Predicting the response rate on Stack Overflow with Google BigQuery

Chapter 1

1.0 Introduction

Stack Overflow is the largest online community for programmers to learn, share their knowledge, and advance their careers. Updated on a quarterly basis, this BigQuery dataset includes an archive of Stack Overflow content, including posts, votes, tags, and badges. This is a real-world data set published publicly on Google BigQuery. This dataset contains 16 tables with the number of rows ranging from 4-32 million. The goal of this project is to predict the response time based on various features of a question posted on Stack Overflow. The project will be carried out in Python on Jupyter Notebook.

Link to the code:
https://github.com/varuni-d/Predictive_Analysis_on_StackOverflow/blob/master/Captone1_Code.ipynb

1.1 Data Acquisition

As the data is published on the Google Cloud Platform, first the data must be acquired in order to be further analyzed. The following steps were carried out:

1) Enable Google BigQuery API on the Google Cloud Platform
2) Get authentication for Jupyter Notebook by obtaining a service account key for the BigQuery in a JSON file
3) Install Google BigQuery API client libraries for Python on local computer
4) Set the environment variable GOOGLE_APPLICATION_CREDENTIALS for your python script

Once these steps were successfully completed, the data could be directly analyzed from your Jupyter Notebook console. Subsequently, the data set was analyzed by applying SQL (either standard or legacy format) on Google BigQuery.

1.2 Querying Google BigQuery

Prior to data analysis, the Stack Overflow data set was accessed by directly querying the Google BigQuery using standard SQL from the Jupyter notebook in the following format.

```
>> query_job = client.query(query1)
>> questions_answered_df=query_job.to_dataframe()
```

Subsequently, a client needs to be created for each query and then convert the query results in the Python Pandas Dataframe. The Dataframe can then be handled for further analysis using Python commands. In the next chapter, more details of the exploratory data analysis on the entire Stack Overflow dataset are discussed.

Chapter 2

2.0 Exploratory Data Analysis on the entireStack Overflow dataset in Google BigQuery

In the above example, the percentage of questions answered for each year was calculated. This is illustrated in Fig 1 where the blue bars represent the number of questions asked each year and the red bard represents the questions with answers. We observe that almost 100% of the questions have been answered in the starting years of StackOverflow till 2010 and then the % of questions without answers increases, with the highest % of unanswered questions in 2019.

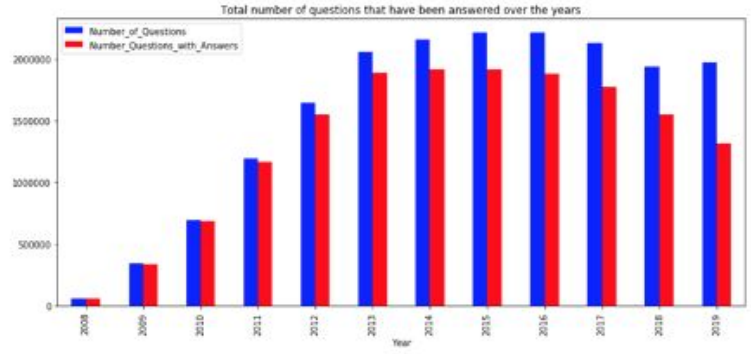


Fig 1: Histogram representing the total number of questions posted on Stack Overflow and the number of questions with answers for each year

This trend can be seen even more clearly in Fig 2, where we directly plot the yearly % of questions with answers. With the drastic drop going from year 2018 to 2019, we can ask whether it's a matter of lower quality questions or lower number of accepted answers posted on Stack Overflow. Let's look into this in more detail below.

