Managing Big Data

Project Report 2021-2022 (1B)

Predicting Accepted Answers using Comments and Votes for

Group 8: *Kushal Kantharajappa(s2406330), Marzieh Adineh (s2548690), Rutuja Dolas(s2592886), Shruthi Sajid (s2551993)*

February 16, 2022

Contents

1 Abstract

Online question and answer (Q&A) forums help foster and build a community that drives interaction among peers on a plethora of topics. One such online forum is Stack Overflow which is a website where programmers can ask and answer questions and it caters specifically towards discussions on programming, software algorithms, coding techniques and software development tools. It has become an increasingly beneficial forum for software developers, programmers, students, and others who want to find viable solutions for debugging errors or identifying better quality of code to support their technical projects.

A particular question on Stack Overflow can be answered multiple times by different experts. Ordering these answers in the answers section of the website is important. By showing more helpful answers on the top, users can more effectively retrieve the desired information. The community votes and the mark of an accepted answer by the author of the question are some of the indicators in determining a good answer. In this paper, we have attempted to accurately predict an 'Accepted Answer' to a question posted on Stack Overflow by making use of comments under the answers and the votes.

2 Introduction

Stack Overflow allows users to post questions on the forum to get a discussion started on various topics. These questions are labeled with tags or keywords that categorize questions according to a particular theme. This makes it easy for other users on the forum to find solutions to similar questions.

StackOverflow encourages user participation by offering incentives in the form of reputation points for good questions and answers. The number of up-votes or down-votes received by a post from registered community members determines its usefulness for this purpose. Furthermore, question askers are encouraged to mark one of the received responses as an accepted answer. Approximately 4.58 million questions (57 percent) have an accepted answer; 2.43 million of these questions had multiple answers.

On the forum, a green-colored tick mark next to an answer indicates an accepted answer by the user who originally posted the question. An important caveat to note in Stack Overflow is that a post can have multiple answers to a question. Users can show support for all the answers posted on the forum by making use of the 'upvote' feature that is available. However, the user who originally posted the question can only deem one of the answers as an 'accepted answer' which essentially means that answer worked best for them and helped resolve the problem. [1]

Our research looks at how the dynamics of community activity on Stack Overflow shape the set of replies and how this affects the final answer. We created two classifiers that can effectively distinguish between an accepted answer and a non-accepted answer. Predicting an Accepted answer will be helpful in two ways. First, reordering the most relevant answer at the top of the list will help the users find their solutions faster. Second, stack overflow can use the predicted accepted answers to improve its Search engine optimization(SEO) results. Search engine crawlers can index only the accepted answer to improve the search results for its users.

Through this project, the Stack Overflow dataset was analysed to accurately predict an accepted answer for a question on the basis of comments and upvotes. This was carried out through an exploratory data analysis that included dealing with missing values and a highly imbalanced dataset. Following this, machine learning algorithms like Logistic Regression, Random Forest, Decision Tree and SVM were employed on the dataset to make predictions. The performance of these algorithms were measured on the basis of F1 scores, Accuracy and the time taken to run a successful execution. The remaining sections of the report consists of the problem statement that we aim to solve followed by the research questions, related work and methodology followed by results and discussion.

2.1 Problem Statement

The problem that we are attempting to address with this project is as follows:

- Users who post questions on Stack Overflow do not always pick an Accepted answer. Which essentially means that, not all the questions on the forum have accepted answers marked under them.
- Furthermore, in some cases, accepted answers may become obsolete as a new answer becomes more relevant. This could be due to changes in the library function, causing the answer to become out of date.

2.2 Research Questions

The following research questions will be investigated as part of this study:

- RQ1- How accurately can we predict an Accepted Answer in Stackoverflow?
- RQ2- How significant are the comments and the number of upvotes while predicting the Accepted Answer in Stackoverflow?

3 Related work

Stack Overflow questions are frequently of scholarly interest. Among related studies, the majority of them have concentrated on Stack Overflow questions with any answer, rather than the accepted answers. In this section, we will look at studies that have been conducted on questions that have no answers as well as questions that have accepted answers.

According to [2], different machine learning models like Random Forest and Logistic Regression can be employed to predict the best answers for questions on Stack Overflow. The accuracy of choosing the best answer, i.e. the percentage of questions for which the models correctly predicted the best answer, was used to evaluate the models.They extracted 44184 questions from the dataset that had at least four and no more than ten answers, and we ensured that the best answer could be determined from the candidates. Then divided the dataset into two parts: training and testing. Classifiers were trained on the training set and tested on the test set using 3-fold cross validation. The logistic regression model gave an accuracy of 34.97%. Further, the authors also applied NLP approaches like tf-idf and LSA to the dataset.

