector which contains all the features. We can use the feature vector from vector assembler to train and test the machine learning algorithms.

After assembling all the features using the vector assembler we have split dataset randomly. 70 percent of dataset is used to train the model using machine learning algorithms and remaining 30 percent of data is used to test the model. We have trained a binary classification model using logistic regression, random forest and support vector machines algorithms. We have tested these models using test data and calculated accuracy using binary classification evaluator. You can see the results in the table 5.

Table 5. Model Accuracy

Algorithm Used	Accuracy of the Model using all the features
Logistic Regression	77.40
Random Forest	77.60
Support Vector Machines	66.32

As you can see from the above table model using the random forest algorithm gave the best results followed by the model using the logistic regression.

We wanted to see how different features like text features, user features, comment features and TF-IDF features contribute to the model and their effects the accuracy. Table 6 gives the details of the features used and their corresponding accuracy. We trained all the below machine learning models using logistic regression algorithm.

Table 6. Model accuracy using different feature sets

Features used for the model	Accuracy
pf+tcf+cf+uf+tf-idf (all features)	77.40
pf+ tcf	75.43
pf+tf-idf	71.41
pf+cf	70.34
pf+uf	68.76
Pf	68.73

pf => features taken from the posts table fields

tcf=> text, code features derived from the body of the post

uf=>features taken or derived from the user table fields

cf=>features taken or derived from the comments table fields

tf-idf=> features from TF-IDF feature extractor for the text part of the question

As you can see from the table when we use features taken from posts table fields the accuracy of the model is 68%. When we added the user features the improvement in the model

accuracy is negligible. From this we can conclude that the user features we have added does not contribute much to the accuracy of the model.

When we added the comment features to the post features the accuracy improved by 1.6%. When we added the features from tf-idf feature extractor to the post features the accuracy of the model is improved by 2.6%.

When we added the features derived from body of the post to the post features the accuracy of the model is improved by 6.7%. From this we can conclude that out of all feature sets, text features contribute more to the accuracy of the model. When we used all the features model has an accuracy of 77.40%.

CHAPTER 6
CONCLUSION

Stack Overflow is very popular question and answer website used by millions of programmers asking thousands of questions every day. To maintain the quality of content, Stack Overflow has some guide lines and the posts which do not follow those guide lines will be closed. A study was conducted to build a machine learning classifier which can predict whether a question will be closed or not. Model trained with random forest algorithm gave the best results with an accuracy of 77.60%.

We also tested for importance of different feature groups like text, code features, user features, comment features, TF-IDF features. We found that features derived from the body of the post contribute more to the accuracy of the model whereas features derived from the users table contribute less. We obtained best accuracy for the model when we used all the features.

From the manual analysis of Stack Overflow website, we found that some of the closed questions are were very popular and receiving lot of attention from the community even though they are closed whereas some of the open questions should actually be closed. This is where user comments, user interaction features will be helpful. We have used some features related to the user, comments but it is possible to improve the model by including the more features.

REFERENCES

[1] E. L. cG Galina and A. M. Kuznetsov, “Predict Closed Questions on StackOverflow.”

[2] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne, “Finding high-quality content in social media,” in *Proceedings of the 2008 international conference on web search and data mining*, 2008, pp. 1

[3] L. GALINA, K. ARTEM, and B. PAVEL, “Learning to predict closed questions on stack overflow,” *Ученые записки Казанского университета. Серия Физико-математические науки*, vol. 155, no. 4, 2013.83–194.

[4] C. Treude and M. P. Robillard, “Understanding Stack Overflow Code Fragments,” in *Software Maintenance and Evolution (ICSME), 2017 IEEE International Conference on*, 2017, pp. 509–513.

[5] B. Li, T. Jin, M. R. Lyu, I. King, and B. Mak, “Analyzing and predicting question quality in community question answering services,” in *Proceedings of the 21st International Conference on World Wide Web*, 2012, pp. 775–782.

