Q1

For both one-hoot and TF-IDF vectorization, I use the FeatureUnion to combine all the fields. Fig 1.0 shows that all fields are vectorized using the same tokenizer, and the same for the TF-IDF way.

```
# Use FeatureUnion to combine the features from text and summary
# One hot
prediction_pipeline = Pipeline([
        ('union', FeatureUnion(
          transformer_list=[
            ('text', Pipeline([
              ('selector', ItemSelector(key='title')),
              ('one-hot', CountVectorizer(tokenizer=tokenize_normalize, binary=True)),
              ])),
            ('all_posts_bodies', Pipeline([
              ('selector', ItemSelector(key='all_posts_bodies')),
              ('one-hot', CountVectorizer(tokenizer=tokenize_normalize, binary=True)),
              ])),
            ('all_posts_ids', Pipeline([
              ('selector', ItemSelector(key='all_posts_ids')),
              ('one-hot', CountVectorizer(tokenizer=tokenize_normalize, binary=True)),
              ])),
        ])
        )
    ])
# TF-IDF
prediction_pipeline_tfidf = Pipeline([
        ('union', FeatureUnion(
          transformer_list=[
            ('text', Pipeline([
              ('selector', ItemSelector(key='title')),
              ('tf_idf', TfidfVectorizer(tokenizer=tokenize_normalize, binary=True)),
              ])),
            ('all_posts_bodies', Pipeline([
              ('selector', ItemSelector(key='all_posts_bodies')),
              ('tf_idf', TfidfVectorizer(tokenizer=tokenize_normalize, binary=True)),
              ])),
            ('all_posts_ids', Pipeline([
              ('selector', ItemSelector(key='all_posts_ids')),
              ('tf_idf', TfidfVectorizer(tokenizer=tokenize_normalize, binary=True)),
              ])),
        ])
        )

    ])
```

Fig 1.0 Feature Union for both one-hot and TF-IDF

The overall result for 6 models is shown down below as a table. The 6th model which I picked is a neural network classifier called MLC. As Fig 1.1 shows, the model with the best performance is Linear Regression with TF-IDF, and the F1 score for each class is also shown below as a bar chart in Fig 1.2. Since the first group of results is tested on the training set, the number is very good-looking for 2 linear regression models and the MLC model.

Summary on training set

	Acc	Marco Precision	Recall	F1
dummy model: most frequent	0.237	0.05	0.012	0.019
dummy model: stratified	0.102	0.059	0.059	0.058
lr: one hot	1	1	1	1
lr: TF-IDF	1	1	1	1
SVC	0.237	0.05	0.012	0.019
MLC	1	1	1	1

Summary on test set

	Acc	Marco Precision	Recall	F1
dummy model: most frequent	0.23	0.05	0.012	0.019
dummy model: stratified	0.126	0.075	0.07	0.069
lr: one hot	0.644	0.515	0.694	0.555
lr: TF-IDF	0.677	0.556	0.742	0.597
SVC	0.23	0.05	0.012	0.019
MLC	0.649	0.491	0.73	0.522

Fig 1.1 Scores of all 6 models

There are 20 classes in the test dataset in total, and the F1 score is listed below.

