STAT 432 Final Project Report

Dev Patel

1.Project Description and Summary

This analysis will use process features from keystroke log data to predict overall writing quality. These efforts may identify relationships between learners' writing behaviors and writing performance. Ultimately, this may help direct learners' attention to their text production process and boost their autonomy, metacognitive awareness, and self-regulation in writing.

Project Goal: The overall goal of the project is to train and build machine learning models to accurately predict overall writing quality and identify the relevant features and what predictors are more important in overall writing quality from interpreting the models.

Project Approach: This project encompasses the following steps.

1. **Data Preprocessing:** This involves using feature engineering to generate new variables that are correlated to predicting the score variable. We are going to condense each user's logs into a single row of data and these new generated features will aim to provide an optimal representation of the quality of the essay. Utilizing some of the variables in the other kaggle notebooks as well as using reasoning regarding the most influential variables in writing quality, we will do an initial selection of the variables. We will perform exploratory data analysis including generating correlation matrices and pairwise plots to select variables that are closely associated with the target score variable.

Feature engineering is one of the concepts that was not covered in class, but that is utilized extensively in this notebook to generate relevant features. Feature engineering is the process of creating new variables from existing features that are helpful in predicting the target score variable. In this project, we are utilizing feature engineering by extracting certain characteristics of variables to use in the training data to fit the model. Specifically, we will be using counts of certain values in the variables and will extract this information through an aggregation. This helps train the model on variables that are more associated and contain more information to help predict the score variable.

2.Unsupervised Learning: We will utilize K-means clustering and hierarchical clustering to find if there exists a clusterable structure within the dataset. We will tune linkage and distance metric parameters in hierarchical clustering and we will utilize the number of categories used as the k value in K means.

3. **Regression and Classification Models:** We will implement ridge regression, support vector machines, and random forest regression. This process will include the following:

1. **Hyperparameter tuning:** Find the optimal parameters for each of the model. For ridge regression, it is the optimal lambda regularization parameter. For support vector machines, it is the optimal C regularization parameter. For random forest, the parameters n_estimators, max_depth, min_samples_split, min_samples_leaf were involved in the hyperparameter tuning process.

2. **Training the model:** 80 percent of the data will be used for training and the remaining 20 percent will be utilized for testing. Additionally, each of the models will be trained with the appropriate hyperparameters and will be fit to the training data. Random forest regression and ridge regression were models covered in this course. However, support vector regression is a model that builds off the support vector classification model that was taught in the course. Support Vector regression seeks to find a hyperplane that best fits the data points in a continuous space. SVR, like

SVMs for classification, aims to find a hyperplane that best represents the relationship between the input features and the output variable. The key idea behind SVR is to find a hyperplane that minimizes the margin violations while still ensuring that the predictions fall within a certain range.

3. **Evaluating the model:** The models will be evaluated using Root Mean Squared Error. R squared will also be utilized and it is a measure of how much of the variance in y can be explained by the predictor variables. R squared can be used as a goodness of fit measure and is commonly used throughout the literature.

4. **Interpreting the model results:** Using feature importances and model coefficients, we are trying to extract how the model is associating the features to the target variable. It helps answer questions such as which features is the model using heavily?

Project Conclusion: After evaluating the models, the support vector machine regression model with hyperparameter C of 2 had the highest performance as it resulted in the lowest RMSE value and highest R squared value. The clustering algorithms did not provide optimal clusterings that would be utilized in the final interpretations. Combining the model interpretations, the variables input_count and q_count provided great information in predicting the score of the target variable.

2. Literature review

Best known accuracy: 0.582 is the best known Root Mean Squared Error. The approach was using an LGBMRegressor to do the model fitting with hyperparameter optimization on n_estimators. First, the essay was constructed using logic from the predictor variables. The feature engineering invovled using sentence and paragraph aggregations and utilizing activity and event counts and creating variables such as largest_lantency, smallest_lantency, median_lantncy, initial_pause, pauss_half_sec, word_time ratio, and event_time ratio variables, and more. The main crux of this approach was to recreate the essay, extract new sentence and paragraph features from the constructed essay, and to utilize the other common feature engineering approaches such as activity and event and text change counts.

Other Approaches:

1. One approach consisted of the introduction of sentence and paragraph features in feature engineering. First, the approach consisted of constructing the essays from the predictor variables, splitting them into sentences and paragraphs, counting the characters and words in the sentences and paragraphs, and computed aggregations (e.g, number of sentences, number of paragraphs, mean sentence lengths). The advantage of this approach was that the feature importance plot revealed that many of these sentences and paragraph features are indeed important in predicting the outcome variable. However, one of the disadvantages could be that it is quite computationally expensive and takes more time for the model to train and do the initial data preprocessing.

2. Another approach did not involve advanced analysis of recreating sentences and paragraph aggregation from the essays. Instead, it simply analyzed the specific frequencies of the counts of the predictors such as activity and text change. It also considered distribution of text after replacement and most common text changes. Analyzing associations was done mostly through exploratory data analysis and through plots and visualizations. One advantage of this approach is it is more computationally efficient than the approach in 1 because it doesn't do advanced data

preprocessing. However, this could mean that certain variables that are more predictive in determining the score variable could be left out.
Many of these advanced feature engineering approaches utilized the idea of constructing the essay from the predictor variables and extracting more features from the constructed essay. Additionally, many of the notebooks had a lot of the common engineered features such as specific activity or text change counts. Other approaches also included natural language processing techniques such as word2vec to create and represent word embeddings.

