## Classifying Amazon Fine Food Reviews

**Team:** Grace Benner, Karen Li, Kennedy Ellison, **Project Mentor TA:** Yiming Huang

<https://github.com/gebenner1/CIS5190Project>

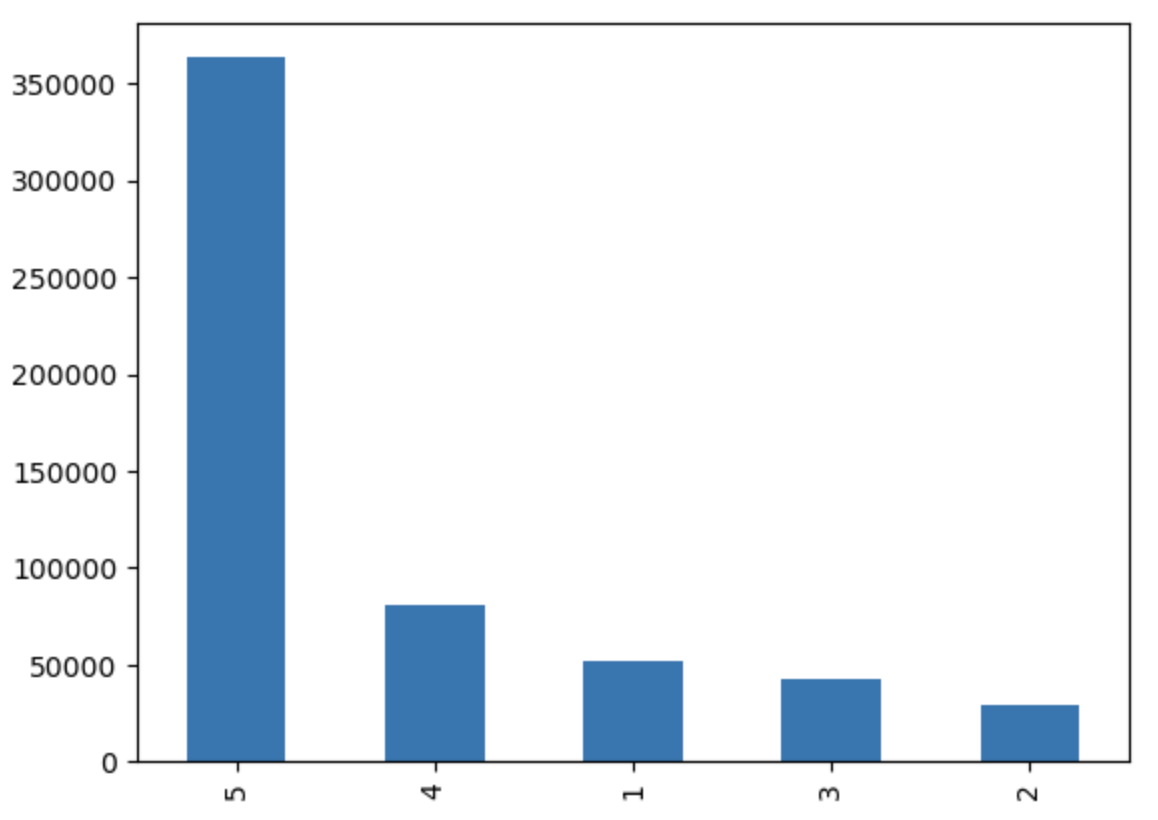
#### **1) Abstract**

This project explores the problem of sentiment analysis using a database of Amazon reviews and their accompanying ratings. This problem involves predicting the positive or negative sentiment of the review, which provides a useful tool to evaluate the natural language input that generalizes across many different grammatical or discourse structures. To study this problem, we used the traditional Word2Vec embedding approach, Bag of Words, a more tabular approach focused on feature engineering reviews, and also compared the BERT and RoBERTa models. This array of approaches allowed us to compare the efficacy of different methods and weigh their strengths and weaknesses. Overall, we found that some of the simpler models and feature structures performed the best. In the multiclass prediction case we found that the feature engineering method with a Random Forest model outperformed the more complex models, and Bag of Words outperformed all others in the binary setting.

**2) Introduction**

Our dataset contains Amazon food reviews from 1999-2012 of over 74k products for a total of over 568k reviews. There are around 256k users and only 260 users have more than 50 reviews so there is a good amount of diversity within the reviews. The dataset contains 10 features. The most important ones we will be focusing on are “Score” (1, 2, 3, 4, 5) which is the rating of the fine food product, and “Summary” and “Text” which are the written reviews. In most of our models we are using word embeddings of “Summary” and “Text” as inputs to predict the “Score” as output, but we are also using feature engineering techniques on “Summary” and “Text” to create additional inputs to traditional ML models.

Specifically, a “Summary” is on average 23.45 characters long, whereas “Text” is on average 436.22 characters long. Looking at the distribution of the “Score” variable, we noticed that it is highly skewed. The vast majority of fine food reviews gave a score of 5, there are over 350,000 records with this score. The second most given score is 4, with around 80,000 reviews associated with that score. Scores of 2 and 3 are the least common, most likely since they are the most middle-grounded. Many people who choose to leave reviews either love or hate the product.



**Figure 1**. Distribution of the review scores.

***Implementation:*** We are pursuing Option 1. We are interested in comparing different model types, as well as comparing the inputs to the same models. In terms of the ML models, in particular, we plan to consider: (i) transformers, (ii) random forests, (iii) XGBoost, (iv) KNN. In training these models we experimented with variations of both the Word2Vec and Bag of Words embeddings, testing the effect of sentiment-specific embedding as compared to general embedding, and even are trying to use feature engineering of the text data to set up a more “traditional” tabular structure to feed as inputs to models.

***Evaluation:*** We evaluate our approach by calculating the accuracy of correctly predicting the ratings accompanying the reviews, as well as investigate other measures like F1-score due to class imbalance. Each model has different associated hyperparameters and we will tune these values based on the metrics of the test dataset. We are predicting review scores, which take on values 1 through 5, also evaluating binary classification by mapping scores less than 4 to a negative bin and scores 4+ into a positive bin.

***How We Have Addressed Feedback:*** The feedback about multiclass classification was particularly helpful, binary prediction is faster and reduces the severity of the class imbalance. Another point of feedback is using the review summary instead of the full review. We mainly evaluated the success of using just the summaries, however some of our models still try to incorporate the full text when it was not too computationally ineffective. Lastly, regarding feedback from milestone two, we are investigating binary and multi-class classification for all of our methods so that we compare the results between methods instead of between binary vs multiclass.

#### 

#### **3) Background**

In terms of transformer models, we use a roBERTA transformer model below(A). We also found a BERT transformer model which we thought would be interesting to evaluate and compare to the other transformer model since the BERT model is specifically on reviews. Another difference is that BERT is not trained on English specific data like the RoBERTA model.

In addition to the work described above, some of the relevant work that we built on is shown below:

1. Sentiment Analysis using roBERTa’

<https://www.kaggle.com/code/ivanvitales/sentiment-analysis-using-roberta>

This codebase is relevant to our project because it provides a baseline example of how to use a transformer model to form predictions. We use this as a baseline to see what improvements we can make using the same model as well as compared with another transformer model.

