Predictive Analysis of the response rate of Stack Overflow on 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.

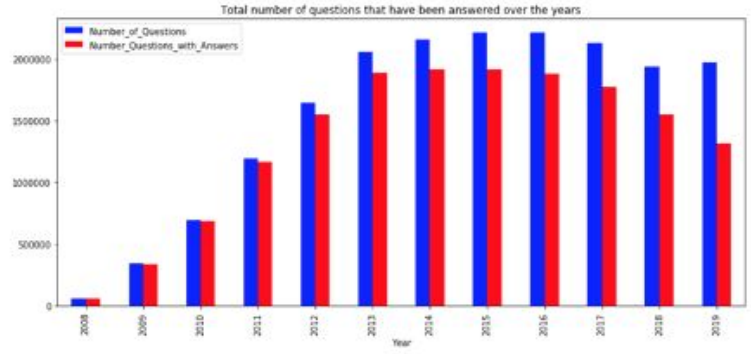


Fig 1: Histogram representing the total number of questions posted on Stack Overflow and the number of questions with answers for each year

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.

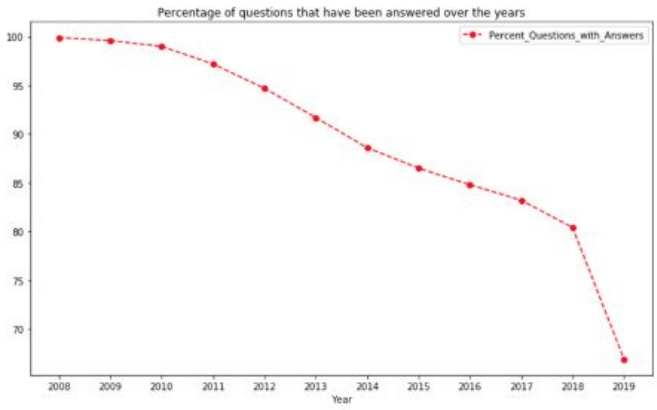


Fig 2: Illustration of the percentage of questions with answers for each year

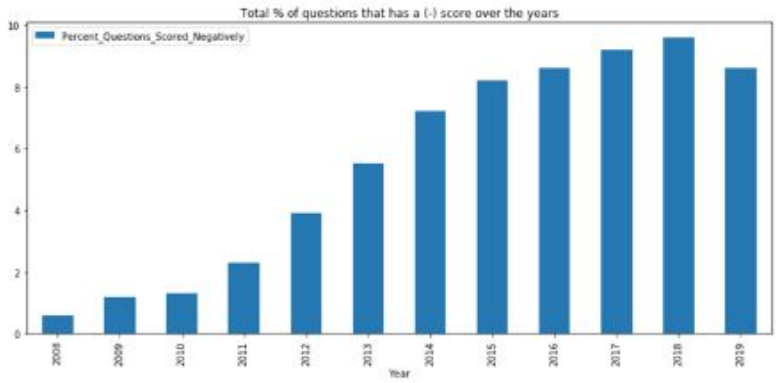


Fig 3: Percentage of questions that have been negatively scored for each year

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.

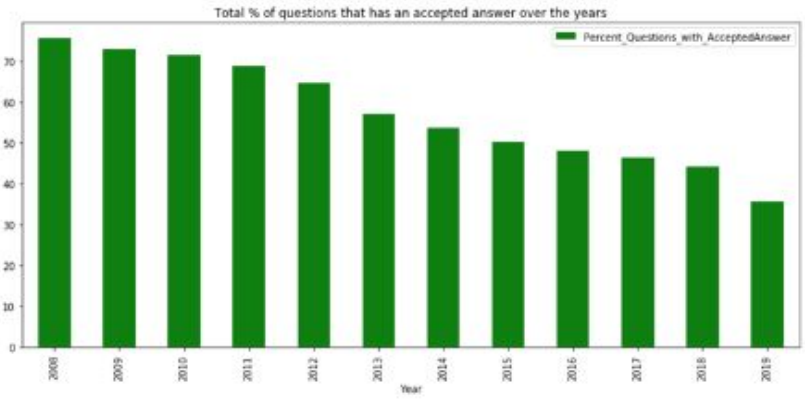


Fig 4: Percentage of questions with an accepted answer for each year

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.

