Stack Overflow: Answer Time Prediction

Team: Group 8

Abhishek Madan (011408969)

Gaurav Misra (011449815)

Tanmay Bhatt (011499072)

Introduction

StackOverflow.com is an online question-and-answer site for programmers. Started in fall 2008, its rich feature set brought rapid popularity: users can ask and answer questions, collaboratively tag and edit questions, vote on the quality of answers, and post comments on individual questions and answers. The website serves as a platform for users to ask and answer questions, and, through membership and active participation, vote questions and answers up or down and edit questions and answers.

Motivation

StackOverflow is one of the most helpful websites for every programmer and/or computer science engineer. Most of the topics related to software development, computer science and related fields have been a part of the posts which are present on StackOverflow.
However, a typical question which remains unanswered is the amount of wait-times a user can expect after posting a question. This statement is the motivation behind our project work, through which we have tried to estimate approximate answer times based on the question and tags provided by the user.
The StackOverflow dataset is a rich dataset available on google cloud, and we have used it during the development of our project. Updated on a quarterly basis, this BigQuery dataset includes an archive of Stack Overflow content, including posts, votes, tags, and badges.

Objective

“Based on the accepted answer times of questions available, predict an approximate accepted answer time of a new question asked by the user”.
Following are the tasks involved:

1. Retrieve the data from Bigquery tables.
2. Clean and transform the data.
3. Develop three regression models to predict answer times.
4. Compare the results obtained.

Literature Review

StackOverflow dataset has been used in multiple publications, some of which are:

1. Azad, Shams, Peter C. Rigby, and Latifa Guerrouj. **Generating API Call Rules from Version History and Stack Overflow Posts.** ACM Transactions on Software Engineering and Methodology (TOSEM) 25.4 (2017): 29.

2. Chunyang Chen, Sa Gao, Zhenchang Xing. **Mining Analogical Libraries in Q&A Discussions — Incorporating Relational and Categorical Knowledge into Word Embedding**. The 23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER 2016).
3. Chunyang Chen, Zhenchang Xing. **Mining Technology Landscape from Stack Overflow**. The 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2016)
4. Vinayakarao, Venkatesh, et al. **Anne: Improving source code search using entity retrieval approach.** Proceedings of the Tenth ACM International Conference on Web Search and Data Mining. ACM, 2017.

System Design and Implementation Details

Algorithms considered

In order to solve the problem we explored as number of different regression algorithm. After evaluating the results from all we considered to improve on the following three algorithms-

1. **KNN Regression-** It is non-parametric algorithm which finds the k "nearest" points to a given point, and returns the prediction with the highest proportion. KNN was a suitable algorithm for solving this problem as questions of related domain and tags often have similar answer times, similar question description, thus can be used for predictions.

2. **Passive Aggressive Linear Regression-** The passive-aggressive algorithms are a family of algorithms for large-scale learning. They are similar to the Perceptron in that they do not require a learning rate. However, contrary to the Perceptron, they include a regularization parameter C.

3. **SVM Regression**- It maps the input data onto a m-dimensional feature space and then applies linear/non-linear regression, depending on the kernel, to predict the output.

Technology and Tools used

Programming language: Python 2.7
Development tool: Jupyter notebook
Libraries: Numpy, Scipy, Pandas, nltk, sklearn, matplotlib
Cloud: GCP Bigquery and storage buckets