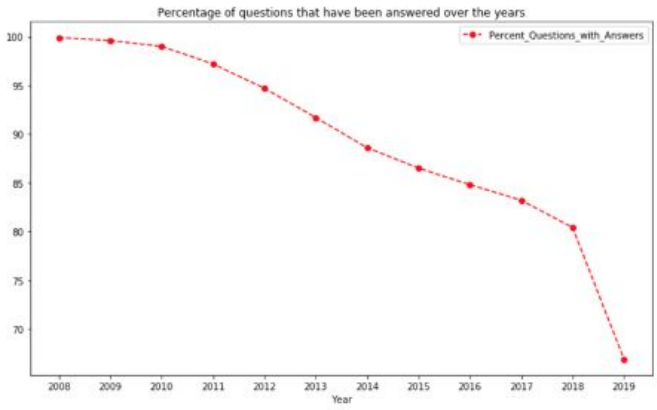


Fig 2: Illustration of the percentage of questions with answers for each year

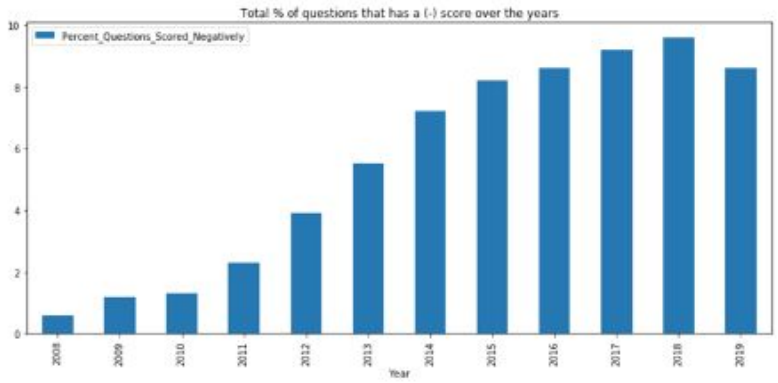


Fig 3: Percentage of questions that have been negatively scored for each year

Users on Stack Overflow have the capability to give a negative rating to posted questions depending on its quality. The site also keeps track of and reports on the number of up and down votes and the number of views received, which indicate how valuable and visible the content was to the community. In Fig 3, we see that the number of negatively scored questions shows a general upward trend till 2018 and actually decreases in 2019.

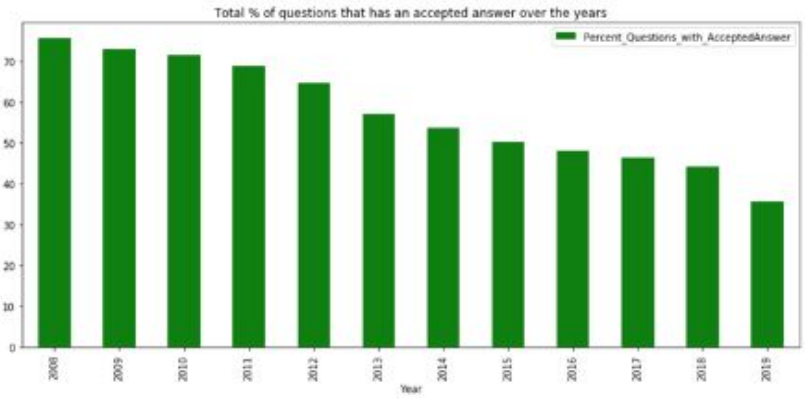


Fig 4: Percentage of questions with an accepted answer for each year

While there is a general downward trend of the percentage of questions with an accepted answer from 2008 (Fig 4), there is a significant drop going from 2018 to 2019. Thus, we can conclude that the cause for a high percentage of unanswered questions in 2019 is due to the lack of unaccepted answers and not as a result of negatively scored questions.

The site provides a variety of means by which users classify and evaluate content, which serves to organize the massive amount of content on the site and make it easier for users to find what is most relevant and useful. These include tags that identify the programming language, operating system, framework, library, and other technologies that a question addresses. Additionally, a given question can have multiple tags. If we are interested in studying the distribution of questions across domains, we need to separate tag under its own category.

```
#Top ten most popular tags
tags_query = """
        SELECT tag,COUNT(*) count
        FROM (
              SELECT body AS question, SPLIT(tags, '|') AS tags
                FROM `bigquery-public-data.stackoverflow.posts_questions`
            ), UNNEST(tags) tag
        GROUP BY 1
        ORDER BY 2 DESC
        LIMIT 10"""
query_job = client.query(tags_query)
popular_tags_df=query_job.to_dataframe()
popular_tags_df
```

In the above query, the top ten most popular tags are listed in Table 1.

Table 1: List of ten most popular tags on Stack Overflow and the number of questions under each tag

Tag	Count
Javascript	1909607
Java	1613271
C#	1363741
Php	1320047
Python	1297726
Android	1237162
Jquery	9714118
Html	863493
C++	644513
Ios	615317

We observe that Javascript is the most popular tag on Stack Overflow with over a 18% margin over the second most popular tag, Java. It is rather interesting that three out of the ten most popular tags are related to Java-the Java community seems to have a lot of technical questions!!

Furthermore, the average length for questions under each of these most popular tags were studied. This was implemented by executing a parameterized SQL query.

```
tags_list=popular_tags_df['tag'].values.tolist()
print(tags_list)
question_length_query="""
        SELECT tag, question
        FROM (
              SELECT body AS question, SPLIT(tags, '|') AS tags
                FROM `bigquery-public-data.stackoverflow.posts_questions`
            ), UNNEST(tags) tag
        WHERE tag IN UNNEST(@tagnames)
        LIMIT 5000"""

job_config = bigquery.QueryJobConfig(
    query_parameters=[
        bigquery.ArrayQueryParameter("tagnames", "STRING", tags_list),
    ]
)
```

The question length distribution for each tag is shown in Fig 5. We observe that most questions are less than 1000 characters long. It is rather interesting that a couple of questions with 10000 characters are categorized under Java.