In [3], the authors elucidate that a question posted on any community-based forum may receive multiple answers from people across different demographics and experiences and the user who asked the question can deem the response that best solves the question as 'accepted'.To further support this, they extracted various features from the question, answers and the user profile available on the forum and trained classifiers to select the best answer. The 13 aspects were the independent variables, and the asker's decision about the quality of an answer was the dependent variable. The dataset, as previously described, consisted of 120 questions with a total of 600 answers. Once again, an answer was considered high quality if (1) the asker selected it as the best answer and (2) it received a rating of 3 or higher.They were able to correctly classify 80.33 percent of the time. A 10-fold cross-validation with this model yielded a classification accuracy of 79.50%, with nearly all of the best answers classified in the incorrect class.

In [5], addresses that currently Stack Overflow users do not receive feedback on their questions before asking them, so they may ask questions that are unclear, unreadable, or inappropriately tagged. They evaluated the proposed features with predictive models trained on the features of 18 million questions. They can use this tool of prediction to modify their questions and tags in order to see the different results

of the tool and to purposefully improve their questions in order to get accepted answers. There AUC score was of 0.71.The AUC scores show a significant relationship between the characteristics of questions and getting expected answers.

Different from the cited work, we investigate around 27 million answers with comments. To the best of our knowledge, mostly all the studies were to related to the Prediction of Questions without Accepted Answers, also some studies related to the prediction of the best answers for questions on Stack Overflow. There were no similar studies to our approach, but the prediction models in the studies assisted us in understanding the approach to the machine learning algorithms in our project. Our new proposed features is the prediction of answer status based on a data set of comments and votes.

4 Methodology

We used the Stack overflow data set available at [4] for the project. The data set contains different xml files for posts, comments, votes, tags and badges. All the xml files were zipped using 7zip. The entire data set consists of approximately 82 GB of compressed data, of which the comments and the votes data together make up to approximate 43GB uncompressed in xml format.

To predict the accepted answers, we used two of the above-mentioned datasets individually. First, there's the *comments* dataset, which contains comments for each answer to specific questions. Second, there's the *votes* dataset, which includes the number of upvotes, downvotes, and whether or not the response is accepted. Accepted responses are encoded as "1", whereas non-accepted answers are encoded as "0." The class labels will be based on this.

The Figure 1, shows the position of *comments* and *votes* on a typical stack overflow webpage. The number of upvotes is highlighted as a green box at the top of the page, a green tick indicates an accepted answer, and downvotes are not displayed on the page but are recorded in the database. The comments section is marked with a blue box.

Figure 2, gives an overview of the workflow we followed to achieve the results using comments.

Figure 3, gives an overview of the workflow we followed to achieve the results using votes. Furthermore, with both data sets, the methodology section is detailed separately, which includes data preparation, data exploration, data pre-processing, and the classification methods employed.

4.1 Prediction using Comments Dataset

The data set was downloaded from [4]. The uncompressed file size of the Comments data set was approximately 20.4 GB. There are 27,738,642 answers with comments in total. Since the comments data set lacked the target variable (i.e. accepted answer status), an inner join was performed with the votes dataset, with the PostID serving as the primary key. Appendix B contains the database schema.

4.1.1 Data Preparation

Data preparation is the process of collecting, cleaning, and combining data to yield better insights. It is imperative to put the data into context, make sense of it, remove irrelevant and noisy data before proceeding to apply modeling techniques and algorithms. Following were the steps followed:

1. The data was decompressed and uploaded to HDFS.
2. The xml files were read into an RDD and the python library "ElementTree" was used to convert the xml data into tuples.

Figure 1: Snapshot from Stack overflow webpage

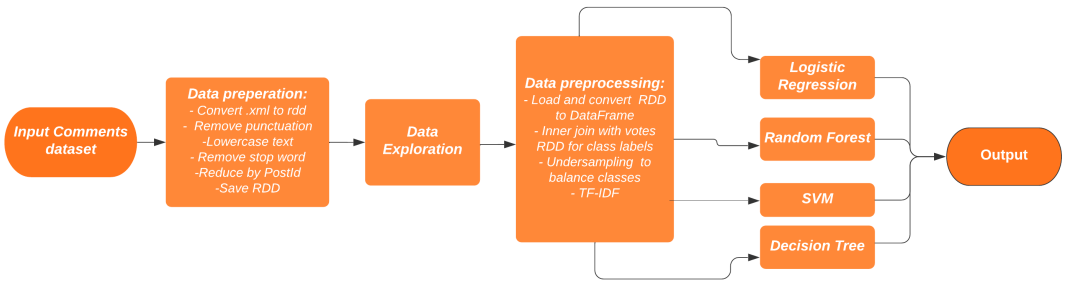

Figure 2: Comments Workflow

3. Unwanted columns were dropped and missing values were imputed. Along with removing punctuation, lowercase text, removing stop words and tokenizing the words.