1. Text Data Feature Engineering <https://www.kaggle.com/code/shivamb/extensive-text-data-feature-engineering/notebook>

This codebase is relevant to the feature engineering done on the dataset. It gave good ideas of what ways we could transform the Summary and Text columns into numeric features that could be used for prediction of score.

1. Vectorization on Amazon Fine Food Reviews

<https://www.kaggle.com/code/mharika/vectorization-on-amazon-fine-food-reviews/notebook>

This codebase offered perspective on data cleaning prior to vectorization and how to integrate multiple word embeddings to create one vector per review. It was also one of several notebooks that provided ideas about subsampling and evening training data, which were helpful during exploration and development.

#### 

#### **4) Summary of Our Contributions**

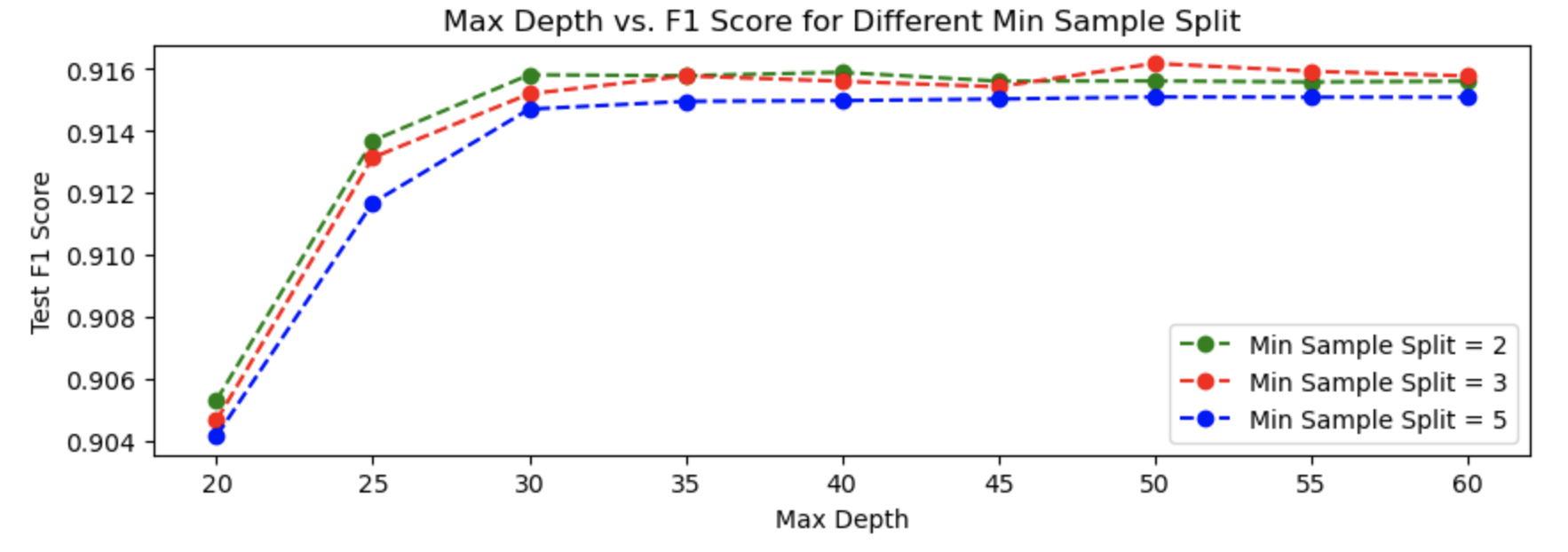
Our contributions in terms of implementation were building different structures of input features using feature engineering, Bag of Words and Word2Vec. These inputs were then run with standard supervised machine learning models to predict the Amazon find food review scores. On top of these traditional models we explored transformers, specifically BERT and RoBERTa. After hyperparameter tuning using test sets, we evaluated and compared our models with weighted F1 scores to find the best performing models in both the multiclass and binary settings.

#### **5) Detailed Description of Contributions**

##### ***5.1 Implementation Contributions***

##### One of the first questions we are asking is if traditional, tabular modeling, where we have inputs that are not word embeddings, can predict as well as Bag of Words or Word2Vec under the same ML models. We decided to feature engineer with the initial dataset to use in predicting score. The features we built were: proportion of people who found the review helpful, character and word counts of the summary and text, stopword and punctuation count of the text, punctuation count of the summary, word density (characters / words) for summary and text, and the polarity, subjectivity (TextBlob), and sentiment score (Afinn) of summary and text.

Next we are feeding these features into 3 different models, KNN, Random Forest, and XGBoost, which we hyperparameter tuned. The hyperparameters we focused on were the number of neighbors for KNN, and multiple different parameters for XGBoost and RF (Figure 1), but most importantly maximum depth, minimum samples split, and number of estimators. Before running KNN we made sure to standardize the features. We explored both multiclass classification of the score as well as binary classification (1, 2, 3 vs. 4, 5). Another very simple method we tried along with feature engineering was Bag of Words using the summary of the food reviews. This method counts the frequency of words for each of the summaries in the training dataset, removing both stopwords and punctuation. This resulted in a table with nearly 40,000 attributes. Since this is an extremely high dimensional dataset, which was computationally expensive and potentially prone to too much sparsity, we also experimented using only 5,000 of the most commonly occurring words, however this underperformed compared to the original feature set. Again, the Bag of Words table was fed into KNNs and RFs for both multiclass and binary classification. Unfortunately, the data was too big to run with XGBoost.



**Figure 2**. Feature engineered data performance predicting binary scores for differing hyperparameters.

To contextualize the utility of these “non-deep” methods of feature engineering and Bag of Words, we additionally experimented with the baseline of the Word2Vec embeddings on the review text. With these vector representations, we again trained our three models: KNNs, Random Forests, and XGBoost. Of these, Random Forests and XGBoost were the most effective. Based on the confusion matrices in the supplemental materials, we saw the strength of the random forest predictions using this approach. However, we additionally see a bias towards predicting a high/5 score, which reflects the skew noted in the introduction towards five star reviews. Compared to the other models, KNNs were less effective, which was likely due to the large dimension of our embedding vector (100). However, these results show that the Word2Vec approach can produce significantly better than chance classifications, even from raw word embeddings that do not necessarily indicate positive or negative sentiment. As with the previous tabular approach, we also considered binary classification, considering 4 and 5 as high scores and the 1, 2, and 3 as low scores. This change once again resulted in higher predictions from all models, about one percentile higher. Finally, we also experimented with embedding using the Doc2Vec variation of Word2Vec. This yielded positive results, but did not improve upon the Word2Vec approach.