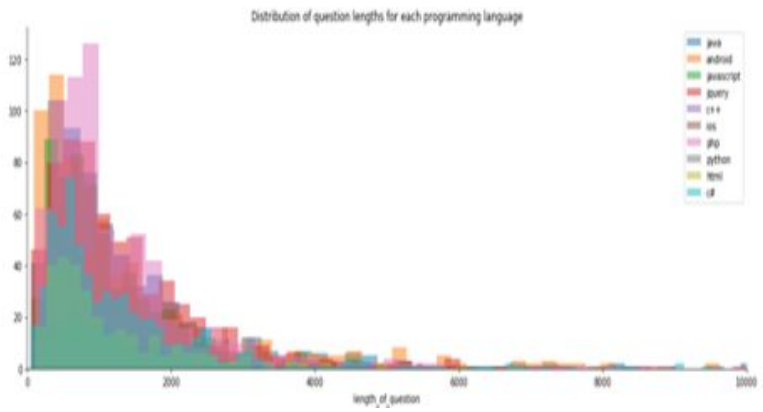


Fig 5: Distribution of question length for the top ten most popular tags

Users can build their reputation points through submitting questions and answers that other users vote up and by submitting an answer that the user who submitted a question accepts as the best answer. There are also several other means to gain reputation, such as suggesting edits that are accepted, submitting documentation that is accepted or voted useful, and accepting another user's answer to one's own question. Users lose reputation points when others vote that their questions are not useful, if

others withdraw up votes on their questions or answers, or if they repeatedly spam the site or engage in other conduct that harms the community. As users seek to build their reputation on the site, they receive incentives to participate in ways that others find helpful. Gaining in reputation can also afford users more permissions on the site; for example, new users cannot make comments until they have acquired at least 50 points of reputation. Users can also spend some of their reputation points on bounties, which allow other users to gain extra points by answering a question of particular interest or importance to the person offering the bounty. These mechanisms encourage users to offer value in their answers and actions, especially to the extent that they are motivated to keep "playing" the site like a game. It also discourages antisocial behavior such as spamming.

Users on Stack Overflow can earn different badges based on their activity. The top ten easiest gold badges to earn were queried as follows, with resulted listed in Table 2:

```
#Top ten easiest gold badges to earn

badges_query= '''
        SELECT name, COUNT(user_id) AS count
        FROM `bigquery-public-data.stackoverflow.badges`
        WHERE class=1 --1=gold 2=silver 3=bronze
        GROUP BY name
        ORDER BY count DESC
        LIMIT 10

'''

query_job = client.query(badges_query)
badges_df=query_job.to_dataframe()
badges_df
```

Table 2: List of ten easiest gold badged to earn on Stack Overflow and the number of users who have already earned that badge

Badge name	Count
Famous Question	668653
Great Answer	77521
Great Question	35936
Fanatic	33217
Unsung Hero	22120
Electorate	21201
Populist	20924

Steward	14158
Stellar Question	7725
Socratic	4070

By extracting the ending character of questions, we can calculate how many questions actually end with a question mark. We note that only 21% of all questions posted on Stack Overflow end with a question mark. This low percentage could be due to the fact that most users add more useful details like a code snippet, output, or error message to the question after posting the actual question.

We also study the starting word of questions, the time and week of day questions are posted, and user account creation year. These are chosen as the features that would directly impact the probability and time it would take to get an answer for a question on Stack Overflow. More details of these features will be discussed in the next section.

The exploratory data analysis completed above gives us a clear picture of the questions posted on Stack Overflow. More thorough data analysis can be carried out such as user information, tags, and badges. However, as we are only concerned with the questions and answers, the above analysis is sufficient. Subsequently, a refined dataset was obtained by applying several conditions and analyzed further. Several regression machine learning algorithms were applied to the dataset in order to accurately predict the time it would take for an answer to be posted for a question.

Chapter 3

3.0 Data wrangling

The entire dataset comprising 4-32 million rows was refined with several conditions to make it more manageable. The relevant data was obtained by joining the following three tables:

1) bigquery-public-data.stackoverflow.posts_questions
2) bigquery-public-data.stackoverflow.posts_answers
3) bigquery-public-data.stackoverflow.users

The above tables were joined so that we can extract information such as the number of answers, time taken for the first answer, and user information for each question posted. Each question was further categorized by its length, starting word, whether it ended with a question mark and its temporal information. The data set was cleaned based on the following conditions:

1) first word: 'how','is','why','what','can', 'i'
2) Tags are relevant to data science: 'python', 'r','mysql', 'machine-learning','data-science'

Feature engineering involved categorizing whether each question ended with a question mark, length of each question (short if <700 characters, medium if <1700 characters, long if >1700 characters), day of week and time of day it was posted, user account creation date, time taken for the first answer and computing the chance of getting an answer. The wrangled dataset containing ~65k rows is ready to be further analyzed.

3.1 Data cleaning and Handling missing values

We need to pay attention to data types and any unaccepted/ missing values.

