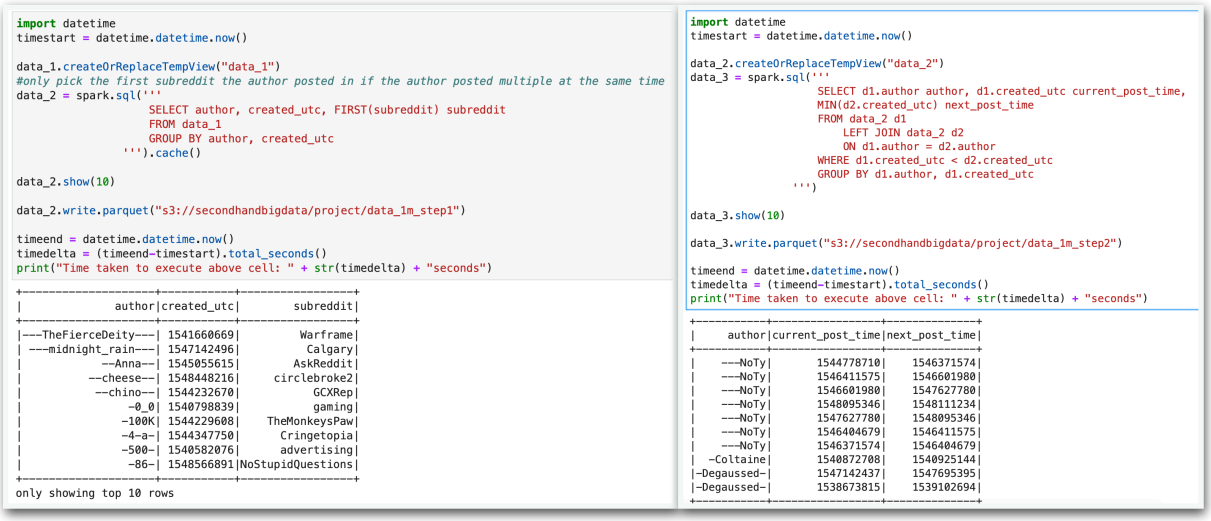
There are many features in the reddit comment dataset. Feature author, author_cakeday, author_created_utc, author_flair_background_color, author_flair_css_class, author_flair_richtext, author_flair_template_id, author_flair_text, author_flair_text_color, author_flair_type, author_fullname, and author_patreon_flair are the features which have the users information. Body feature consists the body of comments. Archived, can_gild, can_mod_post, collapsed, distinguished, edited, gilded, is_submitter, no_follow, and stickied are some boolean type features which can be explained by their names. Subreddit, subreddit_id and subreddit_name_prefixed consist the information of the subreddit the comment/post is in. Subreddit_type shows whether it is private or public. Score is the difference of likes and dislikes of the comment. Send_replies shows the number of replies each comment has. Controversiality indicates the controversiality of the comment/post in integers. Created_utc indicates the created time of the comment/post in unix time. Retrieved_on shows the retrieved time of the comment/post in unix time.

When I first saw the the dataset, off the top of my head, I thought it would be interesting to discover what subreddit each user/row would next engage with. And at what time would they post the next comment/post. The following analysis would be explaining the process of data cleaning, data visualization, modeling and prediction.

Since I was only allowed 4 m4.xlarge core instances, I sampled the huge reddit dataset into 10 million rows. For data cleaning, first of all, I removed all the rows of deleted users ('[deleted]'), because I would be join tables using author feature as the anchor. Then I replaced all the null cells in categorical features with a new category "U" and replaced all the "True" and "False" with 1 and 0 in all the boolean type features, respectively. Lastly, I dropped features which are not so relevant, e.g. author_flair_css_class, for faster process.

