Prediction of Community Approval of Reddit Posts

Runxuan Jiang *
runxuanj@umich.edu

Tianyi Li *
litianyi@umich.edu

Tianchen Ye *
jackye@umich.edu

Abstract

In this study, we predict of the approval of a new post in a specific community within the social media platform Reddit. We divide this task into three sub-tasks: predicting whether the net amount of upvotes on the post will surpass a certain threshold, predicting directly the upvote ratio the post will receive, and predicting which of three ranges the upvote ratio will fall into. We attempt to build models for these tasks using only textual data from the Reddit posts themselves, including the title and text, avoiding any features that may only be collected after the post has been released such as comment count. We also introduce a custom loss function for dealing with the imbalanced nature of Reddit upvote data. We train and test our models on several university subreddits and find that our models perform reasonably well with respect to a random baseline.

1 Introduction

The task of deterimining the attitudes and approval of a certain community towards an idea is often very useful, and can lead to insights in economics, finance, and public policy. At the same time, this task can be very difficult since it involves gathering the opinions of a large amount of people within one community, and obtaining an accurate representation of someone's "attitude" is a non-trivial task. The first issue can be resolved with the rise of social media, specifically Reddit, where forums are already organized into communities surrounding a single topic. In particular, university subreddits have grown exponentially in the past few years, with subreddits like r/berkeley reaching over 100,000 members. While it is still difficult to determine users' attitudes using this data, the problem can be simplified to instead analyze the approval of users in a subreddit towards a specific post. The

goal of this project is thus to create and analyze models that, given a textual post, predicts the approval of the post within a certain subreddit.

Previous works have conducted similar exploration on Reddit data, almost all of which attempt to predict the net upvote count (the number of upvotes minus downvotes) of a post using regression or binary classification (Segall and Zamoshchin, 2012; Shuaibi, 2019; Terentiev and Tempest, 2014). However, many of these studies collect their data as a set of posts from a "pool" of different subreddits, and create models across all these subreddits. Often times such models perform poorly, especially on regression of the net upvote count, where the RMSE across such models is almost always higher than 100 (Segall and Zamoshchin, 2012; Shuaibi, 2019). Additionally, some works use features that can only be known after the post has been created, such as the number of comments (Shuaibi, 2019).

To address the task of predicting the approval of a post within a certain subreddit, we first define two metrics that we believe are correlated with a posts' approval: net upvote count (upvotes - downvotes) and upvote ratio (upvotes / total votes). We focus more heavily on upvote ratio since it is more related to the notion of a post's "approval". We specifically study three sub-tasks involving these metrics:

1. We pose the prediction of net upvote count as a binary classification task, where we predict whether the net upvote count of a post will exceed a certain threshold.

2. We pose the prediction of upvote ratio as a regression task, where we predict directly the upvote ratio of a post.

3. We also consider the prediction of upvote ratio as a classification task, where we predict which of three ranges the upvote ratio of a post will fall into.

---
*University of Michigan Department of EECS

In the rest of this paper, we may sometimes refer to each of these three sub-tasks as "task 1", "task 2", and "task 3", respectively.

For the regression task, we further introduce a custom loss function and smoothing techniques for handling imbalanced data. We then train subreddit-specific models for each task using text data collected from university related subreddits including r/uofm, r/berkeley, r/uwaterloo, and r/UTAustin. We specifically chose university subreddits since we felt that the type of posts in these subreddits will be more homogenous over time, as compared to subreddits like r/news or r/politics which will vary greatly by current events. The models used include feed-forward neural networks for regression, and k-nearest neighbors (KNN), gaussian mixture models (GMM), random forest classifier (RFC), and neural networks for classification. In terms of feature generation, our models are trained solely using the textual information (title and text) of each post.

We then evaluate our model performance on posts not seen before during training, and interpret the models to identify trends within each subreddit. For the first task (binary classification of net upvote count), our neural network model achieved an accuracy of over 0.7 for each subreddit. For regression of upvote ratio, our neural network model consistently had a mean absolute error (MAE) of around 0.2. And for three-class classification of upvote ratio our GMM model was able to achieve an accuracy of 0.5 for most subreddits. Note that our models are trained solely on textual post data and does not use any features that depend on future interaction with the post (such as number of comments). Furthermore, after a thorough search we have not found any other works that focus on the prediction of upvote ratio, so our models for regression and classification of upvote ratio may act as a baseline for future work in this area.

The paper is organized as follows. We will first discuss in more detail related works. We will then discuss how we collected and cleaned our data in the "data" section and presetn visualizations of our data. Next in "methods" we discuss how we chose our metrics and features. In "algorithms", we will discuss each of the learning algorithms we used for each of the three tasks. We then discuss the performance of the models and how we interpreted the models in "results and discussion". Finally, we discuss some ethical issues related to this project

in "ethical considerations".

2   Related work

There have been several previous works involving supervised learning tasks on Reddit data, specifically with predicting Reddit post popularity (number of upvotes - downvotes of a post).

Segall and Zamoshchin (2012) studied the prediction of net upvote count both as a multi-class classification problem (where the classes are different ranges of net upvote count) and a regression problem. The study used a random sample of over 2 million reddit posts, and used features including title and text embedded using TF-IDF, author, time created, and whether the post is marked as "over 18". The study tried several techniques, including naive bayes and multi-class SVM for classification, and linear regression for regression.

Shuaibi (2019) also studied the prediction of net upvote count, focusing mainly on regression techniques. The study only utilized the title of the each post as the only textual feature, and utilized other features like number of comments and whether the post was given any Reddit rewards. The study also used engineered features such as the length of the title and the sentiment of the title. Three regression models were trained on posts across several subreddits from the first 6 months of 2018, including linear regression, KNN regression, and random forest regression.

3   Data

3.1   Data collection

We collected all posts prior to March 2022 in the subreddits r/uofm, r/berkeley, r/uwaterloo, and r/UTAustin using Reddit's API, using the pushshift.io library (Baumgartner et al., 2020). For each post, we collected the title, text , upvote scores, upvote ratio, text_only, and number of comments of each post. Note that text_only is a boolean that is set to true if a post does not have any non-textual content (like images). The distribution of net upvote count for r/uofm is shown in figure 1. Histograms for other subreddits can be found in A.1. Using the upvote ratio and net upvote count, we were also able to calculate the total number of votes in general (upvotes + downvotes) for each post.

3.2   Data cleaning and filtering

Next, we filtered the data by only keeping posts where text_only is set to true, since we plan on
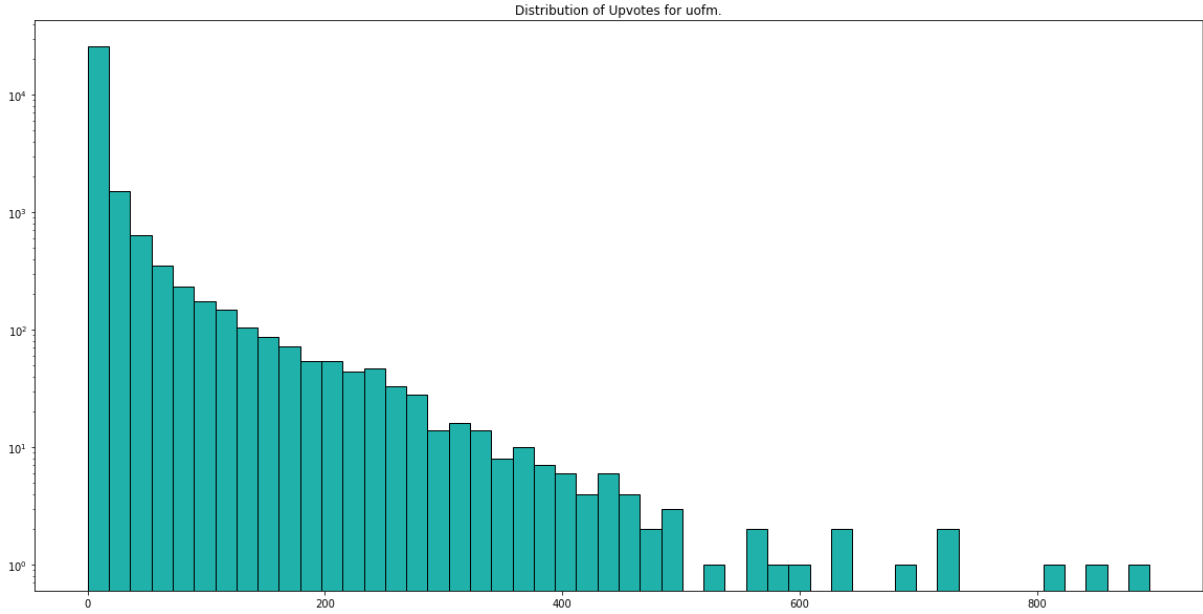
2

Figure 1: Distribution of net upvote count for r/uofm.

working only with posts that use textual data. We also filtered out posts with less than 5 comments and less than 16 total votes (upvotes + downvotes). This is because by default posts will have exactly one upvote (from the user who posted it) and would thus have an upvote ratio of 1. However, this does not reflect the community as a whole and thus we only want to maintain posts with a certain number of community interactions.

