Course Evaluation Prediction – Milestone 2 Final Report

Ani Madurkar

Feb. 4th, 2021

Introduction

Objective

Given a dataset of course evaluations written by students across a myriad of courses at UM, I will be attempting to model the features (‘Division’, ‘Type’, and ‘Comment.Text.Processed’) to predict a real-value target variable (‘score’). The model evaluation metric will be mean squared error. I will be conducting data cleaning, exploratory data analysis to understand the data available, and feature engineering and preprocessing to get the training and validation sets prepared to model. I plan to then assess a wide range of modeling techniques, but for this report will focus on two primary types – Linear and Ensemble. I will conduct hyperparameter sensitivity analyses, feature importance through ablation testing, residual normality tests, and failure analysis.

For the Unsupervised Learning portion, I will explore two techniques to topic model evaluations (Latent Dirichlet Allocation and Non-negative Matrix Factorization) and assess ways to empower the supervised learning portion. I will attempt to unpack model performance through viewing the output and metrics like Coherence Score and Jaccard Similarity.

Supervised Learning

Supervised Learning Methods

For the categorical variables, ‘Division’ and ‘Type’, I OneHotEncoded them as opposed to LabelEncoding because there is no hierarchical structure where a given Division or Type should be granted preference by a higher number value. This resulted in a sparse matrix, but since we are working with text data in the ‘Comment.Text.Processed’ field, that was largely unavoidable.

For the text variable, ‘Comment.Text.Processed’, I first started by normalizing and cleaning the unstructured data. From each evaluation value, I stripped and lowercased extracted words, removed stop words, and lemmatized remaining values. This new ‘text_clean’ feature was sent through TfidfVectorizer with ngrams ranging from unigrams to bigrams, max_features set to obtain 15,000 features which was found using learning curves to find an appropriate number, and use_idf was set to False. The idf weighting was initially resulting in top feature importance’s that did not seem to make too much sense. After some analysis, I found that the document corpus may not be large enough and that rare words may not be more interesting for this specific problem. Due to these assumptions, just using term frequency was deemed enough[1].

I found that there are many students who use similar language with common words like ‘class ‘, ‘felt like’, ‘great, ‘question’ instead of complex, unique words so I assumed using min_df to remove words that are too infrequent would serve to be meaningful but trying a range of min_df values with the Linear Regression model showed a different story.

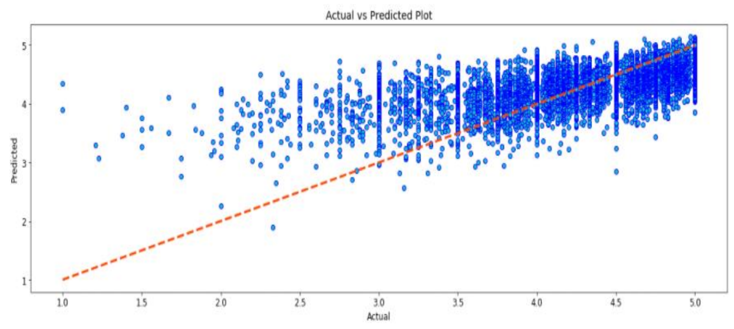
Additionally, I created 4 numerical features after examining the ‘Comment.Text.Processed’ field: character count, average word length, average sentence length, and capital letters normalized by sentence length. These numerical features were created after assessing a random sample of evaluations and noticing that character count and length of words/sentences can be a distinguishing factor to identify lower scores. Additionally, typing words in capitals is commonly known to be considered as ‘angry’ so I believed it would also assist in learning evaluations at lower scores. Creating features that distinguished lower score evaluations was critical for my modeling as there are so few examples. These features were processed via RobustScaler() which removes the median and normalizes by the IQR. This scaler offers more robustness for outliers than StandardScaler(), which can be especially crucial here; for example: there are a decent number of outliers of students who write extremely long or short responses. Then I passed the output through PowerTransformer() to result in a more ‘Gaussian-like’ distribution for the numerical attributes. The PowerTransformer() was tuned to use the Yeo-Johnson method as our target values can potentially include 0s and Box-Cox method is bounded to only positive values[2].

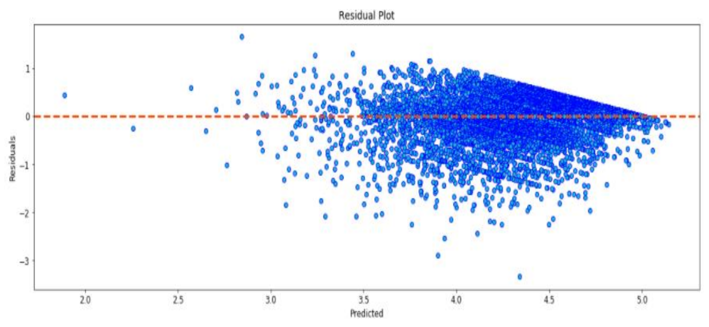
I also noticed through initial exploration of the data that the target variable was left skewed and the distribution was not normally distributed. Due to this, the full pipeline was finally passed through TransformedTargetRegressor to transform the target variable with PowerTransformer() to make its distribution more Gaussian-like with standardization without allowing for data leakage[3].

This final pipeline was used to fit and transform the training set and only transform the validation set.

I explored 5 different Supervised Learning methods for this regression task: Multiple Linear Regression, Support Vector Machine, Decision Tree Regressor, Random Forest Regressor, and eXtreme Gradient Boosted Regressor. I chose the Linear Regression model due to its ability to handle multiple independent variables effectively to predict a real value target variable. I also believed that it may serve as a good baseline to compare the effectiveness of more complex models against. I chose Support Vector Machine due to its ability to handle high dimensional data and go beyond linear decision boundaries. I was aware of the risk of overfitting with adding complexity to these models and I expected to use regularizing to control model complexity. Both linear models do well with sparsity too. Decision Tree were chosen as a relatively simpler model that would be interpretable to get a better idea of what influential features lie in the dataset. Unfortunately, Decision Tree models are prone to overfitting even with careful tuning, so I also decided to try Random Forest models which train each tree independently on a random bootstrapped sample of trees which makes them more prone to the overfitting. Additionally, both the tree- and ensemble-based models do not handle sparsity well, so I used them with a grain of salt. Although I tuned and explored each of the 4 models, The LinearSVR resulted in fairly similar results as the Linear Regression so I will be focusing on the Linear Regression and Random Forest Regressor/ eXtreme Gradient Boosted Regressor models for this report.