The next question we asked was how these more classical approaches to NLP compare to transformer based models which have revolutionized NLP over the last few years. We investigated two different transformers, roBERTa and BERT which have slight differences which we noted in our background section. The main steps in this was to pass the reviews into a tokenizer to establish embeddings. These embeddings were then passed to each model for a score prediction. In the binary case, we followed the same process as before (low vs high score). In the multi-class scenario, we used whatever the model was trained to predict. For roBERTa it was trained to predict negative, positive, or neutral sentiment (-1,1,0 respectively), and for BERT it was just a 1-5 prediction directly related to the label score. For evaluation, due to the class imbalance of the dataset, weighted F1 score was used.

##### 

##### ***5.2 Evaluation Contribution***

Overall, we were extremely curious about how the different ways we structured the inputs would impact model performance. We asked the questions: Is there much benefit to added complexity and lack of interpretation by using deeper and deeper embeddings or models? Or does basic feature engineering obtain just as high metrics with much less computational energy and much more explainability? We explore these ideas for both multiclass and binary prediction models.

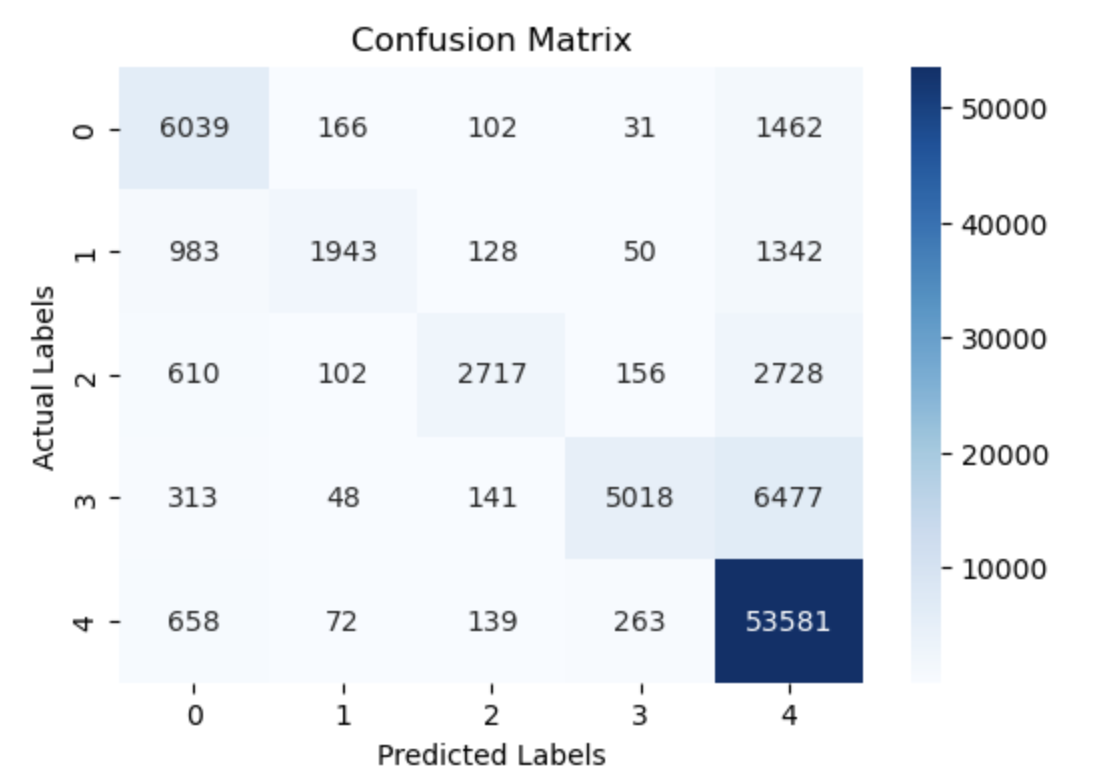
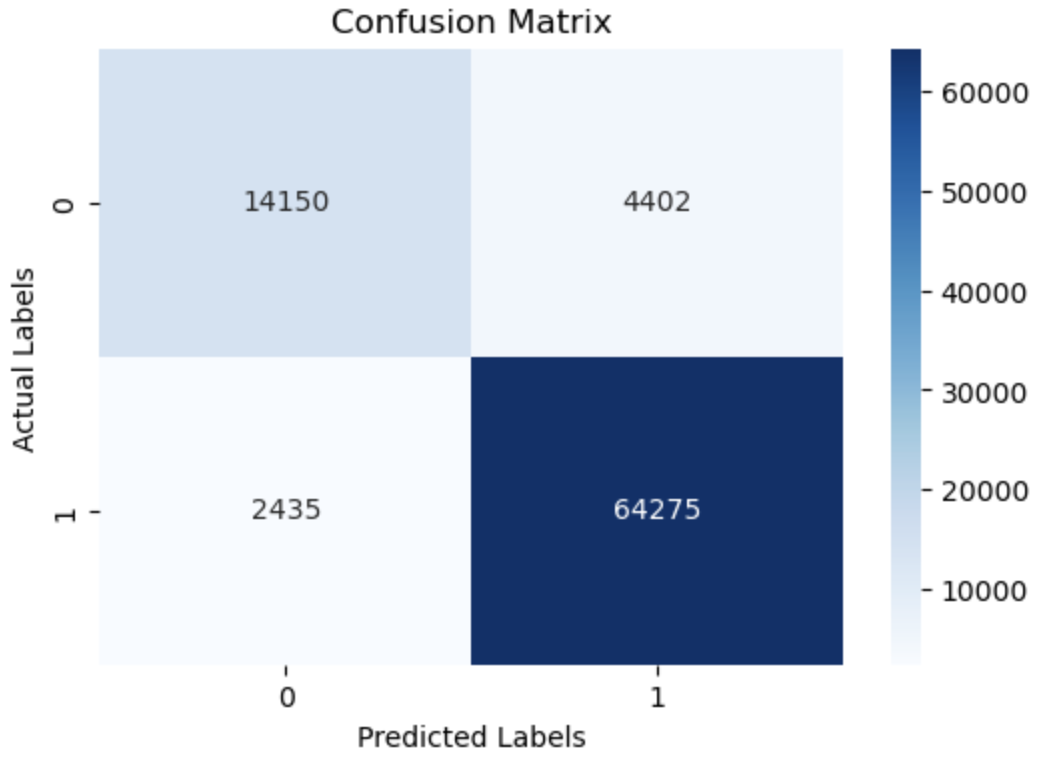
***Multiclass:*** Starting from the most basic model, using feature engineered data, the highest performing model was a Random Forest model with 100 estimators, a maximum depth of 40, and a minimum sample split of 3 had a weighted F1 score of 0.7916 and accuracy of 0.8127 on the test set. This was the highest weighted F1 score of all the models, yet it was the simplest set of features and used a basic ML model. Next, using the Bag of Words vector as the input to the models, we found the best results when we used the entire set of words rather than a reduced number. The highest performing model was also a Random Forest model, with 100 estimators, a maximum depth of 2000, and a minimum sample split of 2. It had a weighted F1 score of 0.7641 and accuracy of 0.7819 on the test set. We were surprised to have such a high maximum depth, but the feature space is so large it is what performed the best on the test set. The BOW underperformed compared to the feature engineered data, however due to long running times, less hyperparameters were tested.