We then cleaned our data by concatenating the title and text of each post, since some posts only have a title. We also converted any posts where the text or title were '[deleted]' or '[removed]' to an empty string, and converted all text to lowercase. A histogram of upvote ratios for r/uofm is shown in figure 2.

We then split our data into a training set and test set for each subreddit by first shuffling the data and then choosing the first 80% of the data to be the training set and the remaining 20% to be the test set. When training models for each of our experiments, we exclusively used the training set, and we evaluted each model using the test set.

4 Methods

4.1 Metrics Used

We decide to use both net upvote count and upvote ratio as our metric for a post's approval. We chose to use net upvote count because it is the same metric used by Reddit's algorithm for determining which posts are "popular" and will be more likely to show up on the front page of the subreddit. However, this metric does not directly align with a post's approval since a post with a high net upvote count could have no downvotes, or it could have a significant amount of downvotes. Furthermore, the previous work by Shuaibi (2019) has shown that predicting net upvote count is difficult, which is likely caused by the fact that the distribution of net upvote count is highly skewed, as seen in figure 1. Thus, we only chose to predict net upvote count as a binary classification task and focus more on upvote ratio. Upvote ratio directly measures the percentage of users who approved of the post, conditioned on the fact that the user interacted (or voted) on the post. Since we are not able to access data on how many people viewed the post (but did not interact with the post), we believe this is the closest metric we could obtain for gauging the approval of the subreddit towards the post.

4.2 Feature Generation

In order to be used as input to our models, we converted our textual posts into fixed length numerical vectors using a pretrained attention model called sentence Bert (or SBert) (Reimers and Gurevych, 2019), which achieves state-of-the-art performance on sentence-pair regression tasks like semantic textual similarity. The embedding method applied bi-directional attention layers, which guides the embedding layer to try to "focus" on words that contain more information, and then filter out the
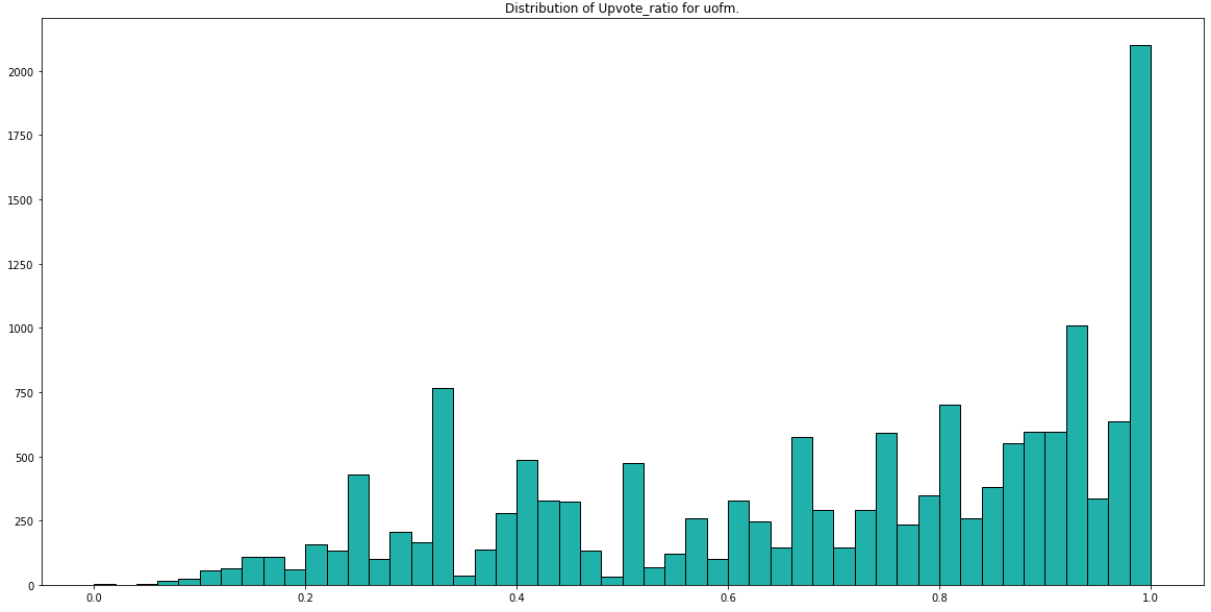
3

Figure 2: Distribution of upvote ratio after data cleaning for r/uofm.

information through unsupervised learning. In this particular case, we applied the pre-trained model all-MiniLM-L6-v2, which converts each post into a 384-dimensional vector. Each vector is of unit length and the model is designed to assign vectors of similar cosine similarity to similar input text.

5 Algorithms

We now describe in detail each sub-task as well as the algorithms used.

5.1 Task 1: Binary Classification of Net Upvote Count

In this experiment, we consider our problem as a classification task to tell whether a certain post will be popular or not based on the number of net upvotes. We define posts that are being popular and under hot discussions as those with net upvotes larger or equal to 5, and those being tedious with net upvotes less or equal to 1. The threshold of tediousness being one is because each post will automatically receive one upvote from the user itself, so posts with one upvote basically indicates there is no other user looking into the post. Because of the skew of the dataset, there are many more posts with the number of upvotes in the tedious category, and thus we selected the same amount of data points from the two categories to deal with the imbalance. Next, we applied a deep neural network to solve the classification problem. Visually, the model follows the pipeline shown in figure 3.

Between each layer, we use a ReLU activation function to add non-linearity, and a dropout layer to enhance the generality of the model. At the end of the model, we applied sigmoid for the classification task. The model is trained using a binary cross-entropy loss function.

5.2 Task 2: Regression of Upvote Ratio

In this task, we try to predict the upvote ratio for a post by directly predicting the upvote ratio value. We train a neural network identical to the one in task 1 shown in figure 3, except for the last layer maps to a single scalar value representing the upvote ratio, rather than a 2-dimensional vector. At first, we train our neural network using the mean absolute error (MAE) as our loss function:

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |\hat{y}_i - y_i|.$$

where n is the size of one batch, $\hat{y}_i$ is the predicted upvote ratio for datapoint i, and y_i is the true upvote ratio for datapoint i. For this task only, we further filter our data to only include posts with a total number of votes greater than 20, to further reduce the skewness of the distribution of upvote ratio.

However, by comparing the histograms of the actual upvote ratio and the predicted upvote ratio, we find that most of the predicted values are in the interval [0.3, 0.4] and [0.9,1.0], as seen in figure 4. This is mainly due to the imbalanced training

400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
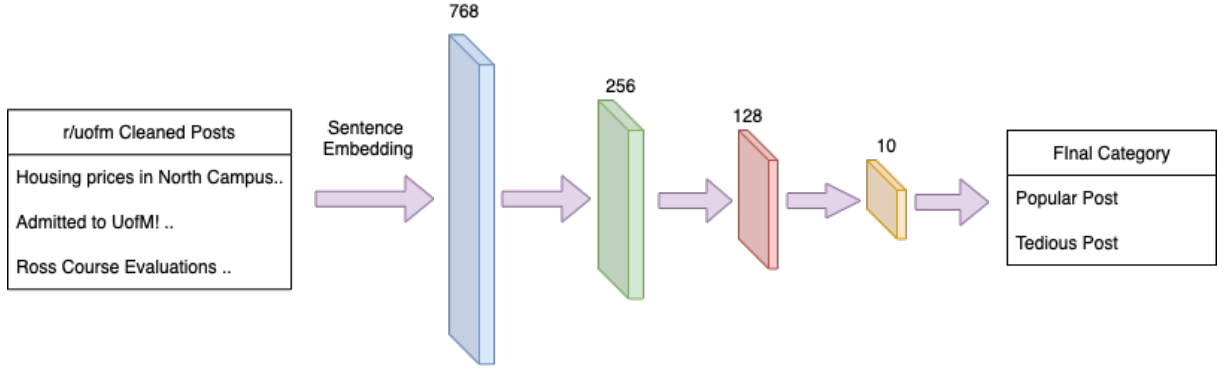488
489
490
491
492
493
494
495
496
497
498
499

