Part 1

a) Model 1 – n-grams

Accuracy	Precision	Recall	F1	Specificity	AUC
0.614	0.320	0.416	0.362	0.685	0.551

Preprocessing:
To extract the top 500 unigrams and top 500 bigrams from the posts (the 'request_text' column) in the training set which make more sense and help achieve better classification performance, I used several data pre-processing techniques before actually started extracting n-grams features.

First, I lowercased the text of posts, because for the tasks of identifying unigrams and bigrams and even for model 3 and 4, there is no need to distinguish the capital words and small case words. Then, I used NLTK Library's *word_tokenize* to break the text into a list of tokens (i.e. words). This step is important for extracting n-grams as well as the later word lemmatization task.

Furthermore, I also removed stop-words and non-alphabet words and performed word lemmatization at the same time. Stop-words and non-alphabet words would contribute nothing to the classification task but only cause worse performance, especially for the n-grams model because they are likely to appear in the top 500 unigrams. For words that are not in the English stop-words library and are the alphabet, I then used the *WordNetLemmatizer* provided by NLTK to lemmatize the words to somehow normalize them but at the same time keep the resulting words meaningful and valid. These lemmatized request posts will be used to extract n-grams and do regular expression matching tasks for model 3 and 4.

Before preparing training and test data sets, I first encoded the target label (i.e. 'requester_received_pizza'). Then I used *model_selection.train_test_split* provided by the sklearn library to split the data into training and test data sets. The parameter *test_size* has been set to 0.1 because we wanted to have 90% of the data (i.e. 5104 posts) to be our training data set. Also, to make sure the training and test sets are consistent across different classification models, I set the *random_state* to be 42.

Feature Extraction:
After data pre-processing, I extracted all words from X1_train into a list and used *set()* to get unique unigrams. I then counted the frequency of each unigrams in the posts of the training set and extracted the top 500 most frequently occurring unigrams. Next, I used bigrams from NLTK Library to extract all of the bigrams of each post in the training set, and I also used *set()* to get unique bigrams. I then counted the frequency of each bigrams in the posts of the training set and extracted the top 500 most frequently occurring bigrams.

Furthermore, I used CountVectorizer from the sklearn library to convert the top 500 unigrams and bigrams to feature names. Then, I vectorized each post by transforming the training and test data into a matrix respectively based on unigrams and bigrams. As a result, for each post, I got a 1000-dimension vector, which counted the frequency of occurrence of each unigram and bigram in each post.

In terms of the linear SVM model for the four classifiers, I set the *C* parameter to 1.0 and chose a balanced class weight.

b) Model 2 – Activity and Reputation

Accuracy	**Precision**	**Recall**	**F1**	**Specificity**	**AUC**
0.658	0.416	0.745	0.534	0.628	0.686

Preprocessing:
I first encoded an activity's label (i.e. 'post_was_edited'). Then, for each activity or reputation data (21 in total), I split the data into training and test data sets. Again, the parameter *random_state* is 42 to make sure the training and test sets are consistent across different classification models.

Feature Extraction:
For the 'requester_subreddits_at_request' feature, I used CountVectorizer to convert all subreddits included in the 'requester_subreddits_at_request' of training post set to feature names. Then, I vectorized each post's 'requester_subreddits_at_request' by transforming the training and test data into a matrix respectively based on subreddits. As a result, for each post, I got an 8395-dimension vector (because 8395 subreddits in total appear in the training set), which takes notes of whether the requester used a certain subreddit at request for each post. Next, I converted the remaining activity and reputation data frames to matrices and then joined the 21 matrices into a single matrix for the training set and test set respectively.

c) Model 3 – Narratives

Accuracy	**Precision**	**Recall**	**F1**	**Specificity**	**AUC**
0.609	0.325	0.456	0.380	0.663	0.560

Preprocessing & Feature Extraction:
I first used *split('\n')* to construct five lists that contain the narrative-associated words for each narrative. Then, I used *'|'.join()* to construct five strings that are used as the regular expressions for later matches.

Furthermore, I performed regular expression match between all words corresponding to the narrative and those corresponding to each post using the strings constructed previously. The post data I used here to do the regular expression match is the lemmatized posts (i.e. 'review_final') to make sure the same words of different forms can all be matched. Then, I calculated the ratio of the number of matches for each narrative to the total number of white-spaced words in the posts and stored them in five newly added columns respectively. These would be the features extracted for this classifier.