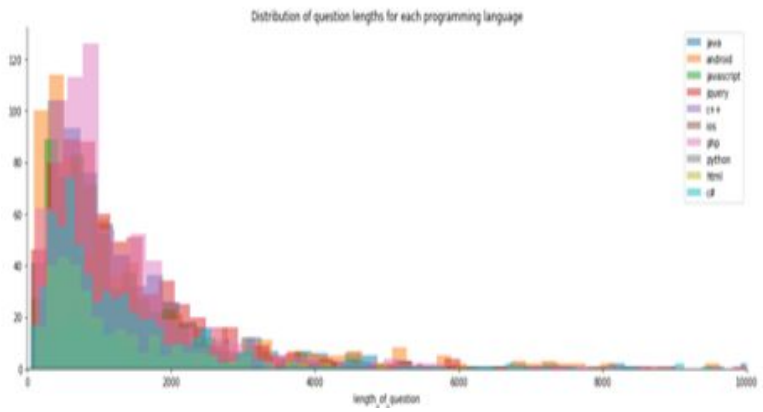


Fig 5: Distribution of question length for the top ten most popular tags

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'
3) Questions posted in 2017 and later
4) Only the questions which received answers within a month are considered

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 contained ~240k rows was saved as csv file, which was subsequently read as a Pandas DataFrame. This dataset was acquired from the SQL query listed below:

```
WITH answers AS (
SELECT *,
REGEXP_EXTRACT(LOWER(title), '[a-z]+') first_word,
SUBSTR(title, LENGTH(title))='?' ends_question,
GREATEST(1, TIMESTAMP_DIFF(answers.first, creation_date, minute)) first_answer_minutes,
answers.c > 0 was_answered
FROM (
SELECT creation_date, title,
CASE
WHEN LENGTH(body)<700 THEN 'short'
WHEN LENGTH(body)<1700 THEN 'medium'
ELSE 'long'
END question_length,
(SELECT MIN(creation_date) first, COUNT(*) c
FROM bigquery-public-data.stackoverflow.posts_answers
WHERE a.id=parent_id ) answers,
(SELECT EXTRACT(year FROM creation_date) account_creation_year
FROM bigquery-public-data.stackoverflow.users
WHERE a.owner_user_id=id ) user,
SPLIT(tags, '|') tags,
comment_count,
view_count
FROM
bigquery-public-data.stackoverflow.posts_questions a ))
```

```
  SELECT
  EXTRACT(year FROM creation_date) AS question_posted,
  first_word,
  ends_question,
  tag,
  view_count,
  comment_count,
  FORMAT_TIMESTAMP('%H', creation_date) hour_utc,
  FORMAT_TIMESTAMP('%A', creation_date) weekday_utc,
  user.account_creation_year,
  question_length, first_answer_minutes
  FROM
  answers,
  UNNEST(tags) tag
  WHERE
  tag IN ('python','r','mysql','machine-learning','data-science')
  AND first_word IN UNNEST(['how','is','why','what','can', 'i'])
  AND creation_date >= timestamp('2017-01-01')
  AND first_answer_minutes<=43800
  --1 month
```

3.1 Data cleaning and Handling missing values

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

>> data.info()

```
RangeIndex: 243178 entries, 0 to 243177
Data columns (total 11 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   question_posted      243178 non-null  int64
 1   first_word           243178 non-null  object
 2   ends_question        243178 non-null  bool
 3   tag                  243178 non-null  object
 4   view_count           243178 non-null  int64
 5   comment_count        243178 non-null  int64
 6   hour_utc             243178 non-null  int64
 7   weekday_utc          243178 non-null  object
 8   account_creation_year  238825 non-null  float64
 9   question_length      243178 non-null  object
 10  first_answer_minutes  243178 non-null  int64
dtypes: bool(1), float64(1), int64(5), object(4)
```

We observe some null values for the ‘ account_creation_year’ feature. Null values were probably due to a data logging error.

>> data.isnull().sum()

```
question_posted          0
first_word               0
ends_question            0
tag                      0
view_count               0
comment_count            0
hour_utc                 0
weekday_utc              0
account_creation_year    4353
question_length          0
first_answer_minutes     0
dtype: int64
```

3.2 Exploratory Data Analysis

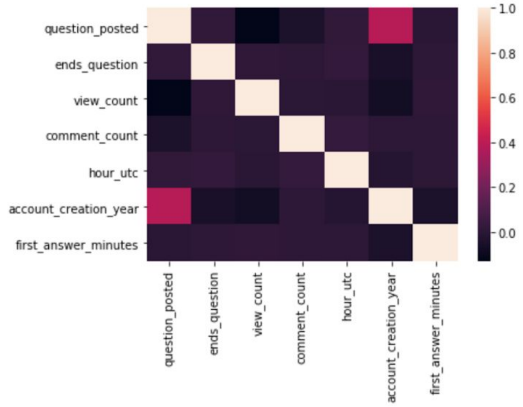
Let's take a look at the numerical data distribution.