Figure 3: Feed-forward neural network architecture used in task 1.
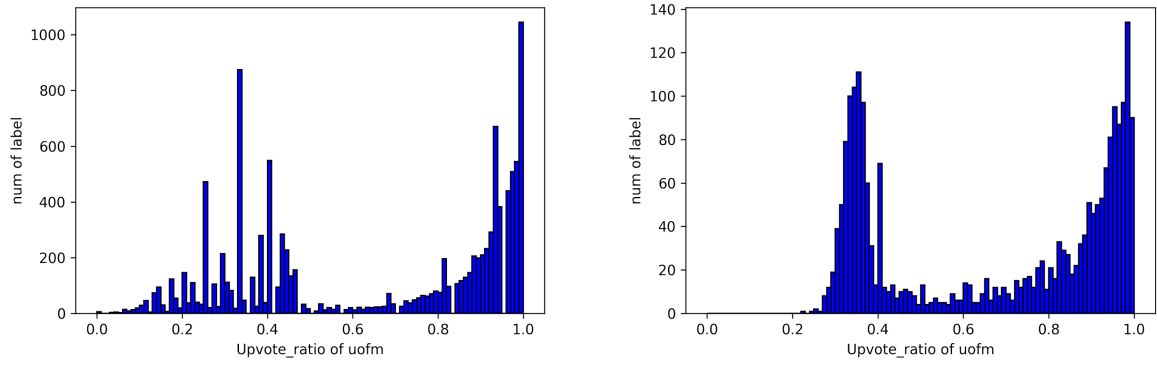

Figure 4: True (left) and predicted (right) distributions of upvote ratios for r/uofm using model trained on MAE loss.

dataset. In order to minimize the loss function (i.e. Mean Absolute Error), our model tends to predict a value that has the highest occurring probability. However, the model trained is highly biased and failed to recognize the actual pattern of the data, as there are almost no predictions below 0.2. A model like this is not desired because it failed to predict values with low occurring probability.

To solve this problem, we decided to design a new loss function that can lower the influence of data with high occurrence. We refer to the new loss function as the Weighted Mean Absolute Error (WMAE), and it is calculated as follows:

$$WMAE = \frac{1}{n} \sum_{i=1}^{n} \frac{|\hat{y}_i - y_i|}{W(y_i)}.$$

where

$$W(y_i) = \frac{\log P(y_i)}{\sum_{j=1}^{n} \log P(y_j)}.$$

where $P(y_i)$ is the probability of y_i occuring among the true labels. Since the true labels occur within a continuous range, we fit a continuous probability distribution over the distribution of our true labels. We first smooth the data by adding or
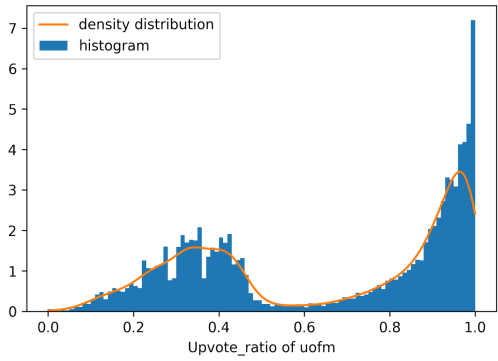
Figure 5: Smoothed distribution of upvote ratio (labels) and fitted kernel density for r/uofm.

subtracting 0.03 at random to the true upvote ratios. Then, we utilize the Kernel Density function provided by scikit-learn (Pedregosa et al., 2011) to fit a density distribution over the upvote ratios. The fitted density distribution for r/uofm can be seen in figure 5. We then use the value of the density function at y_i for $P(y_i)$.

We then trained the neural network using this loss function. We see that the result distribution

5

Figure 6: Distribution of predicted upvote ratio after training using weighted MAE.

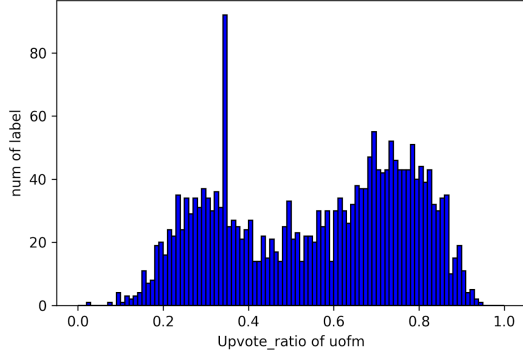| subreddit / range	(0, 0.5)	(0.5, 0.8)	(0.8, 1)
r/uofm	3694	3220	5514
r/berkeley	6591	6724	11963
r/uwaterloo	15247	19470	22255
r/UTAustin	3229	3410	7189

Figure 7: Distribution of upvote ratio of posts across three different classification ranges for each subreddit.

of predicted upvote ratios has higher occurances in ranges with low label occurances, as seen in figure 6.

For figures of the distribution of predicted upvote ratio for each of the four subreddits, see A.3.

5.3 Task 3: Classification of Upvote Ratio

In this experiment, we pose the predict of upvote ratio as a classification problem where the goal is to predict which of three ranges of values the upvote ratio for a post falls into. The ranges chosen for this task are (0, 0.5), (0.5, 0.8), and (0.8, 1). The first range corresponds to posts that have more downvotes than upvotes and thus can be categorized as "negative approval". The second category has slightly more upvotes than downvotes and can be categorized as "mild approval" and the third category can be categorized as "high approval". We specifically chose these ranges so that each range can be reasonably correlated with a level of approval, while keeping the data somewhat balanced between the classes. Note that due to the skewness of the data, the highest range still remains with the largest proportion compared to the other ranges. The distribution of the test data for each subreddit among these classes are detailed in figure 7

For each subreddit, we train four different classifiers on the training set and evaluate on the test set.

Subreddit	Accuracy	F-1 score
r/uofm	0.7391	0.7477
r/berkeley	0.7207	0.7327
r/uwaterloo	0.7113	0.7318
r/UTAustin	0.7615	0.7704

Figure 8: Accuracy and F-1 score of neural network model trained on binary classification of net upvote count (task 1).

The classifiers include (hyperparameters selected after tuning):

- K nearest neighbors classifier - the predicted label for each input is the most commonly seen label of the 20 datapoints closest to the input (by Euclidean distance) in the training set.

- Random forest classifier - ensemble of 128 decision trees.

- Feed forward neural network trained using softmax function and Adam optimizer with the following layers:

  - linear(384, 128) $\rightarrow$ ReLU $\rightarrow$ dropout
  - linear(128, 64) $\rightarrow$ ReLU $\rightarrow$ dropout
  - linear(64, 3) $\rightarrow$ ReLU

- Guassian mixture model where each class is modeled by a GMM with 2 components.

6   Results and Discussion

In this section, we discuss the performance of our models in each sub-task.

6.1   Task 1: Binary Classification of Net Upvote Count

The results for the neural network model on each of the four subreddits are shown in 8.

The results are promising, with relatively high accuracy between 71.1% - 76.1% and F1-score between 73.2% - 77.0%, whereas a random model will perform at around 50%. We also applied the same model on some larger subreddits like r/askreddit, which the accuracies can even boost higher to about 83%. This can be an indication that with larger dataset and more number of posts collected, the performance of the model can be further improved, which can be one of the improvements we can focus on in future tasks.

Subreddit	MAE (MAE loss)	MAE (WMAE loss)
r/uofm	0.1763	0.2207
r/berkeley	0.1760	0.2166
r/uwaterloo	0.1756	0.1932
r/UTAustin	0.1751	0.2385

Figure 9: Mean absolute deviation for each subreddit of neural network model trained on task 2, trained using either MAE or WMAE as loss function.

	KNN	RFC	Neural Net	GMM
Acc.	0.530	0.526	0.531	**0.536**
F-1	0.493	0.435	0.398	**0.514**
UAR	0.499	0.502	0.346	**0.518**

Figure 10: Results for each model trained on task 3 in r/uofm (RFC refers to random forest classifier, Acc. refers to accuracy).