Citations and Links:

https://www.kaggle.com/code/yongsukprasertsuk/writing-processes-to-quality-0-584
https://www.kaggle.com/code/prashantkumarsundge/linking-writing-processes-to-writing-quality
https://www.kaggle.com/code/chuboy/simple-essay-reconstructor
https://www.kaggle.com/code/yuriao/fast-essay-constructor
https://www.kaggle.com/code/suraj520/word2vec-voting-gpu-k-fold-cv-optuna

Application to your Project: From reviewing these approaches, I can implement a few of these practices in this project. First, I am going to implement feature engineering in this project, which is a technique to create new variables using existing variables that are better associated with the target variable. I am going to implement some of the features that were in the literature review and present in the approaches in the other notebooks. Specifically, the features involving activity_nonproduction_count, activity_input_count, and further splitting up activity variable as well as the count of text change values will be considered. Additionally, I will implement correlation matrices and pairwise plots that was also seen during the literature review process of analyzing the relationship between some of the predictors and the target variable.

3. Data Preprocessing

In the data preprocessing, we will identify the features that are most correlated with the target score variable. Correlation plots will be generated to identify these features. Additionally, we do not need to process null values because there are no missing values in the dataset. We need to identify possible features to extract in our final condensed dataframe where we condense all the events for a single id into one row. We will check which possible features to include in the model through a correlation matrix of some possible features.
Below is the reasoning behind many of the features that are selected and detailed guide in the approach of selecting these variables:

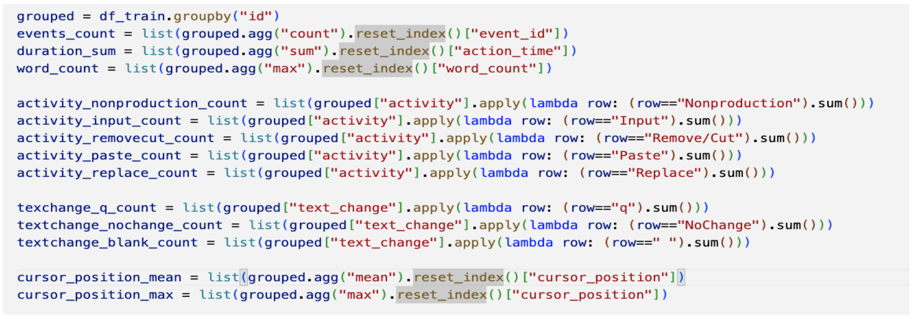
- Number of events represents the number of changes that have occurred in the essay and it is reasonable to anticipate the quantity of edits to have an effect on the quality of the essay.
- The time to write the essay may also have an influence on the quality of the essay. It's reasonable to expect better essays to take longer duration in general.
- The number of words also could reflect upon the quality of the essay. More words may mean that more ideas and thoughts are present in the essay, perhaps leading to a better overall essay.
- Activity NonProduction, Activity Input, Activity RemoveCut, Activity Paste, Activity Replace counts - these were the most common values in activity and were use heavily in the other notebooks during literature review.
- Text change - q count, nochange, and blank count were also the most common values in the text change column and were used by other approaches on kaggle.

These are just potential features that we are using. There will be additional testing to see if they are actually related to the target score variable.

We will now illustrate the process of feature engineering that was used to create these new variables.

- Number of events: Group by id and aggregate by count and extract “event_id”
- Sum of duration : Group by id and aggregate by sum and extract “action_time”
- Word Count: Group by id and aggregate by max and extract “word count”
- Number of Non Production, input, remove cut, paste, and replace activities: All were grouped by id extract “activity” and apply lambda function and get sum of value
- Text change for q, no change, and blank are done using lambda functions
- Cursor position max and mean: Grouped by id and aggregate by mean/max and extract cursor_position