To generate the features of next post time and next engaged subreddit, I joined the dataset with itself on author, and selected the least created_utc which was larger than current created_utc as next_post_time. And I made the current created_utc into current_post_time. Current_subreddit and next_subreddit were created in a similar method. Queries are shown in the screenshots below.



```
import datetime
timestart = datetime.datetime.now()

data_1.createOrReplaceTempView("data_1")
#only pick the first subreddit the author posted in if the author posted multiple at the same time
data_2 = spark.sql('''
                    SELECT author, created_utc, FIRST(subreddit) subreddit
                    FROM data_1
                    GROUP BY author, created_utc
                   ''').cache()

data_2.show(10)

data_2.write.parquet("s3://secondhandbigdata/project/data_1m_step1")

timeend = datetime.datetime.now()
timedelta = (timeend-timestart).total_seconds()
print("Time taken to execute above cell: " + str(timedelta) + "seconds")
```

```
+--------------------+-----------+-----------------+
|              author|created_utc|        subreddit|
+--------------------+-----------+-----------------+
|---TheFierceDeity---| 1541660669|         Warframe|
| ---midnight_rain---| 1547142496|          Calgary|
|           --Anna--| 1545055615|        AskReddit|
|         --cheese--| 1548448216|     circlebroke2|
|          --chino--| 1544232670|           GCXRep|
|              -0_0-| 1540798839|           gaming|
|             -100K-| 1544229608|    TheMonkeysPaw|
|              -4-a-| 1544347750|      Cringetopia|
|              -500-| 1540582076|      advertising|
|               -86-| 1548566891|NoStupidQuestions|
+--------------------+-----------+-----------------+
only showing top 10 rows
```

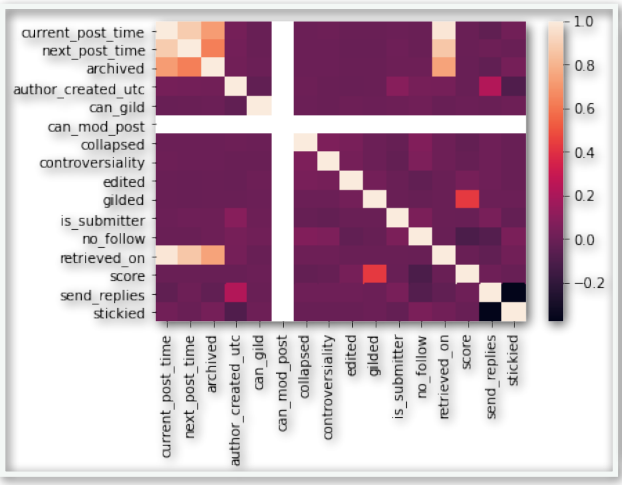
```
import datetime
timestart = datetime.datetime.now()

data_2.createOrReplaceTempView("data_2")
data_3 = spark.sql('''
                    SELECT d1.author author, d1.created_utc current_post_time,
                    MIN(d2.created_utc) next_post_time
                    FROM data_2 d1
                        LEFT JOIN data_2 d2
                        ON d1.author = d2.author
                    WHERE d1.created_utc < d2.created_utc
                    GROUP BY d1.author, d1.created_utc
                   ''')

data_3.show(10)

data_3.write.parquet("s3://secondhandbigdata/project/data_1m_step2")

timeend = datetime.datetime.now()
timedelta = (timeend-timestart).total_seconds()
print("Time taken to execute above cell: " + str(timedelta) + "seconds")
```