For Word2Vec embeddings, Random Forests and XGBoost were the most effective at about 0.78 accuracy. The best model was a Random Forest model, with 200 estimators and a minimum sample split of 1. The confusion matrices for this model can be found in the supplementary materials. Notably, the XGBoost models could likely have been tuned for a better accuracy and F1 score, but due to their longer running time it was more difficult to fine tune the hyperparameters. The least effective models were the KNNs at around 0.66 accuracy, a score that was generally invariant to the tuning of the number of neighbors evaluated. As previously mentioned, this was likely due to the large dimension of our embedding vector (100). Because the word embeddings were not a priori semantically interpretable (for instance, because no one index would necessarily correspond to negation), the size of this vector may have muddled the distances between vectors of similar sentiment.

Lastly, for the transformer models, the best multi-class result for roBERTa was 0.771 weighted F1 score and 0.658 weighted F1 for BERT. This was surprising because our initial hypothesis was that BERT would outperform roBERTa since it was trained on review data. It seems the multilingual nature of the BERT dataset was an impediment to performance. Another reason for poor performance could be due to BERT predicting on five classes while roBERTa only predicts on three classes (less difficult).

***Binary:*** Switching from predicting the scores 1, 2, 3, 4, and 5 all separately, we now investigate the binary setting where we look at low scores of 1, 2, 3 compared to high scores of 4 and 5. The results of the models using the feature engineered data had the highest performance with a random forest model with 100 estimators, a maximum depth of 50, and a minimum sample split of 3 had a weighted F1 score of 0.9162 and accuracy of 0.9187 on the test set. Next, using the BOW vector as the input, we again found the best results when we used the entire set of words rather than a reduced number. The highest performing model was also a Random Forest model, with 100 estimators, a maximum depth of 300, and a minimum sample split of 3. It had a weighted F1 score of 0.9181 and accuracy of 0.9198 on the test set. This was a slight improvement to the feature engineered data.

With the Word2Vec approach, we considered 4 and 5 as high scores and 1, 2, and 3 as low scores for binary classification. As with the tabular approach, this simplification was easier for all the model types to predict, with accuracy in the 0.85-0.89 range for all models. The most effective model was once again a Random Forest model with 200 estimators, but this model’s 0.89 accuracy only narrowly improved upon XGBoost’s 0.88 accuracy. The f1 score for the most accurate model, the random forest, was 0.89, but this was not optimized because the model hyperparameters were adjusted based on accuracy. Lastly for the transformer based models, it is important to note that our roBERTa results significantly outperformed the Kaggle dataset we based our work on which only had an accuracy of 0.06 compared to our best roBERTA accuracy was 0.881. The best roBERTa result was a weighted F1 score of 0.889 and the best BERT result is 0.895 weighted F1 score. This slightly validates our belief that the poor performance for multi-class is due to five compared to three labels. In the binary scenario, we are seeing that BERT is slightly better than roBERTa, as we originally predicted.



**Figure 3 and 4.** The confusion matrix for the best performing models (both Random Forest).

| **Model** | **Binary Test F1** | **Multi-Class Test F1** |
| --- | --- | --- |
| Roberta | 0.889 | 0.771 |
| Bert | 0.895 | 0.658 |
| Word2Vec | 0.896 | 0.751 |
| Bag of Words | 0.918 | 0.764 |
| Feature Engineering | 0.916 | 0.792 |

**Table 1.** Comparison of weighted F1 test scores across all models, with highest F1s highlighted.

#### 

#### **6) Compute/Other Resources Used**

To complete this project we used AWS SageMaker and Google Colaboratory to run the notebooks. We used varying libraries for sentiment analysis (see notebooks in Git), but used PyTorch and Scikit-Learn to implement and evaluate the models. Some Kaggle codebases also aided our project, see above.

#### **7) Conclusions**

***Outcomes:***Overall, we found the results of this project to be very interesting. Some of the most simple models and formats of inputs outperformed even the transformer models, though not significantly. We attribute this success to the ability to hyperparameter tune better with simpler models, as well as being able to try more hyperparameters since these simpler models took much less time to run compared to transformer models.

***In Hindsight:*** Some roadblocks encountered in the baseline modeling using both the feature engineered data and the Bag of Words vector as the input was the running time of the models. We were not able to use cross validation to hyperparameter tune, but could only use one train, test split. This was acceptable since the size of the dataset is so large, our test set still had nearly 100,000 records. The size of the BOW vector also impacted the ability to try XGBoost as one of the models. One roadblock we encountered related to transformers was time. Due to slow result output, we elected to stick with the first 100k samples for BERT. For the roBERTa model, the prior work only used 500 samples. We increased it to about 34k, but ran into failures when using more on the transformer side.

***For the Future:*** One potential way to extend our work would be to see how finetuning of transformer models improves performance. We achieved great similar results to the classical forms of ML we used without fine tuning, but it would have been interesting to see how performance would change with it. However, given the BERT model was trained specifically on reviews, there might not be a huge improvement by using fine tuning other than getting more English focused examples. We briefly explored finetuning but had memory issues even after increasing memory on the AWS side so we elected to move that to future work. It is important to note that we had not included fine tuning in our proposal so our investigation into finetuning was just nice to have. However, given more time it would be interesting to explore how finetuning can be implemented, especially as it is more memory intensive than just using the models out of the box.

***Ethical Considerations, and Broader Social and Environmental Impact:*** In general, NLP has been known to introduce bias that was not necessarily intended, so when working with text it is always important to evaluate the results of models. Given that these models will generate predictions along one dimension (ratings), we would be wary of using it for some purpose other than review assessment. In particular, with human interactions, there are many other factors (formality, emotional state) to take into account that cannot be captured by this singular dimension. We also should consider the bias that could be introduced due to the class imbalance in the dataset and how significantly reviews can differ from person to person.