6.2 Task 2: Regression of Upvote Ratio

We report the mean absolute error for each of the four subreddits for both the neural network model trained using MAE and WMAE on the test set in figure 9, as discussed in algorithms.

Although the resulting MAE was higher for the model trained using the WMAE loss function, we believe that it did succesfully reduce the impact of datapoints with labels that are more likely than others. Future work could further explore how different weighting techniques can increase model generalizability, particularly with labels that have low representation in the dataset.

6.3 Task 3: Classification of Upvote Ratio

We report the the accuracies, F-1 score, and unweighted average recall (UAR) for each of the four models trained on task 3 for r/uofm in figure 10. For results on each of the other four subreddits, see A.4.

Note that since there are three classes, a random baseline would achieve an accuracy of 0.33. We notice that in several cases each of the models have similar performance, and the GMM model seems to consistently perform well especially with respect to F-1 score and UAR. In general it appears that the GMM is relatively better at identifying true positives for each class, even for classes that have less representation among the labels. However, these results are not significantly higher than a random

baseline. This may be caused by the fact that the text embeddings generated are all of unit length. Thus, all text embeddings lie on the Euclidean ball in 384-dimensional Euclidean space. Such embeddings may perform poorly with models that utilize the l_2 norm for determining simlarity, such as KNN or GMM. Future work to improve accuracy of these models may look into ways of transformer the embeddings, or using additional features like the time a post is generated, and the length of the text.

To further analyze the GMM model, we inspect the top five submissions with the highest probability of having an upvote ratio in the range of $(0, 0.5)$, and within the range $(0.8, 1)$. The results are shown in figure 11

We see that some trends exist within the posts. For example, the posts that are likely to be sampled from the $(0.8, 1)$ GMM model seem to mention housing or acceptance into the university. On the other hand, four posts likely to be sampled from the $(0, 0.5)$ GMM model mentioned Ross (the business school at Michigan).

7 Ethical Considerations

We now disucss three ethical considerations relating to this project.

We first discuss the biases and assumptions used within in our experiment. Because of the limit of time and computational resources, we are mainly focusing and discussing subreddits within the four universities, namely UofM, Waterloo, UC Berkeley, and UT Austin. In other words, our experiments and conclusions can only be drawn from the four universities, which cannot reflect the situations in other subreddits such as r/USC. Since the prediction of posts on upvotes and upvote ratio is highly correlated with the subreddit, all implications and applications are only reasonable to be interpreted inside the four universities, and the situation should not be generalized to other universities or subreddits. Furthermore, our models should not be used to make judgements about any of these universities or their students/faculty/alumni in general.

The second ethical implication is about privacy. Since we are dealing with a large number of posts and related data, prevention of data leakage will be one of the important ethical issues we have dealt with seriously. To prevent user information, including biographical information, username, and their relative posts from leaking, we kept all using posts anonymous, and avoided using sensitive informa-

Titles of posts with embeddings with highest probability of being in range (0.8, 1) according to GMM model.	Titles of posts with embeddings with highest probability of being in range (0, 0.5) according to GMM model.
"First year Graduate Housing Options"	"UMich Undergraduate Ross Direct Admission"
"Off Campus Housing - too late?"	"CS Eng Freshman Schedule Question"
"Sophomore Transfer - Just committed to UMich!"	"Is it possible to transfer from LSA to Ross at ..."
"EECS 203 and 280 spring semester"	"Applying to Ross"
"Just got accepted as a transfer student!!!"	"LSA-CS + Ross?"

Figure 11: Titles of posts with embeddings that have the highest likelihood of being randomly sampled from the GMM of the range (0.8, 1) and (0, 0.5).

tion such as the user's gender, age, and regional status. In the meantime, we collected our data from the official Reddit API and strictly followed Reddit's api terms of use as developers. Finally, our source data is deployed privately on Google Drive, which is one of the largest cloud servers where data security can be guaranteed.

The third ethical implication we would like to discuss is preventing potential harm to participants. Making predictions on upvotes or upvote ratio might be instructive to users that are preparing to send a post, but also can be misleading when our model is giving a wrong prediction. In such circumstances, both the user and the subreddit will receive negative impacts. For the user, he or she might receive a result of upvote consequence that doesn't fit his estimate because of the model's prediction, which might undermine his self-confidence and also his trust towards our model. On the other hand, if some users receive predictions that their posts will get the brush-off, all they will do is to delete their drafts and choose not to send their posts, which in turn will affect the interest of Reddit itself. In order to prevent such negative impacts from happening, in our further investigations we have to deal seriously with the accuracy of our model so that it can deliver considerable results that can satisfy both users and Reddit. Meanwhile, before our model can be delivered as a commercial software that will officially provide services for users, we should discuss with the Reddit platform to reach an agreement on dealing with potential issues and arrangements.

8 Conclusion

In this work, we established three models that can make predictions on whether a post can become popular in three different ways: classification on upvote numbers, classification on upvote ratio, and regression on upvote ratio. All three models delivered promising results and can be used to convey some hints on whether a post will be popular or not. In future works, we would like to improve our three models to reach better accuracies and encapsulate them within packages so that real users can engage with our model.

References

Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. 2020. The pushshift reddit dataset. *CoRR*, abs/2001.08435.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084.

Jordan Segall and Alex Zamoshchin. 2012. Predicting reddit post popularity.

Ahmed Shuaibi. 2019. Predicting the popularity of reddit posts.

Andrei Terentiev and Alanna Tempest. 2014. Predicting reddit post popularity via initial commentary.

A Appendices

A.1 Histograms of net upvote count for each subreddit.

Distribution of Upvotes for uofm.
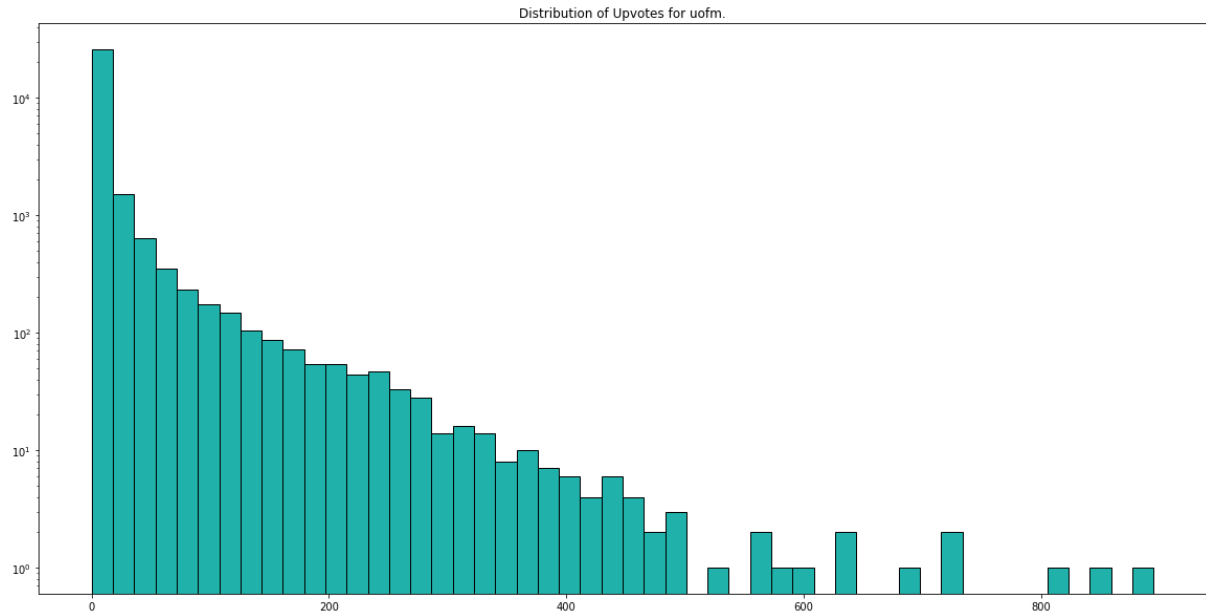
Figure 12: Distribution of net upvote count for r/uofm.

Distribution of Upvotes for berkeley.