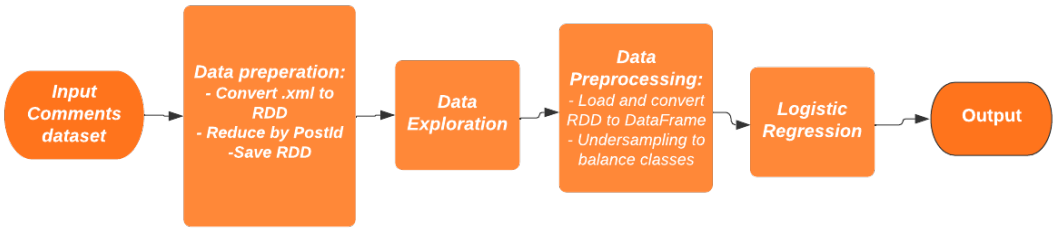

Figure 3: Votes Workflow

4.1.2 Data Exploration

Data exploration is the process of exploring and visualizing data in order to discover insights and identify areas or patterns to investigate further.

Figure 4, shows the top 10 Tags - 'javascript' is the top most popular tag followed by 'python'. A tag is a keyword or label that categorizes your question with other similar questions. Using the right tags makes it easier for others to find and answer your question.

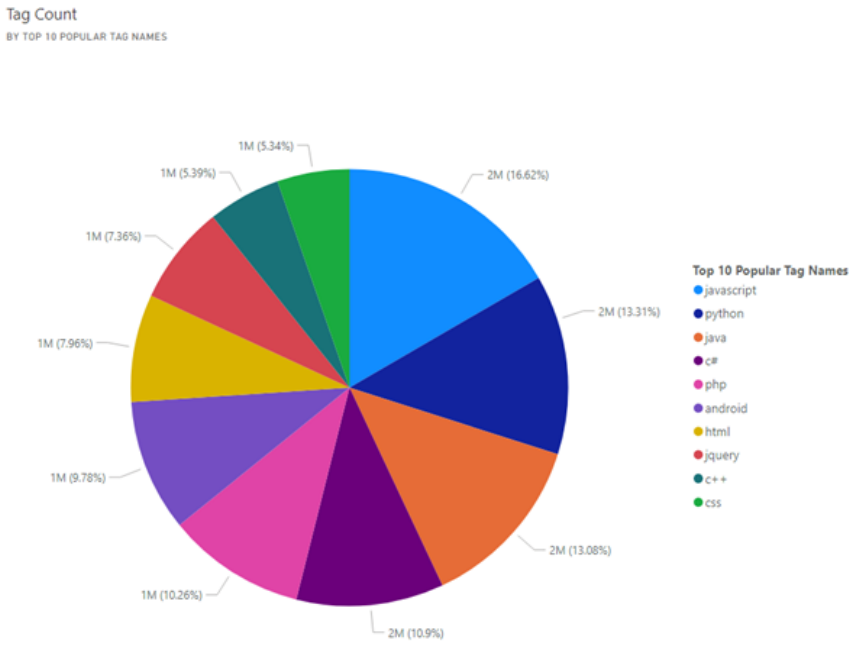

Figure 4: Top 10 popular tag names

A high magnitude of class imbalance in Big Data can adversely affect the training of the machine learning algorithms. In our first run of training the classifier on comments data, we obtained a test accuracy of 77% but the classifier classified all samples into one class due to skewness. Upon further data exploration, it was noted that the ratio of classes was 77:23 as shown in Figure 5. To overcome this problem, we under-sampled class 0 data and matched its count to the count of class 1.