[6] A. Baltadzhieva and G. Chrupala, “Question quality in community question answering forums: a survey,” *Acm Sigkdd Explorations Newsletter*, vol. 17, no. 1, pp. 8–13, 2015.

[7] D. Correa and A. Sureka, “Fit or unfit: analysis and prediction of ‘closed questions’ on stack overflow,” 2013, pp. 201–212.

[8] M. Duijn, A. Kučera, and A. Bacchelli, “Quality questions need quality code: Classifying code fragments on stack overflow,” in *Proceedings of the 12th Working Conference on Mining Software Repositories*, 2015, pp. 410–413.

[9] Kaggle Stack Overflow competition ranked 10th on leader board.

https://github.com/saffsd/kaggle-stackoverflow2012

[10] What is a day in life of a stack overflow moderator?

http://meta.stackoverflow.com/a/166630/214223, February 2013

[11] Stack exchange data dump. http://www.clearbits.net/torrents/2076-aug-2012, August 2012.

[12] Working of Random Forest Algorithm

https://medium.com/@Synced/how-random-forest-algorithm-works-in-machine-learning-3c0fe15b6674.

[13] C.-Y. J. Peng, K. L. Lee, and G. M. Ingersoll, "An introduction to logistic regression analysis and reporting," *The journal of educational research*, vol. 96, no. 1, pp. 3–14, 2002.

[14] J. Ali, R. Khan, N. Ahmad, and I. Maqsood, "Random forests and decision trees," *IJCSI International Journal of Computer Science Issues*, vol. 9, no. 5, pp. 272–278, 2012.