Figure 13: Distribution of net upvote count for r/berkeley.
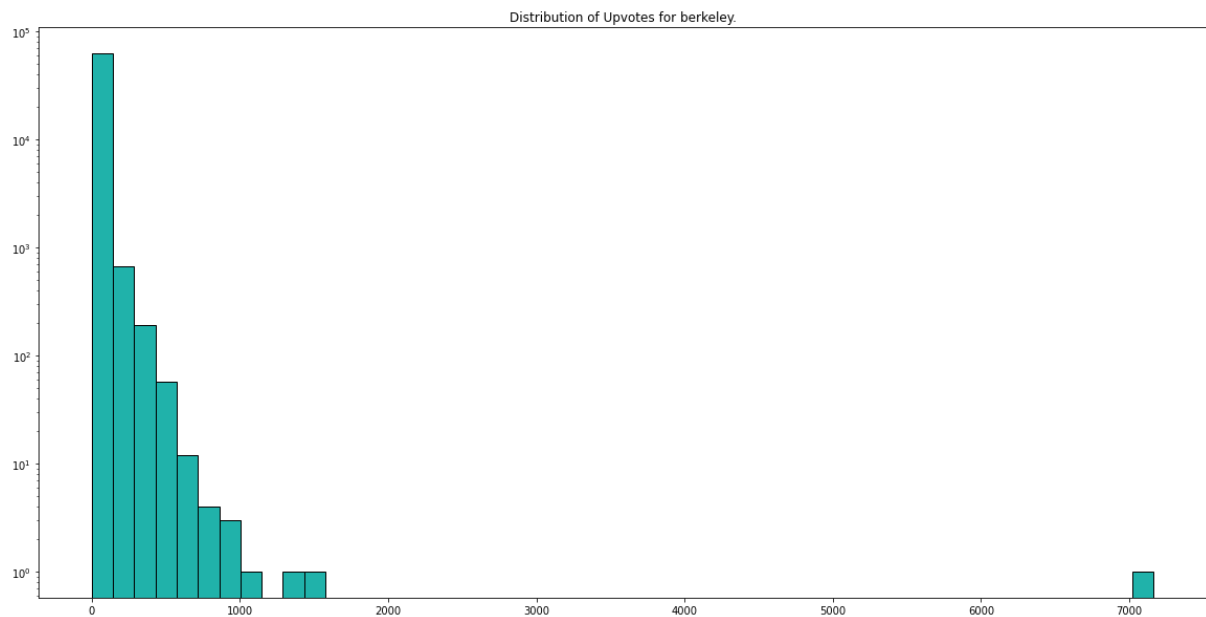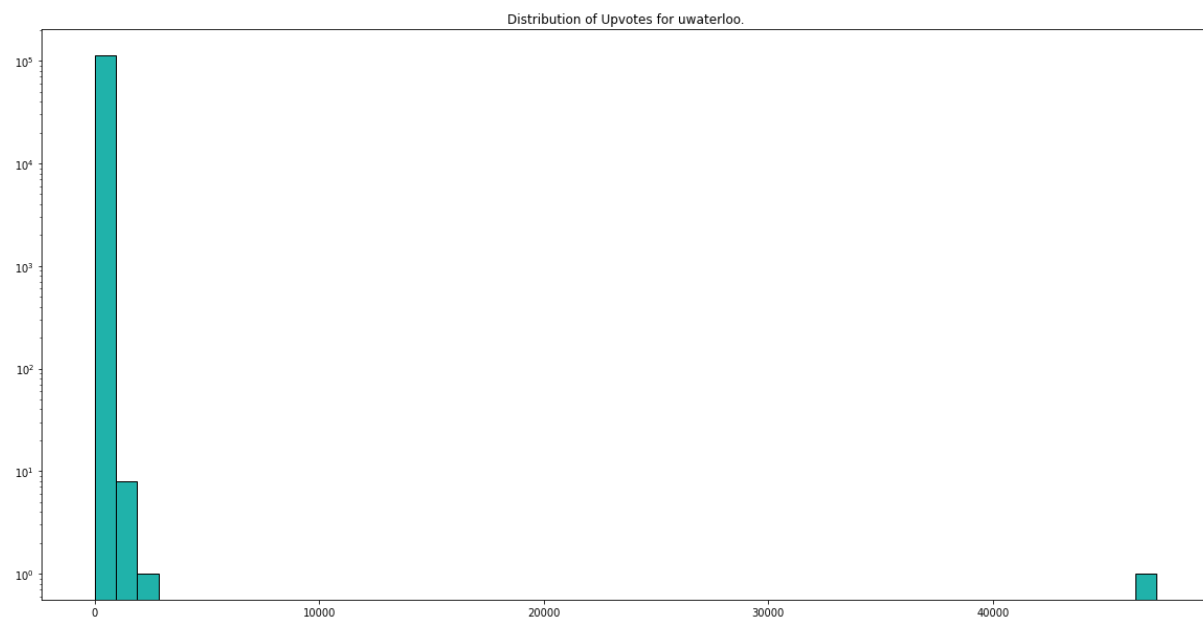
Figure 14: Distribution of net upvote count for r/uwaterloo.
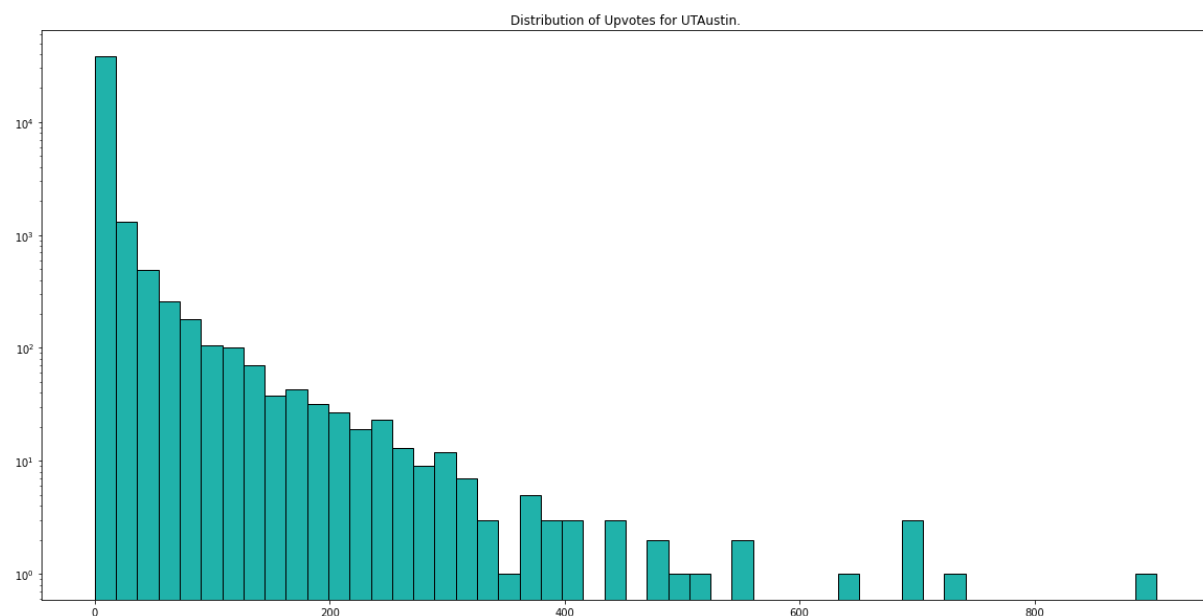
Figure 15: Distribution of net upvote count for r/UTAustin.