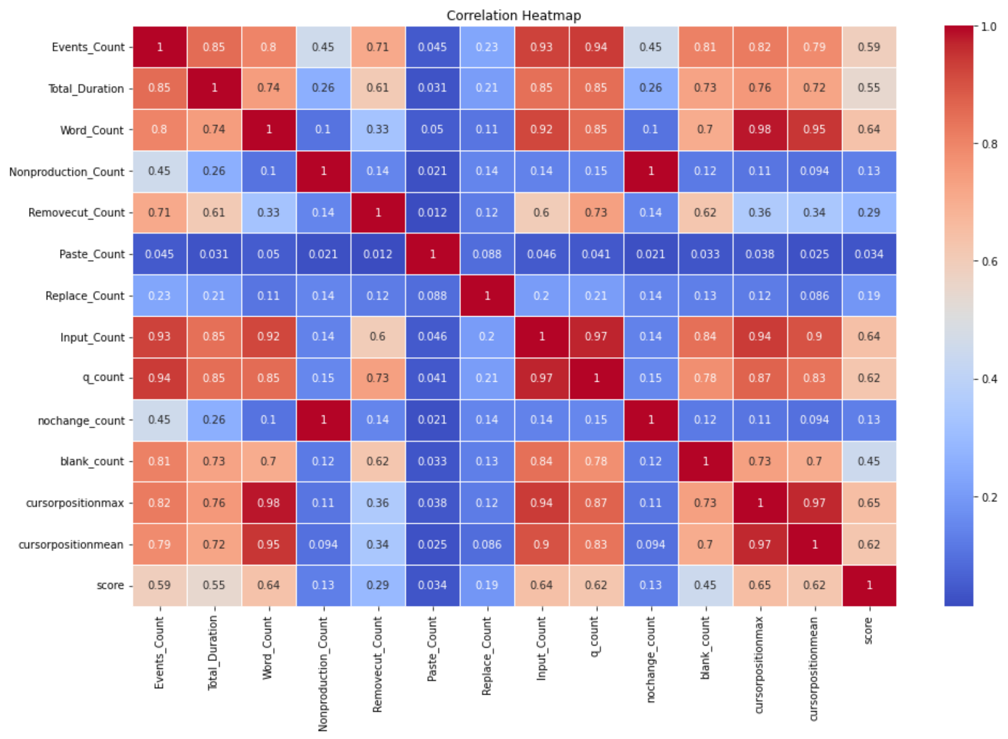
The following code snippet showcases how the feature engineering was done through group bys, aggregations, and lambda functions:



```
grouped = df_train.groupby("id")
events_count = list(grouped.agg("count").reset_index()["event_id"])
duration_sum = list(grouped.agg("sum").reset_index()["action_time"])
word_count = list(grouped.agg("max").reset_index()["word_count"])

activity_nonproduction_count = list(grouped["activity"].apply(lambda row: (row=="Nonproduction").sum()))
activity_input_count = list(grouped["activity"].apply(lambda row: (row=="Input").sum()))
activity_removecut_count = list(grouped["activity"].apply(lambda row: (row=="Remove/Cut").sum()))
activity_paste_count = list(grouped["activity"].apply(lambda row: (row=="Paste").sum()))
activity_replace_count = list(grouped["activity"].apply(lambda row: (row=="Replace").sum()))

texchange_q_count = list(grouped["text_change"].apply(lambda row: (row=="q").sum()))
textchange_nochange_count = list(grouped["text_change"].apply(lambda row: (row=="NoChange").sum()))
textchange_blank_count = list(grouped["text_change"].apply(lambda row: (row==" ").sum()))

cursor_position_mean = list(grouped.agg("mean").reset_index()["cursor_position"])
cursor_position_max = list(grouped.agg("max").reset_index()["cursor_position"])
```

By creating these features using group bys and aggregations, we are condensing each user’s log, which usually spans several thousands rows, into a single row of data.

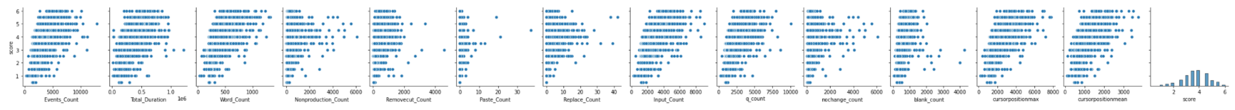
Next in order to measure the correlation and association between these new features and the score variable, we will create a correlation plot to aid us in feature selection of our new features.



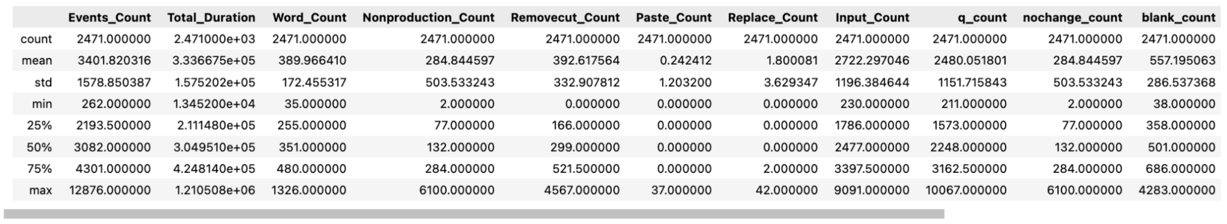
From the above correlation matrix, we see that there are many variables that have moderate correlation to the score variable. This means that it is possible that those variables can help us determine score and could potentially be useful in our model. The variables that have a moderately strong correlation with score are the following:

1. Number of events
2. Total duration
3. Word Count
4. Input Count
5. q_count
6. Cursorpositionmax

These might be the variables that have most influence in the models, but we will include the variables that we extracted from feature engineering above to gain a more holistic picture.



Additionally, after analyzing the pairwise plots of these features, it does not appear that there are any apparent and obvious outliers, so we do not need to implement strategies to handle those cases.