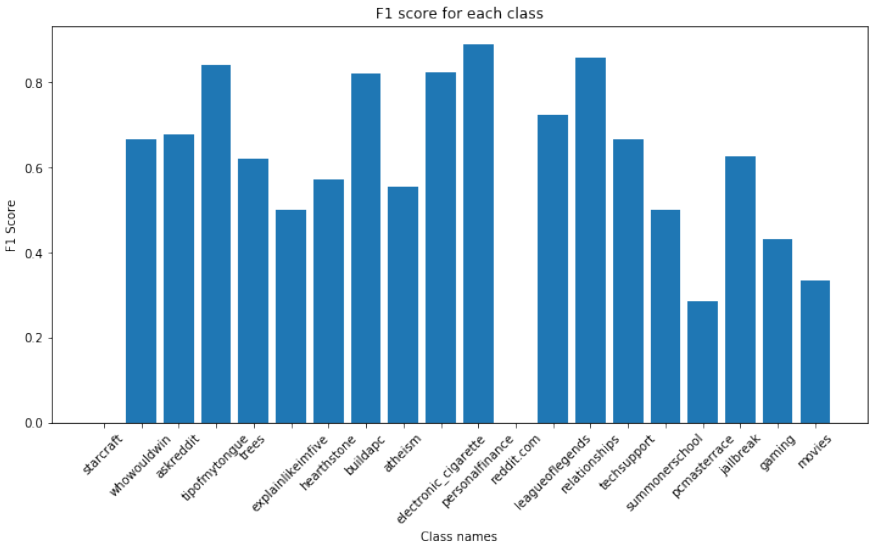

Fig 1.2 F1 score for each class within the MLC model

The MLC model and the linear regression model with one-hot vectorization have the best performances on accuracy. From the perspective of F1 score, the winners are still these two models, no matter on the training dataset or on the test dataset. Both dummy models perform so badly, even on training set, but the SVC model is not much better than these two, with only 23.7% accuracy on training set and 23% accuracy on test dataset.

For the sixth model, MLC, the vectorization method for this model is one hot. Since it is a multilayer perceptron classifier, it can learn so deep to discover the pattern of the training data. However, the dataset is not so big, the number of the hidden layers is only 50 and other parameters are following the default numbers. That is might the reason why a neural network performs worse than a linear regression model.

Q2

For this question, I gradually searched the best value for each parameter and then apply it, make it fixed, then turn to the next one. Firstly, I used the grid search method to find the best parameter of Regularization C value. The range for that value is [0.001, 0.01, 0.1, 1,10,100,1000,10000,100000], and the best number is 100000. And step by step, I each parameter is tuned to its best: max_features = 5000, sublinear_tf = False, norm = None. The final accuracy for the tuned model is 0.697, which is 2% higher than the original one. Other scores are shown below in Fig 1.3.

```
Evaluating LR TF-IDF
LR TF-IDF summary on test set
              Acc    Marco Precision    Recall     F1
---------  -----  -----------------  --------  -----
LR TF-IDF  0.697              0.557     0.758  0.629
```

Fig 1.3 Scores of tuned LR model

For the error analysis, here is the confusion matrix of the tuned model.