For each narrative data frame, I split the data into training and test data sets. Next, I vectorized each narrative feature by converting the five data frames to matrices and then joined them into a single matrix for the training set and test set respectively.

d) Model 4 – Moral foundations

Accuracy	**Precision**	**Recall**	**F1**	**Specificity**	**AUC**
0.722	0.263	0.034	0.060	0.967	0.500

Preprocessing & Feature Extraction:
I first constructed a list that contains each line of the moral foundations dictionary. I also used strip() to remove the white spaces at the beginning and end of lines to make the list look neater. I then sliced the list to only contain the term lines. Next, I constructed eight vocabulary lists for each moral foundations dimension, using regular expression match to capture all the terms indexed by the same integer (which is mapped to a moral foundations dimension) into a list. Then, I used *'|'.join()* to construct eight strings that are used as the regular expressions for later matches.

Furthermore, I performed regular expression match between all terms corresponding to the moral foundations dimension and those corresponding to each post using the strings constructed previously. The post data I used here to do the regular expression match is the lemmatized posts (i.e. 'review_final') to make sure the same words of different forms can all be matched. Then, I calculated the ratio of the number of matches for each moral foundation dimension to the total number of white-spaced words in the posts and stored them in eight newly added columns respectively. These would be the features extracted for this classifier.

For each narrative data frame, I split the data into training and test data sets. Next, I vectorized each moral foundation dimension feature by converting the eight data frames to matrices and then joined them into a single matrix for the training set and test set respectively.

Part 2

a)
Among the four classifiers, even though the moral foundations classifier has the highest accuracy (0.722), its recall is extremely low, indicating that there is a large number of false negatives (i.e. 144). Considering that the test data set has imbalanced cases (i.e. 73.77% of the data set is negative cases), it is hard to tell that the moral foundations classifier performed the best, and in fact, it performed the worst. Among the remaining three classifiers, the Activity and Reputation classifier performed the best given that most of its evaluation metrics are higher than others.

b)
The differences in performance of the four classifiers are fundamentally driven by the different features extracted from posts that are used to train models. According to Althoff, Danescu-Niculescu-Mizil, & Jurafsky's (2014) study, some textual factors (i.e. narratives, reciprocity, length, gratitude), temporal factors (i.e. community age), and social factors (i.e. status) are all predictive of the success of altruistic requests.

The n-grams model extracted the top 500 unigrams and bigrams as features, which might be able to capture many textual factors, like narrative, reciprocity, and gratitude. The Activity and Reputation model extracted a number of features that could precisely, to some extent, capture the users' status in the ROAP subcommunity and the Reddit community. The Narratives model extracted five narrative features, which could help identify which or which combination of narrative(s) a post has used. Since the factors captured by these three models are proven to be significantly associated with the success of altruistic requests, they all show relatively good performance with an accuracy around 0.61. The Activity and Reputation model performed a bit better, because the social factors have a relatively higher statistical significance compared to others according to the logistic regression model developed in Althoff et al.'s (2014) study. The Moral foundations model, however, extracted moral foundations features, which are not mentioned in Althoff et al.'s (2014) study, and thus their effectiveness in predicting success is unknown, making the model perform the worst.

c)
Model 3 and 4 performed worse than model 1 and 2. Specifically, even though there is no significant difference between the performance of model 1 and 3, model 4 has much lower precision, recall (0.034), and F1, while the recall of model 2 is 0.745 (i.e. it has a high number of true positives). Model 1 performed slightly better than model 3, because model 3 only captured the narrative factors, while model 1 could capture other textual features like gratitude and reciprocity apart from narrative. Although people across cultures show many similarities in their morality, and whether offering a free pizza to a stranger online is something related to morality or intuitive ethics, there is no evidence that moral foundations are good indicators of the success of altruistic requests. Whereas model 3 extracted 21 activity and reputation features that could characterize a user's status in the community. For instance, the *'requester_upvotes_minus_dow- nvotes_at_request'* helps define the user's overall status in the Reddit community, and the *'number_of_downvotes_of_request_at_retr- ieval'* and *'number_of_upvotes_of_request_at_retrieval'* help define the user's status in the ROAP subcommunity. Both status factors are highly correlated with success (Althoff et al. 2014).

Between model 3 and 4, model 3 performed much better than model 4. Model 4 predicted most of the test data to be "fail to receive a pizza" (549 in 568), which resulted in 144 false-negative cases and 5 true-positive cases, whereas model 3 only has 81 false-negative cases and 68 true-positive cases. Therefore, model 3's recall and F1 are much higher than model 4. The reason behind these differences is where the lexicons for each narrative or moral foundations dimensions come from. In Althoff et al.'s (2014) study, they first did a topic modeling to identify the topics of request posts and then built the lexicons for each narratives in requests by picking the topics with higher success rates and referring to related LIWC categories. This process could ensure that the topics and lexicons did occur frequently in the request posts and are associated with the success of altruistic requests. In contrast, the lexicons for moral foundations are not derived from the posts, and there is no study that confirms these moral foundations dimensions did occur in the posts. As a result, there are 10129 matches between the words corresponding to the narratives and those corresponding to training posts, while there are only 5568 matches between the words corresponding to the moral foundations and those corresponding to training posts.