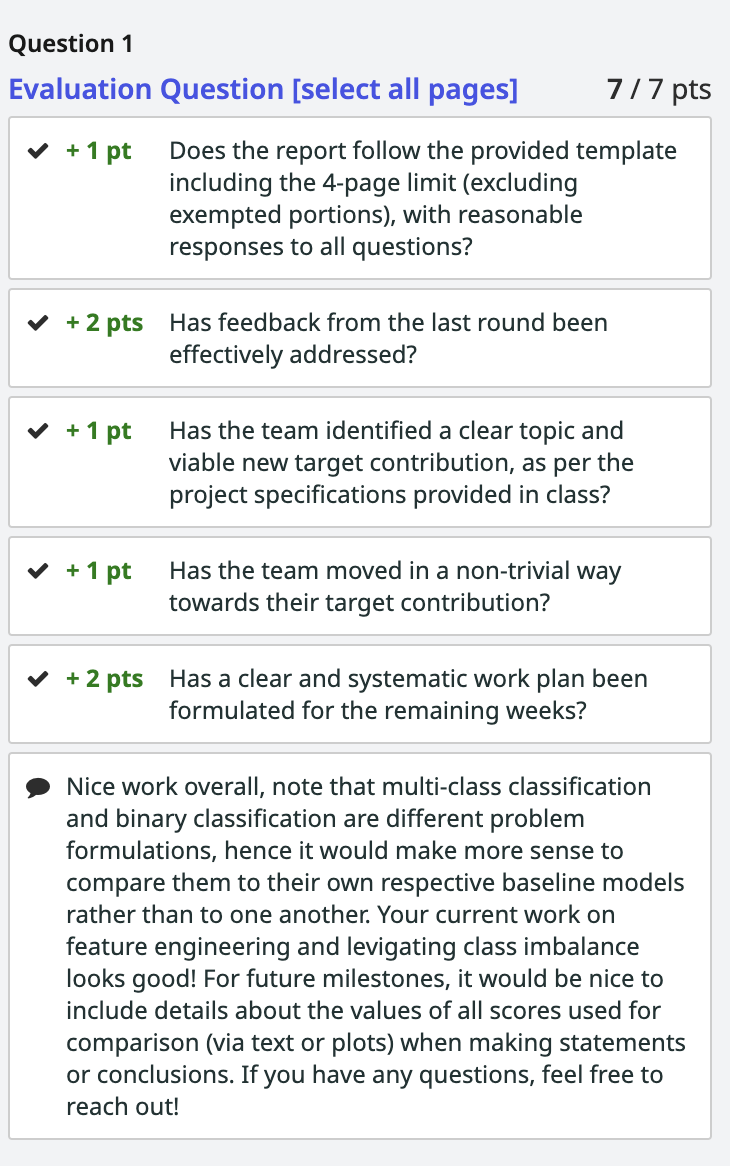
**8) Roles of team members (1-2 sentences each):**

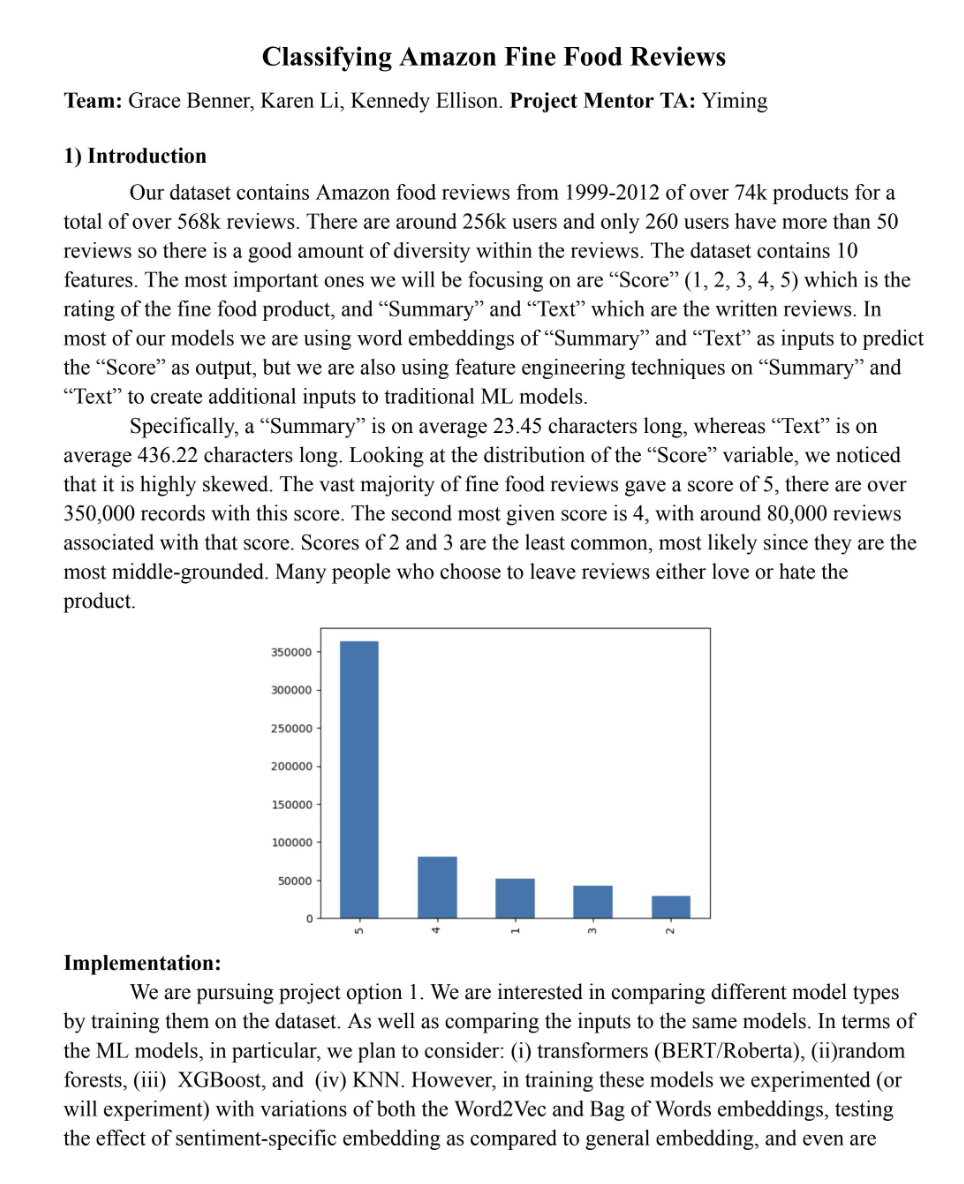
**Grace Benner:** Worked mainly on the feature engineering of the text and summary columns, as well as Bag of Words implementation. Built and tuned multiclass and binary models to predict score using both types of inputs.