A.2   Histograms of upvote ratio for each subreddit after data cleaning and filtering.
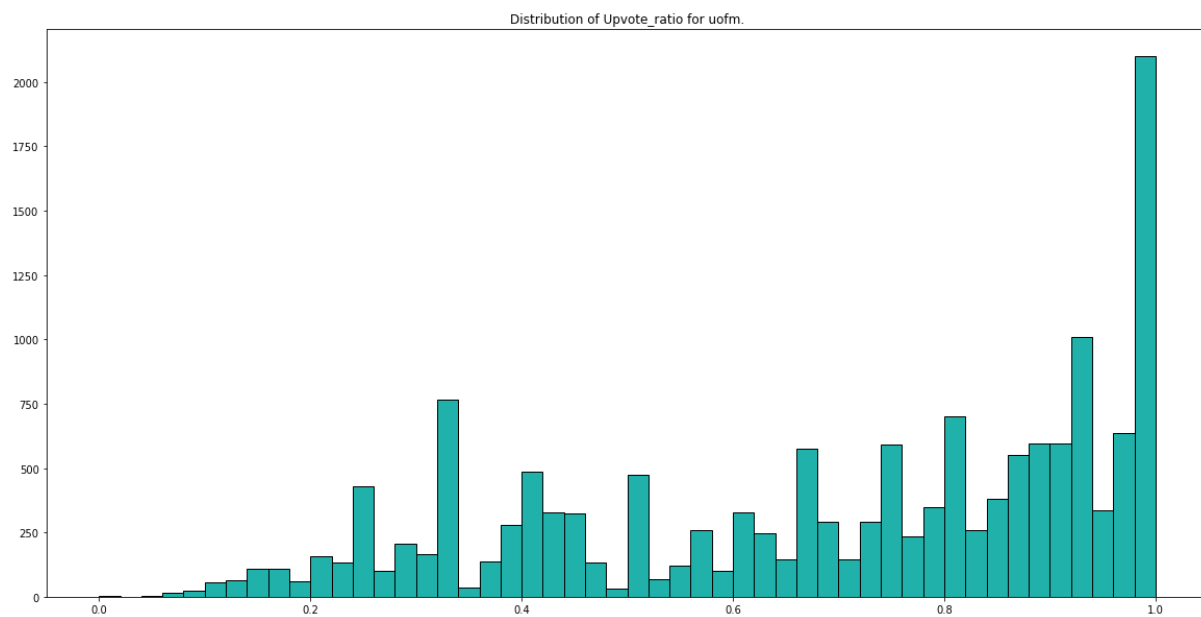
Figure 16: Distribution of upvote ratio after data cleaning for r/uofm.

Figure 17: Distribution of upvote ratio after data cleaning for r/berkeley.

11

1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199

Figure 18: Distribution of upvote ratio after data cleaning for r/uwaterloo.
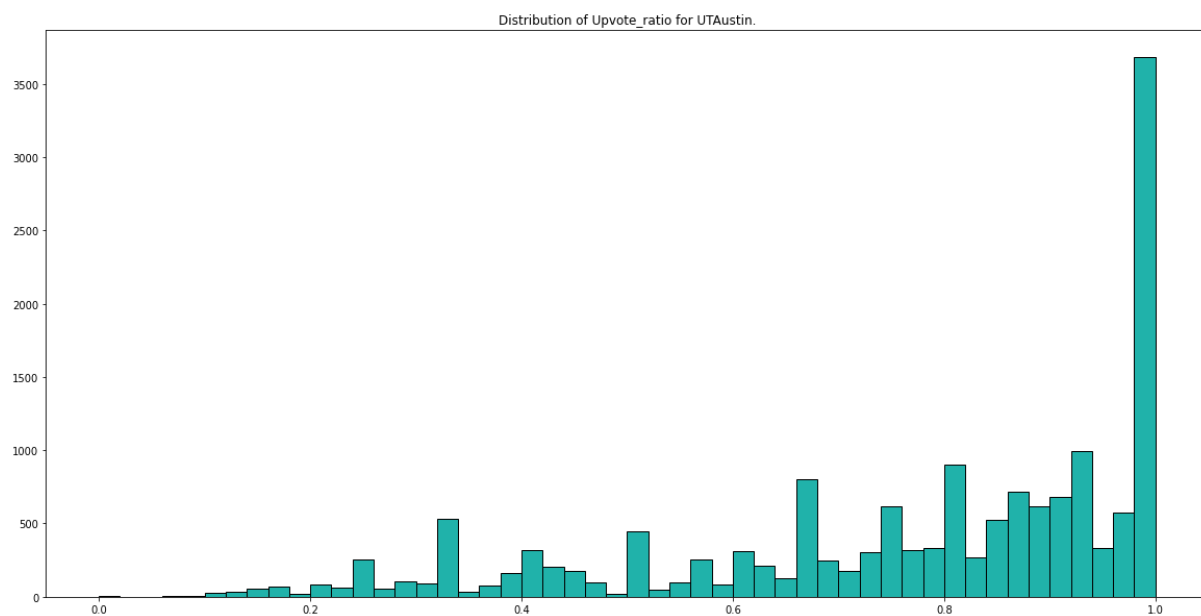
Figure 19: Distribution of upvote ratio after data cleaning for r/UTAustin.

1200

1201

1202

1203

1204

1205

1206

1207

1208

1209

1210

1211

1212

1213

1214

1215

1216

1217

1218

1219

1220

1221

1222

1223

1224

1225

1226

1227

1228

1229

1230

1231

1232

1233

1234

1235

1236

1237

1238

1239

1240

1241

1242

1243

1244

1245

1246

1247

1248

1249

1250

1251

1252

1253

1254

1255

1256

1257

1258

1259

1260

1261

1262

1263

1264

1265

1266

1267

1268

1269

1270

1271

1272

1273

1274

1275
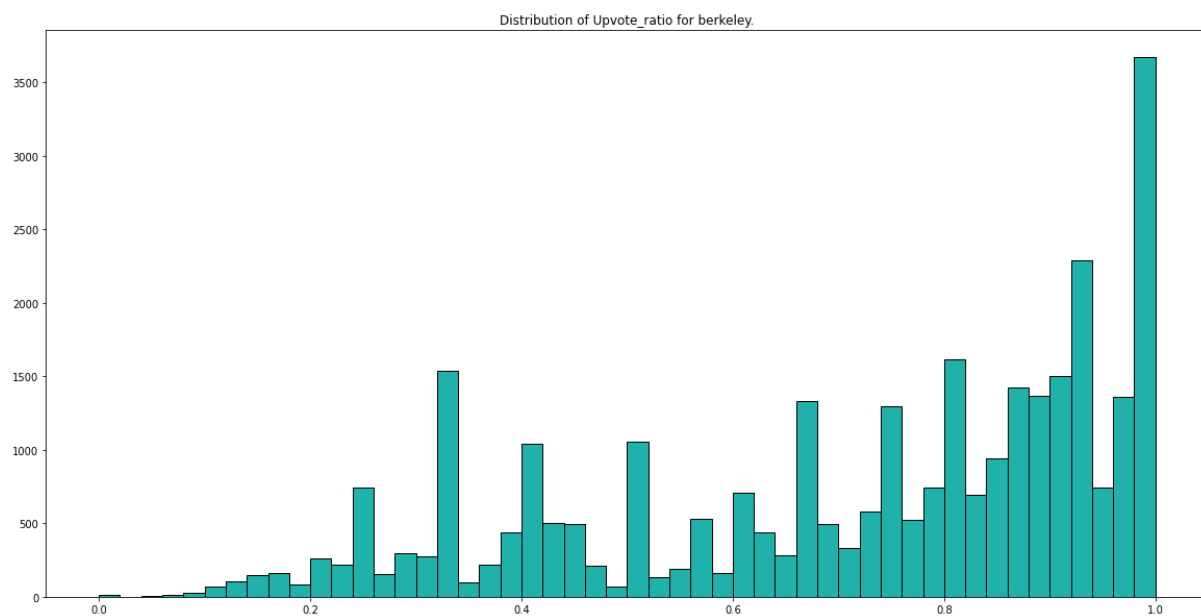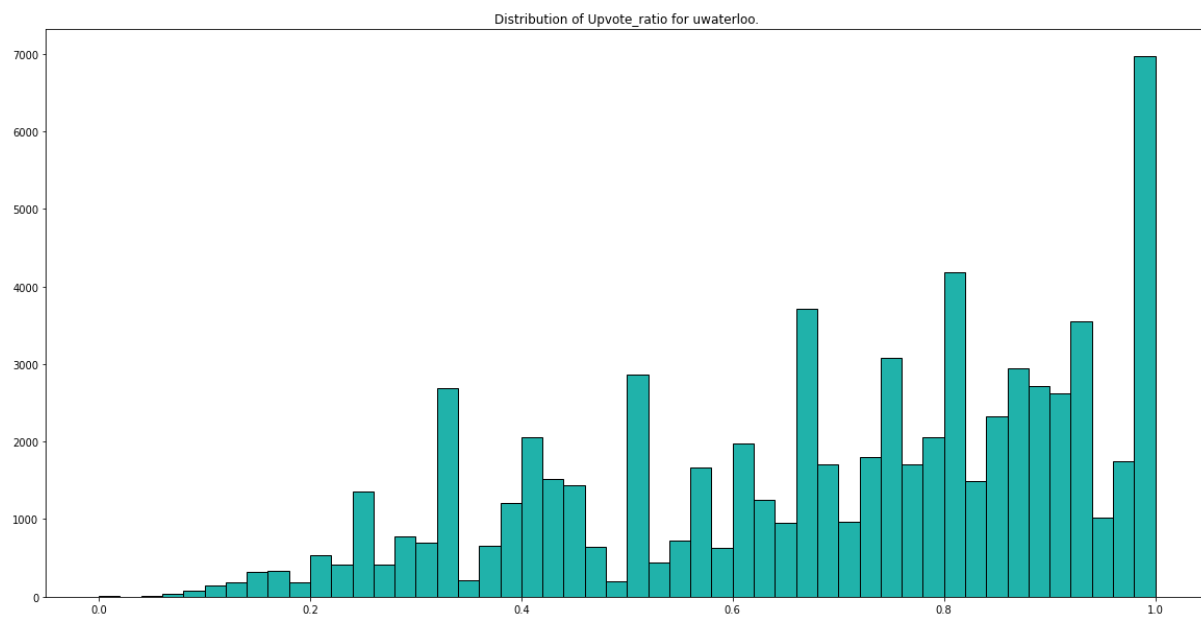
1276