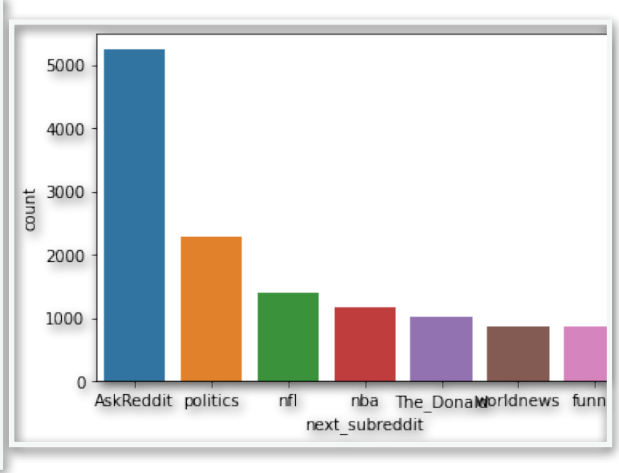
```
+----------+------------------+--------------+
|    author|current_post_time|next_post_time|
+----------+------------------+--------------+
|    ---NoTy|        1544778710|    1546371574|
|    ---NoTy|        1546411575|    1546601980|
|    ---NoTy|        1546601980|    1547627780|
|    ---NoTy|        1548095346|    1548111234|
|    ---NoTy|        1547627780|    1548095346|
|    ---NoTy|        1546404679|    1546411575|
|    ---NoTy|        1546371574|    1546404679|
|  -Coltaine|        1540872708|    1540925144|
|-Degaussed-|        1547142437|    1547695395|
|-Degaussed-|        1538673815|    1539102694|
+----------+------------------+--------------+
```

```
import datetime
timestart = datetime.datetime.now()

data_3.createOrReplaceTempView("data_3")
data_4 = spark.sql('''
                    SELECT d1.author author, d1.current_post_time current_post_time, d1.next_post_time next_post_time,
                    d2.subreddit current_subreddit, d3.subreddit next_subreddit,
                        d2.archived archived, d2.author_created_utc author_created_utc,
                        d2.body body, d2.can_gild can_gild, d2.can_mod_post can_mod_post,
                        d2.collapsed collapsed, d2.controversiality controversiality,
                        d2.edited edited, d2.gilded gilded,
                        d2.is_submitter is_submitter, d2.no_follow no_follow,
                        d2.retrieved_on retrieved_on, d2.score score, d2.send_replies send_replies,
                        d2.stickied stickied, d2.subreddit_type subreddit_type
                    FROM data_3 d1
                        LEFT JOIN data_1 d2
                        ON d1.author = d2.author AND d1.current_post_time = d2.created_utc
                        LEFT JOIN data_1 d3
                        ON d1.author = d3.author AND d1.next_post_time = d3.created_utc
                ''')

data_4.show(10)

data_4.write.parquet("s3://secondhandbigdata/project/data_1m_step3")

timeend = datetime.datetime.now()
timedelta = (timeend-timestart).total_seconds()
print("Time taken to execute above cell: " + str(timedelta) + "seconds")
```

```
+--------------------+----------------+--------------+--------------------+--------------+--------+-------------------+------
-------------+--------+------------+---------+---------------+------+------+------------+---------+-----------+-----+--
---------+--------+-------------+
|              author|current_post_time|next_post_time|current_subreddit|next_subreddit|archived|author_created_utc|
body|can_gild|can_mod_post|collapsed|controversiality|edited|gilded|is_submitter|no_follow|retrieved_on|score|send_replies
|stickied|subreddit_type|
```

In order to find out which features are best to use to predict next engaged subreddit and time, I had to find out the correlations of each feature. I transferred the dataframe into pandas dataframe using .toPandas() function. I chose pandas because it was easier to do data visualization using pandas. I created the correlation plot for next_post_time (shown in the heat map below). According to the heat map, cunrrent_post_time, archived and retrieved_on showed strong correlations to next_post_time. Therefore, I would be using them as the features in the linear regression model. The reason that I chose linear regression was because the output was numerical. Fun fact: AskReddit is the largest subreddit by the number of engaged users.