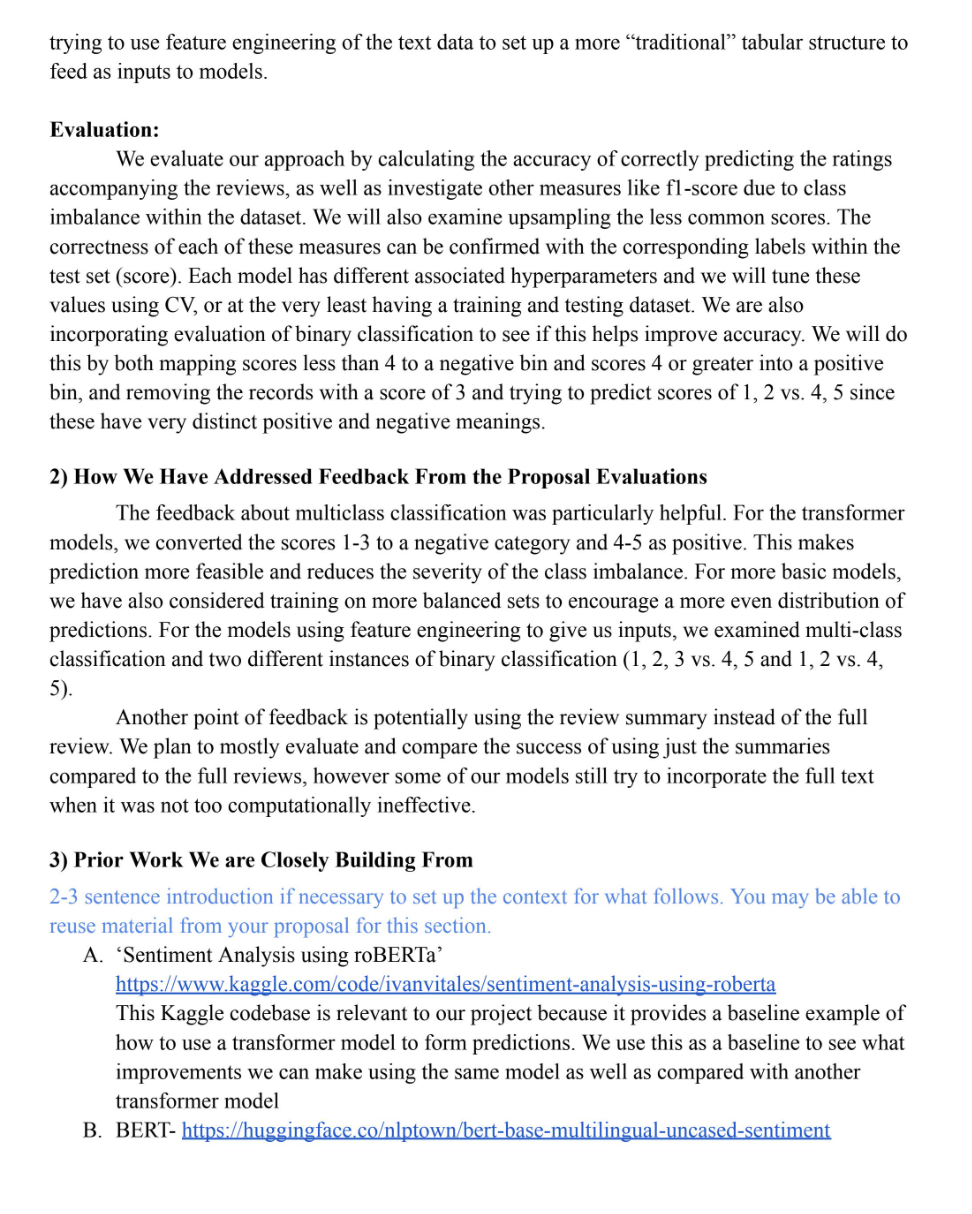
**Kennedy Ellison:** Worked on evaluating the two different transformer based models (how they are used and the different outputs they present as well as determining how best to evaluate them). Eventually settled on weighted F1 scores for both the binary and multi-class scenarios.

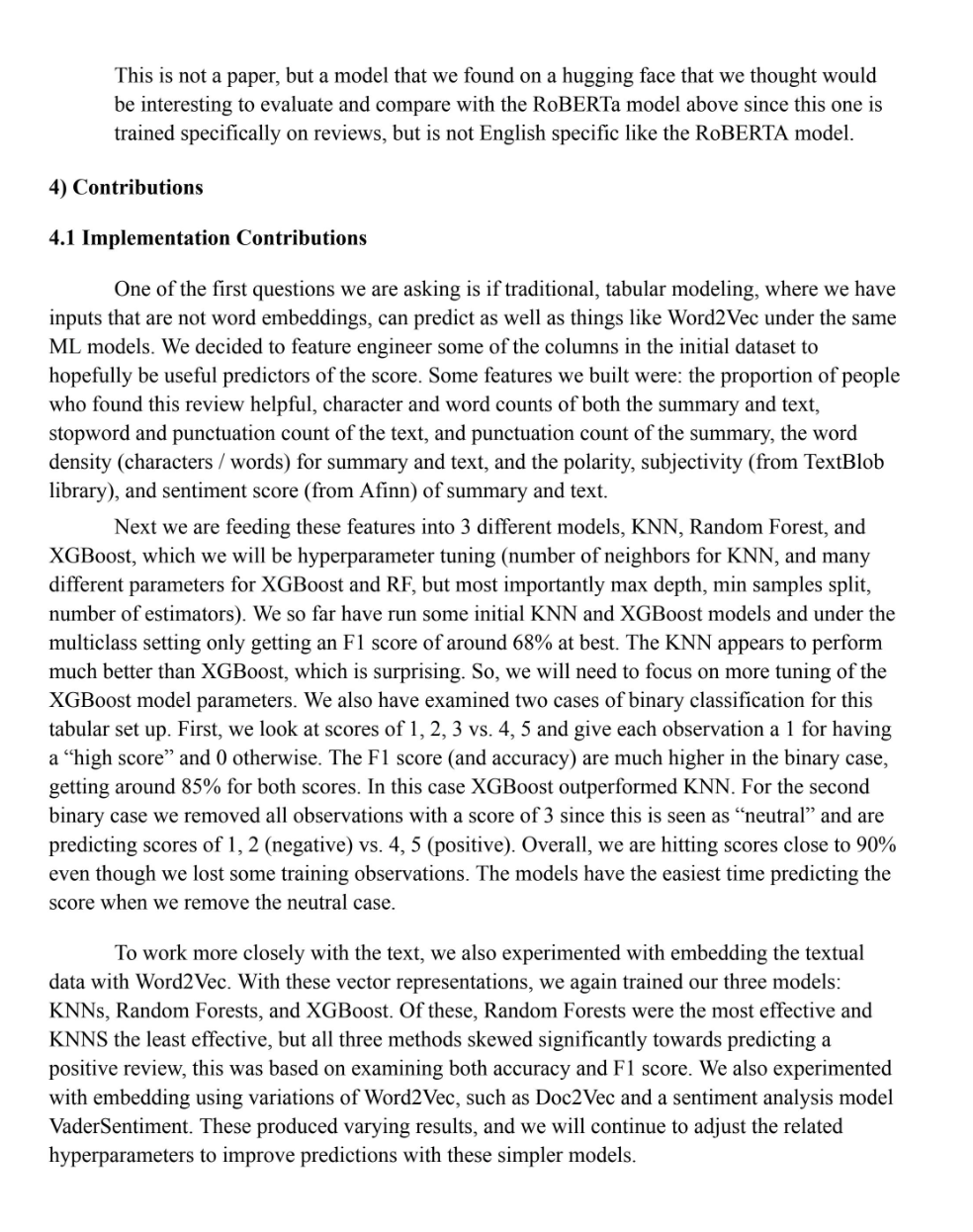
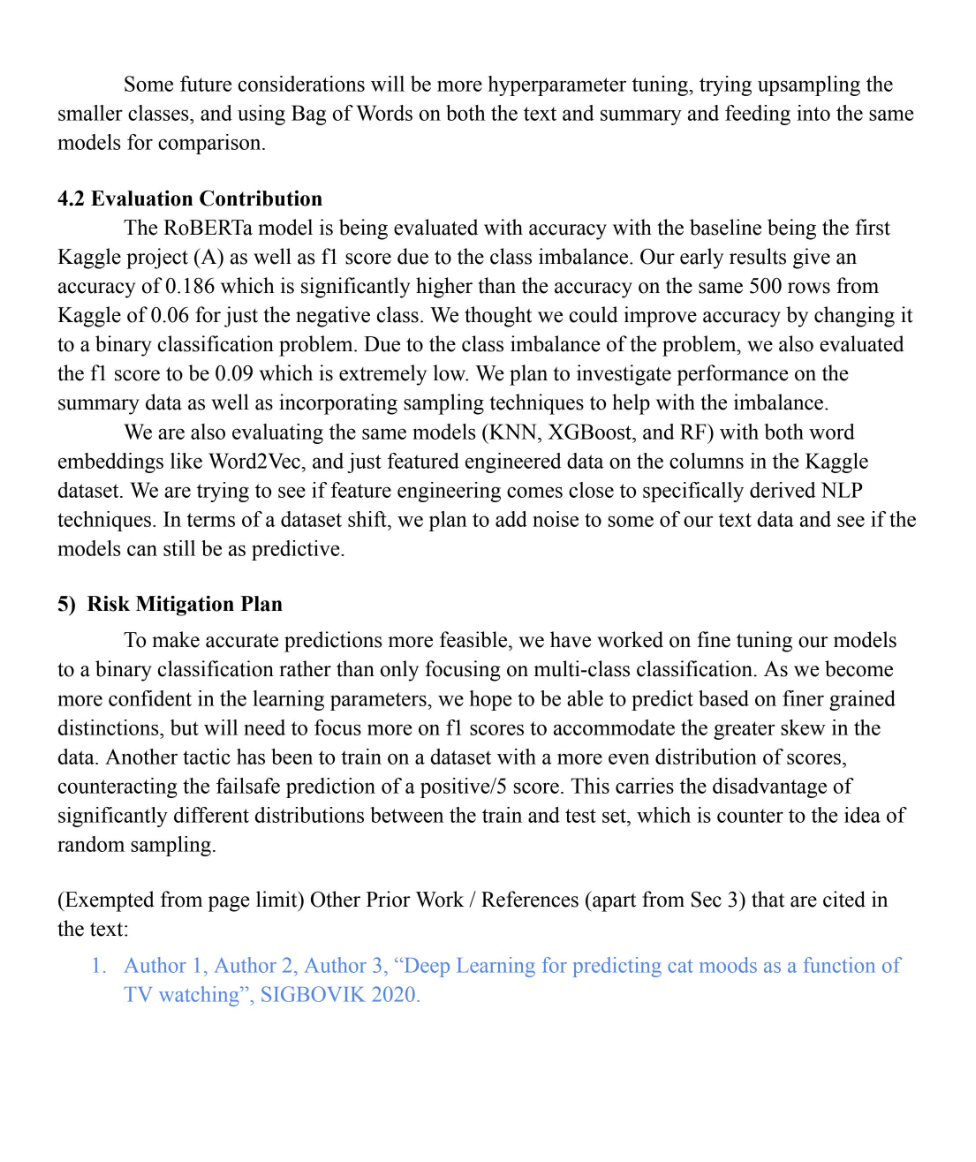
**Karen Li:** Worked on implementation of the Word2Vec approach and weighed various embedding approaches and model types. Tested models on binary and multiclass classification, tuning hyperparameters for best accuracy.