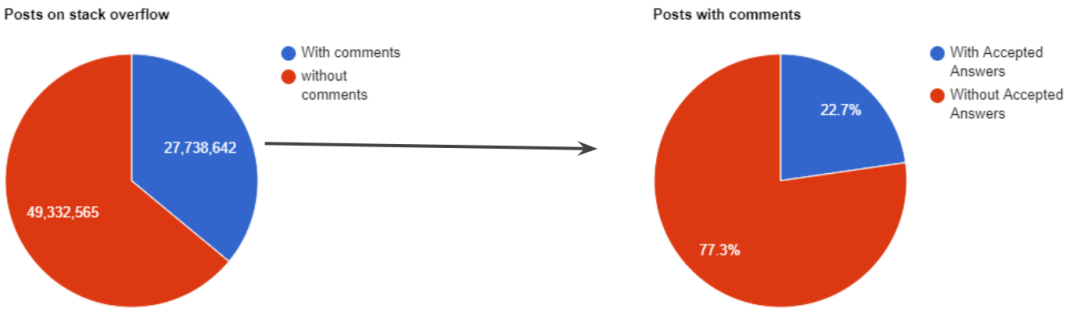

Figure 5: LEFT: Total posts on stack overflow. RIGHT: Posts with comments

4.1.3 Data Preprocessing

- RDD was transformed to a dataframe after the data was loaded.
- "Comments.xml" (Columns - Id and PostId) and "Votes.xml" (Columns - Id, PostId and VoteTypeId) were merged with PostId as unique value.
- The text column for comments.xml was preprocessed, which included calculating the TF-IDF scores
- To balance the classes, undersampling was used.

4.1.4 Classification methods

In the following part of this section, we describe multiple classifiers that were trained to classify the answers based on comments data. The performance of the classifiers was also tabulated and compared. The code which was used for implementation is given in Appendix A.

The total posts with comments after under sampling is approximately 12 million, divided equally for both class values 0 and 1. For the comments data, the input is a sparse matrix of the TF-IDF matrix which represents the comments under each answer. The output is the class values 0 and 1.

We implemented five different binary classifier algorithms using library 'mllib'. The features is only TF-IDF and labels are the Answer status. TF-IDF is a method to quantify words in a set of documents. The implemented classifiers are as follows :

- Binary Logistic Regression which is optimized using Stochastic Gradient Descent. We randomly split the training and testing data into 80% and 20% respectively to perform the logistic regression on the data.
- Random Forest was performed with 10 trees.
- Decision Tree was performed with a max tree length as 20
- Support Vector Machines classified the data by determining the best hyperplane that separates all data points from one class from those from the other.
- Naive Bayes algorithm did not run because the input was a high-dimensional sparse matrix and the algorithm did not have enough memory to compute all of the probabilities.

For each classifier, Accuracy and F1 scores were measured and the time taken for execution was also noted.

4.2 Prediction using Votes Dataset

The uncompressed file size of the Votes data set was approximately 23 GB. The Votes data set consisted of ID, PostId and VoteTypeId. Appendix B contains the database schema.

4.2.1 Data Preparation

- The xml files were read into an RDD and the python library "ElementTree" was used to convert the xml data into tuples.
- The RDD was then reduced by 'PostID' using the reducebyKey function and saved for further data analysis.

.

4.2.2 Data Exploration

Figure 6, shows the distribution of answers taken from the votes data. The x-axis represents the number of upvotes and the y-axis represents the number of downvotes. Both the axes are on a logarithmic scale. The blue data points indicate non-accepted answers while the red data points indicate accepted answers. The red data point cluster is distributed towards the bottom-right portion of the graph which indicates that the accepted answers tend to have a high number of upvotes and a low number of downvotes.