CODE

```
import org.apache.spark.ml
import org.apache.spark.ml.linalg.Vectors
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.feature.Normalizer
import org.apache.spark.ml.classification.LogisticRegression
import org.apache.spark.mllib.util.MLUtils
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.ml.evaluation.BinaryClassificationEvaluator
import org.apache.spark.ml.feature.StringIndexer
import scala.xml._
import org.apache.spark.ml.feature.{HashingTF, IDF, Tokenizer}
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.classification.{RandomForestClassificationModel,
RandomForestClassifier}
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.ml.feature.{IndexToString, StringIndexer,
VectorIndexer}
import org.apache.spark.ml.classification.LinearSVC
//imported the table from mysql
val posts = spark.read.format("jdbc").option("driver",
"com.mysql.jdbc.Driver").option("url","jdbc:mysql://localhost:3307/posts?ve
rifyServerCertificate=true&useSSL=true").option("dbtable",
"postsuserscomments").option("user", "root").option("password",
"deepu.555$").load()
//filled the null values with 0 in integer fields
val posts1=posts.na.fill(0)
//filled the null values string fields with 0
val posts2= posts1.na.fill("0")
//user defined functions to create a label based on closed date
val label:java.sql.Timestamp=>Int=(arg:java.sql.Timestamp)=>{if(arg==null)
0 else 1}
val labeludf = udf(label)
// udf to check wheteher age, locaton, aboutme filled or not
val filled:String=>Int=(arg:String)=>{if(arg==null) 0 else 1}
val filledudf = udf(filled)
//UDF to seperate code and text
case class
bodyresults(text:String,code:String,codeblocks:Int,FirstSen:String,LastSen:
String,FirstCode:String,LastCode:String)
val bodyudf = udf{ (body: String)  =>  try {
        val xmlElems = xml.XML.loadString(s"""<?xml version="1.0"
encoding="utf-8"?> <!DOCTYPE body [<!ENTITY nbsp " "> <!ENTITY ndash
"&#8211;"><!ENTITY mdash "&#8212;">]><body>${body}></body>""")
        val code = (xmlElems\\"body"\\"code").text
        val text = (xmlElems \\ "body").text.replace(s"$code" ,"" )
        val codeblocks = body.count(_=="<code");
        var FirstSen = "0"
        var LastSen  = "0"
        var FirstCode = "0"
        var LastCode = "0"
        if(text!=null){
         val textsen = text.split("\n")
         FirstSen = textsen(0)
         LastSen = textsen.last
```

```
          }
         if(code!=null){
          val codesen = code.split("\n")
          FirstCode = codesen(0)
          LastCode = codesen.last
          }

bodyresults(text,code,codeblocks,FirstSen,LastSen,FirstCode,LastCode)
        }catch{
          case e: Exception => bodyresults("0","0",0,"0","0","0","0")
         }
      }



// udf to get text features
case class textresults(NoOfI:Int, NoOfYou:Int, NoOfQuestionMark:Int,
NoOfExclamationMark:Int, NoOfInterWords:Int, NoOfFind:Int, NoOfRecc:Int,
NoOfPls:Int, NoOfHelp:Int, NoOfInternet:Int)
val wordcount:String=>textresults = (text:String)=>{
var NoOfI = 0;
var NoOfYou = 0;
var NoOfQuestionMark = 0;
var NoOfExclamationMark =0;
var NoOfInterWords =0;
var NoOfFind = 0;
var NoOfRecc = 0;
var NoOfPls = 0;
var NoOfInternet = 0;
var NoOfHelp = 0;
 for(rawWord <- text.split(" "))
                   {
                   val word = rawWord.toLowerCase
                    if(word.contains("recommend"))
                    NoOfRecc = NoOfRecc+1
                    if(word=="find"||word=="looking")
                   NoOfFind = NoOfFind+1
                   if(word=="i")
                   NoOfI= NoOfI+1
                   if(word=="you")
                   NoOfYou= NoOfYou+1
                   if(word.contains("?"))
                   NoOfQuestionMark= NoOfQuestionMark+1
                   if(word.contains("!"))
                   NoOfExclamationMark = NoOfExclamationMark+1
              if(word=="what"|| word=="when" || word=="how" || word
=="where" || word == "why" || word =="whom")