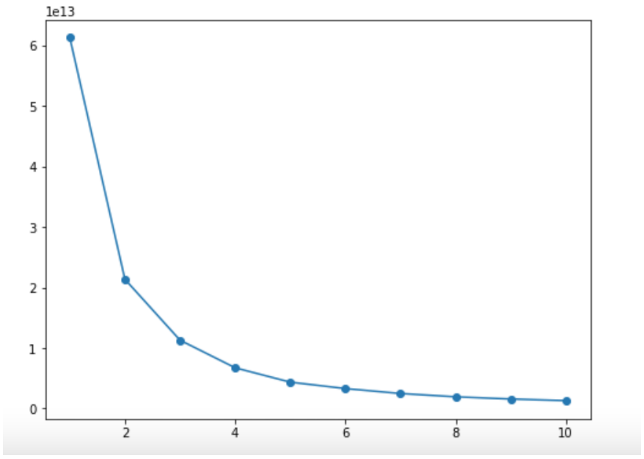
	Events_Count	Total_Duration	Word_Count	Nonproduction_Count	Removecut_Count	Paste_Count	Replace_Count	Input_Count	q_count	nochange_count	blank_count
count	2471.000000	2.471000e+03	2471.000000	2471.000000	2471.000000	2471.000000	2471.000000	2471.000000	2471.000000	2471.000000	2471.000000
mean	3401.820316	3.336675e+05	389.966410	284.844597	392.617564	0.242412	1.800081	2722.297046	2480.051801	284.844597	557.195063
std	1578.850387	1.575202e+05	172.455317	503.533243	332.907812	1.203200	3.629347	1196.384644	1151.715843	503.533243	286.537368
min	262.000000	1.345200e+04	35.000000	2.000000	0.000000	0.000000	0.000000	230.000000	211.000000	2.000000	38.000000
25%	2193.500000	2.111480e+05	255.000000	77.000000	166.000000	0.000000	0.000000	1786.000000	1573.000000	77.000000	358.000000
50%	3082.000000	3.049510e+05	351.000000	132.000000	299.000000	0.000000	0.000000	2477.000000	2248.000000	132.000000	501.000000
75%	4301.000000	4.248140e+05	480.000000	284.000000	521.500000	0.000000	2.000000	3397.500000	3162.500000	284.000000	686.000000
max	12876.000000	1.210508e+06	1326.000000	6100.000000	4567.000000	37.000000	42.000000	9091.000000	10067.000000	6100.000000	4283.000000

From the summary of the newly created dataframe, we can see that there is a large discrepancy in the scale of the variables. This can be seen by the large variance in the mean and standard deviation of all the variables. As a result, we will scale this dataframe using StandardScalar() in python. This will also help make the dataset more well suited for the unsupervised and supervised algorithms that are going to be applied below. For running each of the below models, we go through this data preprocessing steps of feature engineering and scaling.

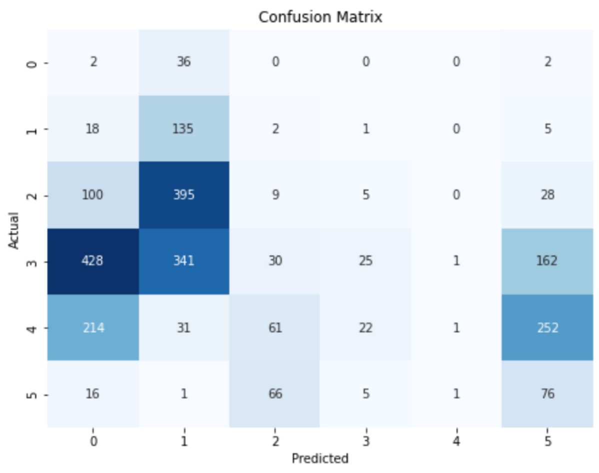
5. Unsupervised Learning

1. K means clustering

We will treat the different levels of the score variable as 6 categories. Typically, the tuning process for K means involves finding the optimal number of clusters k. In this case, the optimal number of clusters would be from the elbow plot below plotting the k means inertia vs the number of customers where the elbow is around k = 2 clusters. Elbow plot is a methodology to help determine the optimal clusters for a dataset. The idea is to plot the explained variation as a function of the number of clusters and look for an "elbow" point where adding more clusters does not significantly improve the model's performance.

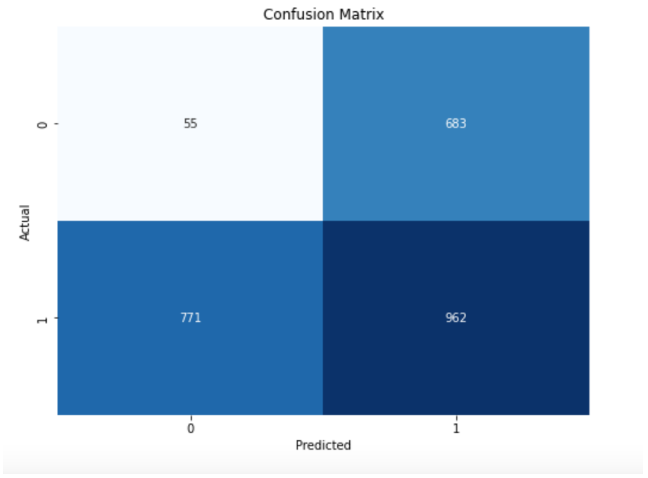


However, because we already have a predefined set of true labels, we will use that number which would be k = 6 clusters. To evaluate this k means clustering, we can compare the clusters with the true labels or categories of the score variable.



	0	1	2	3	4	5
0	2	36	0	0	0	2
1	18	135	2	1	0	5
2	100	395	9	5	0	28
3	428	341	30	25	1	162
4	214	31	61	22	1	252
5	16	1	66	5	1	76

From this confusion matrix, we can see that this is not an optimal clustering result. Majority of our predictions do not align with the true values. This could be because of a multitude of reasons either k = 6 is a large number of clusters to do the evaluation or there are no intrinsic characteristics within the data that can distinguish between categories (the data may not be clusterable).