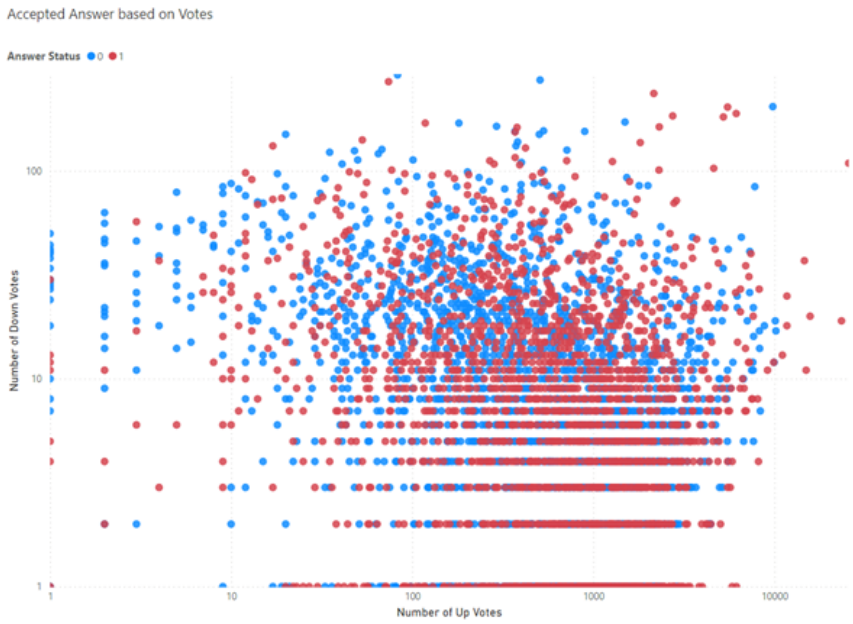

Figure 6: Accepted Answers based on voted

Figure 7, shows the frequency of words according to each class taken from user comments. The image on the left shows the top words for user comments under the accepted answer. The words highlighted indicate the associated positive connotation for the words. The image on the right displays the top words in the comments under non-accepted answers. The number of words with positive connotations is missing here. Instead, we can see some words with a possible negative connotation.

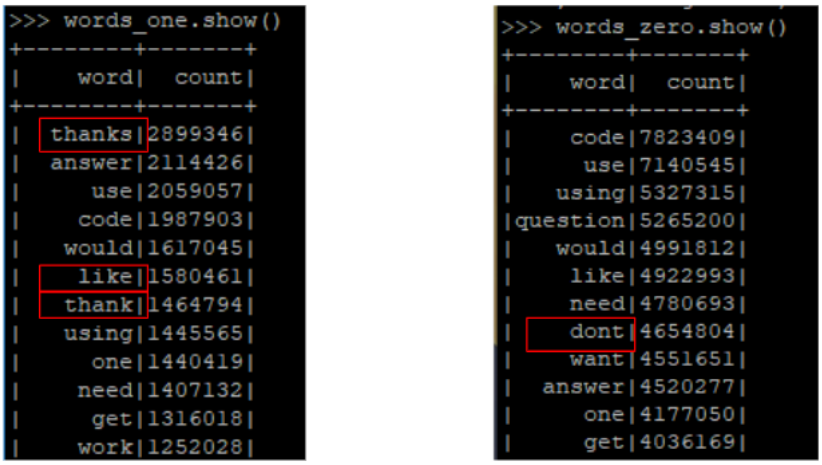

Figure 7: Left: Frequency of words for accepted answer. Right: Frequency of words for non-accepted answers.

4.2.3 Data Preprocessing

- RDD was transformed to a dataframe after the data was loaded.
- To balance the classes, undersampling was used.

4.2.4 Classification methods

We constructed a model with the three features in the dataset viz. upvotes and downvotes and ratio. Ratio is calculated as follows, (upvotes/(downvotes+1)), where 1 is to avoid "divide by zero error". The Labels are the Answer status. For the training and testing of the model the data was divided into 70% and 30% respectively.

We ran a binary logistic regression with SGD optimizer to infer the difference between both the classes. Quantitative classifiers were implemented using the library 'mllib' and we used 10 executors with 2GB memory to run the program. The resulting model had an accuracy of 66% and indicates that all three parameters significantly influence its prediction. Later we calculated the Accuracy, F1 score and time taken to execute which is discussed in the results section. Refer to Appendix A, upvotes.py for the code.

5 Results

5.1 For comments

Figure 8, compares the results of the classifiers trained on the comments data. Logistic Regression took 4 minutes to train and displayed accuracy of 63.42%. Figure 9 , shows the confusion matrix of Logistic Regression. This was the best-performing classifier. Although Support Vector Machines is compute intensive, it provided an accuracy of 63.50% which is comparable to logistic regression. The Naive Bayes algorithm failed to run since the input was a high dimensional sparse matrix and the algorithm did not have sufficient memory to compute all the probabilities. Table 1, summarises the results.