>>data.describe()

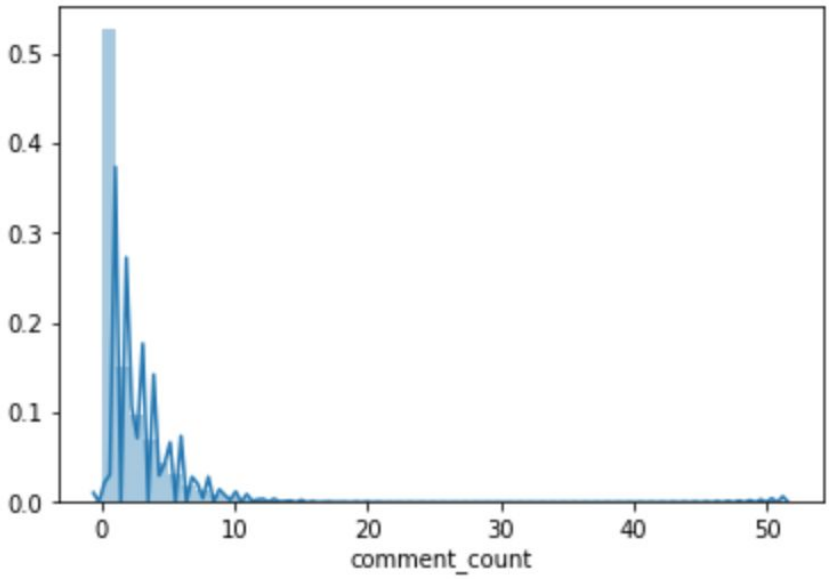
	question_posted	view_count	comment_count	hour_utc	account_creation_year	first_answer_minutes
count	238825.000000	238825.000000	238825.000000	238825.000000	238825.000000	238825.000000
mean	2018.461430	558.580383	2.042994	12.310847	2016.603576	756.950872
std	1.023267	3563.192185	2.582340	6.263594	2.502777	3400.491044
min	2017.000000	5.000000	0.000000	0.000000	2008.000000	1.000000
25%	2018.000000	41.000000	0.000000	8.000000	2015.000000	7.000000
50%	2019.000000	72.000000	1.000000	13.000000	2017.000000	22.000000
75%	2019.000000	279.000000	3.000000	17.000000	2019.000000	115.000000
max	2020.000000	479597.000000	51.000000	23.000000	2020.000000	43784.000000

We see that 50% of the questions get a response within 22 minutes and for 75% of the questions within 2 hours.

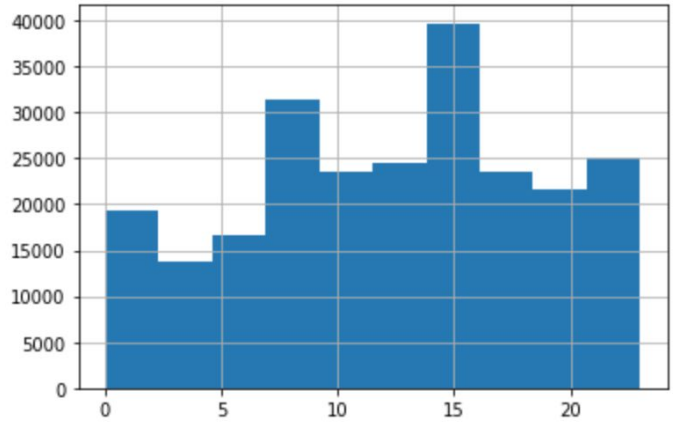
	question_posted	ends_question	view_count	comment_count	hour_utc	account_creation_year	first_answer_minutes
question_posted	1.000000	0.019314	-0.130346	-0.042280	0.017934	0.387422	-0.004301
ends_question	0.019314	1.000000	0.011625	0.003311	0.021084	-0.053595	0.008166
view_count	-0.130346	0.011625	1.000000	0.001896	-0.003321	-0.070637	0.012859
comment_count	-0.042280	0.003311	0.001896	1.000000	0.028429	0.005547	0.008597
hour_utc	0.017934	0.021084	-0.003321	0.028429	1.000000	-0.019836	0.002377
account_creation_year	0.387422	-0.053595	-0.070637	0.005547	-0.019836	1.000000	-0.042989
first_answer_minutes	-0.004301	0.008166	0.012859	0.008597	0.002377	-0.042989	1.000000



The correlation map between the numerical features, we don't see strong correlations between any features. Account_creation_year and no. of questions posted seems to be somewhat correlated. The response variable shows weak correlation to the user account creation year.



From the density plot of the number of comments above, we can see that most of the questions have less than 10 comments/answers. More specifically, 75% of the questions have about 3 or less number of comments.