**9) Appendix:** Midway Report and Feedback







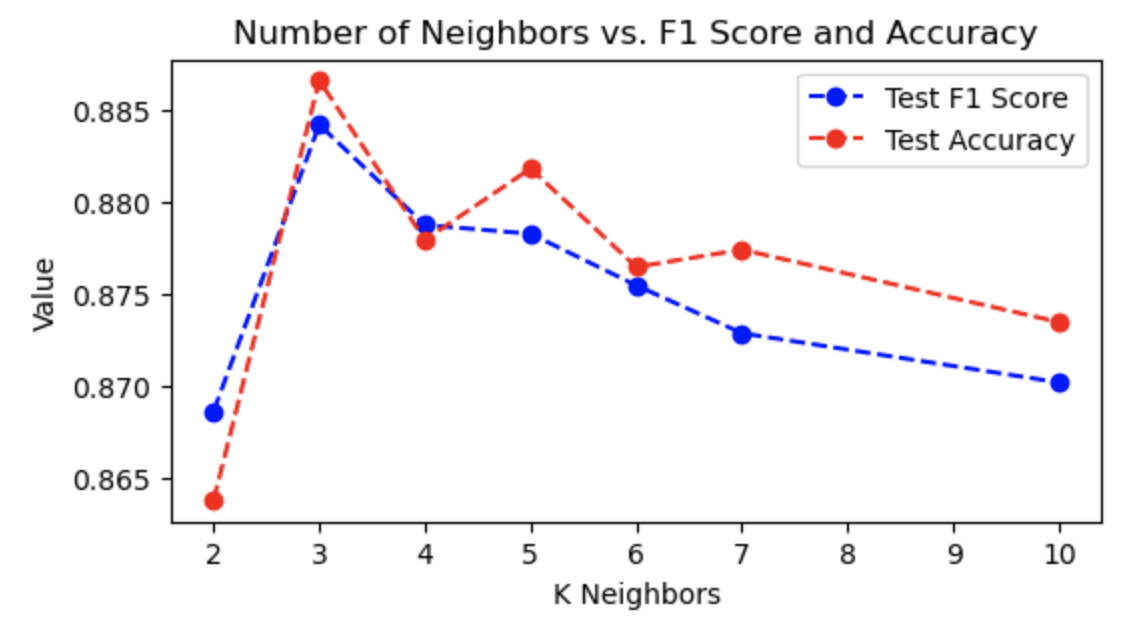


#### 

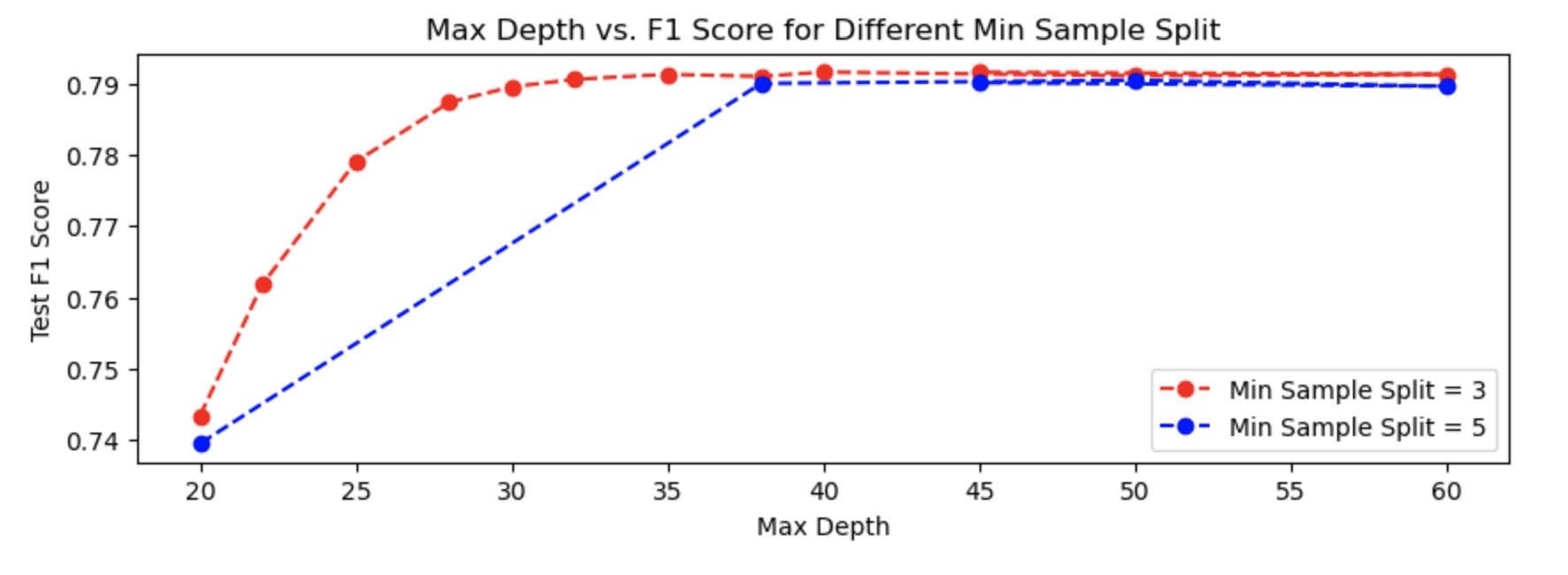
#### **Supplementary Materials (not guaranteed to be considered during evaluation):**