>> data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 69016 entries, 0 to 69015
Data columns (total 10 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   first_word            69016 non-null  object
 1   ends_question         69016 non-null  bool
 2   tag                   69016 non-null  object
 3   hour_utc              69016 non-null  int64
 4   weekday_utc           69016 non-null  object
 5   account_creation_year  66640 non-null  float64
 6   question_length       69016 non-null  object
 7   time_to_answer        60358 non-null  float64
 8   chance_of_answer      69016 non-null  float64
 9   questions             69016 non-null  int64
dtypes: bool(1), float64(3), int64(2), object(4)
```

The data types of the features are as expected, expect the 'account_creation_year'. This must be converted into an integer/datetime object.

>> data.isnull().sum()

```
first_word              0
ends_question           0
tag                     0
hour_utc                0
weekday_utc             0
account_creation_year    2376
question_length         0
time_to_answer          8658
chance_of_answer        0
questions               0
```

We observe some null values for the 'account_creation_year' and 'time_to_answer' feature. Null values were probably due to a data logging error. The null values in the dependent variable means that those questions did not receive an answer. These rows will need to drop as there cannot be any missing values in the dependent variable and metric interpolation (i.e. replace NaN with mean or median) would not be suitable.

3.2 Outlier Removal

Since the 'time_to_answer' feature varies over several orders of magnitude, from 60-8.8e7, we have applied percentage based outlier removal at the 95% percentile.

```
# Find the 95th quantile
quantile = data['time_to_answer'].quantile(0.95)

# Trim the outliers
trimmed_df= data[data['time_to_answer'] < quantile]

trimmed_df.info()

# The original histogram
data[['time_to_answer']].hist()
plt.show()
plt.clf()

# The trimmed histogram
trimmed_df[['time_to_answer']].hist()
plt.show()
```

This removed about about 2,900 rows but the resulting improvement in the distribution of 'time_to_answer' is significant. The distribution is highly right-skewed.

a) b)

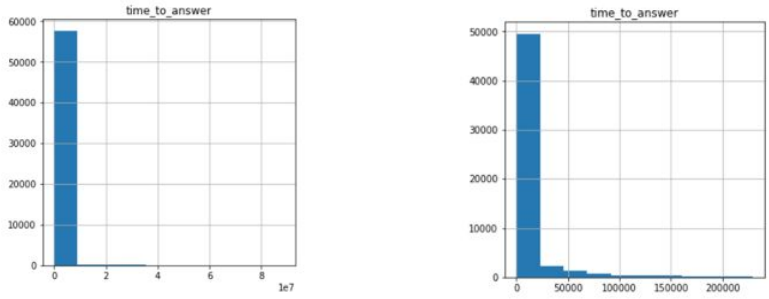


Figure 6: a) Before and b) after applying outlier removal to the 'time_to_answer' feature

3.3 Exploratory Data Analysis

3.3.1 Numerical Data Analysis

Let's take a look at the numerical data distribution.

>>data.describe()

	hour_utc	time_to_answer	chance_of_answer	questions
count	58256.000000	5.825600e+04	58256.000000	58256.000000
mean	11.972539	3.291258e+05	0.857342	4.170456
std	6.597645	3.014012e+06	0.181865	6.733131
min	0.000000	6.000000e+01	0.125000	1.000000
25%	7.000000	7.200000e+02	0.800000	1.000000
50%	12.000000	2.100000e+03	1.000000	2.000000
75%	18.000000	8.253362e+03	1.000000	4.000000
max	23.000000	9.851344e+07	1.000000	74.000000

We see that 50% of the questions get asked before noon and 75% of the questions get asked before 5pm. Additionally, more that 50% of the questions have a 100% chance of getting answered.

3.3.2 Categorical Data Analysis

The categorical features, 'first_word', 'tag', 'weekday_utc', 'question_length', were handled by applying one-hot encoding. This was carried out by creating dummy variables. This resulted in a dataset with 55345 rows and 27 columns.

```
data.question_length.value_counts(normalize=True)
```

```
medium    0.378025
short     0.345309
long      0.276666
Name: question_length, dtype: float64
```

```
data.weekday_utc.value_counts(normalize=True)
```

```
Wednesday    0.157609
Thursday     0.156614
Tuesday      0.154726
Friday       0.150915
Monday       0.145971
Saturday     0.117460
Sunday       0.116705
Name: weekday_utc, dtype: float64
```

We observe that most questions are of medium length (i.e. total number of characters between 700-1700) and most questions get posted on Wednesday. On the other hand, long questions (over 1700 characters) are not as common and least number of questions get posted over the weekend.

As mentioned before, this study only focused on questions belonging to the following five tags:

```
data.tag.value_counts(normalize=True)
```

```
python              0.496721
mysql               0.224690
r                   0.192180
machine-learning    0.073243
data-science        0.013166
Name: tag, dtype: float64
```

We see that Python questions are most common, while not many questions are tagged under ‘data-science’. Furthermore, we see below that many questions under ‘data-science’ and ‘machine-learning’ tags are short, while all the other tags under consideration have medium length questions.