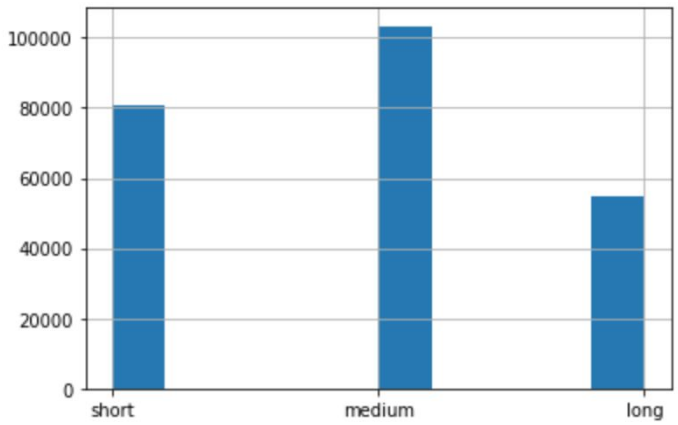


We see an increase in the number of questions posted from mid morning till late night, with a peak the the mid afternoon. We see that 50% of the questions get asked before 1pm and 75% of the questions get asked before 5pm.

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

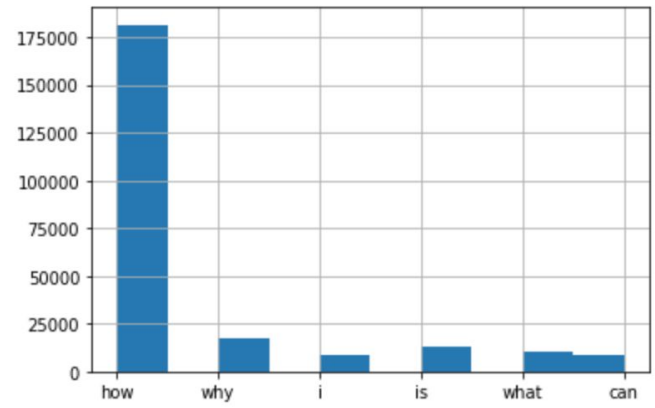
```
: data.weekday_utc.value_counts(normalize=True)
```

```
: Wednesday    0.168383
  Thursday     0.165963
  Tuesday      0.162898
  Friday       0.151400
  Monday       0.150457
  Sunday       0.101170
  Saturday     0.099730
  Name: weekday_utc, dtype: float64
```

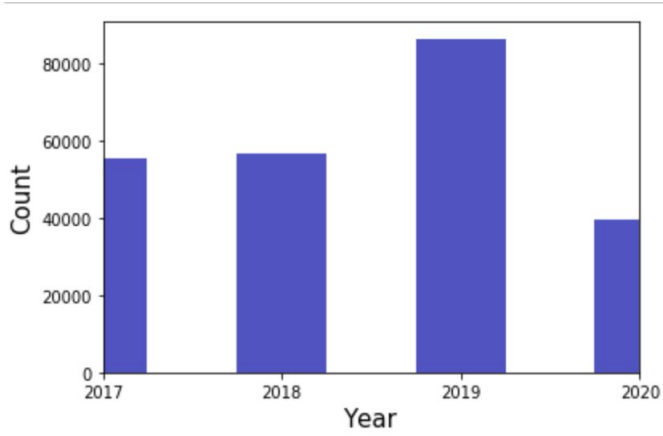


Out of the selected tags, we see that Python questions are most common, while not many questions are tagged under ‘data-science’. Furthermore, we see below that many questions under ‘data-science’ and ‘machine-learning’ tags are short, while all the other tags under consideration have medium length questions.

```
tag                question_length
data-science       short              0.426349
                   medium             0.397987
                   long               0.175663
machine-learning   short              0.380033
                   medium             0.369884
                   long               0.250083
mysql              medium             0.414755
                   short              0.329801
                   long               0.255444
python             medium             0.433623
                   short              0.345938
                   long               0.220439
r                  medium             0.457849
                   short              0.300967
                   long               0.241184
Name: question_length, dtype: float64
```



A significant majority of the questions start with 'how', while the others are on an order of magnitude.



We observe a gradual increase in the number of question posted with the highest so far in 2019. It should be noted that only the first 6 months on 2020 are included.

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 238825 rows and 28 columns.

