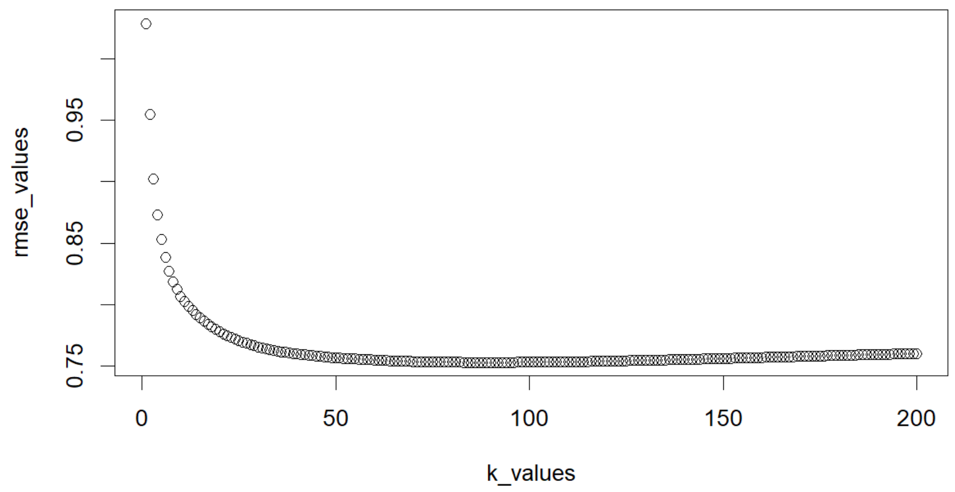
The regression / classification models we decided to run on the data are the following: elastic net model, k-nearest neighbors regression, random forest regression, and support vector machines classification. Our evaluation criteria for all of our models was root-mean-square error (RMSE).

For the first model, we decided to use elastic net regression. As a balance of Ridge and LASSO regression, the elastic net regression model is useful for handling both multicollinearity issues and also doing variable selection (i.e. removing variables that may not be as important in predicting the outcome). As noted in the Data Processing section, many variables (e.g. “total_events” and “total_input_events” and “final_word_count” and “total_text_added”) are highly correlated, simply by the nature of how these variables were processed from the original dataset. Furthermore, since we have not performed variable selection yet, this aspect of elastic net regression will be helpful in our analysis.

Using a sequence of alpha values from 0 to 100, we fit models with the alpha values, found the optimal lambda for these models, and compared their performance in prediction in RMSE. The optimal combination of alpha and lambda that resulted in the lowest possible RMSE were as follows: alpha = 0.98, lambda = 0.002004471, with an RMSE = 0.7546587. What is interesting to note is that alpha = 0.98, which is very close to alpha = 1, which is just LASSO regression.

Inspecting the coefficients, we see that elastic net removed the variables "sd_action_time" and "max_cursor_movement" indicating that they may not be important for predicting the score of the row. Noting the largest coefficients by magnitude, we note that the variables "avg_text_change_length", "avg_cursor_movement", "min_action_time" seem important for predicting the score.

For the second model, we decided to use k-nearest neighbors. Similarly, we tune k by creating models with k = 1 to k = 200 and compare the performance of all of these models. We found that the optimal k is k = 87, which returned RMSE = 0.7528342.