```
bytag_length=data.groupby("tag").question_length.value_counts(normalize=True)
bytag_length
```

tag	question_length	
data-science	short	0.418514
	medium	0.389831
	long	0.191656
machine-learning	short	0.378720
	medium	0.365831
	long	0.255449
mysql	medium	0.379908
	short	0.335752
	long	0.284339
python	medium	0.372486
	short	0.350093
	long	0.277421
r	medium	0.393980
	short	0.326367
	long	0.279653

We observe that nearly ~40% of questions under the tags ‘data-science’ and ‘machine-learning’ are less than 700 characters, thus categorized as short. On the other hand, nearly ~40% of questions under the tags ‘mysql’, ‘python’ and ‘r’ are categorized to be of medium length, between 700-1700 characters.

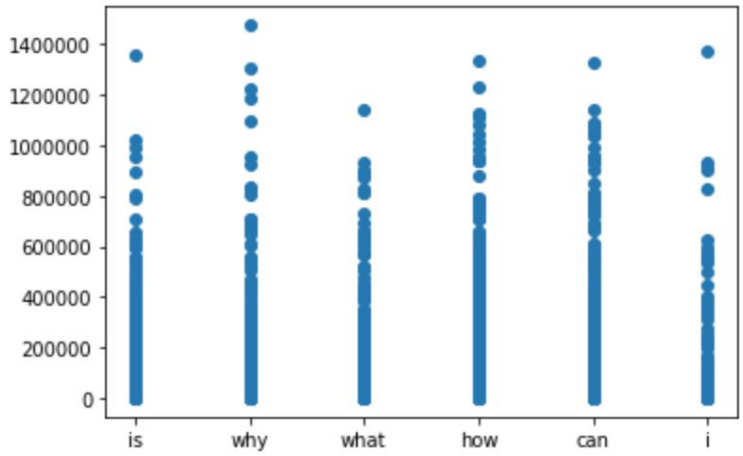


Fig 7: Distribution of time taken for an answer vs first word of the question

The boxplot above illustrates how long it takes for questions to get an answer when categorized by their beginning word. We observe the questions beginning with 'why' has the largest distribution, while questions beginning with 'I' show the smallest distribution, with an outlier by a large margin.

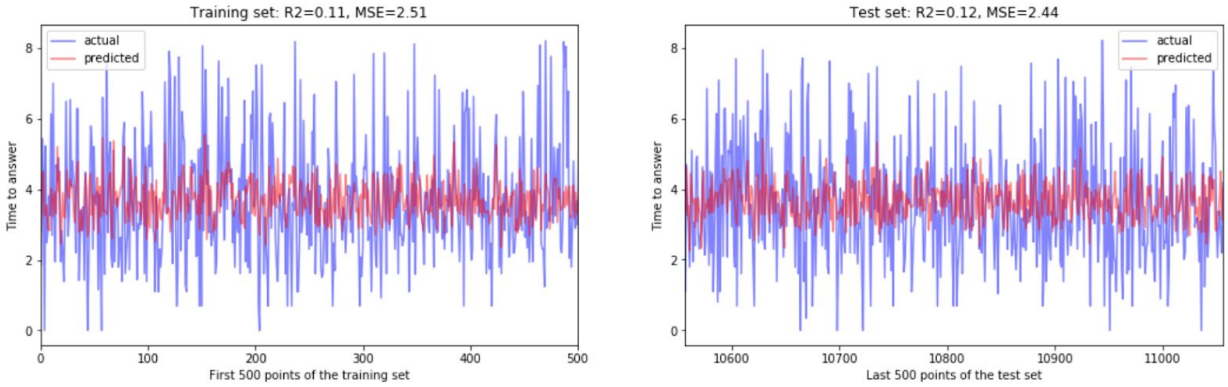
Chapter 4

4.0 Machine Learning Models

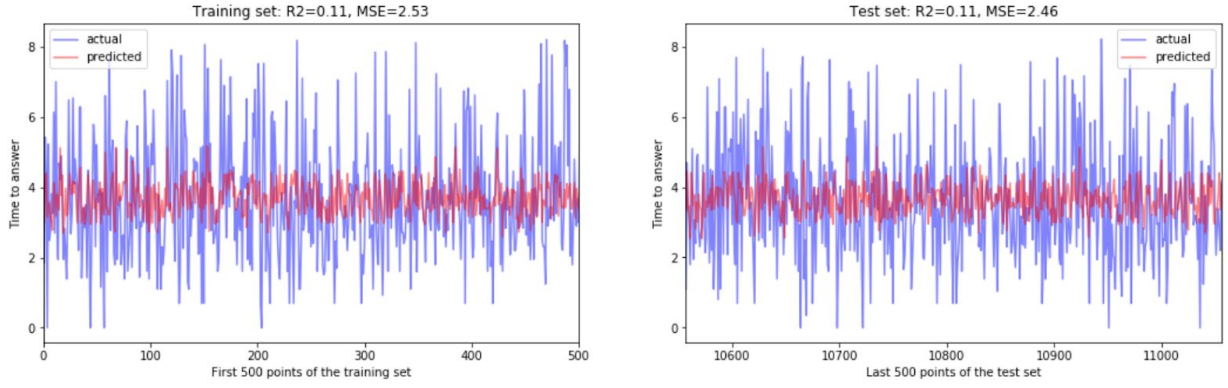