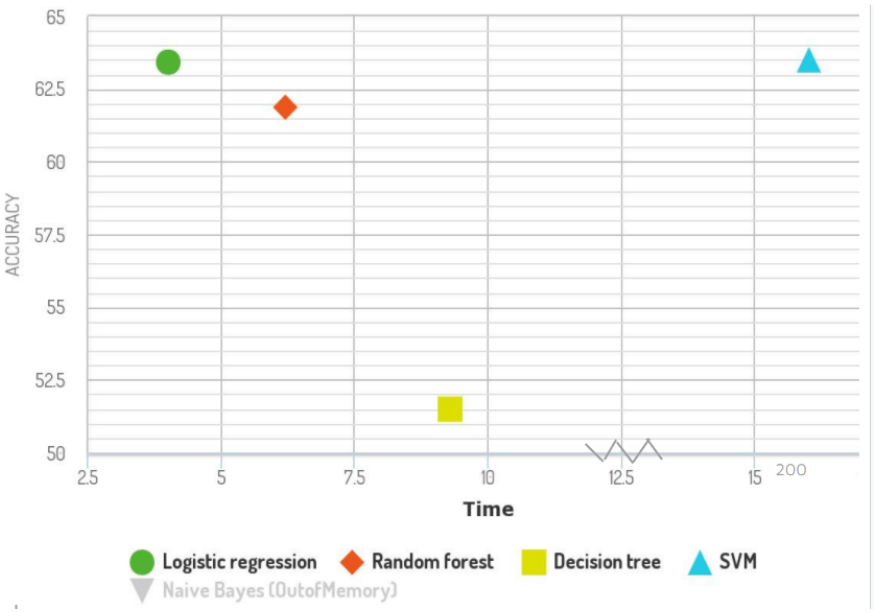

Figure 8: Comparison of classification algorithms

Measures	Logistic regression	Random Forest	Decision tree	SVM	Naive bayes
Accuracy	63.42%	61.87%	51.49%	63.50%	-
F1 Score	60.89%	59.04%	60.81%	60.00%	-
Time in m	4	6.22	9.17	210	-

Table 1: Performance of classifiers

		Predicted
Actual	840400	418974
	565473	692857

Figure 9: Confusion matrix for logistic regression using comments

5.2 For votes

Logistic regression with an accuracy of 66% and an F1 score of 58.01% was obtained for votes data. Figure 10, shows the confusion matrix for logistic regression using votes data.

6 Future Work

To further improve the accuracy of prediction, advanced text classification methods using Deep Neural Networks can be used. Moreover, creating a single classification model with both the comments and votes data can yield better results. To align the project towards more practical use, a probabilistic output, instead of binary output can be retrieved to predict the confidence score. This can be used to accurately

		Predicted
Actual	988215	2446802
	141057	3290713

Figure 10: Confusion matrix for logistic regression using votes

compare the answers using a quantitative approach.

7 Conclusion

Stack Overflow has gained traction over the years in terms of helping users across the globe to connect and discourse on multiple topics and questions. While it has its benefits, it is also imperative to get the best possible solution for the questions posed on the forum. Through this paper, we have attempted to accurately predict accepted answers for questions posted on the forum on various topics. This was carried out using the comments and votes data from the archive of Stack Overflow content which also includes tags, posts and badges. Based on an initial exploratory data analysis that was performed to make sense of the data, we decided to make use of classification algorithms like Logistic Regression, Random Forest, Decision and SVM. The results of the predictions were tabulated and compared on the basis of F1 Score and Accuracy. From the results, Logistic Regression seems to have a better F1 Score, Accuracy and shorter execution run-time as compared to the other algorithms.

References

[1] Amies, Alex. "Does that Answer the Question? Classification of Stackoverflow Comments with Supervised Learning Udacity ML Nanodegree Capstone Project". In: May 2017. DOI: 10.13140/RG.2.2.33315.04647.

[2] Lin, Jun. "Predicting the Best Answers for Questions on Stack Overflow". In: 2018.

[3] Shah, Chirag and Pomerantz, Jefferey. "Evaluating and Predicting Answer Quality in Community QA". In: New York, NY, USA: Association for Computing Machinery, 2010. ISBN: 9781450301534. DOI: 10.1145/1835449.1835518. URL: https://doi.org/10.1145/1835449.1835518.

[4] *Stack exchange dataset*. https://archive.org/details/stackexchange. Accessed: 2021-28-12.

[5] Yazdaninia, Mohamad, Lo, David, and Sami, Ashkan. "Characterization and prediction of questions without accepted answers on stack overflow". In: Mar. 2021. URL: https://arxiv.org/abs/2103.11386.