              NoOfInterWords = NoOfInterWords+1
if(word=="please")
 NoOfPls = NoOfPls+1;
if(word=="help")
NoOfHelp = NoOfHelp+1;
if(word=="internet")
NoOfInternet = NoOfInternet+1;
}
textresults(NoOfI,NoOfYou,NoOfQuestionMark,NoOfExclamationMark,NoOfInterWor
ds,NoOfFind,NoOfRecc,NoOfPls,NoOfHelp,NoOfInternet)
}
val wordcountudf = udf(wordcount)
// udf to get features from the comments text
```

```
case class commentsresults(NoOfDuplicate:Int, SOUrl:Int, NoOfThanks:Int,
NoOfAns:Int, NoOfIntWords:Int, NoOfSorry:Int)
val comments:String=>commentsresults=(text:String)=>{
       var NoOfDuplicate = 0;
       var SOUrl = 0;
       var NoOfThanks = 0;
       var NoOfAns =0;
       var NoOfIntWords = 0;
var NoOfSorry = 0;
       for(rawWord <- text.split("[ .\n?:]+"))
             {
             val word = rawWord.toLowerCase
             if(word=="duplicate")
             NoOfDuplicate = NoOfDuplicate+1
             if(word=="stackoverflow"||word=="overflow")
             SOUrl = SOUrl+1
       if(word=="thanks"||word=="thank")
       NoOfThanks= NoOfThanks+1
       if(word=="answer"||word=="answers")
       NoOfAns = NoOfAns+1
       if(word=="what"|| word=="when" || word=="how" || word =="where" ||
word == "why" || word =="whom")
       NoOfIntWords = NoOfIntWords+1
if(word=="sorry")
NoOfSorry = NoOfSorry+1
       }

commentsresults(NoOfDuplicate,SOUrl,NoOfThanks,NoOfAns,NoOfIntWords,NoOfSor
ry)
     }
val commentsudf = udf(comments)
// udf to get no.of tags
val tagcount = udf{(tags:String)=>
      var NoOfTags = tags.count(_=='<');
      NoOfTags
      }
// udf to get no of sentences in code and text
val NoOfSent = udf{(arg:String)=>{
      var NoOfSent = arg.split("[!?.:]+").length;
      NoOfSent
      }

     }
//udf for reputation
val reputationudf = udf{(Reputation:Int)=> if(Reputation<1000) 0 else 1}
val posts3 = posts2.withColumn("codetext",
bodyudf($"Body")).withColumn("days",
coalesce(col("ClosedDate"),current_timestamp())).withColumn("labels",labelu
df(col("ClosedDate"))).withColumn("AgeFilled",filledudf(col("Age"))).withCo
lumn("AboutMeFilled",filledudf(col("AboutMe"))).withColumn("LocationFilled"
,filledudf(col("Location"))).withColumn("WebsiteUrlFilled",filledudf(col("W
ebsiteUrl"))).withColumn("NoOfTags",tagcount($"Tags"))
val posts4 = posts3.filter($"codetext.text" =!= "0")
val posts5 =
posts4.withColumn("all_text",concat($"codetext.text",$"Title",$"comments_te
xt")).withColumn("commentslen",length(col("comments_text"))).withColumn("co
mmentsresl",commentsudf($"comments_text")).withColumn("reputation",reputati
onudf($"Reputation")).withColumn("firstsenlen",length(col("codetext.FirstSe
n"))).withColumn("lastsenlen",length(col("codetext.LastSen"))).withColumn("
lastcodelen",length(col("codetext.LastCode"))).withColumn("firstcodelen",le
ngth(col("codetext.FirstCode"))).withColumn("codelength",length(col("codete
```

```
xt.code"))).withColumn("textlength",length(col("codetext.text"))).withColum
n("titlelength",length(col("Title"))).withColumn("textfeatures",wordcountud
f($"codetext.text")).withColumn("codesente",NoOfSent($"codetext.code")).wit
hColumn("textsente",NoOfSent($"codetext.text"))
val columns =
List("pid","all_text","comments_text","firstsenlen","commentslen","lastsenl
en","firstcodelen","lastcodelen","Score","ViewCount","NoOfTags","commentsre
sl.NoOfDuplicate","commentsresl.SOUrl","commentsresl.NoOfSorry","commentsre
sl.NoOfAns","commentsresl.NoOfThanks","commentsresl.NoOfIntWords","codetext
.code","codetext.FirstSen","codetext.LastSen","codetext.FirstCode","codetex
t.LastCode","codelength","textlength","titlelength","codetext.text","textfe
atures.NoOfI","textfeatures.NoOfYou","textfeatures.NoOfQuestionMark",
"textfeatures.NoOfInterWords","textfeatures.NoOfRecc","textfeatures.NoOfFin
d","textfeatures.NoOfHelp","textfeatures.NoOfPls","textfeatures.NoOfInterne
t","codesente", "textsente",