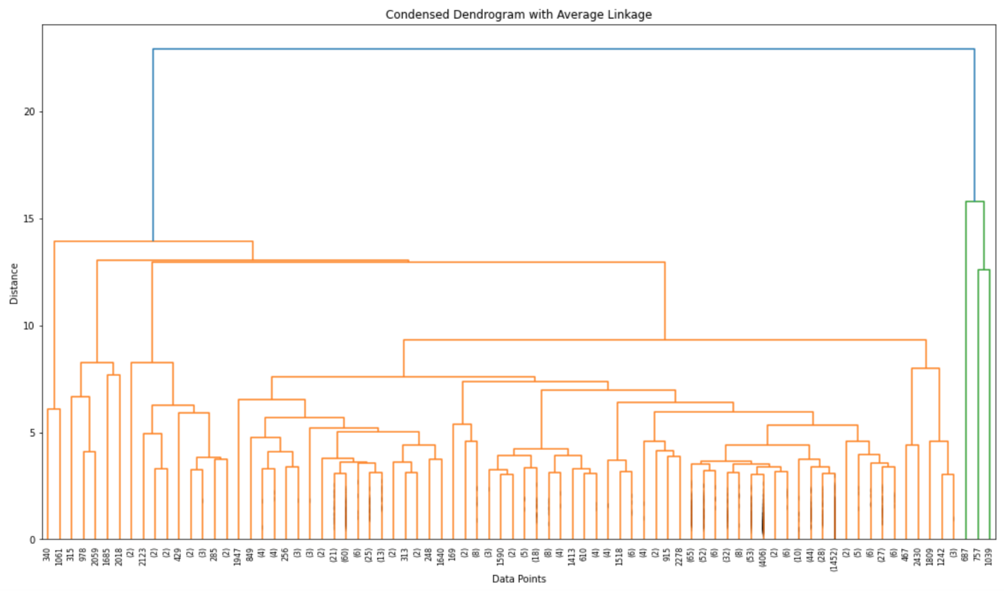


Even when we use the optimal k = 2 clusters for the K-means clustering algorithm, the confusion matrix is not indicative of a strong clustering when we compare with the true labels.

Interpretation: Due to the poor performance of the K-means clustering algorithm, it might be reasonable to interpret that our dataset is not highly clusterable by K-means clustering. We can see from the confusion matrices that the clustered labels are not clearly associated with the true labels of the dataset. We are not able to find common patterns within the data itself that distinguish from one category to another. This may mean that for the supervised learning approach, we may not want to utilize the predicted clusters as features in the model.

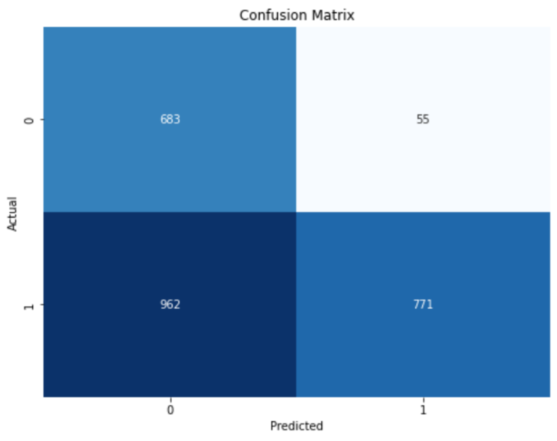
2. Hierarchical Clustering

We will tune the distance metric and the linkage parameter in the hierarchical clustering algorithm based upon the characteristics of our data. One characteristic of our data is that we have unevenly sized clusters. For example, 501 have a 4.0 score while 92 have a 2.0 score. Additionally, from our exploratory analysis in the pairwise plots in the data preprocessing section, outliers should not be an issue. Therefore, we will utilize Average linkage because average linkage is suitable for data with unevenly sized clusters and provides a good compromise between the compactness of complete linkage and flexibility of single linkage. Since our data is scaled, we will use the default Euclidean distance. This is the resulting dendrogram after the clustering.

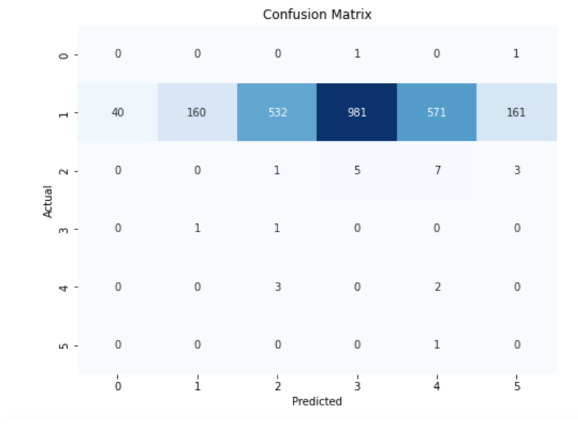


This dendrogram indicates a moderate clustering of the data because there are some clear branches and some clusters are merged at higher heights providing more information about the similarity between clusters. We will evaluate the clustering at k = 2 clusters because those clusters cut off at higher distances. We will utilize confusion matrices as well as sensitivity, accuracy, and specificity to evaluate these clusters with the true labels. Because there are 12 true labels, to do k = 2 clusters, we will split the 12 clusters as follows: [0,1,2] -> 0 and [3,4,5] -> 1. Additionally since we can do 6 true labels or categories of the score variable, we can evaluate the clustering at k = 6 clusters.