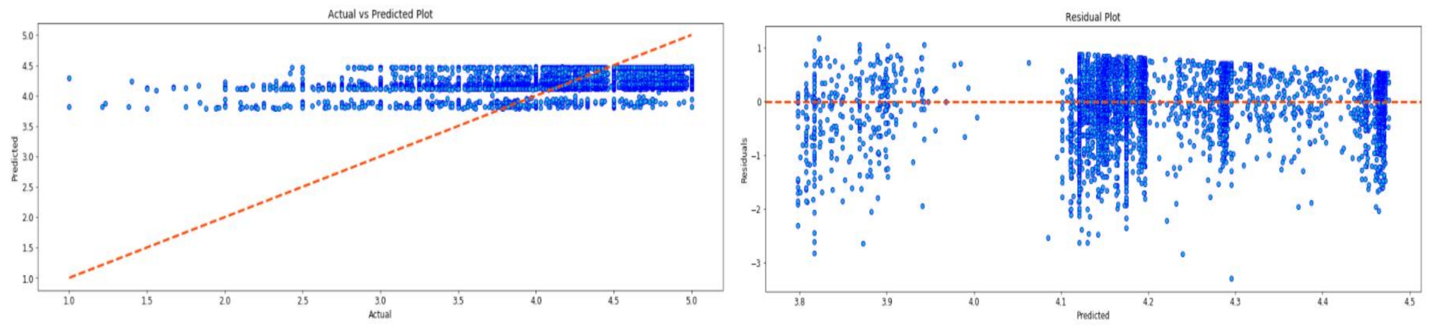
Due to the size of the training set, especially after preprocessing, I quickly ran into my first challenge of quite a large training time. To alleviate this, I trained the linear models through SGDRegressor() to optimize performance and speed. This is due to the models being trained with stochastic gradient descent to find global minima with the loss function. The learning rate schedule is defaulted to 'invscaling' where eta learns at the rate of eta = eta0 / pow(t, power_t). I found that this yielded higher MSE values with cross validation, even after scaling the input and outputs. I found that this was primarily due to having too low of an initial learning rate (eta0) which was causing the model to not find the optimal global minima but get close. Due to this, I increased the eta0 to 0.4 which yielded much stronger MSE results. I utilized ElasticNet to get a blend of L1/L2 regularization, so my model is better able to generalize to new data. I originally tuned the model to be strongly L1 regularized due to the large training set but found that results in higher MSE values and did not perform well on test set. After applying GridSearch Cross Validation, I found it confirmed that a lower L1_ratio (or more L2 regularized) yielded more optimal results when scoring for MSE. I believe this is due to the sparsity of the dataset and having many features with small to medium sized effects. I also found that lower values of alpha performed better on validation set than higher (lower equates to less regularization as this is a constant that multiplies the regularization term). After getting the predictions for the test set and seeing a higher value than the MSE found on validation set predictions, I tuned alpha to a slightly higher value as I noticed that my model was overfitting to the training data. Additionally, I added in early stopping with a holdout validation set of .1 and both stronger regularization techniques resulted in the lowest MSE values on validation and test sets, .2380 and .2816, respectively.



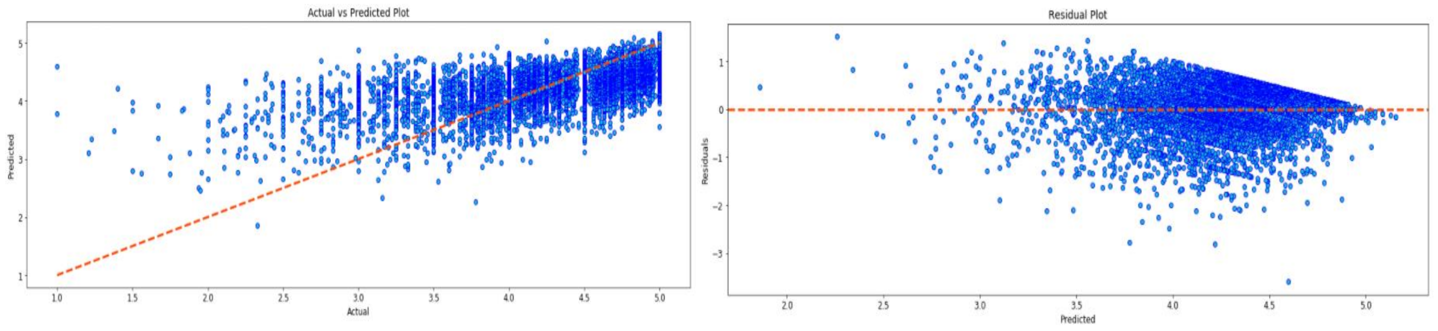


The second model I tried was a RandomForestRegressor(), which is a family of models that performs quite well for a wide variety of tasks, but performed quite poorly here. When it came to preprocessing, this ensemble type of models does not require too careful scaling/normalization so I just OneHotEncoded the categorical features and utilized TfidfVectorizer the same as was done for the linear model but held out on scaling and normalizing the numerical features and target variable. This low amount of preprocessing was primarily done because RandomForest models are known to be 'pretty good out of the box', but we still needed to process the text/categorical features.

In terms of hyperparameters, I tuned the n_estimators to 200 since we are working with a decently large dataset (~15k features, ~25k training rows) and a higher number of trees in each ensemble would reduce overfitting. I tuned max_depth to 2 to add a little more depth/complexity to each tree, hoping this would improve model performance on the training/validation set. The n_estimators and max_depth hyperparameters were found through GridSearchCV, which simply showed the hyperparameters to have marginal impact on our training set. Finally, I bootstrapped each tree sampling to improve generalizability. The model took almost the same time to run (21 min 49s +/- 32.6s) and could not beat the baseline MSE value, resulting with .3337. I found that it was difficult to iterate on different GridSearch hyperparameters of this model due to how long each model took to run, and the time complexity scaled up with larger ensembles/deeper trees.



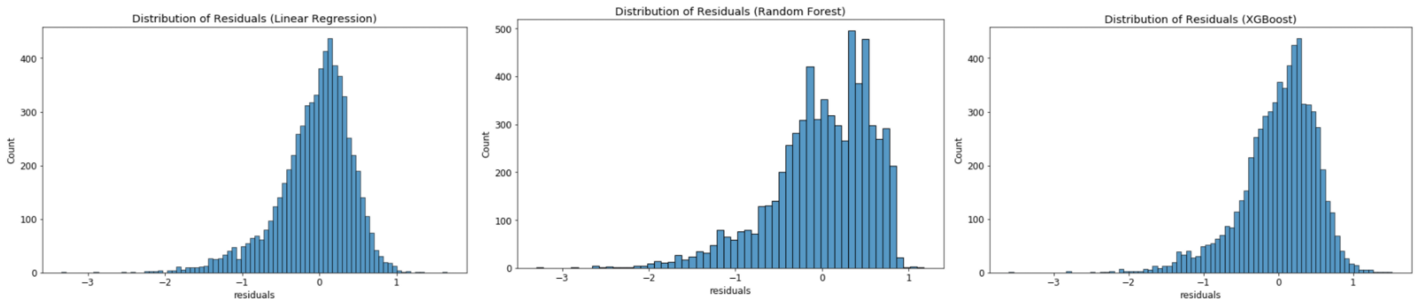
Although Random Forest models were not able to deal with the sparsity as well, all hope is not lost for ensemble models for our application. I also tried a popular variant of Gradient Boosted tree models, XGBoost, and tuned it to perform as a Random Forest model with the booster default parameter set to 'gbtree' and setting subsample to 0.5 to enable random selection of training cases, with bootstrapping. This is another model that is expected to perform 'out of the box' so with minimal tuning I got a MSE of .2515 and .2947 on validation and testing sets respectively:



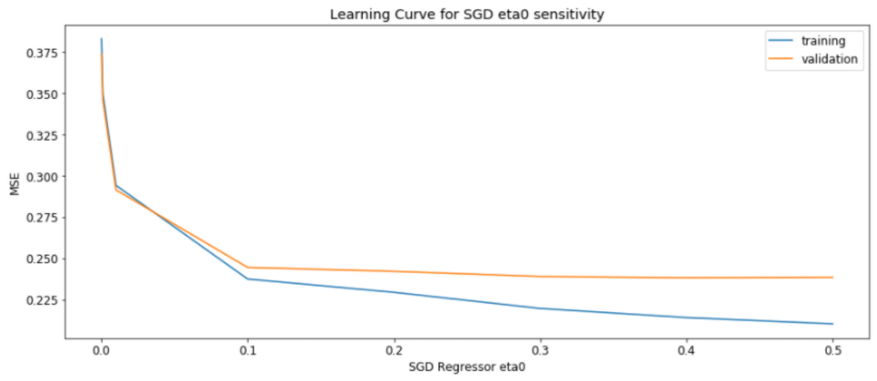
Common challenges I ran into were finding best ways to remove outliers, properly balance the training dataset to ensure models learn lower review scores and dealing with having to wait for models fitting on large datasets (or for GridSearchCV). For outliers, I found there were about 30 reviews that had scores of 0 and the actual evaluations ranged from negative reviews to reviews where 0 was clearly hit by mistake. Due to the low count, I removed the outliers that were a mistake and assigned the remaining negative 0 reviews with a score of 1 so the range is an appropriate 5-point scale. I also noticed quite a skewed target variable and I used robust scaling and power transformation to create Gaussian distributions of 'score' in training and testing sets, this helped models learn a little more nuance in lower scored reviews. For the wait times, I sometimes used a shuffled sample of the training/validation data to run GridSearchCV so it did not run for an abnormally long time and could give me a reasonable idea of successful hyperparameter values.