System design

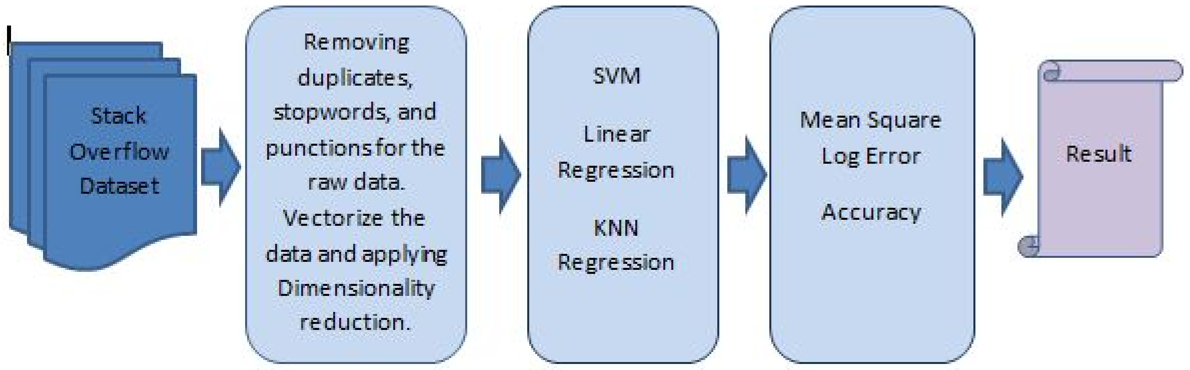
The algorithms were implemented by following these steps-

1. Selecting data- Of all the available tables data was chosen from ‘posts_questions’ and ‘posts_answers’ table. 2 million records were pruned from the available records, exported to the storage buckets, then downloaded to local machines. Following query was used to filter records.

```
SELECT
    questions.id,
    questions.title,
    questions.tags,
    (TIMESTAMP_TO_SEC(answers.creation_date) -
TIMESTAMP_TO_SEC(questions.creation_date)) as time
FROM
  [bigquery-public-data:stackoverflow.posts_questions] AS questions
INNER JOIN
  [bigquery-public-data:stackoverflow.posts_answers] AS answers
ON
  questions.accepted_answer_id = answers.id
WHERE
  questions.accepted_answer_id IS NOT NULL
```

2. The data is then cleaned of any unnecessary punctuations, symbols, digits, html tags etc.
3. The cleaned data is then put into a dataframe, and question titles are appended by tags(with necessary weightage).
4. Testing is done using 10000 records using k-fold validation.
5. Evaluation metrics are generated using necessary functions. We have used two metrics for model evaluation: Accuracy and Mean Square Log Error.
6. Accuracy considers a time span of 10 hours from the actual value, which means a prediction is considered as a true positive if the difference of predicted and actual value is less than 10 hours.
7. Mean Square Log Error: This measurement is useful when there is a wide range in the target variable, and you do not necessarily want to penalize large errors when the predicted and target values are themselves high. It is also effective when you care about percentage errors rather than the absolute value of errors.

System Data Flow


Stack Overflow Dataset
Removing duplicates, stopwords, and punctions for the raw data. Vectorize the data and applying Dimensionality reduction.
SVM
Linear Regression
KNN Regression
Mean Square Log Error
Accuracy
Result

Average evaluation matrix across algorithm

Algorithm	%Acc. within 10hrs	Log Mean Square Error
KNN	65.20	6.55
SVM (Linear)	79.65	7.93
Passive Aggressive Regressor	80.50	6.99

Experiments

Dataset Used

Source: https://bigquery.cloud.google.com/dataset/bigquery-public-data:stackoverflow

1. The dataset on Bigquery was organized in multiple tables of which 2 tables were used to obtain information about the questions and their user accepted answers.
2. The table 'posts_questions' has 14458875 records and the table 'posts_answers' has 22668556 records. The first 2 million question data was taken and their user accepted answer was queried from the 'posts_answers' table.
3. We then calculated the amount of time in which each question was answered, neglecting the questions which lacked an answer, or questions in which the answer time was same as the time of question asked(query shown under system design).
4. The query result was then saved on the cloud and was downloaded as csv split in 30 files.

The attributes in the two tables are -

Questions_posts-

id	**title**	body	**accepted_answer_id**
answer_count	comment_count	community_owned_date	favorite_count
last_activity_date	last_edit_date	last_editor_display_name	last_editor_user_id
owner_display_name	owner_user_id	post_type_id	score
tags	view_count	**creation_date**	

Answers_posts-

id	**creation_date**	comment_count	community_owned_date
body	last_activity_date	last_edit_date	last_editor_display_name
score	owner_display_name	owner_user_id	parent_id
tags	last_editor_user_id	post_type_id	

The highlighted columns were used to obtain the final dataset for this project.