2 clusters



6 clusters



```
Specificity: 0.9254742547425474
Accuracy: 0.5884257385673817
Sensitivity (Recall): 0.4448932487016734
```

From the 6 clusters confusion matrix, we can see that the clustered labels are not at all associated with the true labels of the score variable. From two clusters, we can get a somewhat better association between the true categories and the clustered labels. However, overall, these clusterings are not providing great results in terms of clusterability and association with the true categories. As a result, we may not want to use these in the supervised models.

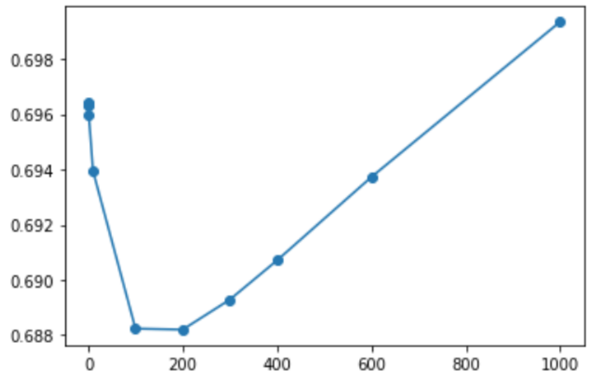
5. Regression/Classification Models

1. Ridge Regression

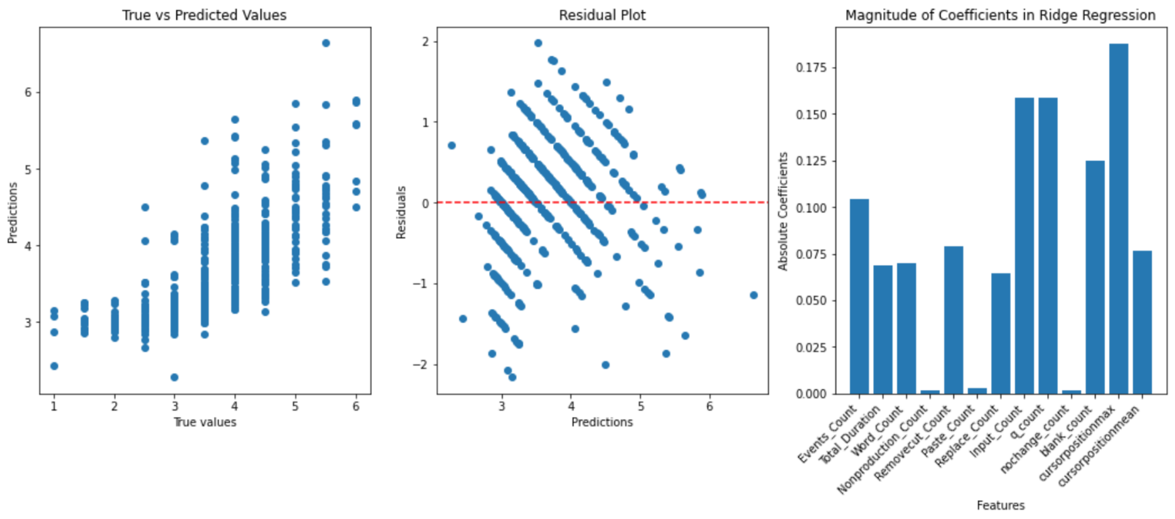
We will utilize Ridge regression because we have high dimensionality with some correlated features. Additionally, we have already selected some features in our model, so we do not necessarily want the model to eliminate certain features (which could be the case in Lasso regression). As a result, we will utilize Ridge Regression. We also want this model to perform well on unseen data (for the Kaggle competition) and reduce overfitting (bias). This can be achieved through the regularization penalty in ridge regression. We will perform parameter tuning for the lambda in ridge regression and select the optimal lambda to use in the model.

The following plot shows the optimal lambda from a range of lambda values in the parameter tuning as well as model performance results from fitting the ridge regression model with the optimal lambda.

```
Optimal lambda:  200
R-squared (R2): 0.5141078124706779
Root Mean Squared Error: 0.68820069729888
```



The following plots further showcase the performance and fit o fthe ridge regression model and the magnitude of the coefficients plot helps to provide some interpretability of the model with regards to what variables had the largest influence in predicting the target score variable.



Model Performance: The Ridge Regression model yields an R squared value of 0.51 and a RMSE of 0.688. This suggests that the model is a moderate fit. The R squared of 0.51 suggests that around 50% of the variance in the dependent variable (score) is explained by the independent variable(s), and the other 50% is still unaccounted for. There is also a moderate association between the true values and the predictions as seen in the plot above. Generally, the points on the plot should fall along a diagonal line, indicating that the predicted values are close to the true values and we see that the points are somewhat along the diagonal line in the plot above. The spread of residuals is generally overall constant across most levels of predicted values, suggesting approximate homoscedasticity. There are also no significant outliers in the residuals plot.