Evaluation

The first evaluation approach I took for each model, I analyzed the residuals by conducting normality testing with the Shapiro-Wilk test. For Linear Regression, the residuals resulted in a test statistic of 0.94758 and a p-value of 2.76e-43 which led to rejecting null hypothesis (because it is less than alpha of 0.05) and the sample of 6552 rows does not look Gaussian. Observing the distribution of the residuals, we see a pattern for the model doing worse at lower scores where the model predicts high values (leading to negative residuals). For Random Forest, the residuals resulted in a test statistic of .93959 and a p-value of 1.40e-45 which is also less than alpha 0.05 so a sample of 6552 rows does not look Gaussian and we can reject the null. Random Forest residuals also show more variation and a potential bimodal distribution. For XGBoost, the residuals resulted in a test statistic of 0.95308 and a p-value of 1.52e-41 which is also lower than alpha of 0.05 so reject the null hypothesis for this model as well and the distribution does not look Gaussian. Although the residuals are similar with XGBoost and Linear Regression, I still find a common pattern of left-skewed distributions with residuals performing poorly on low score evaluations. One important caveat to note is that Shapiro-Wilk test may result in inaccurate p-values for sample sizes > 5000, so I also ran the Anderson-Darling test and it yielded similar results.

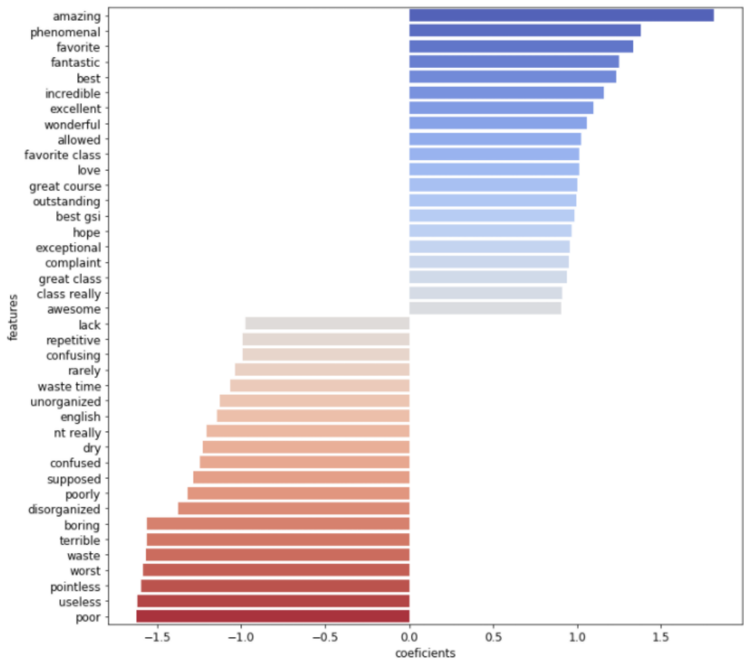


The first hyperparameter I found to make a significant difference on learning, especially on the best model I was able to design, was the eta0 or initial learning rate. I found a good value for this using learning curves as I looped over several eta0 values.

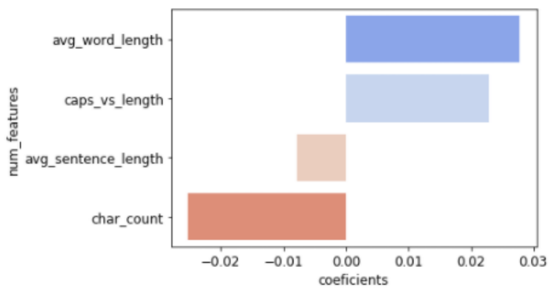


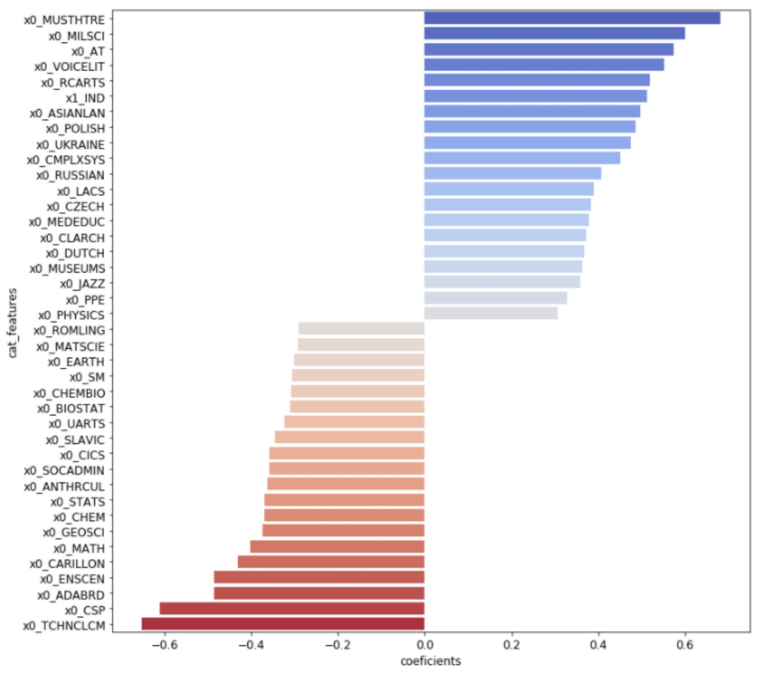
The validation loss started to slowly increase after eta0=0.4 so I used that value in the model. For RandomForestRegressor(), I found it to be insensitive to hyperparameter tunings of n_estimators and max_depth. The MSE consistently resulted in the .32-.35 range. The hypothesis behind this was due to the nature of almost all the features being text or categorical and having such a vast, sparse matrix, which is shown as we get into the feature importance of the models.

Assessing the features that have an impact for the Linear Regression model to logically make sense and it lines up with our initial analysis of students using lots of common words:



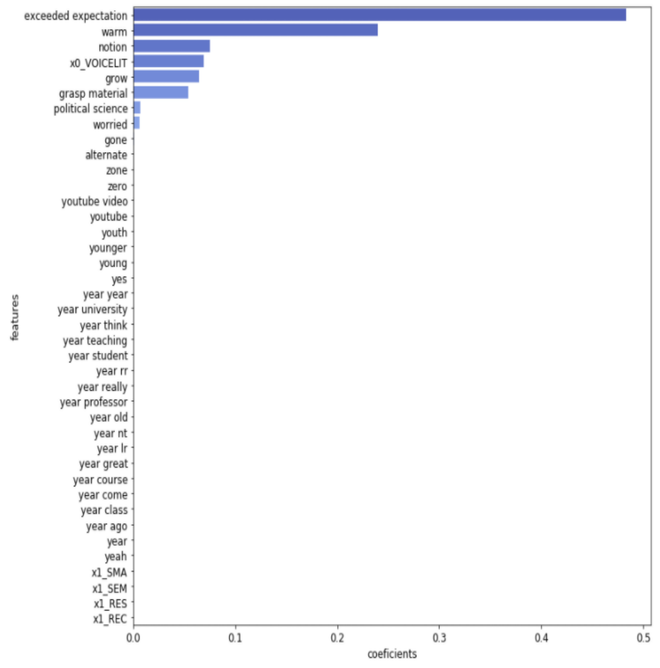
The positive words seem as expected, and it is interesting to see negative features get highlighted such as ‘unorganized’/’disorganized’ or ‘pointless’/’useless’/’waste time’. It highlights the importance of having structure and application in courses. Having a ‘best gsi’ also does significantly help the review. Although the text features dominate, I wanted to see how the other feature representations performed. For numerical and categorical features, we get:

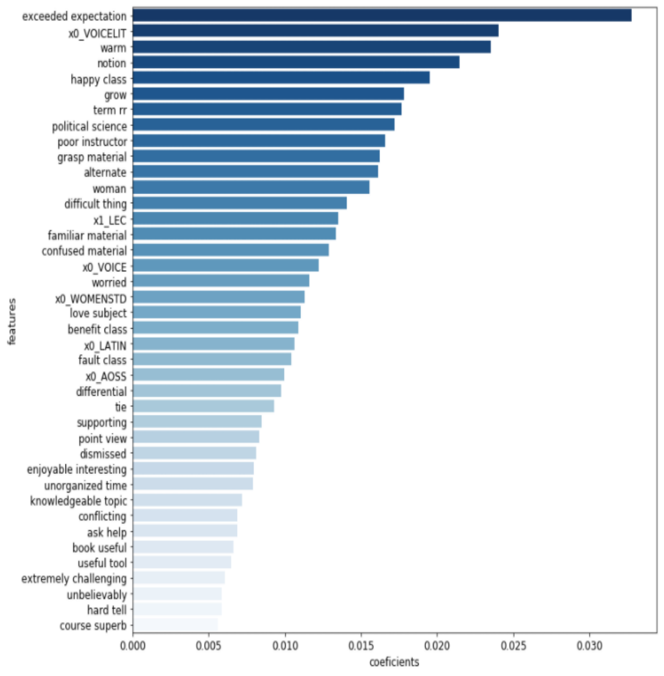




The coefficients are quite small so the numerical features that were engineered do not impact our model with much significance, but it is interesting to find positive reviews to commonly include longer words on average, but negative reviews tend to be longer overall. The categorical features tell a much more insightful story as we see Arts, Literature, and Social Studies Disciplines (Music Theatre, Voice Literature, Polish, Jazz) primarily reside in positive features and STEM Disciplines (Math, Chemistry, Statistics) in negative features. No definitive conclusions can be drawn as there can be a variety of compounding factors here (course difficulty, GSI, engaging professor, time of day courses are administered, etc.), but we do see the model being able to discern patterns of more positive reviews with more artistic/social disciplines and more negative reviews with more scientific/technical disciplines.

Analyzing the features of the Random Forest and XGBoost models confirms suspicions of the Random Forest model's inability to deal with sparsity well, and XGBoost being able to handle sparsity quite effectively:

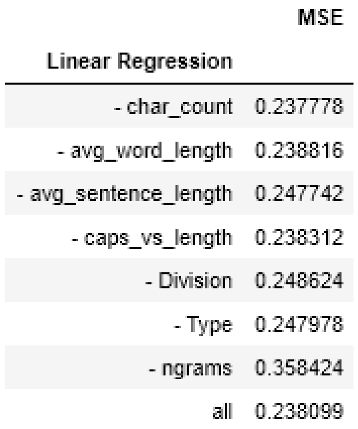




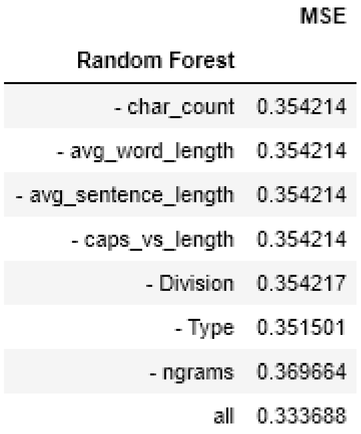
The Random Forest model does not appear to be finding many significant features at all, apart from a select few. As we think about how Random Forest models learn, this does make sense when trying to learn a sparse matrix – it randomly selects a bootstrap subsample of the feature set to create a variety decision trees and then aggregates them together at the end. But if we have a large training set with a lot of 0s, there is a good chance it would learn that most of the features have 'no impact' and thus conclude that only a select few are important. Additionally, as we saw in the Actual vs Predicted plot, the model predicts almost exclusive within the range of 3.5 to 4.5. When looking at the distribution of the score originally, that is not too abnormal for such a skewed distribution.

Random Forest models are known for being quite robust to outliers while handling class imbalance well, so we may be seeing the model’s inability to properly find enough examples at low score values when bootstrapping trees. Although Random Forest model performed okay at best, the feature importance for XGBoost look vastly different. We do see similar top features of ‘exceeded expectations’ and ‘warm’, but XGBoost is able to perform on sparse data immensely better than vanilla Random Forest models and is able to capture quite questionable features. We see ‘familiar material’ and ‘confused material’ right next to each other and ‘poor instructor’ ranking quite high which seems odd especially with the MSE with minimal tuning.

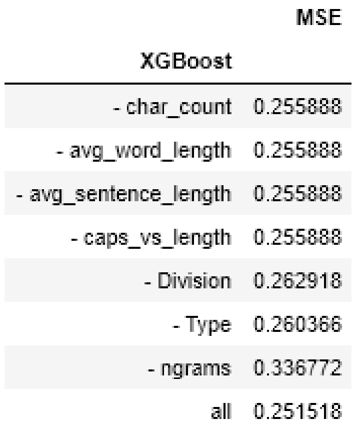
When conducting ablation tests, I noticed a similar pattern of what we see with feature importance – that ngrams have the most impact on the models. For Linear model, Random Forest, and XGBoost respectively, ablation testing resulted in:



Linear Regression	MSE
- char_count	0.237778
- avg_word_length	0.238816
- avg_sentence_length	0.247742
- caps_vs_length	0.238312
- Division	0.248624
- Type	0.247978
- ngrams	0.358424
all	0.238099



Random Forest	MSE
- char_count	0.354214
- avg_word_length	0.354214
- avg_sentence_length	0.354214
- caps_vs_length	0.354214
- Division	0.354217
- Type	0.351501
- ngrams	0.369664
all	0.333688



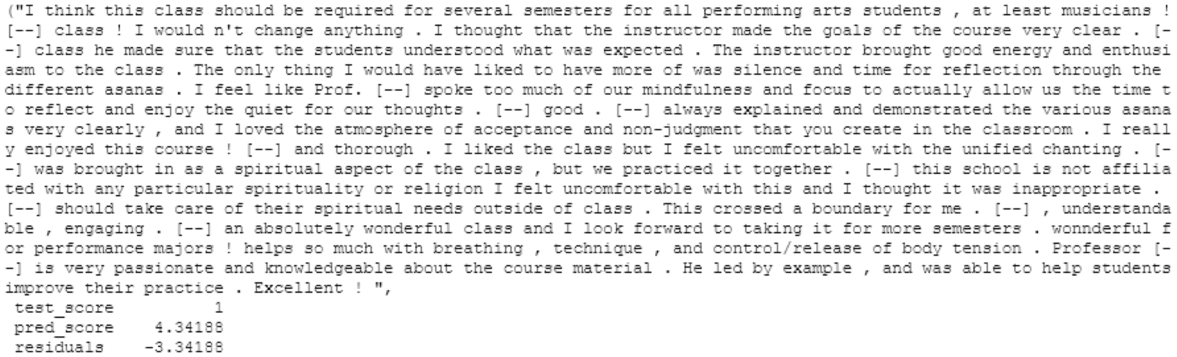
XGBoost	MSE
- char_count	0.255888
- avg_word_length	0.255888
- avg_sentence_length	0.255888
- caps_vs_length	0.255888
- Division	0.262918
- Type	0.260366
- ngrams	0.336772
all	0.251518

What is interesting for the ensemble models is that the numerical features all have equal/insignificance to the model whereas the linear regression model can detect a more complex pattern, albeit marginal, amongst them. Additionally, the categorical features Division and Type also do not have that large of an effect and it appears the only features of significance are in the ngrams. Analyzing the ngrams further, I did find that bigrams and unigrams specifically had the most impact and trigrams added a marginal change and this was why only (1,2) was used for TfidfVectorizer’s ngram_range. A final point of interest was how XGBoost performed without ngrams – out of the box is nearly as good as Random Forest with all features.