Methodology followed

For training the models, following steps were taken:

1. Read the table data containing the records.
2. For each record, append the tags of the record along with the question title.
3. Use Countvectorizer to tokenize the documents and count the occurrences of token and return them as a sparse matrix.
4. The **KNN Regression** model uses WordNetLemmatizer from nltk.stem for lemmatization of features. In an approach with **Passive Aggressive Linear Regression** model, the NLTK PorterStemmer was used for stemming.
5. For cross validation, we divided the dataset into 10 subsets of 20,000 records. Then we took 1000 records from each subset, and calculated results. In all, 10,000 records were used as test set.

KNN Regression: For KNN, the best results were found by using k = 3, and giving weights according to distances of neighbours. This model has also doubled the weight for tags by appending the tags twice to the question title. Various approaches were tried by varying different parameters as well as with Tf-idf, Hashing Vectorizer, out of which CountVectorizer worked best.

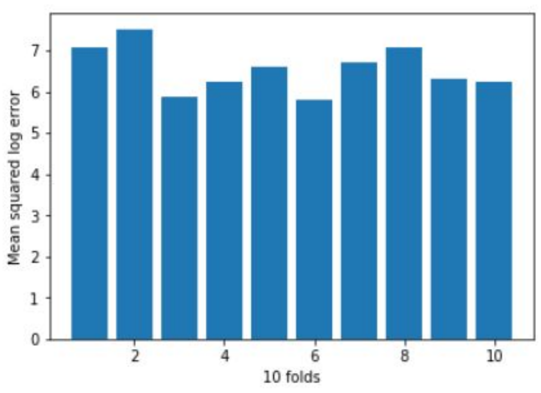
Passive Aggressive Linear Regression: Firstly, CountVectorizer was used for calculating Word Frequency Distribution matrix for Question Titles and Question tags along with NLTK libraries of Stop word removals, punctuation removal and PorterStemmer for removing stemming. Final feature set consisted was combination of Titles and Tags feature vectors with 266065 dimensions. There were mainly two reasons for selecting the Passive Aggressive Regressor:

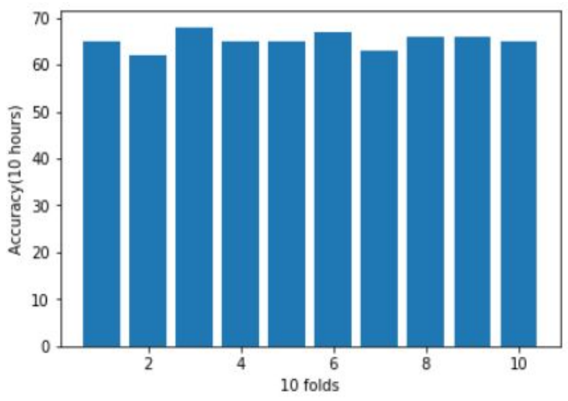
1. As it is a **large-scale learning** algorithm, it's well suited with large dataset with 200000+ features.
2. As it doesn't require a learning rate, it worked faster compared to Logistic regressor.

Linear SVM regressor: "The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences. First of all, because output is a real number it becomes very difficult to predict the information at hand, which has infinite possibilities. In the case of regression, a margin of tolerance (epsilon) is set in approximation to the SVM which would have already requested from the problem. But besides this fact, there is also a more complicated reason, the algorithm is more complicated therefore to be taken in consideration. However, the main idea is always the same: to minimize error, individualizing the hyperplane which maximizes the margin, keeping in mind that part of the error is tolerated" [6]. In this implementation a 'linear' SVM kernel with c = 1.0 was found to give best results.