Model Interpretation: The plot above showcases the magnitudes of coefficients in ridge regression. In Ridge Regression, a larger magnitude of a coefficient means that the model is relying more heavily on that particular feature to make predictions. In this case, it appears that cursor position max, Input_Count, q_Count, and events_count are the features that the model is relying more heavily on to make the score prediction. It is possible that these features with larger coefficients might be more influential. Additionally, it looks like certain variables such as Nonproduction_Count, Replace_Count, and nonchange_count have absolute magnitudes that are relatively lower than the rest of the features, suggesting that they did not have significant influence in the model in making the prediction for score.

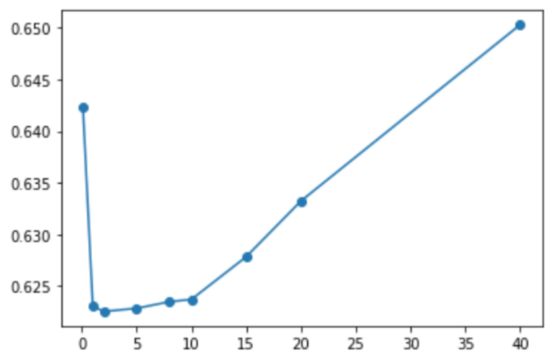
2. Support Vector Machine Regression

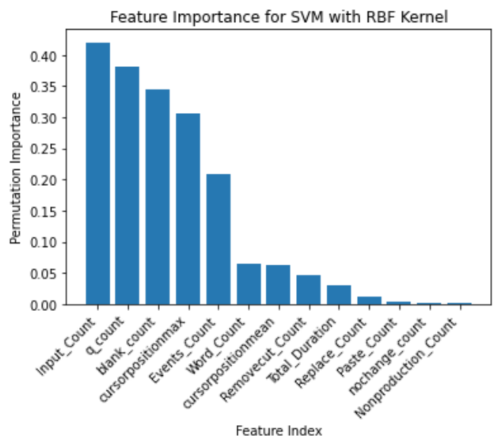
Support Vector Machine Regression builds on Support Vector classification that was discussed in the course. A SVR, like SVMs for classification, aims to find a hyperplane that best represents the relationship between the input features and the output variable. The key idea behind SVR is to find a hyperplane that minimizes the margin violations while still ensuring that the predictions fall within a certain range or epsilon-tube around the true values.

We will utilize the RBF kernel instead of the default linear kernel because we expect the relationships between the features and the score variable. The RBF kernel (also known as the Gaussian kernel) is a popular choice when the relationship is expected to be non-linear or when the data has complex patterns. It allows the SVM to model non-linear decision boundaries effectively. We will use parameter tuning on the regularization parameter (C) to identify the optimal C to use in the final SVM regression model.

The following plot shows the optimal C to use in the final Support Vector regression model as well as the performance of the model.

```
Optimal C:  2
R-squared (R2): 0.6023896931141928
Root Mean Squared Error: 0.6225496123497969
```





To enable interpretability of this SVM model, we will assess feature importance using the permutation importance method to identify the important features used by the model in making the score predictions.

Model Performance: The SVM regression model yields a R squared value of 0.6 and RSME of 0.622. This is considerably better than the performance we were able to receive with the previous ridge regression model. This suggests that the SVM regression model is a moderately strong fit. The R squared value of 0.62 suggests that the SVM model is a moderately strong fit. The R squared of 0.6 suggests that around 60 percent of the variance in the dependent variable (score) is explained by the independent variable(s), and the other 40 percent is still unaccounted for. Additionally, the predicted vs true values plot also appears moderately strong as the overall trend is somewhat close to a diagonal indicating that in general, the predicted values are close to the true values.

Model Interpretation: From the feature importance plot above, we can see that Input_Count, q_Count, blank_count, cursorpositionmax, and Events_counts variable are the variables that have high influence in the model when making the score predictions.

3. Random Forest Regression

We will first engage in the parameter tuning process for random forest. We will tune the following parameters. The criterion for the tuning will be mean squared error. The best parameters after the tuning process are shown below. The final parameters are used and the performance metrics are shown.

1. n_estimators
2. max_depth
3. min_samples_split
4. Min_samples_leaf

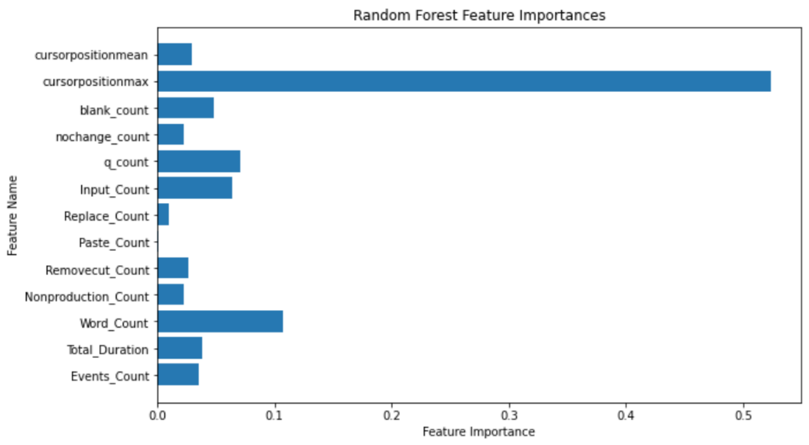
```
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