d)
Based on the prediction performance of model 1, 3 and 4, it is clear that language is able to predict the success of altruistic requests to some extent, depending on which linguistic features are used in the prediction model. The prediction result of model 4 indicates that moral foundations might not be the suitable linguistic factors that are predictive of success. However, the result of model 3 suggests that narratives are highly correlated with the success of altruistic requests, because the success rates of different narratives vary substantially. For instance, rather than requesting a pizza simply with a “desire” narrative, people are more willing to help those who are requesting with a “money” or “job” narrative which indicates the hard times they are experiencing and the urgent need for a pizza. The slightly better performance of model 1 also suggests that there are additional linguistic features such as gratitude and reciprocity that could help the prediction. That is because people are more willing to support those who show gratitude and would like to pay it forward to contribute to the community in the future.

Part 3

a)
Since Althoff et al.’s (2014) study didn’t use moral foundations as features to do the prediction, and I didn’t use temporal features and the combination of different features for the classification, here I only compared the first three models of mine with the unigram, bigram baseline, the text features model, and the social features model of the study.

In terms of the ways to construct the classification model, the previous study and I chose different models: the previous study used logistic regression models, while I used Support Vector Machine models. Moreover, for each model, we chose slightly different features to extract or vectorized them in different ways. For the n-grams model, I extracted the top 500 unigrams and bigrams respectively and combined them to construct a 1000-dimension vector as the n-grams feature for each post, while the study used the standard unigram and bigram model separately and didn’t pick only the 500 for each feature. For the social features model, while the study only included the karma score (i.e. the *‘requester_upvote-s_minus_dow- nvotes_at_request’*) in the model, I extracted 21 features representing the user’s activity and reputation levels for model 2. For example, the ‘*requester_number_of_co-mments_in_raop_at_request’* and *‘requester_number_of_posts_on_raop_at_request’* were used to identify if the user is already a member of the ROAP subcommunity before requesting a pizza, and I also included the *‘requester_subreddits_at_request’* feature which is a list of subreddits the user has already posted at least once at the time of request. In addition, I tried to keep the numeric data as it was when doing vectorization, while the study vectorized the karma score as a decile-coded variable. For the textual features model, while the study included 9 features (i.e. narratives, evidentiality, gratitude, reciprocity, and length), I only used the 5 narratives features. Again, the previous study converted the narratives features to decile-coded variables, while I calculated the ratio of the number of matches for each narrative to the total number of white-spaced words in the post and used it as the features for model 3. Besides, the previous study also tried to combine the different feature sets to see if the performance could be further improved, while I didn't try to combine any of them.

b)

For the n-grams model, my model's AUC score (0.551) is significantly lower than the previous study's (unigram: 0.621, bigram: 0.618). While the paper didn't include any details about how they performed the unigram and bigram models, I suppose the differences are mainly due to the data preprocessing, the choice of classification model, and the model parameters. While I used tokenized the stream of text and removed the stop-words and non-alphabet words and did word lemmatization, the study might use advanced preprocessing techniques like lemmatizing based on part-of-speech tag. Since we chose different classification models and I only added one more parameter 'class_weight = balanced', the study might tune the parameters for better performance.

For the social features model, my model's AUC score (0.686) is significantly higher than the study's (0.576). That is because I have included much more activity and reputation data that could better characterize a user's status not only in the Reddit community but also in the ROAP subcommunity. For example, in addition to the user's karma score, the '*requester_number_of_co- mments_in_raop_at_request'* is also predictive of success because people are more willing to help those who are already a member of the community and have contributed to the community previously. Also, although the study concluded that the similarity (captured by the *'requester_subreddits_at_request' data)* between the requester and giver is not significantly correlated to success, this data might have another potential to help improve the performance, because it might help identify the typical subreddits a successful requester might use.

For the textual features model, my model's AUC score (0.560) is significantly lower than the study's (0.625). One possible reason that could explain this difference is I only extracted the narratives features while the study extracted much more features including length and evidentiality which are significantly correlated to the success of altruistic requests. The different classification models and parameters, and the different ways we vectorized the narratives might also contribute to the difference in performance.

References

Althoff, T., Danescu-Niculescu-Mizil, C., & Jurafsky, D. (2014, May). How to ask for a favor: A case study on the success of altruistic requests. In Proceedings of the International AAAI Conference on Web and Social Media (Vol. 8, No. 1, pp. 12-21).