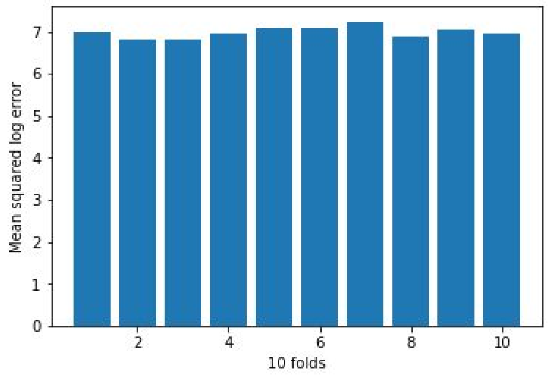
Graph showing comparative results

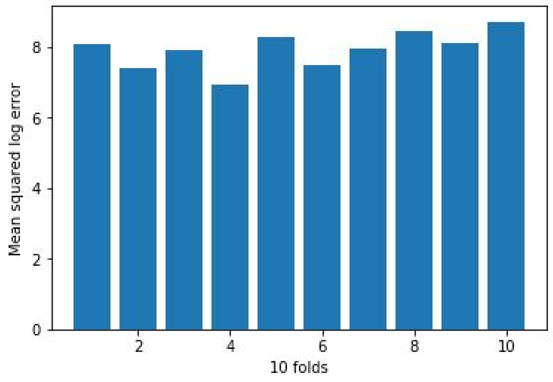
KNN



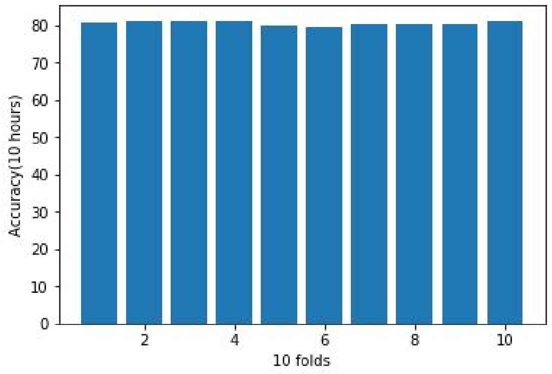


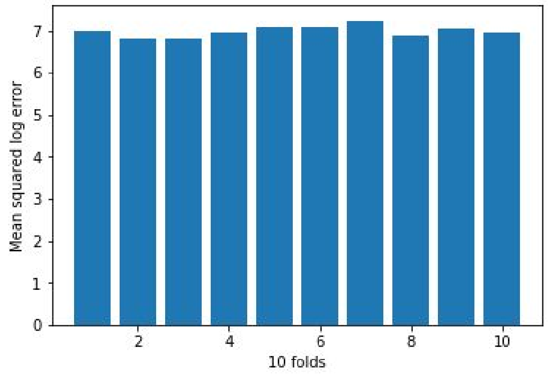
SVM





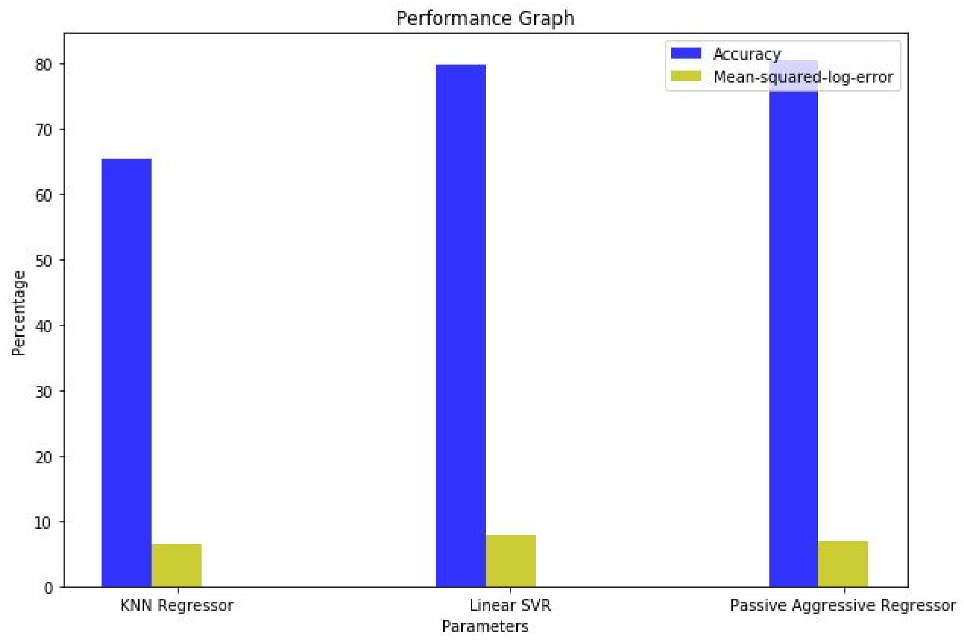
Passive Aggressive





Analysis of result

A comparative look at the performance metric of KNN, SVM and Passive aggressive regressor shows that the Passive aggressive regression algorithm gives a better estimate of the time required for a question to get accepted on Stackoverflow in comparison to the other two. With an error margin of 10 hours, its accuracy was found to be 80.5%. It is better than linear SVR and KNN regressor by 1% and 16% respectively. However, the log mean square errors were approximately in the same range for all three algorithms.



Discussion & Conclusions

Decisions made

We took the following decisions together:

1. Choosing problem statement and dataset: All three of us researched various problem statements and datasets from different domains and chose to work on this problem statement because as software engineers, we found this problem as the most interesting problem to work on. Also, the dataset we found was very rich and huge.
2. Choosing algorithms to work on: We went through numerous regression techniques and zeroed in on the three demonstrated after carefully evaluating pros and cons of each algorithm and the dataset characteristics.
3. Data transformation: We transformed the available data fields. For instance, we found the answer times by subtracting question time from the answer times, changed the time to seconds etc.
4. Choosing metrics: We chose custom accuracy and Mean sq. log error after going through various regression metrics such as MSE, mean absolute error, median absolute error, explained variance and others, as these metrics were able to represent and evaluate our models in a correct way.

Difficulties faced