1277

1278

1279

1280

1281

1282

1283

1284

1285

1286

1287

1288

1289

1290

1291

1292

1293

1294

1295

1296

1297

1298

1299

A.3   Distribution of predicted upvote ratios using WMAE loss function.

Below are the visualizations of both the true and predicted labels (upvote ratio) discussed in task 2 - the regression of upvote ratio. For each subreddit we include four histograms - the first (top left) displays the original distribution of upvote ratio in the test set after data filtering and removing posts with less than and including 20 total votes. The next figure (top right) shows the distribution of predicted upvote ratios on the test set by the neural network model trained using the MAE loss. The third figure (bottom left) shows the distribution of true upvote ratios after smoothing, with a fitted probability density curve. Finally, the fourth figure (bottom right) shows the distribution of predicted upvote ratios using the neural network model trained on the WMAE loss.

1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399

Figure 20: Top left - distribution of true upvote ratios for test set for r/uofm. Top right - distribution of predicted labels of test set using neural network trained on MAE loss for r/uofm. Bottom left - distribution of true upvote ratios for test set after smoothing and fitted density function for r/uofm. Bottom right - distribution of predicted labels of test set using neural network trained on WMAE loss for r/uofm.
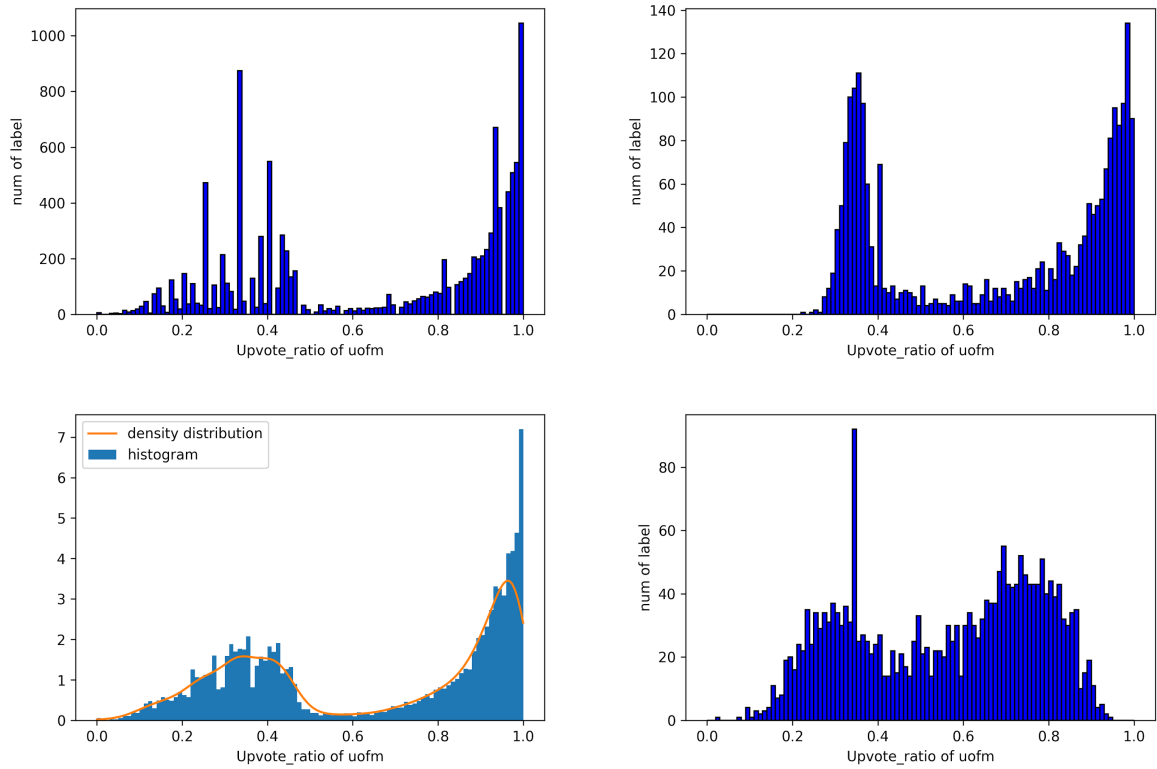
14

Figure 21: Top left - distribution of true upvote ratios for test set for r/berkeley. Top right - distribution of predicted labels of test set using neural network trained on MAE loss for r/berkeley. Bottom left - distribution of true upvote ratios for test set after smoothing and fitted density function for r/berkeley. Bottom right - distribution of predicted labels of test set using neural network trained on WMAE loss for r/berkeley.

1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549

1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
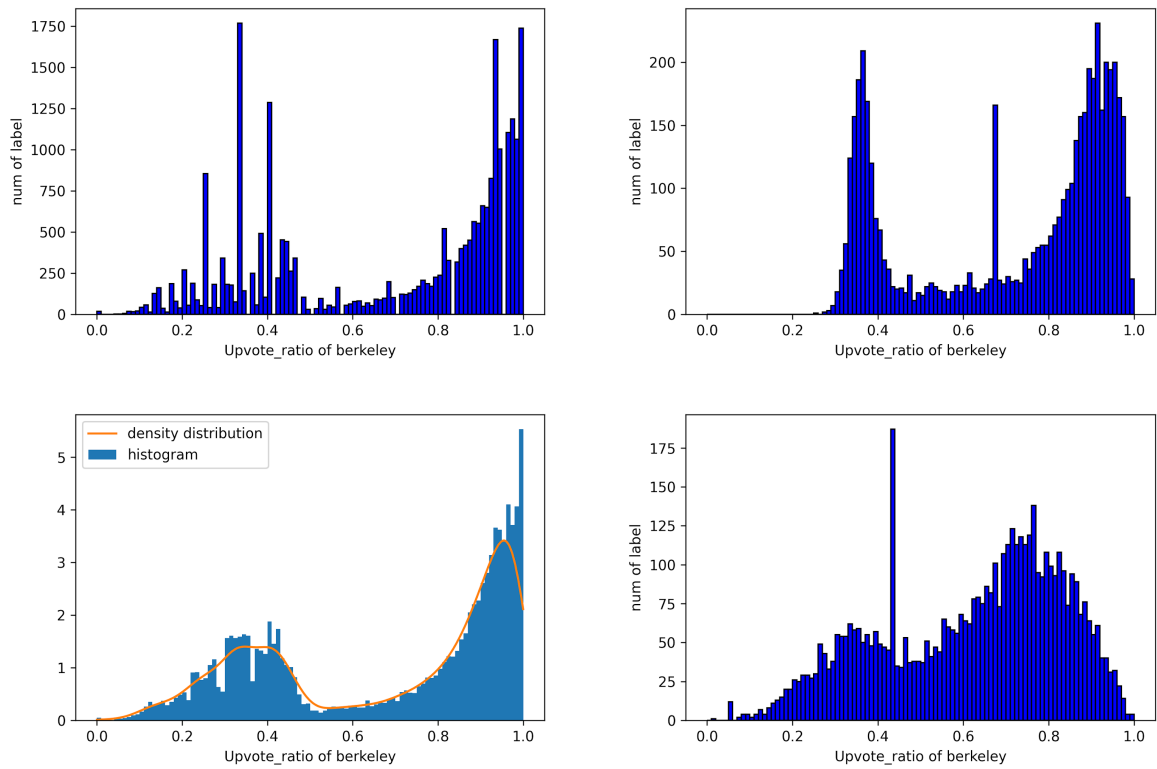1594
1595
1596
1597
1598
1599

Figure 22: Top left - distribution of true upvote ratios for test set for r/uwaterloo. Top right - distribution of predicted labels of test set using neural network trained on MAE loss for r/uwaterloo. Bottom left - distribution of true upvote ratios for test set after smoothing and fitted density function for r/uwaterloo. Bottom right - distribution of predicted labels of test set using neural network trained on WMAE loss for r/uwaterloo.