"textfeatures.NoOfExclamationMark","Title","Tags","ClosedDate","AnswerCount
","FavoriteCount","CommentCount","reputation","WebsiteUrlFilled","LocationF
illed","AgeFilled","AboutMeFilled","Views","UpVotes","DownVotes","labels")
val posts6 = posts5.select(columns.head,columns.tail:_*);
val posts7 =
posts6.withColumn("alltextlen",length(col("all_text"))).withColumn("codetex
tratio",$"codelength"/$"textlength").withColumn("firstcodesenratio",$"first
codelen"/$"firstsenlen").withColumn("lastcodesenratio",$"lastcodelen"/$"las
tsenlen").na.fill(0)
val tokenizer = new
Tokenizer().setInputCol("all_text").setOutputCol("words")
val Tokenized = tokenizer.transform(posts7)
val hashingTF = new
HashingTF().setInputCol("words").setOutputCol("TFfeatures").setNumFeatures(
10)
val tfModel = hashingTF.transform(Tokenized)
val idf = new IDF().setInputCol("TFfeatures").setOutputCol("IDFfeatures")
val idfModel = idf.fit(tfModel)
val posts8 = idfModel.transform(tfModel)
//all features
val assembler = new
VectorAssembler().setInputCols(Array("NoOfDuplicate","NoOfThanks","SOUrl","
NoOfAns","NoOfIntWords","NoOfSorry","NoOfHelp","NoOfPls","NoOfInternet","No
OfRecc","NoOfFind","Score","commentslen","firstcodelen","firstsenlen","last
codelen","lastsenlen","codetextratio","firstcodesenratio","lastcodesenratio
","NoOfTags","Views","codelength","IDFfeatures","textlength","titlelength",
"NoOfI", "NoOfYou", "NoOfQuestionMark", "NoOfExclamationMark",
"NoOfInterWords", "ViewCount", "AnswerCount","codesente", "textsente",
"FavoriteCount","CommentCount", "reputation", "UpVotes", "DownVotes",
"AgeFilled", "LocationFilled",
"AboutMeFilled")).setOutputCol("features_temp")
val df1 = assembler.transform(posts8)
val normalizer = new
Normalizer().setInputCol("features_temp").setOutputCol("features").setP(1.0
)
val df2 = normalizer.transform(df1)
val labelIndexer = new
StringIndexer().setInputCol("labels").setOutputCol("label")
val df3 = labelIndexer.fit(df2).transform(df2)
val featureslabel = List("features","label")
val df4 = df3.select(featureslabel.head,featureslabel.tail:_*)

// logistic regression
val Array(training,test)  = df4.randomSplit(Array(0.7,0.3), seed = 234)
```

```
val lr = new
LogisticRegression().setMaxIter(10).setFeaturesCol("features").setLabelCol(
"label")
val model = lr.fit(training)
val predictions = model.transform(test);
println(s"Coefficients: ${model.coefficients} Intercept:
${model.intercept}")
val evaluator = new
BinaryClassificationEvaluator().setLabelCol("label").setRawPredictionCol("r
awPrediction").setMetricName("areaUnderROC")
val accuracy = evaluator.evaluate(predictions)
//random forest
val labelIndexer = new
StringIndexer().setInputCol("labels").setOutputCol("indexedLabel").fit(df2)
val featureIndexer = new
VectorIndexer().setInputCol("features").setOutputCol("indexedFeatures").set
MaxCategories(4).fit(df2)
val Array(trainingData, testData) = df2.randomSplit(Array(0.7, 0.3))
val rf = new
RandomForestClassifier().setLabelCol("indexedLabel").setFeaturesCol("indexe
dFeatures").setNumTrees(10)
val labelConverter = new
IndexToString().setInputCol("prediction").setOutputCol("predictedLabel").se
tLabels(labelIndexer.labels)
val pipeline = new Pipeline().setStages(Array(labelIndexer, featureIndexer,
rf, labelConverter))
val model = pipeline.fit(trainingData)
val predictions = model.transform(testData)
val evaluator = new
BinaryClassificationEvaluator().setLabelCol("labels").setRawPredictionCol("
rawPrediction").setMetricName("areaUnderROC")
val accuracy = evaluator.evaluate(predictions)
//svm
val lsvc = new LinearSVC().setMaxIter(2)
val lsvcModel = lsvc.fit(training)
println(s"Coefficients: ${lsvcModel.coefficients} Intercept:
${lsvcModel.intercept}")
val predictions = lsvcModel.transform(test);
val evaluator = new
BinaryClassificationEvaluator().setLabelCol("label").setRawPredictionCol("r
awPrediction").setMetricName("areaUnderROC")
val accuracy = evaluator.evaluate(predictions)
```