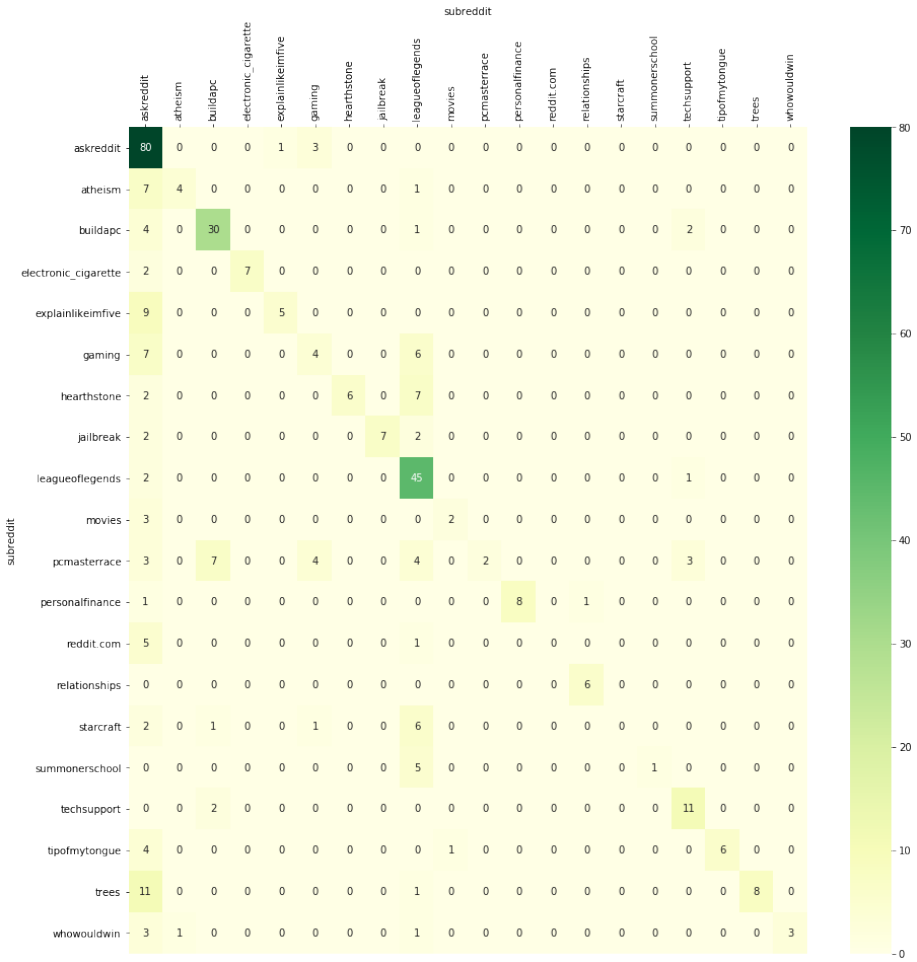

Fig 1.4 Confusion matrix of the LR model

It can be seen that the best predictions are askreddit, buildapc and leagueoflegends. The worst predictions are starcraft and reddit.com. The interesting thing is that for the first row of the confusion matrix, it shows that there are always some threads which are classified as askreddit. Might because of a bias in the training data.

Q3

The extra features I chose are “Post Depth” and “If it is self post”. After adding these two features into the training data, the model performs a little better than before. The scores are shown below in Fig 1.5. Since the previous training data is a sparse matrix, and these two new features are numbers as well, I directly transform these two new features and add these into the training data by connecting these two parts.

LR TF-IDF(plus 1 features) summary on test set

	Acc	Marco Precision	Recall	F1
LR TF-IDF(plus 1 features)	0.698	0.558	0.758	0.629

Fig 1.5 Scores of LR model (plus 1 feature)

```
Evaluating LR TF-IDF(plus 2 features)
LR TF-IDF(plus 2 features) summary on test set
                            Acc    Marco Precision    Recall      F1
--------------------------  -----  -----------------  --------  -----
LR TF-IDF(plus 2 features)  0.701              0.556     0.759  0.653
```

Fig 1.6 Scores of LR model (plus 2 features)

The final accuracy is 0.4% higher and macro F1 score is 0.024 higher. It is easy to understand that different classification might have different mean depth, for the askreddit the depth could be a large number and for the reddit.com it could be a small number. By adding "If it is self post" feature, like the Fig 1.5 shows, it almost does nothing for improving the model's performance. So that feature might be not so relevant with the classification.

It could be helpful if I first analyze the possible relevance for each newly added feature, to decide which one is the most helpful instead of finding it is not so useful after the training and evaluation.

Q4

For the baseline dummy model, Fig 1.7 shows its performance and Fig 1.8 shows the performance of LR classifier.

```
Evaluating Dummy model with stratified strategy
Dummy model with stratified strategy summary on test set
                                    Acc    Marco Precision    Recall      F1
----------------------------------  -----  -----------------  --------  -----
Dummy model with stratified strategy   0.24              0.104     0.104  0.104
```

Fig 1.7 Scores of baseline dummy model

```
Evaluating LR TF-IDF
LR TF-IDF summary on test set
             Acc    Marco Precision    Recall      F1
---------  -----  -----------------  --------  -----
LR TF-IDF  0.474               0.25     0.355  0.271
```

Fig 1.8 Scores of LR model

As for each class, the overall performance is shown below in Fig 1.9. The confusion matrix is shown in Fig 2.0.