8 Appendix

8.1 Appendix A

```
PREPROCESS COMMENTS.py
#Read and split comments.xml from /users/s2406330/project/Comments.xml
from pyspark import SparkContext
```

```
from pyspark.sql.functions import *
import xml.etree.ElementTree as ET
from pyspark.sql.types import IntegerType,BooleanType,DateType
sc = SparkContext(appName="split_comments")
sc.setLogLevel("ERROR")
stopwords = {‘ourselves’, ‘hers’, ‘between’, ‘yourself’, ‘but’, ‘again’, ‘there’, ‘about’,
‘once’, ‘during’, ‘out’, ‘very’, ‘having’, ‘with’, ‘they’, ‘own’, ‘an’, ‘be’, ‘some’, ‘for’, ‘do’,
‘its’, ‘yours’, ‘such’, ‘into’, ‘of’, ‘most’, ‘itself’, ‘other’, ‘off’, ‘is’, ‘s’, ‘am’, ‘or’,
‘who’, ‘as’, ‘from’, ‘him’, ‘each’, ‘the’, ‘themselves’, ‘until’, ‘below’, ‘are’, ‘we’, ‘these’,
‘your’, ‘his’, ‘through’, ‘don’, ‘nor’, ‘me’, ‘were’, ‘her’, ‘more’, ‘himself’, ‘this’, ‘down’,
‘should’, ‘our’, ‘their’, ‘while’, ‘above’, ‘both’, ‘up’, ‘to’, ‘ours’, ‘had’, ‘she’, ‘all’, ‘no’,
‘when’, ‘at’, ‘any’, ‘before’, ‘them’, ‘same’, ‘and’, ‘been’, ‘have’, ‘in’, ‘will’, ‘on’, ‘does’,
‘yourselves’, ‘then’, ‘that’, ‘because’, ‘what’, ‘over’, ‘why’, ‘so’, ‘can’, ‘did’, ‘not’, ‘now’,
‘under’, ‘he’, ‘you’, ‘herself’, ‘has’, ‘just’, ‘where’, ‘too’, ‘only’, ‘myself’, ‘which’,
‘those’, ‘i’, ‘after’, ‘few’, ‘whom’, ‘t’, ‘being’, ‘if’, ‘theirs’, ‘my’, ‘against’, ‘a’, ‘by’,
‘doing’, ‘it’, ‘how’, ‘further’, ‘was’, ‘here’, ‘than’}
file_rdd = sc.textFile("/user/s2406330/project/Comments.xml", minPartitions=5)
rdd1 = file_rdd.filter(lambda c: ’<row’ in c )
def dtoli(c):
tree = ET.fromstring(c.encode(’ascii’,errors=’ignore’).decode(’ascii’)).attrib
text = tree[’Text’]
text= re.sub(r"http\S+", "", text)
text= text.translate(string.maketrans("",""), string.punctuation)
text = text.lower()
text= [w for w in text.split() if not w in stop_words]
return (tree[’PostId’], text)
rdd2 = rdd1.map(dtoli)
rdd3 = rdd2.reduceByKey(lambda a, b: a+’ ’+b)
rdd4.coalesce(10).saveAsTextFile("/user/s2406330/project/votes_rdd")

PREPROCESS VOTES.py
#Read and split votes.xml from /users/s2406330/project/Votes.xml
file_rdd = sc.textFile("/user/s2406330/project/Votes.xml", minPartitions=5)
rdd2 = file_rdd.filter(lambda c: ’<row’ in c )
def dtoli(c):
    tree = ET.fromstring(c).attrib
    return (tree[’PostId’],[tree[’VoteTypeId’]])
rdd1 = rdd2.map(dtoli)
rdd3 = rdd1.reduceByKey(lambda a, b: a+b)
rdd4 = rdd3.map(lambda x:(int(x[0]),1) if ’1’ in x[1] else (int(x[0]),0))
rdd4.coalesce(10).saveAsTextFile("/user/s2406330/project/votes_rdd")

JOIN DATAFRAMES.py
rdd_comments = sc.textFile("/user/s2406330/project/comments_rdd/*")
rdd_votes = sc.textFile("/user/s2406330/project/votes_rdd/*")
rdd_comments1 = rdd_comments.map(lambda x: eval(x))
df_comments = rdd_comments1.toDF(["postid","comments"])
rdd_votes1 = rdd_votes.map(lambda x: eval(x))
df_votes = rdd_votes1.toDF(["postid_votes","AnswerStatus"])
df_join = df_comments.join(df_votes,df_comments.postid==df_votes.postid_votes,"left")
```

```
df_join1 = df_join.na.fill(0)

LOGISTIC REGRESSION.PY
from pyspark.sql import SparkSession
from pyspark import SparkContext
from pyspark.sql.functions import *
import xml.etree.ElementTree as ET
from pyspark.sql.types import IntegerType,BooleanType,DateType
import re
import string
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import HashingTF,IDF,Tokenizer
from pyspark.mllib.evaluation import MulticlassMetrics
sc = SparkContext(appName="split_comments")
sc.setLogLevel("ERROR")
spark = SparkSession(sc)
rdd_comments = sc.textFile("/user/s2406330/project/comments_rdd/*")
rdd_votes = sc.textFile("/user/s2406330/project/votes_rdd/*")
rdd_comments1 = rdd_comments.map(lambda x: eval(x))
df_comments = rdd_comments1.toDF(["postid","comments"])