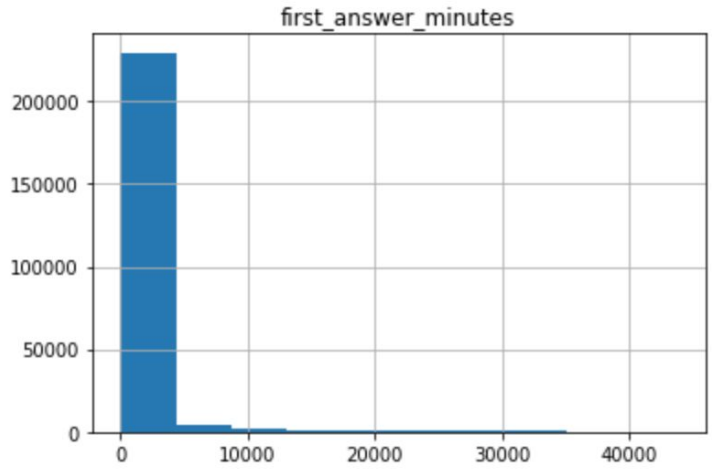
Chapter 4

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. Standard scaling, hyperparameter tuning, and 5-fold cross-validation were applied to each model. The model accuracy was measured by calculating the root mean squared error (RMSE) value and mean absolute 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. The following machine learning algorithms were applied:

1) Linear Regression
2) Lasso Regression
3) Ridge Regression
4) Decision Tree Regression
5) Random Forest Regression

Let's take a look at the distribution of the response variable:

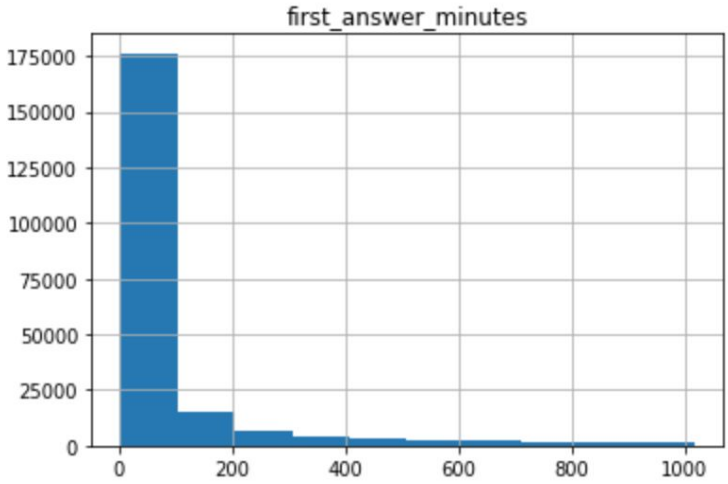


The response variable distribution is extremely right skewed. In order to mitigate the effects of these skewness, the logarithm transformation of the response variable will be considered. We know that 50% of the questions get answered within 20 minutes or so, while 75% of the questions get answered in less than 2 hours. First, we will apply the ML algorithms to the entire dataset and then we will repeat the modeling after removing outliers above the 90th percentile. This is appropriate as Stack Overflow users typically post questions while working on a project and expect not having to wait for over a month for a reply.

	rmse_train	rmse_test	mae_train	mae_test
LinearRegression	3462.06	3507.76	747.07	749.793
Lasso	3462.44	3508.15	747.011	749.696
Ridge	3462.3	3508	747.02	749.73
DecisionTreeRegressor	3458.6	3505.19	745.876	748.992
RandomForestRegressor	3454.31	3505.39	740.22	748.373

We got poor model performance when considering the entire dataset. This is intuitive given that over 75% of the questions get answered within 115 minutes, while the response variable ranges over 43,800 minutes with an extreme right skewed distribution.

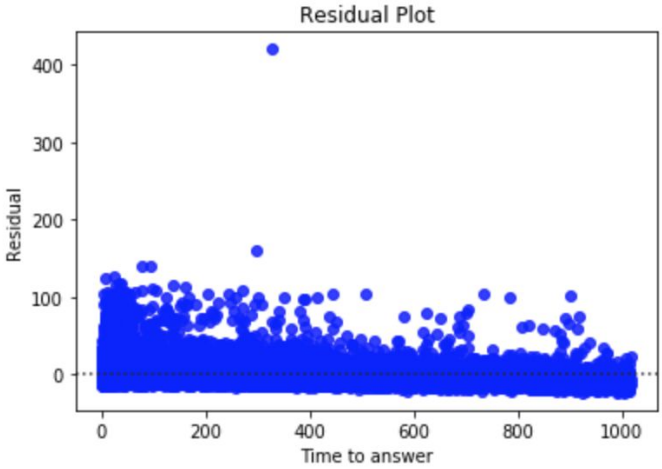
Now, let's consider the dataset after the outliers have been handled. The 90th percentile falls at 1018 minutes (~17 hours). The distribution of the response variable post outlier removal is still extremely right skewed.