When thinking about tradeoffs, I noticed that as I tried more complex models that combine features to find high-dimensional non-linear boundaries, the performance on validation/test set dropped (higher MSE) which resulted in a need for more careful tuning to regularize and prevent overfitting. After an exhausting amount of time spent on tuning complex models and thinking about adding more features to extract .1-.3 more MSE values, I found the Linear Regression model to be more than sufficient for the problem at hand given the data provided. It is a relatively ‘simple’ model and although it makes a decent number of assumptions regarding the residuals, I discerned that the model was performing quite well for this task. To assess the uncertainty of MSE values for the Linear Regression model, I bootstrapped 50% of the training set 150 times the training set to construct prediction intervals, and of all intervals, 95% of them consisted of MSE values between .223 and .265. This confidence band can be narrowed with higher bootstrapped iterations.

Failure Analysis

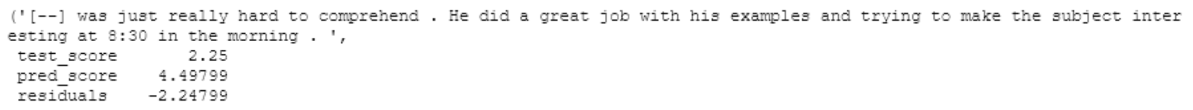
The three common types of failures I found were almost all related to reviews with low scores. The first is evaluations that are largely positive but have one mistake that the student is not able to forgive. An example of this is shown here:



```
("I think this class should be required for several semesters for all performing arts students , at least musicians !
[--] class ! I would n't change anything . I thought that the instructor made the goals of the course very clear . [-
-] class he made sure that the students understood what was expected . The instructor brought good energy and enthusi
asm to the class . The only thing I would have liked to have more of was silence and time for reflection through the
different asanas . I feel like Prof. [--] spoke too much of our mindfulness and focus to actually allow us the time t
o reflect and enjoy the quiet for our thoughts . [--] good . [--] always explained and demonstrated the various asana
s very clearly , and I loved the atmosphere of acceptance and non-judgment that you create in the classroom . I reall
y enjoyed this course ! [--] and thorough . I liked the class but I felt uncomfortable with the unified chanting . [-
-] was brought in as a spiritual aspect of the class , but we practiced it together . [--] this school is not affilia
ted with any particular spirituality or religion I felt uncomfortable with this and I thought it was inappropriate .
[--] should take care of their spiritual needs outside of class . This crossed a boundary for me . [--] , understanda
ble , engaging . [--] an absolutely wonderful class and I look forward to taking it for more semesters . wonnderful f
or performance majors ! helps so much with breathing , technique , and control/release of body tension . Professor [-
-] is very passionate and knowledgeable about the course material . He led by example , and was able to help students
improve their practice . Excellent ! ",
 test_score         1
 pred_score   4.34188
 residuals   -3.34188
```

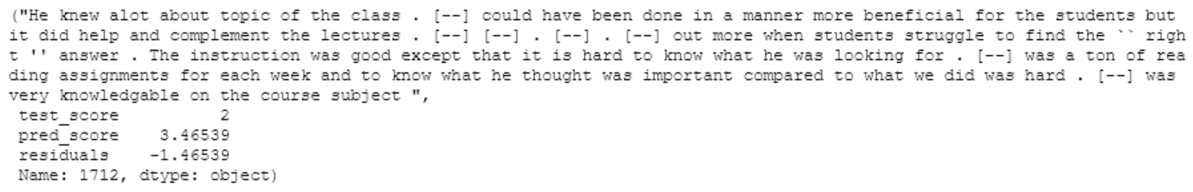
These failures were found commonly around 0-1.5 range and since they have a lot of positive words, the model is unable to properly attribute a score to these types of reviews even though there is a line in there that identifies something that crossed a boundary for a student. Reviews like these are tough to discern because a fatal flaw is commonly drowned out around a lot of positive words, and our TfidfVectorizer is just relying on term frequency so with a lot of positive words the model finds this review to be highly rated.

Another failure type I found were reviews with low word/character count. Since the primary features in our models are coming from the ‘Comment.Text.Processed’ field, if there are not many words given then the models are not able to distinguish what makes those evaluations unique. An example of this can be seen here:



```
('[--] was just really hard to comprehend . He did a great job with his examples and trying to make the subject inter
esting at 8:30 in the morning . ',
 test_score       2.25
 pred_score    4.49799
 residuals    -2.24799
```

A final failure type noticed was almost the inverse of the previous one: very long evaluations that go back and forth between positive/negative sentiment. These are simply confusing evaluations that I believe even humans may have a tough time discerning the appropriate score for:



```
("He knew alot about topic of the class . [--] could have been done in a manner more beneficial for the students but
it did help and complement the lectures . [--] [--] . [--] . [--] out more when students struggle to find the `` righ
t '' answer . The instruction was good except that it is hard to know what he was looking for . [--] was a ton of rea
ding assignments for each week and to know what he thought was important compared to what we did was hard . [--] was
very knowledgable on the course subject ",
 test_score          2
 pred_score    3.46539
 residuals    -1.46539
 Name: 1712, dtype: object)
```