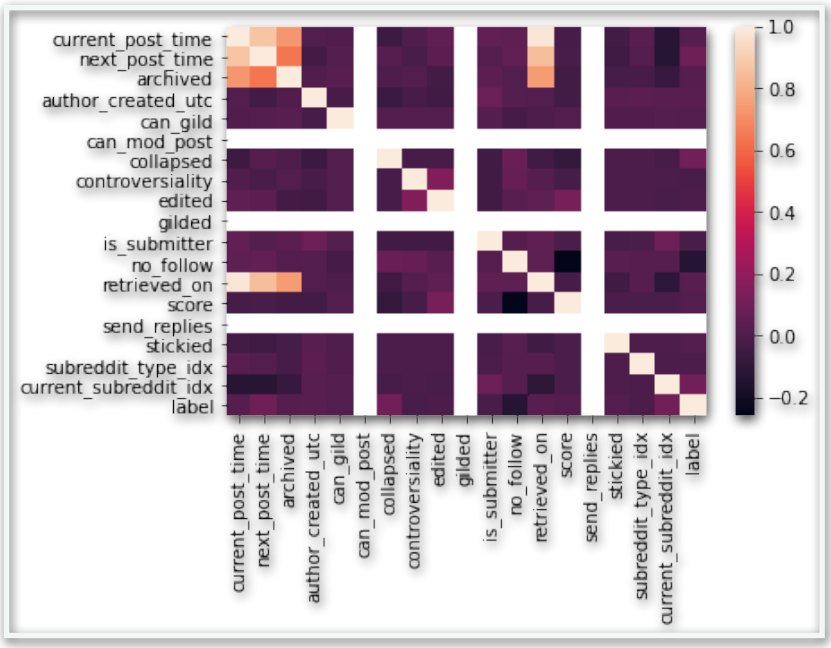




The results of the linear regression model are shown below. The R squares for the train set and test set were both around 0.782 which is pretty close to 1. And the RMSEs for train set and test set were around 1368000 which is almost nothing comparing to 9 digit next_post_time feature. Given the error analysis above, the linear regression model on predicting next_post_time is acceptable. If I transformed the unix time format into date and hour, the prediction would be more accurate.

```
-------------------------------------------------
Number of training records:  18232275
Number of testing records :  7812770
-------------------------------------------------
RMSE: 1369265.547406
r2: 0.781717
-------------------------------------------------
R Squared (R2) on test data = 0.781991
Root Mean Squared Error (RMSE) on test data = 1.36832e+06
-------------------------------------------------
+--------------------+--------------+--------------------+
|          prediction|next_post_time|            features|
+--------------------+--------------+--------------------+
|1.5483137741088765E9|    1548369936|[1.548096415E9,1....|
| 1.545978059971857E9|    1545266512|[1.545266414E9,0....|
| 1.545978059971857E9|    1545266512|[1.545266414E9,0....|
| 1.545978059971857E9|    1545266512|[1.545266414E9,0....|
|1.5425009890438209E9|    1542840143|[1.541086767E9,0....|
|1.5451428507458239E9|    1545189283|[1.54422644E9,0.0...|
|1.5451428507458239E9|    1545189283|[1.54422644E9,0.0...|
|1.5451428507458239E9|    1545189283|[1.54422644E9,0.0...|
|1.5480891356662362E9|    1548689139|[1.547822511E9,1....|
|1.5406988679331768E9|    1539670210|[1.5393293E9,0.0,...|
|1.5454126699173648E9|    1545065497|[1.544562532E9,0....|
| 1.548148423772936E9|    1548609260|[1.547400646E9,0....|
| 1.548148423772936E9|    1548609260|[1.547400646E9,0....|
|1.5402695866756074E9|    1539016085|[1.538789556E9,0....|
| 1.541126270024477E9|    1545948641|[1.539867065E9,0....|
| 1.541126270024477E9|    1545948641|[1.539867065E9,0....|
| 1.546892139797759E9|    1547239773|[1.546351734E9,1....|
|1.5469269051254852E9|    1546403767|[1.546394461E9,1....|
|1.5451486795480993E9|    1544633047|[1.544233638E9,0....|
|1.5451486795480993E9|    1544633047|[1.544233638E9,0....|
+--------------------+--------------+--------------------+
only showing top 20 rows
```