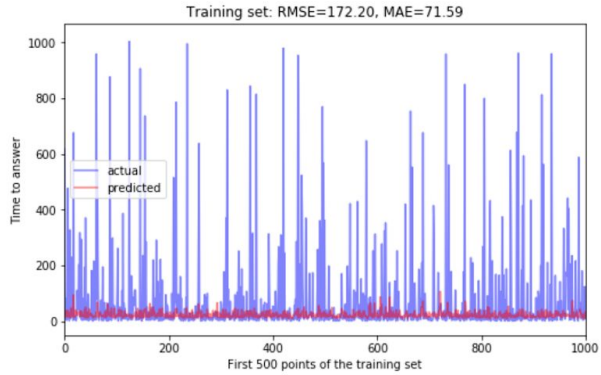


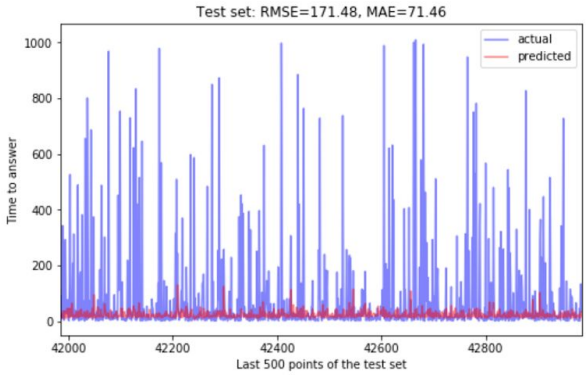
In the following sessions, the actual and predicted values for a portion of train and test data sets along with the residual plots are illustrated.

4.1 Linear Models

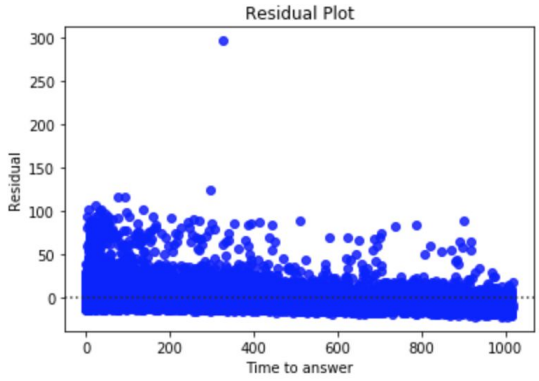
4.1.1 Linear Regression:

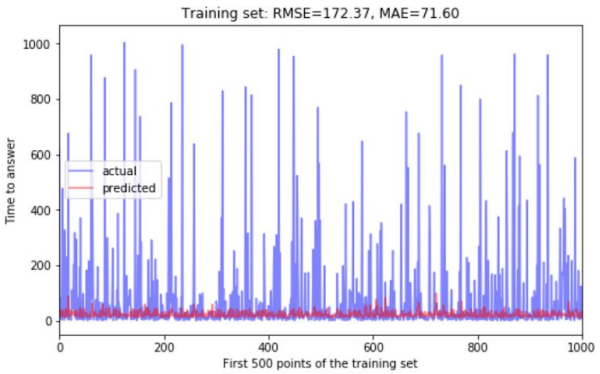


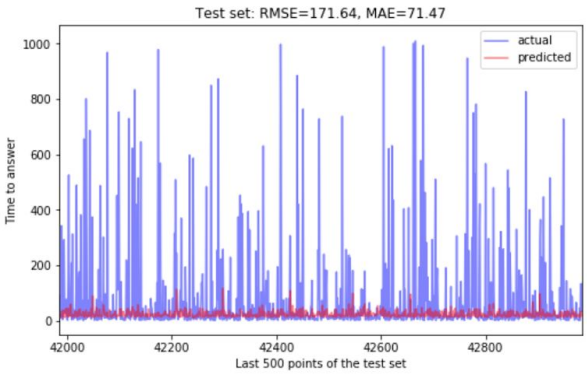




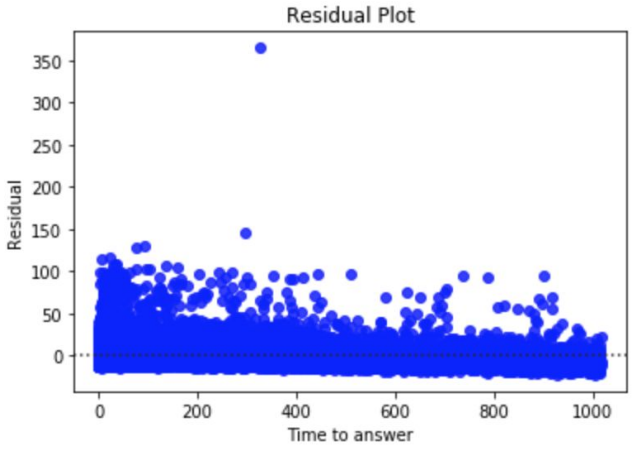
4.1.2 Linear Regression w/ Lasso Regularization:

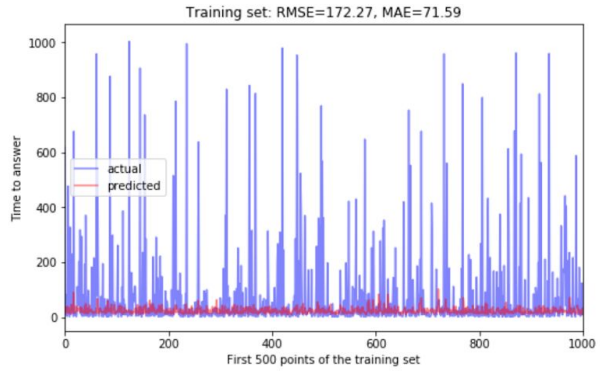


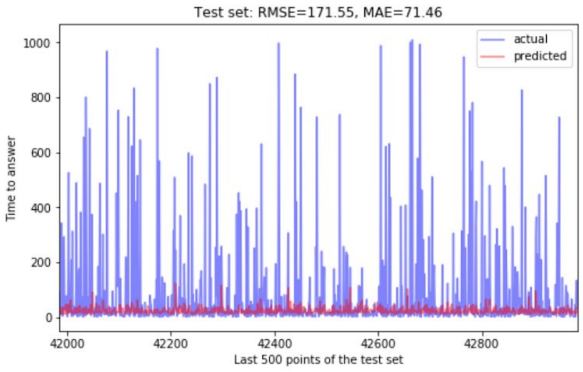




4.1.3 Linear Regression w/ Ridge Regularization:

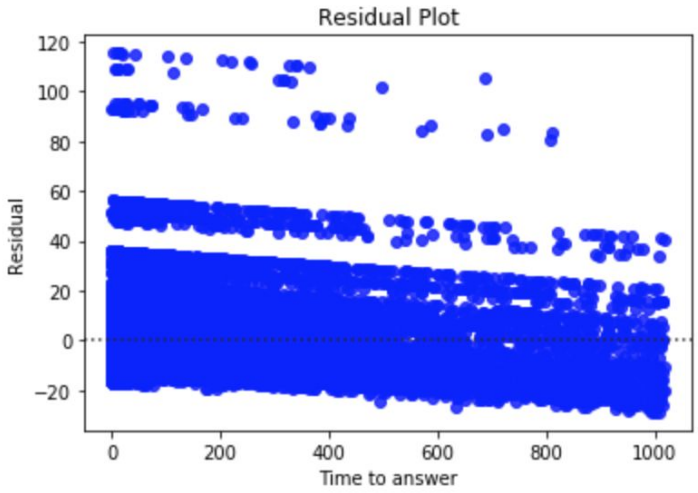


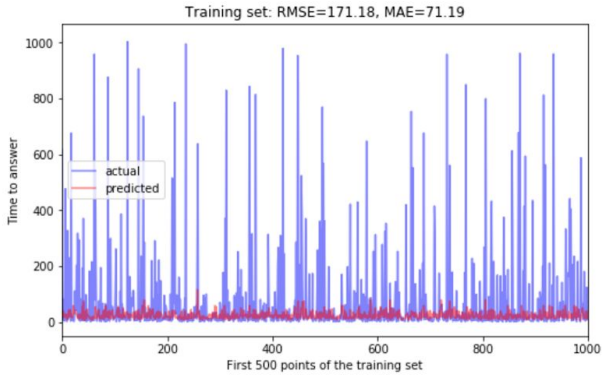


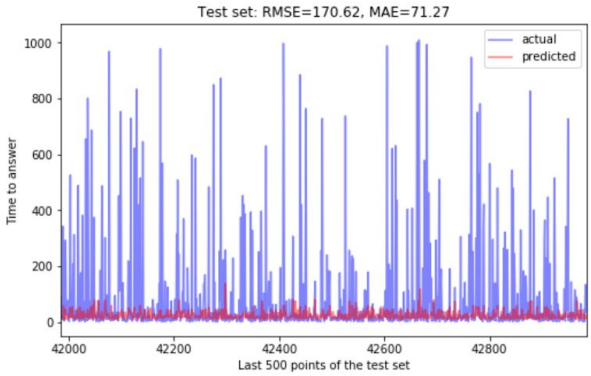


4.2 Tree-based models

4.2.1 Decision Tree:

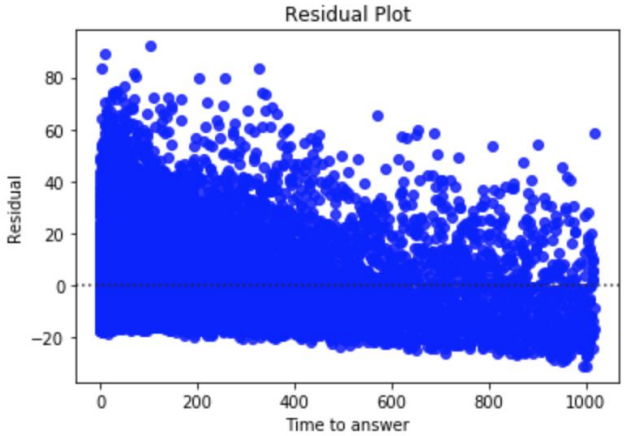


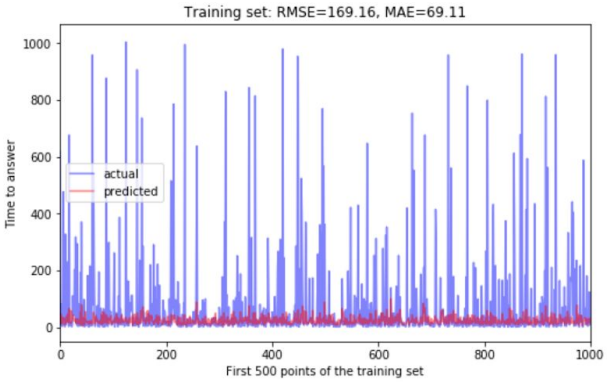


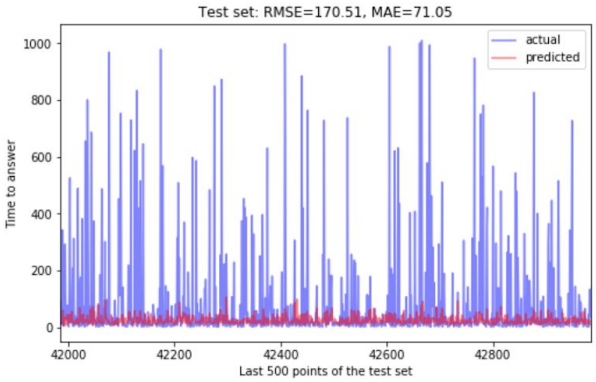


Best hyperparameters: {'dt_model__max_features': 'auto', 'dt_model__max_leaf_nodes': 150, 'dt_model__min_samples_leaf': 50}

4.2.2 Random Forest:







Best hyperparameters: {'rf_model__max_features': 'sqrt', 'rf_model__min_samples_leaf': 10, 'rf_model__n_estimators': 300}

Feature importances:

view_count	0.237805
hour_utc	0.137114
C_long	0.103029
comment_count	0.080505
C_short	0.076324
account_creation_year	0.073867
question_posted	0.047582
C_machine-learning	0.036875
C_medium	0.032513
ends_question	0.022301
C_mysql	0.013829
C_python	0.013576
C_r	0.012740
C_Tuesday	0.011522
C_Wednesday	0.011428
C_Thursday	0.011275
C_Monday	0.011048
C_how	0.010888
C_Friday	0.010452
C_Saturday	0.009894
C_Sunday	0.009367
C_why	0.009027
C_is	0.005512
C_can	0.003963
C_what	0.003731
C_i	0.002711
C_data-science	0.001122

4.3 Model Metrics

	rmse_train	rmse_test	mae_train	mae_test
LinearRegression	172.202	171.483	71.588	71.4566
Lasso	172.372	171.639	71.6015	71.4651
Ridge	172.274	171.548	71.5868	71.4551
DecisionTreeRegressor	171.179	170.62	71.1881	71.2653
RandomForestRegressor	169.162	170.51	69.1129	71.0462

4.5 Results and Discussion

We have evaluated multiple supervised machine learning algorithms, and evaluated each model for accuracy and efficiency. The random forest algorithm gives the best results with lowest MAE and RMSE. As we are trying to predict how long it would take for a question to get an answer, the error scales linearly, where MAE is a good metric. Hence, with the random forest model, we can predict with time with an error of one hour. Given the range of values of the response variable, this is excellent.

Additionally, there seems to be overfitting on a very small scale, which can be addressed by increasing the number of cross validation folds. We can conclude that the view count, the time of day the question is posted and length of the question have the greatest impact on how long it would take to get a response, while start_word and tag seem to have the lowest impact.

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. By identifying these features, can perhaps flagging the user to 'improve' the question just before posting it, will lead to faster response times. Thi will lead to higher user engagement, thus increasing the revenue for Stack Overflow via pushing more ads to users.

Future Work

This project can also be addressed as a classification task, where questions that get answered within the first 30 minutes are classified as ‘quick’, <2 hours as ‘medium’ and >2 hours as ‘slow’ response times. Additionally, the body of the question can be analyzed further using NLP techniques, instead of only considering the first word of the questions.