All the decisions which have been listed above can be considered as difficulties we faced during development.

Apart from this, getting data from cloud can also be considered another difficulty, as the dataset was huge and contained many unneeded features.

Model training and testing was difficult due to the huge dataset. We had to code in an efficient way, to improve execution times, and we had to test a lot of variations in order to get models which fit the train data properly.

Things that worked

1. The decision to choose this dataset and problem statement improved our data mining skills as well as our understanding of existing libraries, algorithms etc.
2. Successfully handled huge number of records(2 million) without huge processing times.
3. Successfully dealt with huge number of features which were a part of the dataset.
4. Our regular sessions together helped us understand and solve some common problems.

Things that didn't work

1. SVM couldn't handle huge number of records for training, thus the SVM model used 150,000 records for training.
2. Clustering was also an option for solving this problem, but we couldn't implement it properly.

Project Plan

	Team	Date/Time
Project topic discussion and Proposal	all	Sep 10, 2017 to Sep 20, 2017
Project analysis	all	Sep 21, 2017 to Oct 16,2017
Project implementation	Abhishek Madan: SVM regression Gaurav Misra: KNN regression Tanmay Bhatt: Passive Aggressive regression	Oct 17, 2017 to Dec 03, 2017

References

1. https://en.wikipedia.org/wiki/Support_vector_machine
2. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
3. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.PassiveAggressiveRegressor.html
4. https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/
5. https://machinelearningmastery.com/how-to-prepare-data-for-machine-learning/
6. http://www.saedsayad.com/support_vector_machine_reg.htm
7. https://cloud.google.com/bigquery/public-data/stackoverflow
8. http://www.cs.duke.edu/courses/spring14/compsci290/assignments/lab02.html