	Class_Name	Acc	Marco_Precision	Recall	F1
agreement	agreement	0.010	0.069	0.139	0.093
announcement	announcement	0.002	0.027	0.094	0.042
answer	answer	0.283	0.236	0.174	0.200
appreciation	appreciation	0.050	0.192	0.237	0.212
disagreement	disagreement	0.001	0.015	0.058	0.023
elaboration	elaboration	0.044	0.081	0.093	0.086
humor	humor	0.001	0.007	0.062	0.013
negativereaction	negativereaction	0.001	0.018	0.089	0.031
other	other	0.002	0.034	0.085	0.048
question	question	0.080	0.155	0.153	0.154

Fig 1.9 Overall scores of each class

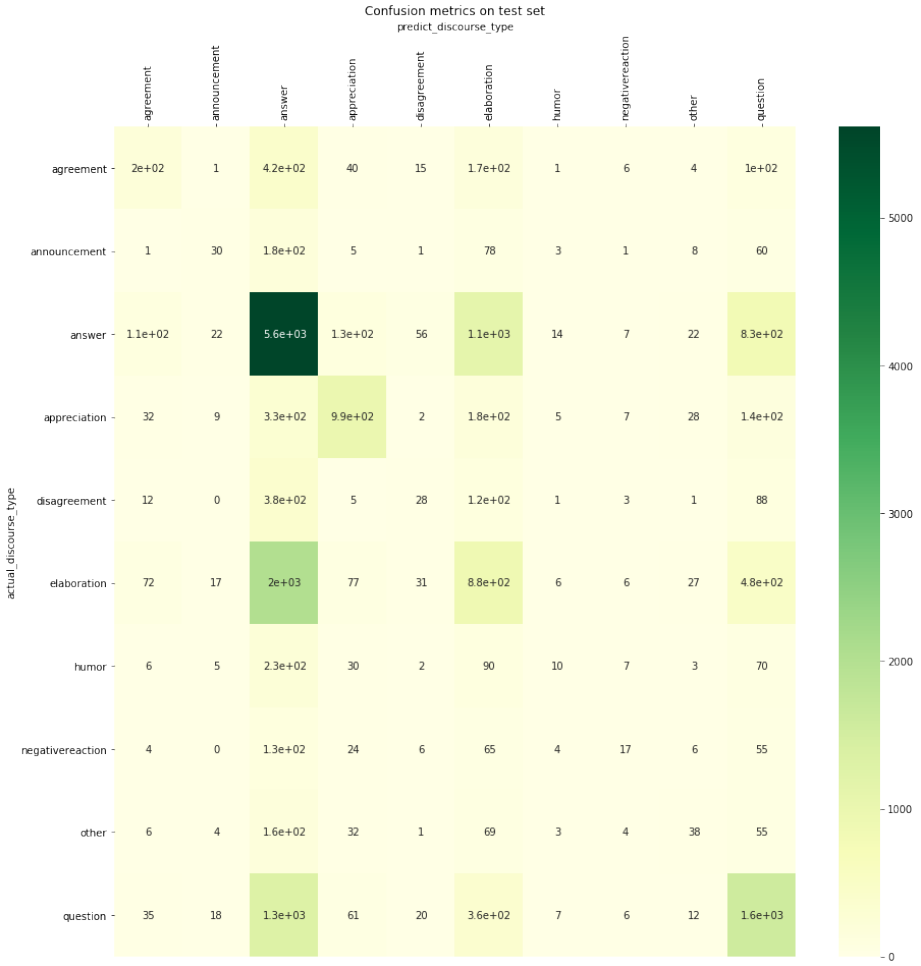

Fig 2.0 Confusion matrix on teat dataset

And in the final model, the 20 most important features and their indices are shown in Fig 2.1.

	0		
40270	1.391554e-13	**15958**	2.982059e-14
73085	7.822631e-14	**11850**	2.914474e-14
13148	7.034373e-14	**66415**	2.886580e-14
34991	6.772360e-14	**73105**	2.861045e-14
67331	5.853096e-14	**7377**	2.741141e-14
38454	4.480860e-14	**41111**	2.677858e-14
66421	3.983480e-14	**72996**	2.602363e-14
36672	3.816947e-14	**66540**	2.570721e-14
72938	3.508305e-14	**48369**	2.553235e-14
62475	3.400613e-14	**62233**	2.531308e-14

Fig 2.1 20 most important features

As all the graphs show that this model with only three features performs not so good. The accuracy on the test set is only 47.4%. And the confusion matrix shows that most posts are

misclassified as “answer”, “elaboration” and “question” (5130 for “answer”, 2232 for “elaboration” and 1878 for “question”).