Next, I would be predicting users' next engaged subreddit (next_subreddit) using logistic regression, random forest classifier and decision tree classifier because I would add in categorical features and the output was categorical as well. In order to add categorial features and text feature into the model, StringIndexer and Tokenizer are needed. I made subreddit_type_idx and current_subreddit_idx based on subreddit_type and current_subreddit using StringIndexer. And I made words based on body using Tokenizer. The heat map below was made using pandas as well. It showed that next_subreddit (label) did not have correlation with any other features.



However, I still did logistic regression model, random forest model and decision tree model to predict the next_subreddit. And the results are shown below. As shown, the accuracy of the three models were 0.44, 0.64 and 0.51, with the random forest model being the best. Only the random forest model passed the 0.6 threshold, which could be considered somewhat accurate.

```
import datetime
timestart = datetime.datetime.now()

from pyspark.ml.classification import LogisticRegression
lr = LogisticRegression(featuresCol = 'features', labelCol = 'label', maxIter=25)
lrModel = lr.fit(train_df)
predictions = lrModel.transform(test_df)
from pyspark.ml.evaluation import BinaryClassificationEvaluator
evaluator = BinaryClassificationEvaluator()
print('Area Under the Curve for LogisticRegression:', evaluator.evaluate(predictions))

timeend = datetime.datetime.now()
timedelta = (timeend-timestart).total_seconds()
print("Time taken to execute above cell: " + str(timedelta) + "seconds")
```

```
Area Under the Curve for LogisticRegression: 0.43569148936170216
Time taken to execute above cell: 331.153202seconds
```

```
import datetime
timestart = datetime.datetime.now()

from pyspark.ml.classification import RandomForestClassifier
rf = RandomForestClassifier(featuresCol = 'features', labelCol = 'label')
rfModel = rf.fit(train_df)
predictions = rfModel.transform(test_df)
from pyspark.ml.evaluation import BinaryClassificationEvaluator
evaluator = BinaryClassificationEvaluator()
print("Area Under the Curve for RandomForestClassifier: " + str(evaluator.evaluate(predictions, {evaluator.metricName: "areaUnderROC"})))

timeend = datetime.datetime.now()
timedelta = (timeend-timestart).total_seconds()
print("Time taken to execute above cell: " + str(timedelta) + "seconds")
```

```
Area Under the Curve for RandomForestClassifier: 0.637556561085973
Time taken to execute above cell: 274.083157seconds
```

```
import datetime
timestart = datetime.datetime.now()

from pyspark.ml.classification import DecisionTreeClassifier
dt = DecisionTreeClassifier(featuresCol = 'features', labelCol = 'label', maxDepth = 20)
dtModel = dt.fit(train_df)
predictions = dtModel.transform(test_df)
from pyspark.ml.evaluation import BinaryClassificationEvaluator
evaluator = BinaryClassificationEvaluator()
print("Area Under the Curve for DecisionTreeClassifier: " + str(evaluator.evaluate(predictions, {evaluator.metricName: "areaUnderROC"})))

timeend = datetime.datetime.now()
timedelta = (timeend-timestart).total_seconds()
print("Time taken to execute above cell: " + str(timedelta) + "seconds")
```

```
Area Under the Curve for DecisionTreeClassifier: 0.5074706510138741
Time taken to execute above cell: 612.715372seconds
```

In conclusion, the model used for predicting next_post_time was acceptable and the models used for predicting next_subreddit were somewhat feasible. For future work, I would apply some natural language process analysis on words feature. I would make an index based on the words used in the comment, which would probably be a better feature to use to predict next_subreddit or next_post_time. And I would do a network analysis on all the subreddits to find out the distance between each subreddit and create distance index between current_subreddit and next_subreddit, which could be a reliable feature to use to predict next_subreddit or next_post_time.