Unsupervised Learning

Motivation

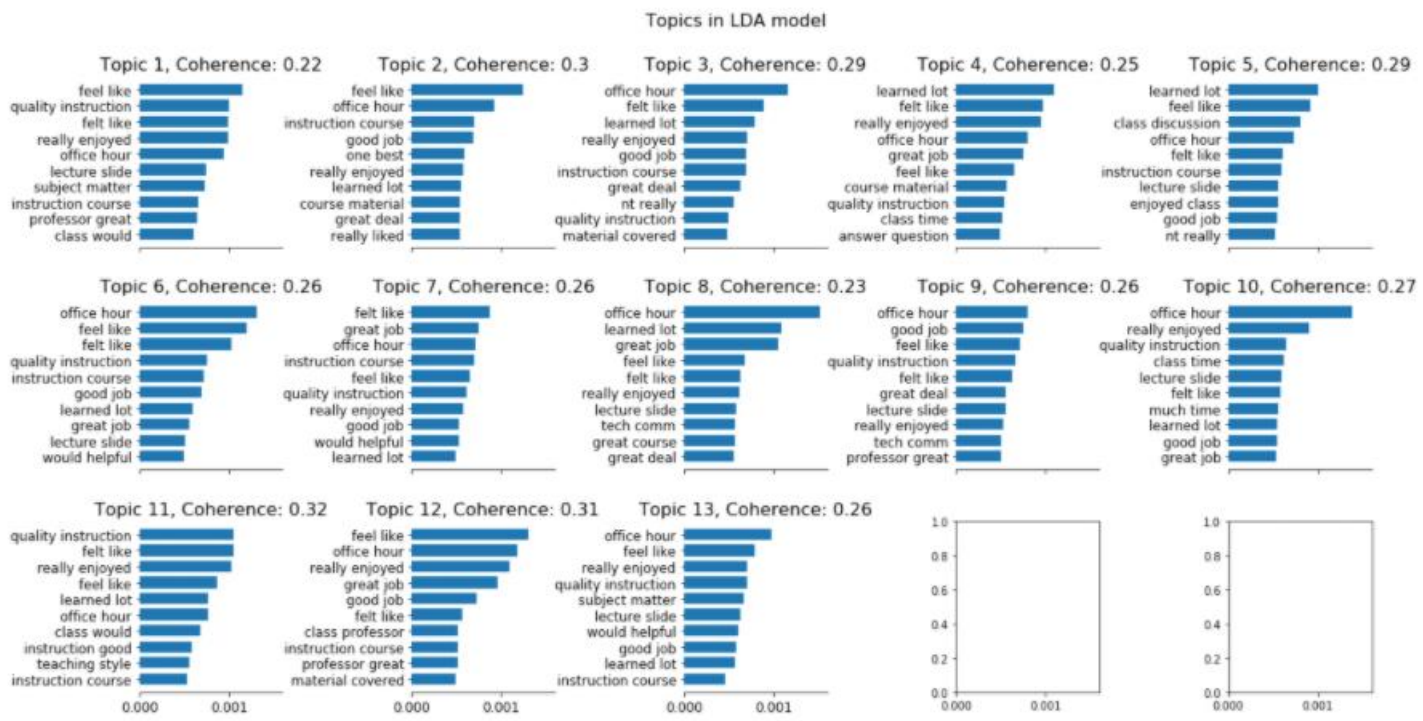
A common challenge that was found in the supervised learning modeling was there was not enough features that were distinguishing enough for the lower score features. I believe that unsupervised techniques like topic modeling could be used before supervised modeling to improve the quality of the predictions. Additionally, since we only have a small number of features to begin with, TfidfVectorizer has a very large impact on our models especially if we use idf weighting or not. Due to these findings, I wanted to assess which topic modeling technique, Latent Dirichlet Allocation or Non-negative Matrix Factorization, would yield topics with higher coherence, with the hopes that these topics may identify crucial features appropriately. Each of these models take in slightly different inputs as LDA takes in document-term frequency counts (not idf weighting) and NMF takes in tf.idf values so I deemed this question to be highly relevant and valuable for our problem.

Data Source

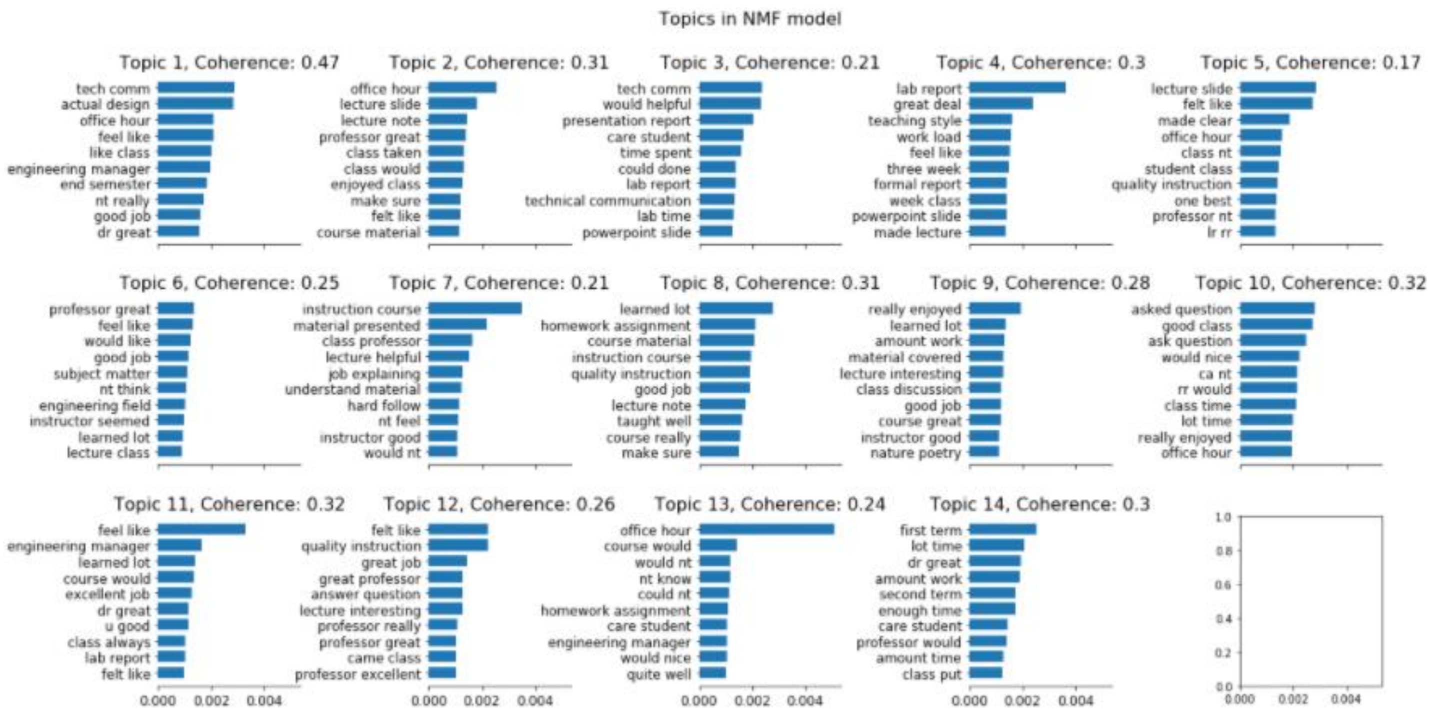
The dataset is course evaluations with instructor demographic information (job title, gender, tenure-track) that was provided in the Kaggle project. The dataset is a subset with about 5000-6000 rows.

Unsupervised Learning Methods

I first normalized the ‘Comment.Text.Processed’ field by splitting and extracting the words and stripping for any trailing spaces. Then I removed stop words and used NLTK to lemmatize the text. Then I created a list of lists for each text containing term-frequencies of each lemmatized text. Passing this as a Dictionary through Gensim made it possible to use their doc2bow() function for each text in the corpus. This creates a ‘bag of words’ representation of each document, which creates a term-frequency representation of text features. When initially passing this into Gensim’s LdaModel() with 5 topics to just see output out of the box, I see lots of common and similar words in multiple topics. The primary hyperparameter is num_topics, or the number of topics to choose for each of these models. Ideally, the topics should yield low similarity to each other and high coherence. To understand best tuning options, I looped through 15 values to assess how the Jaccard Similarity and Coherence Score tracks at each number for each model to find the optimal number of topics. The result of this analysis yielded these topic models, for LDA (13 topics):

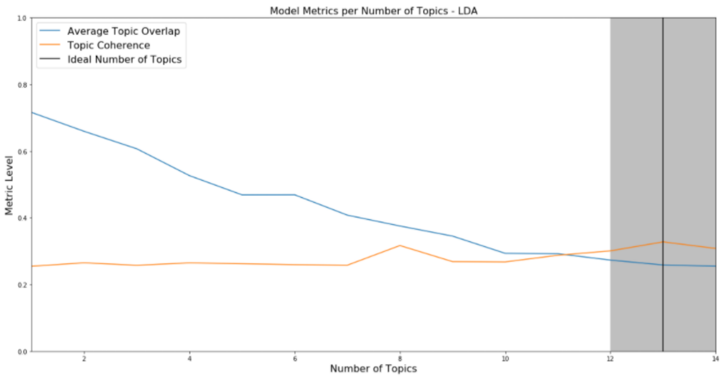


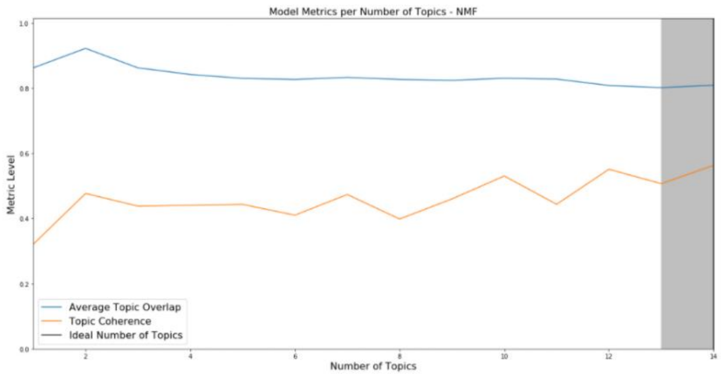
and NMF (14 topics):