Q5

The features I chose are shown in Fig 2.2.

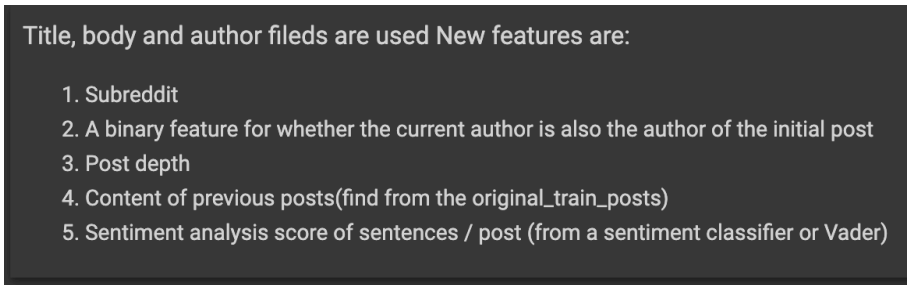

Fig 2.2 Additional 5 features

For the first feature, it can be directly extracted from the dataframe. I use the TF-IDF vectorizer and limit the max_features to 955 to unify the column number of training data and test data. For the second feature, I compare the field “id” with the field “majority_link” to define whether the author of this post is the same author of the initial post. 1 represents true and 0 represents false. For the third feature, it can be directly extracted from the original dataset as well, the name of that column is “post_depth”. For the fourth feature, it becomes a little complex. Since the training data is random, I need to find the index number of that row and map it back into the original ordered dataset. Besides I also need to reindex that ordered dataset because some rows might be dumped at first because it is empty. After all these steps, I can simply find a previous post by looking for the index and minus it with 1. Then use the TF-IDF to vectorize it for further processing. For the fifth feature, use the Vader classifier, I can easily generate the sentiment scores for each post and form a dataframe to store them. There are 4 scores for each post, which are positive, compound, neutral, and negative.

After transforming all features into sparse matrix form and connect them, the training data is now well prepared. The reason for picking up all these features are in consideration of adding both numerical features and text features. The second and third features might help classify the “question” and “answer”, and the fifth feature might be helpful for classifying the post with strong emotion (like “disagreement”, “appreciation”, and “negativereaction”). As for the first and fourth features, obviously they can provide much more information for the model and I think it is the foundation of all the training data.

After combining all features and training the model, the performance is shown below in Fig 2.3. And the confusion matrix is shown in Fig 2.4.

```
Evaluating LR TF-IDF Q5
LR TF-IDF Q5 summary on test set
                Acc    Marco Precision    Recall    F1
------------  -----  -----------------  --------  ----
LR TF-IDF Q5  0.522               0.29     0.418  0.32
```