1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649

1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
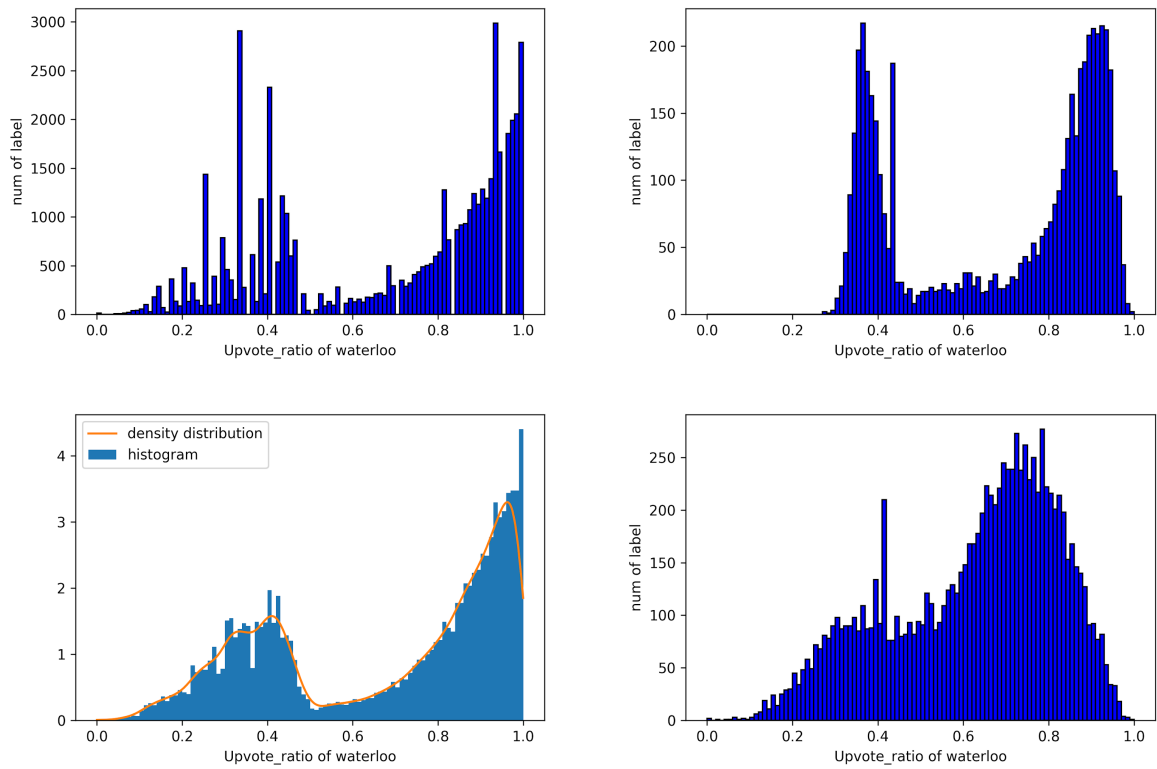1688
1689
1690
1691
1692
1693
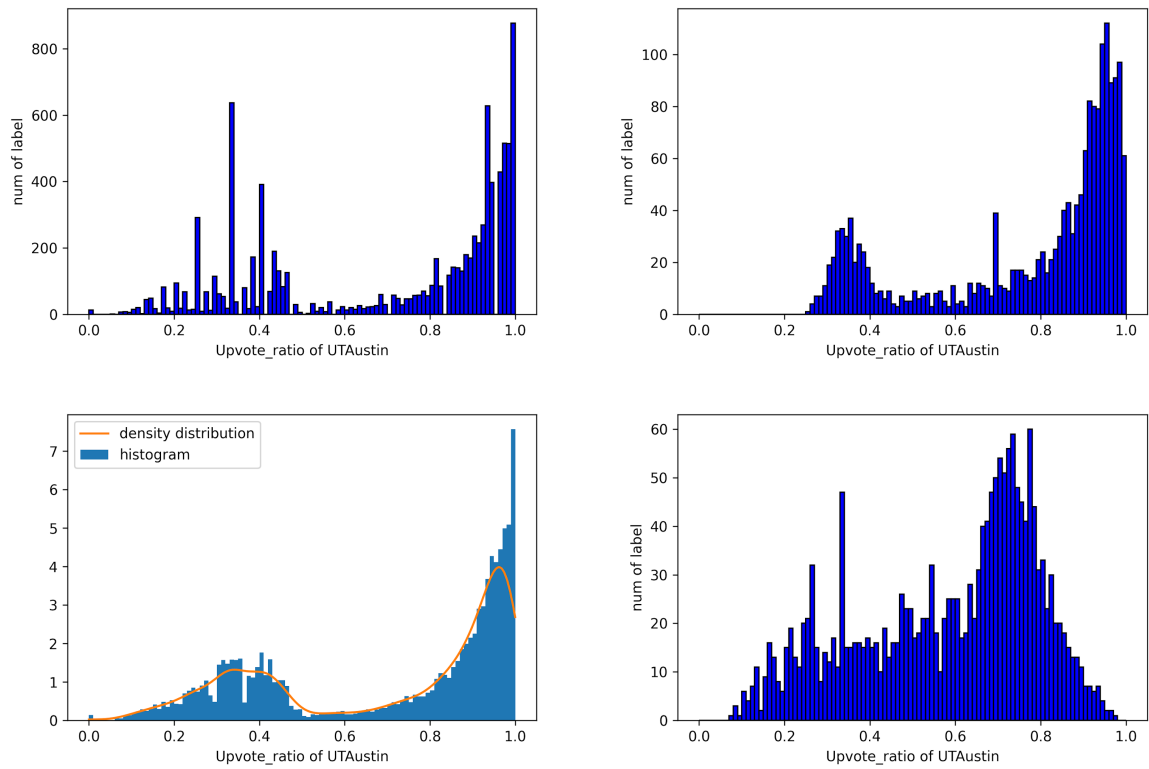1694
1695
1696
1697
1698
1699

Figure 23: Top left - distribution of true upvote ratios for test set for r/UTAustin. Top right - distribution of predicted labels of test set using neural network trained on MAE loss for r/UTAustin. Bottom left - distribution of true upvote ratios for test set after smoothing and fitted density function for r/UTAustin. Bottom right - distribution of predicted labels of test set using neural network trained on WMAE loss for r/UTAustin.

17

A.4 Results for the classification of upvote ratio.

Included below are the full results for each subreddit of each model discussed for task 3 - the three class classification of upvote ratio.

	KNN	Rand. Forest	Neural Net	GMM
Acc.	0.530	0.526	0.531	**0.536**
F-1	0.493	0.435	0.398	**0.514**
UAR	0.499	0.502	0.346	**0.518**

Figure 24: Results for each model trained on task 3 in r/uofm (Acc. refers to accuracy).

	KNN	Rand. Forest	Neural Net	GMM
Acc.	0.539	0.518	**0.546**	0.518
F-1	0.474	0.386	0.446	**0.497**
UAR	0.488	0.479	0.498	**0.501**

Figure 25: Results for each model trained on task 3 in r/berkeley (Acc. refers to accuracy).

	KNN	Rand. Forest	Neural Net	GMM
Acc.	0.465	0.449	**0.474**	0.465
F-1	0.458	0.415	0.459	**0.460**
UAR	0.458	0.444	**0.476**	0.459

Figure 26: Results for each model trained on task 3 in r/uwaterloo (Acc. refers to accuracy).

	KNN	Rand. Forest	Neural Net	GMM
Acc.	0.553	0.553	**0.563**	0.522
F-1	0.410	0.272	0.331	**0.462**
UAR	0.481	**0.496**	0.484	0.480

Figure 27: Results for each model trained on task 3 in r/UTAustin (Acc. refers to accuracy).