The dataset was split into an 80% train and a 20% test set. For this regression analysis, we focus on linear and tree-based Machine Learning algorithms as the features are not on the same scale. Standardization, hyperparameter tuning, and 5-fold cross-validation were applied to each model. The model accuracy was measured by calculating the R2 value and mean squared error (MSE) for each model for the train and test set. Additionally, the actual and predicted values are plotted for the train and test set for each model.

4.1 Linear Models

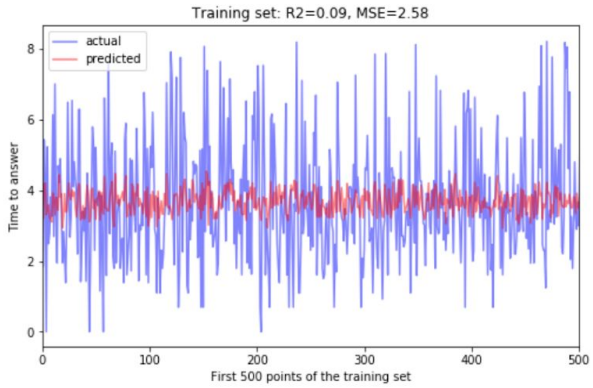
4.1.1 Linear Regression:

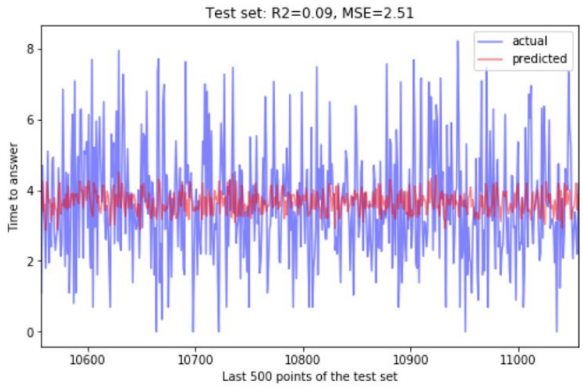


4.1.2 Linear Regression w/ Lasso Regularization:



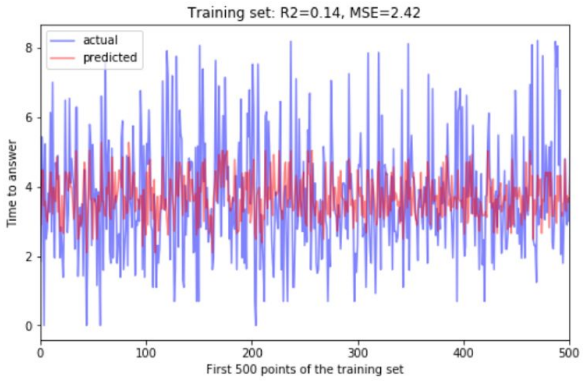
4.1.3 Linear Regression w/ Ridge Regularization:

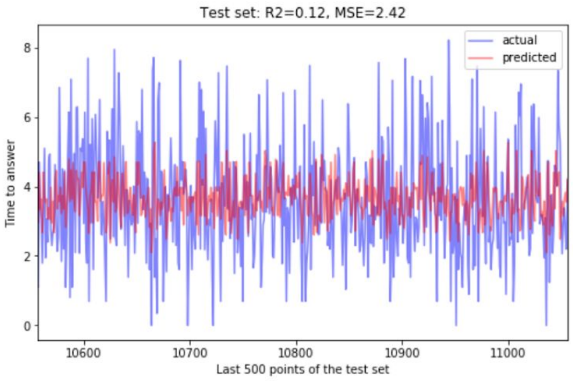




4.2 Tree-based models

4.2.1 Decision Tree:



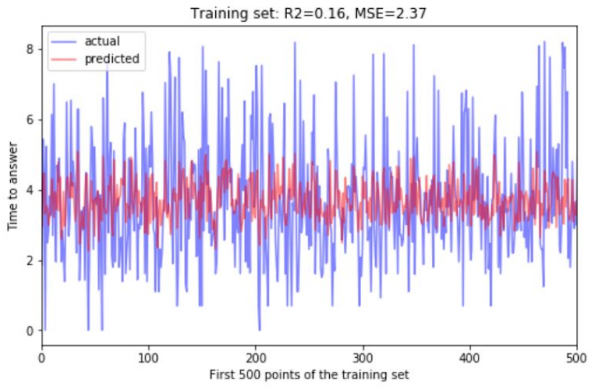


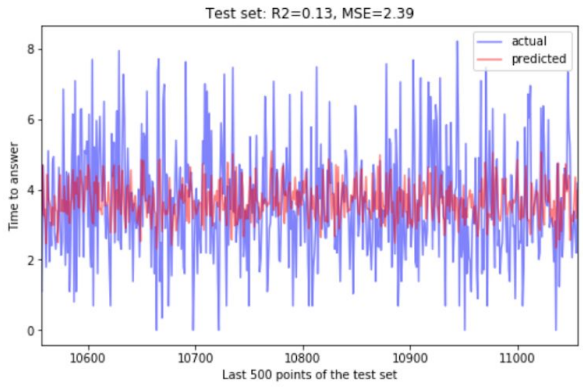
Best hyperparameters: {'dt_model__max_features': 'auto', 'dt_model__max_leaf_nodes': 150, 'dt_model__min_samples_leaf': 50}

Feature importances:

C_long	0.331117
C_machine-learning	0.157824
C_mysql	0.108888
C_short	0.088699
hour_utc	0.054360
C_how	0.051054
account_creation_year	0.044616
C_why	0.036593