Fig 2.3 Scores of LR model in Q5

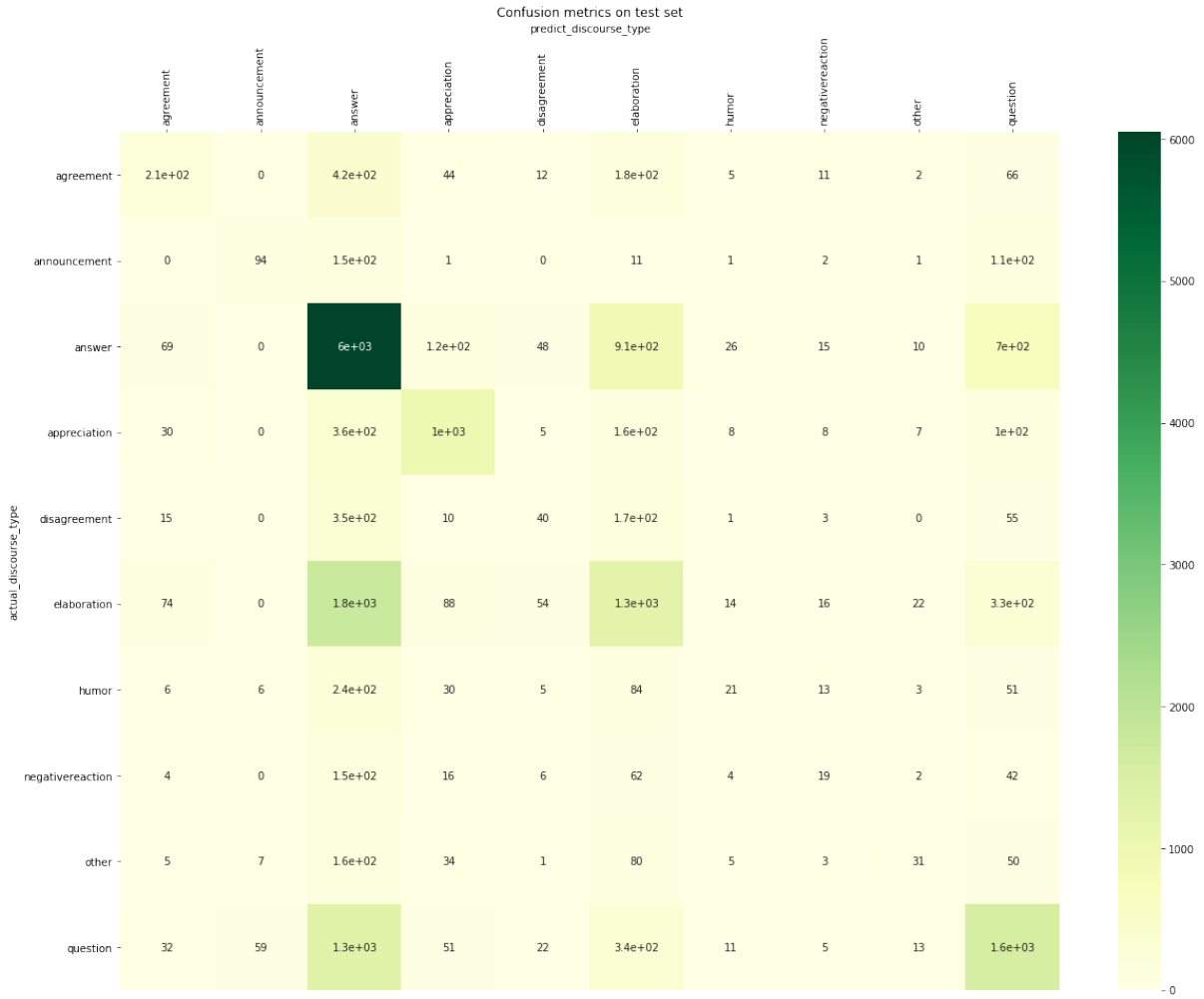

Fig 2.4 Confusion matrix of LR model in Q5

As the figures show, the accuracy and other scores are all improved. And the misclassified posts are decreased as well (4930 for “answer”, 1997 for “elaboration” and 1504 for “question”). And the posts with relatively strong emotions are classified more precisely.

By computing the coefficient of the model, it can be seen from Fig 2.5 down below that the feature1 (Subreddit) and feature4 (previous post) are the most helpful features. As for the numerical features, the most important one is feature3(post depth) and the most unhelpful feature is feature2 (whether the current author is the initial post’s author).

	feature name	coef
0	feature 1	-1.216129e-16
3	feature 4	-1.693821e-16
2	feature 3	-4.872826e-15
4	feature 5	-1.507287e-13
1	feature 2	-4.758799e-12

Fig 2.5 Feature importance

In general, all five features have more or less improved the performance of the model. By involving more information, the ability of the model has been improved to a certain extent. However, it can also be seen that the final performance of the model is not particularly good, some new methods may be applied in the early processing stage of features to improve the model. In addition, the model parameters have not been tuned carefully, and the performance of the model may also be affected by this aspect.