Unsupervised Evaluation

Assessing the topic models of each strategy was quite insightful when viewing the Jaccard Similarity and Coherence Scores to find the best number of topics. The metrics plot also highlights a significant tradeoff for utilizing these models on this data. For LDA and NMF, respectively:





These plot show that both models have 'optimal' number of topics around the same values, and neither are still that good in terms of low average Jaccard Similarity and high Coherence. We do see that LDA has relatively low Coherence, but it does slowly rise with more topics, and it is Jaccard Similarity decreases quite significantly. For NMF models, we see a higher Coherence on average that trends up a little more, but also quite a high and immovable Jaccard Similarity. This does hold when we think about how the models work and their inputs. LDA uses document term frequencies to estimate the model parameters that 'best explain' the observed words in the corpus, given the model. So, having a large corpus of similar language written by students with low complexity/uniqueness to the words would lead to a steep drop in similarity as the topics increase since majority of the similar words can be spread out to best explain the corpus. Since majority of these words are still not distinguishing enough to be unique topic models, a low coherence score persists.

In contrast, NMF attempts to factor data matrix X into two k-dimensional factors W and H that, when multiplied, approximate X. It utilizes a document term matrix of tf-idf values, or values that prioritize words used in a large frequency and less often. As we saw in the supervised portion, our corpus of student evaluations has a large frequency of words that also get used often. Average topic similarity barely fluctuating makes sense here as well as there simply are not enough differentiating tokens in large frequencies in the corpus but of the ones the model is able to find, we see a relatively higher Coherence due to those tokens.

Discussion

As I conducted Part A, I learned most about: the value of simple solutions, the significant effort to design and implement meaningful features, the computational challenges to be able to iterate on ideas quickly, and how to consistently balance tradeoffs. Even with trying out a variety of different models, the Linear Regression (which was the first one tried) ended up with the best results at the end. As model complexity grew, I spent more time in tuning the models to find that they still would not beat the Linear Regression one. Occam's Razor really stuck out here of deferring to simpler solutions/explanations unless more complexity is a necessity. Secondly, I also learned that the features had a far more meaningful impact in the results and when designing/creating features that could differentiate low scores more specifically, I learned how much effort that would require. It was a venture that would require more time with the data and understanding where the data comes from to be able to construct non-generic features. This reminded me of the significance of domain value knowledge in data science projects. Additionally, as I was running a variety of models through via grid search and cross validation, it became clear how long some took and how limiting that was. I had to rethink the hyperparameter tuning to not just include as wide of a range for any features but be more specific and intentional in my choices. This also proved to be valuable when attempting to explain the models at the end. Finally, I learned that assessing tradeoffs for modeling is not only relevant when assessing model performance with metrics, but it is something that should be done throughout the whole process. Knowing when exhaustively tuning a model or preprocessing technique or creating more features is an art as much as a science. I had to intentionally think of when to stop tuning a model for marginal performance improvements and only focus on what mattered (feature vectorizer tuning or feature engineering). I was primarily surprised at the effectiveness of Linear Regression models and the impact that TfidfVectorizer tuning had on model performance. In hindsight, this does seem to make sense as there are a limited number of features, but it was interesting to work through a problem that required you to actually think about the data and not just throw all the 'recommended strategies' at it. For ex. When I was applying idf weighting, the features at the end seemed a little nonsensical. After only leveraging term frequencies, model performance increased and was more explainable.

With more time, I would extend the solution in these ways: a logistic regression model to first differentiate the data into binary 'positive' and 'negative' sentiment classifications before the regression modeling, utilizing more finely tuned topic modeling (such as turbo topics) to create more meaningful features that differentiate the data before the regression modeling, and tuning XGBoost to optimize performance. I believe logistic regression would be a suitable model to find high and low scores with a balanced set of classes (could use SMOTE or random sampling to keep high and low score evaluation classes balanced) and the topic modeling could illuminate latent structural patterns that emerge in lower scores which are unable to discern easily. Additionally, I believe that there are more performance improvements that could be achieved through tuning XGBoost considering how well it performed out of the box.

Implementing Part A as a solution could raise some ethical concerns. The first is that students from varied backgrounds convey language in different ways. What is sarcastic in one culture may be less hurtful elsewhere. Trying to only use language and words expressed makes it hard to differentiate whether this may be a cause. Additionally, the question is still outstanding of who this solution is really to serve and how? We should check whether there is a pattern in the learned models to detect for age, gender, and race. There could be the case that some groups have a specific issue with the class/professor but are drowned out due to the low frequency of representation. Finally, there is the issue of privacy and informed consent. As students are entering these values, they are under the impression that it is completely anonymous. It could be likely that a student may not have entered the same response in if they believed it would somehow get back to the professor or faculty. Some ways to address these issues would be to run homogeneity tests to check if there is balance amongst race, age, and gender variables. Showing significant patterns in these features could indicate that our model is learning some implicit patterns we do not intend it to. Additionally, including linguistic professionals and social scientists in the conversation would be helpful to identify if there are patterns in the language that may be improperly interpreted across different cultures. Finally, I would add an 'Opt In' option on the course evaluation surveys that clearly and concisely inform the student how the data would be used and provide them the option to allow their entered data to be used. Students may also choose after some time to change their minds, so each student who opted in should receive an email with clear instructions on how to opt out if they wish to which would purge their stored data.

As I conducted Part B, I learned most about: the difficulty in topic modeling with a large frequency of similar words, the impact in model learning based on using term frequency or tf-idf weighting, and the tradeoffs with using LDA and NMF. Latent Dirichlet Allocation is a probabilistic (generative) model is finding the distribution of words that best explain the original corpus, but if the original corpus has quite a skewed distribution then you will naturally result in what we saw – a lot of topic models showing similar words with the highest term frequency. Non-Negative Matrix Factorization attempts to decompose a matrix into a small number of matrix factors and the columns of one of these matrix factors can be considered topics. Since this takes tf-idf weighting as an input, I found that some of the values in each topic were more varied than just top term frequencies but still had too similar words amongst each topic model. It did surprise me that the models did not perform as well, and the amount of effort needed to carefully tune them properly.