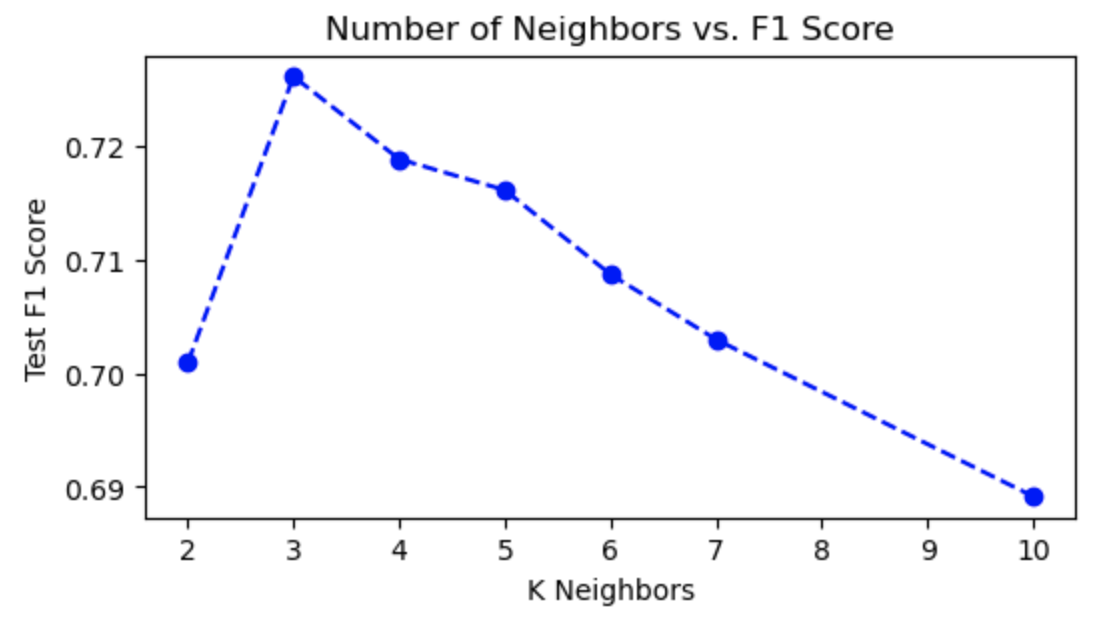
Evaluation of Hyperparameters:

*Binary*

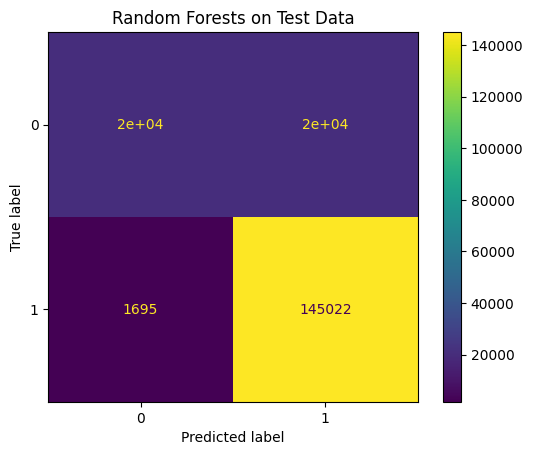
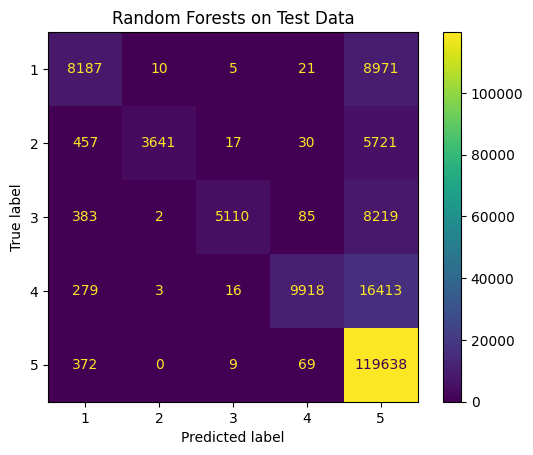


*Multiclass*

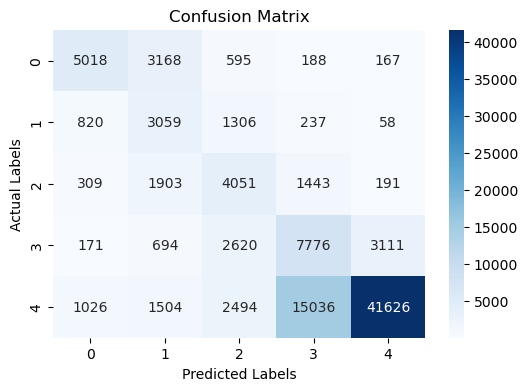
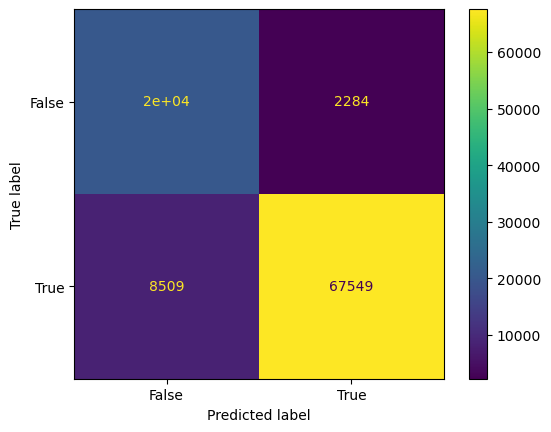




Confusion matrices for the best Word2Vec models:



Confusion matrices for the best BERT model:



Confusion matrices for the best roBERTa model