questions	0.035504
C_python	0.019148
chance_of_answer	0.015342
C_Saturday	0.008093
C_Sunday	0.007937
C_r	0.006220
C_what	0.005969
ends_question	0.005026
C_data-science	0.004620
C_i	0.003883
C_Thursday	0.003566
C_is	0.003235
C_Friday	0.003168
C_Tuesday	0.001973
C_can	0.001386
C_Wednesday	0.001005
C_Monday	0.000772
C_medium	0.000000

4.2.2 Random Forest:



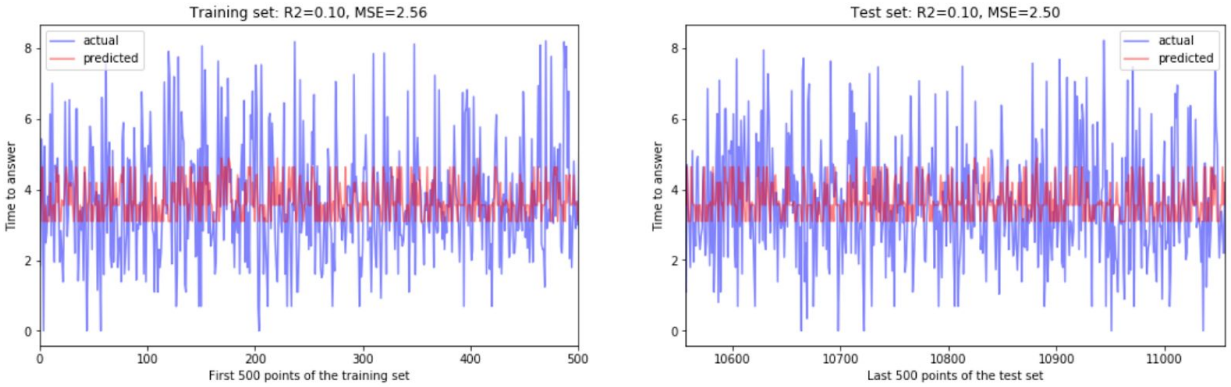


Best hyperparameters: {'rf_model__max_features': 'sqrt', 'rf_model__min_samples_leaf': 30, 'rf_model__n_estimators': 300}

Feature importances:

C_long	0.199185
C_short	0.130256
C_machine-learning	0.113168
C_mysql	0.083648
hour_utc	0.072842
account_creation_year	0.058029
C_medium	0.051555
questions	0.049044
C_how	0.040040
C_python	0.028548
C_why	0.027599
chance_of_answer	0.023812
C_r	0.021820
ends_question	0.015269
C_Saturday	0.009280
C_is	0.009179
C_i	0.008517
C_what	0.008455
C_Sunday	0.007849
C_Tuesday	0.007241
C_Wednesday	0.006952
C_can	0.006647
C_Friday	0.006477
C_Monday	0.005829
C_Thursday	0.005824
C_data-science	0.002936

4.2.3 Adaboosting:

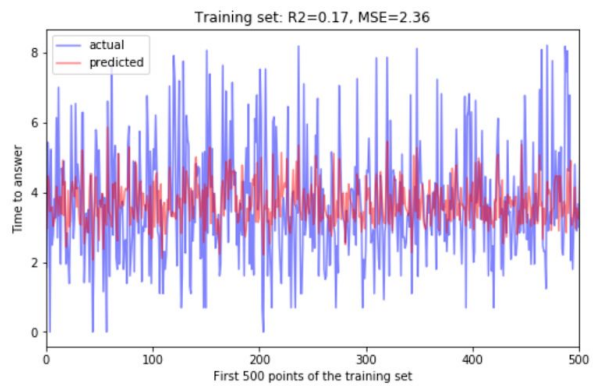


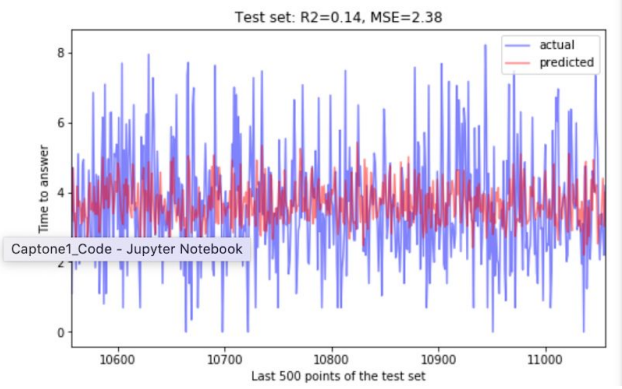
Best hyperparameters: {'ada_model__learning_rate': 0.001, 'ada_model__n_estimators': 100}

Feature importances:

```
C_long                  0.495559
C_machine-learning      0.226740
C_short                 0.131044
C_mysql                 0.098747
C_how                   0.038777
hour_utc                0.006794
account_creation_year   0.000762
C_Sunday                0.000327
chance_of_answer        0.000234
C_Thursday              0.000185
C_Friday                0.000183
C_i                     0.000172
questions               0.000138
C_is                    0.000100
C_Tuesday               0.000066
C_what                  0.000056
C_Saturday              0.000047
C_Wednesday             0.000044
C_why                   0.000025
C_medium                0.000000
C_python                0.000000
C_r                     0.000000
C_data-science          0.000000
C_can                   0.000000
ends_question           0.000000
C_Monday                0.000000
```

4.2.4 Gradient Boosting:





Best hyperparameters: {'gb_model__learning_rate': 0.01, 'gb_model__max_depth': 5, 'gb_model__n_estimators': 1000}

Feature importances:

C_long	0.281056
C_machine-learning	0.137722
C_mysql	0.094009
hour_utc	0.073104
account_creation_year	0.063368
C_short	0.054355
C_how	0.042681
questions	0.041098
C_why	0.031796
C_medium	0.029947
chance_of_answer	0.026769
C_python	0.016964
C_Saturday	0.011681
C_can	0.011165
C_r	0.010420
C_Sunday	0.009492
ends_question	0.009449
C_data-science	0.009076
C_is	0.008477
C_i	0.007943
C_what	0.006979
C_Tuesday	0.006558
C_Friday	0.005793
C_Wednesday	0.003599
C_Monday	0.003417
C_Thursday	0.003082

4.3 Model Metrics

	r2_train	r2_test	mse_train	mse_test
LinearRegression	0.112513	0.115753	2.51319	2.44304
Lasso	0.107216	0.110074	2.52819	2.45873
Ridge	0.0890805	0.0907337	2.57955	2.51216
DecisionTreeRegressor	0.144187	0.122418	2.4235	2.42462
RandomForestRegressor	0.164307	0.133187	2.36652	2.39487
AdaBoostRegressor	0.096405	0.0963203	2.55881	2.49673
GradientBoostingRegressor	0.168331	0.137636	2.35513	2.38258

4.5 Results and Discussion

In this project, we have picked the most important features to predict how long it will take to answer your Stack Overflow questions. Out of the seven regression models that were used, the Gradient Boosting algorithm displays the highest accuracy with the highest R-squared value (R-squared is a statistical measure of how close the data are to the fitted regression line; R-squared = Explained variation / Total variation; the higher the R-squared, the better the model fits your data) and lowest mean squared error, for both train and test sets. However, this model took almost 5.5 hours to complete. On the other hand, the Random Forest model shows only 1.2% decrease in accuracy (w.r.t mean-squared-error) but was executed in 5 mins. Therefore, perhaps efficiency might be of more value than the small gain in accuracy.

Furthermore, by analyzing the importance of different features, the top three features that affect the time it would take for an answer for a particular question seems to be: length of the question, question tag and the time of day the question is posted, respectively.

Chapter 5

Conclusion

In conclusion, we have successfully developed a model to predict the response rate for a question posted on Stack Overflow. Stack Overflow is a platform used by millions of computer scientists worldwide, and usually post questions while working on a project, thus anticipating an answer as soon as possible. It is helpful to know if there are some aspects of the question that would lead to a faster helpful answer, thus enabling the poster to design the question as such. We have evaluated multiple supervised machine learning algorithms, and evaluated each model for accuracy and efficiency. We can conclude that the length of the question, question tag and the time of day the question is posted have the greatest impact on how long it would take to get an answer.