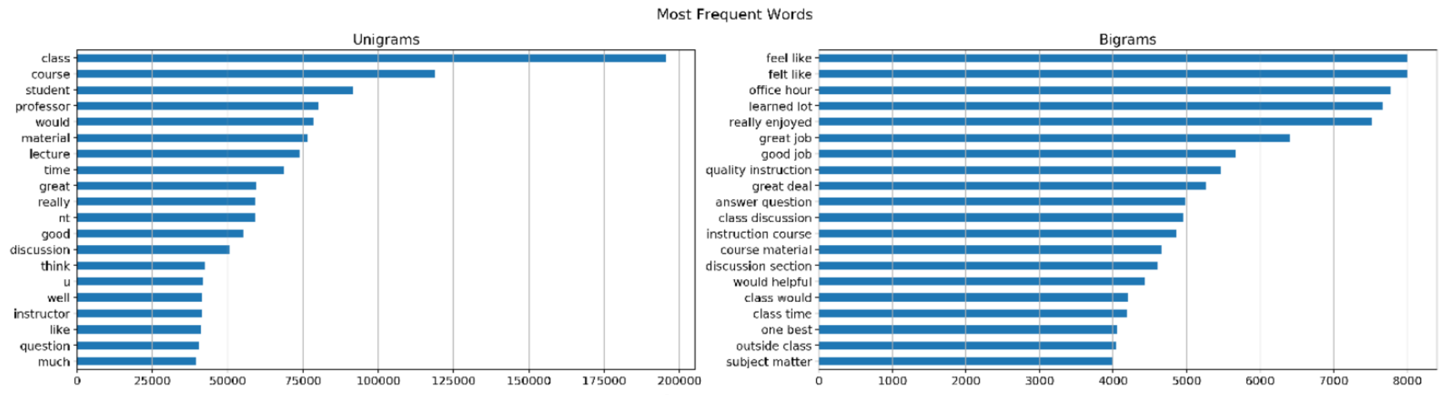
With more time, I would extend the solution in these ways: a robust study on hyperparameter sensitivity analysis to see which hyperparameters affect each model strongly, utilizing turbo topics to get more intelligible topics, and diving deeper into the demographics to find more interesting patterns amongst different professors.

Implementing Part B as a solution could also raise significant ethical concerns. Part B contains the same privacy & informed consent issue, so I believe a similar strategy to address it would be appropriate – adding an ‘Opt In’ option where students are clearly aware of what is being used, how, why, and how to change their mind if needed. Part B would require careful monitoring especially on the language ethical concerns as it is looking to find latent structure that cannot be easily visible. This would mean ensuring that underlying data distributions match the training data and are not misrepresenting the information provided is a critical factor. I would address this by adding a human in the loop to routinely monitor the topic models and a random subsample of the training data to see if the topic models are reasonably accurate along with the metrics such as Coherence Score and Jaccard Similarity.

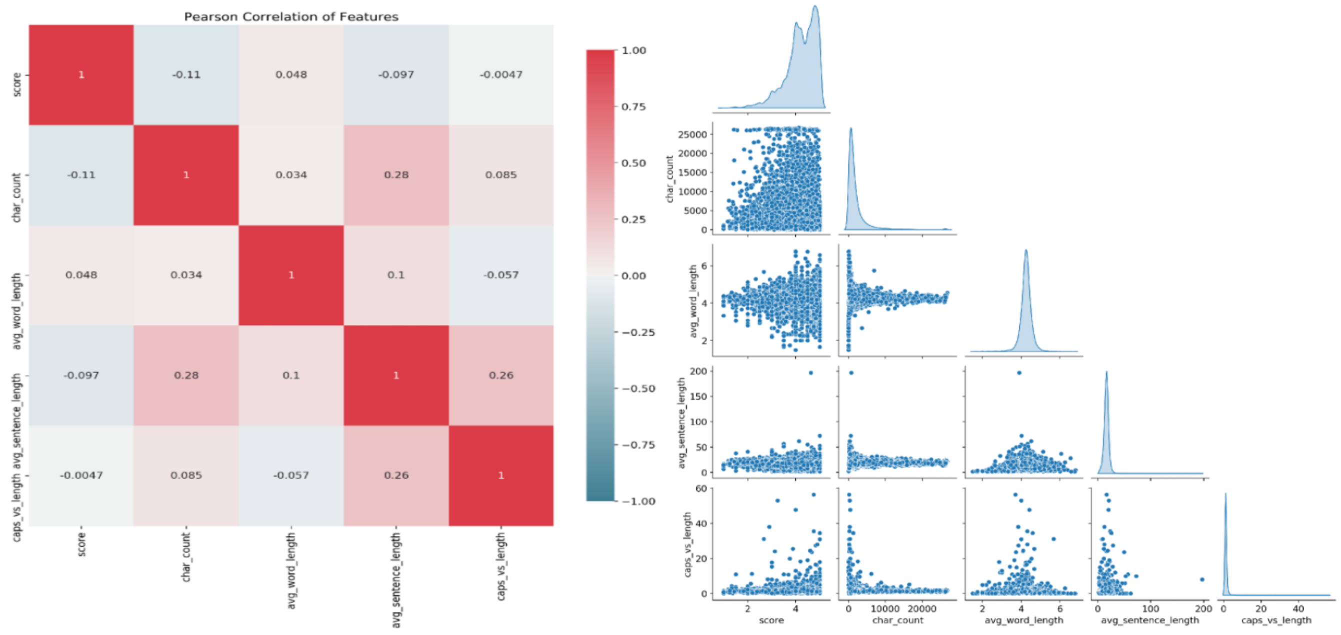
Statement of Work

All data cleaning, manipulation, visualization, preprocessing/feature engineering, Supervised and Unsupervised modeling, analysis, and final report write up was done by Ani Madurkar.

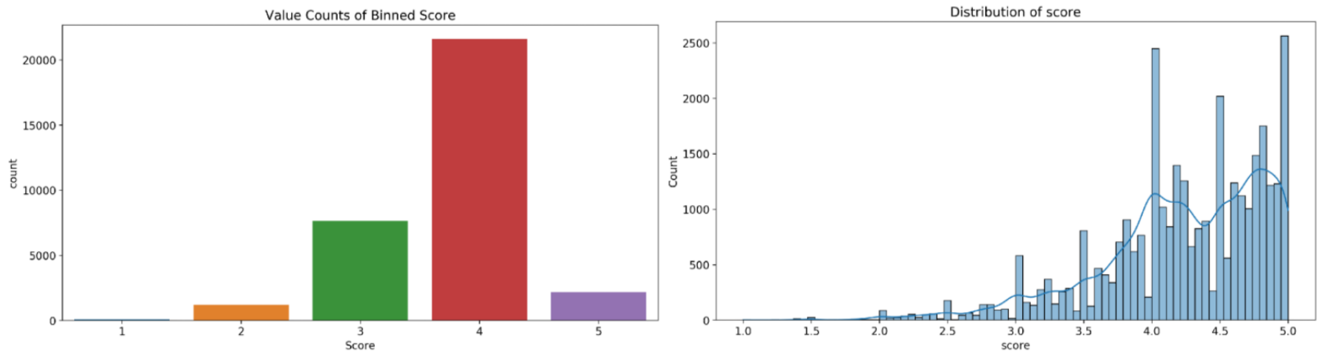
Appendix


Most Frequent Words
Unigrams
class
course
student
professor
would
material
lecture
time
great
really
nt
good
discussion
think
u
well
instructor
like
question
much
0
25000
50000
75000
100000
125000
150000
175000
200000
Bigrams
feel like
felt like
office hour
learned lot
really enjoyed
great job
good job
quality instruction
great deal
answer question
class discussion
instruction course
course material
discussion section
would helpful
class would
class time
one best
outside class
subject matter
0
1000
2000
3000
4000
5000
6000
7000
8000

1.


Pearson Correlation of Features
score
char_count
avg_word_length
avg_sentence_length
caps_vs_length
1
-0.11
0.048
-0.097
-0.0047
-0.11
1
0.034
0.28
0.085
0.048
0.034
1
0.1
-0.057
-0.097
0.28
0.1
1
0.26
-0.0047
0.085
-0.057
0.26
1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
score
char_count
avg_word_length
avg_sentence_length
caps_vs_length

2.


Value Counts of Binned Score
count
Score
1
2
3
4
5
0
5000
10000
15000
20000
Distribution of score
Count
score
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
0
500
1000
1500
2000
2500

3.