rdd_votes1 = rdd_votes.map(lambda x: eval(x))
df_votes = rdd_votes1.toDF(["postid_votes","AnswerStatus"])
df_join = df_comments.join(df_votes,df_comments.postid==df_votes.postid_votes,"inner")
df_join1 = df_join.na.fill(0)
print(df_join1.groupBy('AnswerStatus').count().show())
fract = df_join1.groupBy('AnswerStatus').count()
zz = df_join1.select('AnswerStatus').where(df_join1.AnswerStatus==0).count()
oo = df_join1.select('AnswerStatus').where(df_join1.AnswerStatus==1).count()
zero = df_join1.filter(df_join1.AnswerStatus==0).sample(fraction= float(cc[1][0])/float(cc[0][0]) )
one = df_join1.filter(df_join1.AnswerStatus==1)
data = one.union(zero)
train_df, test_df = data.randomSplit([0.8,0.2])
hashingTF = HashingTF(inputCol="comments", outputCol="hash",numFeatures=100)
idf = IDF(inputCol=hashingTF.getOutputCol(),outputCol="features",minDocFreq=5)
pipeline = Pipeline(stages=[hashingTF,idf])
model = pipeline.fit(train_df)
train_df = model.transform(train_df)
test_df = model.transform(test_df)
lr = LogisticRegression(labelCol= "AnswerStatus", featuresCol="features")
model_lr = lr.fit(train_df)
pred_lr = model_lr.transform(test_df)
evaluator = BinaryClassificationEvaluator(labelCol="AnswerStatus")
accuracy = evaluator.evaluate(pred_lr)
print("Accuracy",accuracy)
pred_metric = pred_lr.select(['prediction','AnswerStatus'])
pred_metric = pred_metric.withColumn("prediction",pred_metric.prediction.cast('double'))
pred_metric = pred_metric.withColumn("AnswerStatus",pred_metric.AnswerStatus.cast('double'))
metrics = MulticlassMetrics(pred_metric.rdd.map(tuple))
```

```
print(metrics.confusionMatrix().toArray())
print("Precision",metrics.precision())
print("Recall",metrics.recall())

UPVOTES.PY
from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.sql.functions import *
from pyspark.ml.feature import OneHotEncoderEstimator, StringIndexer, VectorAssembler
from pyspark.mllib.evaluation import MulticlassMetrics
sc = SparkContext(appName="split_comments")
sc.setLogLevel("ERROR")
spark = SparkSession(sc)
rdd_votes = sc.textFile("/user/s2406330/project/upvotes/*")
rdd_votes = rdd_votes.map(lambda x: eval(x))
df_votes = rdd_votes.toDF(["up","down","status"])
df_votes = df_votes.withColumn('ratio', expr("up/(down+1)"))
nc = ["up","down","ratio"]
ass = VectorAssembler(inputCols = nc,outputCol = "features")
stages = [ass]
pipeline = Pipeline(stages = stages)
pipelineModel = pipeline.fit(df_votes)
df = pipelineModel.transform(df_votes)
selectedCols = ['status', 'features']
df = df.select(selectedCols)
train, test = df.randomSplit([0.7, 0.3], seed = 2018)
lr = LogisticRegression(featuresCol = 'features', labelCol = 'status', maxIter=10)
lrModel = lr.fit(train)
predictions = lrModel.transform(test)
evaluator = BinaryClassificationEvaluator(labelCol="status")
accuracy = evaluator.evaluate(predictions)
print(accuracy)
pred_metric = predictions.select(['prediction','status'])
pred_metric = pred_metric.withColumn("prediction",pred_metric.prediction.cast('double'))
pred_metric = pred_metric.withColumn("status",pred_metric.status.cast('double'))
metrics = MulticlassMetrics(pred_metric.rdd.map(tuple))
print(metrics.confusionMatrix().toArray())
```

8.2 Appendix B

The main fields we are considering from the data set are the comments and the votes file. Below is the data schema: **comments**.xml

- Id
- PostId

votes.xml

- Id

- PostId
- VoteTypeId
    - ‘ 1‘: AcceptedByOriginator
    - ‘ 2‘: UpMod
    - ‘ 3‘: DownMod
    - ‘ 4‘: Offensive
    - ‘ 5‘: Favorite - if VoteTypeId = 5 UserId will be populated
    - ‘ 6‘: Close
    - ‘ 7‘: Reopen
    - ‘ 8‘: BountyStart
    - ‘ 9‘: BountyClose
    - ‘10‘: Deletion
    - ‘11‘: Undeletion
    - ‘12‘: Spam
    - ‘13‘: InformModerator