```
Best Hyperparameters: {'max_depth': 10, 'min_samples_leaf': 4, 'min_samples_split': 5, 'n_estimators': 200}
```

```
R-squared (R2): 0.5977897671970029
Root Mean Squared Error: 0.6261403734270747
```

The model performance encompasses an R squared value of around 0.6 and RMSE of 0.626 which is similar to the performance results that were obtained from the support vector regression. The R squared value of 0.62 suggests that the random forest model is a moderately strong fit. The R squared of 0.6 suggests that around 60 percent of the variance in the dependent variable (score) is explained by the independent variable(s), and the other 40 percent is still unaccounted for.

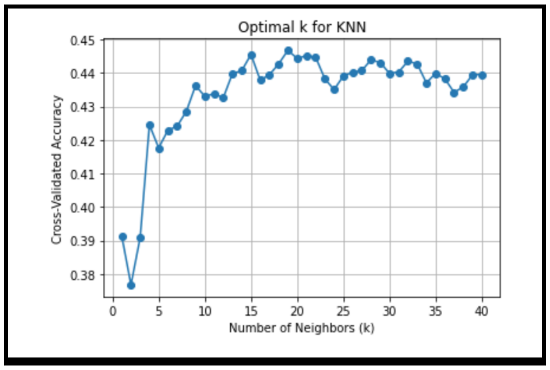
We can do model interpretability with random forest by utilizing feature importance. Feature importance in a Random Forest refers to a score assigned to each feature (variable) indicating the relative importance of that feature when making predictions.



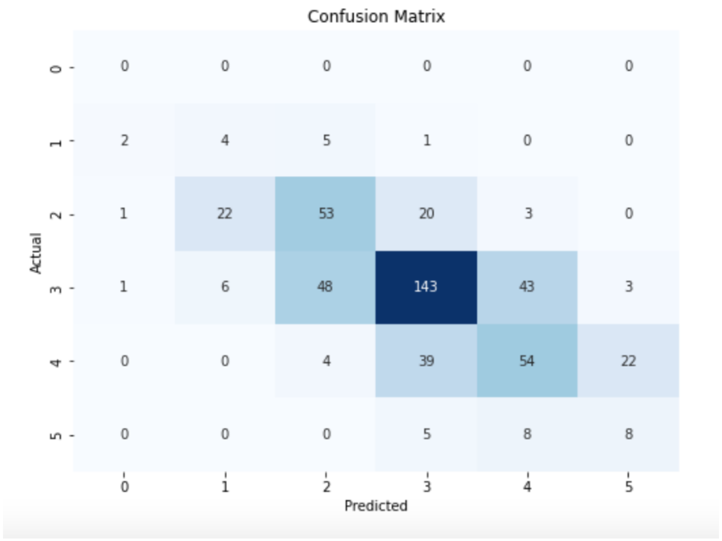
This shows that by a great deal, cursorpositionmax was the most important and influential variable in the model to help determine the target score variable. All of the variables do not seem to compare to the magnitude of cursorpositionmax's influence. This falls in agreement with what we see from the interpretation of support vector regression where cursorpositionmax variable also exemplified high impact.

4. K Nearest Neighbors (Classification Algorithm)

In the KNN classification algorithm, we will tune the k = number of nearest neighbors parameter. This is the most important parameter in K nearest neighbors. When you set a specific value for "k," the algorithm looks at the "k" nearest neighbors of a data point and assigns the class label based on majority voting. We will utilize the accuracy metric as evaluation criteria for k because this is a classification problem. The results of the tuning are below and the optimal k that results in the highest accuracy will be utilized in the final model. The range of k values that were used was 1 to 41 because a common rule of thumb for the k parameter in KNN is to consider k values from 1 to square root of the number of points in the dataset. The sample size affects the number of K to experiment and ultimately use. In this case, due to the size of the dataset, max experimented k value was 41.



The optimal k that maximizes the accuracy is k = 19 which will be used in the final model. After fitting the model and testing it on the test data, the confusion matrix and accuracy were found to evaluate the performance of the model.


Confusion Matrix

Actual \ Predicted	0	1	2	3	4	5
0	0	0	0	0	0	0
1	2	4	5	1	0	0
2	1	22	53	20	3	0
3	1	6	48	143	43	3
4	0	0	4	39	54	22
5	0	0	0	5	8	8

The accuracy is 52.9 percent meaning the algorithm will correctly classify the data point into the correct category around 53 percent of the time. Considering that this is used for 6 classes, this is a moderately fit model. From this model performance, we can conclude that there are certain patterns and characteristics of the features in the dataset that are associated with a particular label. The model is able to associate some of the features in the model to the categories in the score variable.

Conclusion: Overall, it appears that the support vector regressor produces the most optimal performance in terms of MSE and R squared values. From looking at the feature importance plots and interpretations, cursor position max, input count, and blank count were the most influential variables in the models. Secondly, events count and word count also were important variables across all the models when analyzing the magnitude of coefficients and feature importances. Number of inputs and blanks could be related to more activity in the generation of the essay and cursor position max may be related to the overall length and progression of the essay. These variables seem to be the most predictive of essay quality and predicting the score variable. Further interpreting these variables, we can conclude from the results of the models that the length or progress of the essay and the type of inputs in the essay play a significant role in determining the overall quality of the essay.