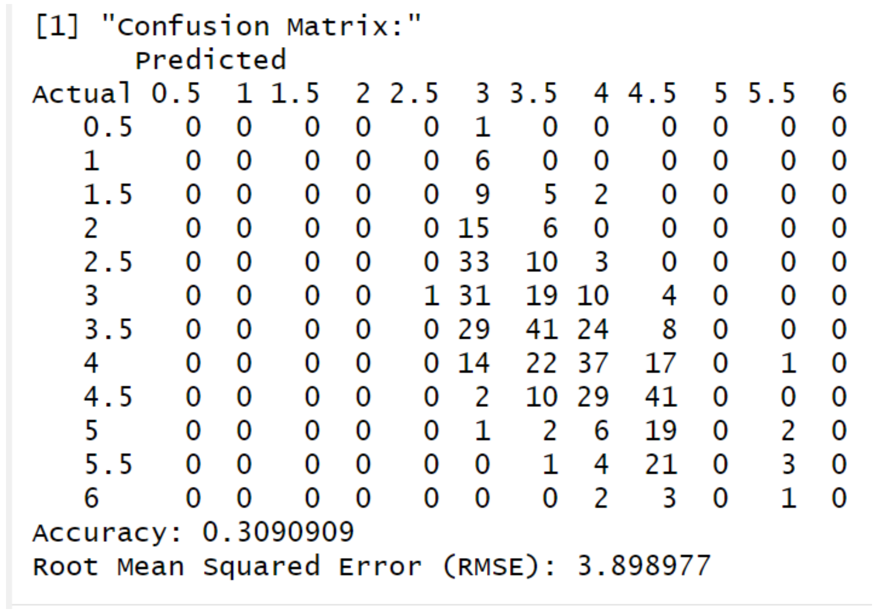


We can see that the RMSE has a sharp drop in the beginning as the bias increases and variance sharply decreases. However, the RMSE levels off and eventually starts increasing as bias continues to increase and the model overfits, demonstrating the bias-variance tradeoff involved in tuning k for k-nearest neighbors algorithm.

The next model we tried was random forest regression. We note that random forest has more parameters to tune than the previous algorithms: namely, ntree, sampsize, mtry, and nodesize. Using a grid search to search every possible combination of these parameters and comparing all of their performances in prediction, we found that the optimal combination of parameters was as follows: ntree = 700, sampsize = 110, mtry = 13, nodesize = 1, with RMSE = 0.6829625. This is the best RMSE we have gotten thus far.

Inspecting variable importance from the model, we find that the three most important variables are "final_word_count", "total_events", and "max_cursor_movement", in terms of Increase in Node Purity (IncNodePurity).

The classification model we fit was a support vector machine classification. We did this using the svm() function from the e1071 library, which does classification when the outcome variable is specified as a factor. Cost is a parameter that we can tune for this model, and it affects the "cost" of misclassification. In other words, with a larger cost, the model may have a more complex and jumpy decision boundary to avoid misclassification. Optimizing this parameter with RMSE as we did with the previous models, we got an optimal cost = 1 with an RMSE = 3.898977.



```
[1] "Confusion Matrix:"
      Predicted
Actual 0.5  1 1.5  2 2.5  3 3.5  4 4.5  5 5.5  6
   0.5   0  0   0  0   0  1   0  0   0  0   0  0
   1     0  0   0  0   0  6   0  0   0  0   0  0
   1.5   0  0   0  0   0  9   5  2   0  0   0  0
   2     0  0   0  0   0 15   6  0   0  0   0  0
   2.5   0  0   0  0   0 33  10  3   0  0   0  0
   3     0  0   0  0   1 31  19 10   4  0   0  0
   3.5   0  0   0  0   0 29  41 24   8  0   0  0
   4     0  0   0  0   0 14  22 37  17  0   1  0
   4.5   0  0   0  0   0  2  10 29  41  0   0  0
   5     0  0   0  0   0  1   2  6  19  0   2  0
   5.5   0  0   0  0   0  0   1  4  21  0   3  0
   6     0  0   0  0   0  0   0  2   3  0   1  0
Accuracy: 0.3090909
Root Mean Squared Error (RMSE): 3.898977
```

Looking at the classification confusion matrix and accuracy = 30.90909%, we see that performance of this model is quite bad. The model tends to confuse many of the scores in the 2.5 to 4.5 range, which, as we noted in the Data Processing section tends to be where the majority of data points lie.

Just for the sake of curiosity, we also fit an svm regression model and did the optimization for cost. This model performed much better with an optimal cost = 75 correspondent to an optimal RMSE = 0.7580726.

	Model <chr>	RMSE <dbl>
3	random forests	0.6829625
2	k nearest neighbors	0.7528342
1	elastic net	0.7546587
5	svm regression	0.7580726
4	svm classification	3.8989770

The performances of all of our models relative to each other are summarized in the table above. We see that random forests has the lowest RMSE, while svm classification has the largest RMSE by a significant margin.

Elastic net coefficients:

```
(Intercept)                2.285071e+00
total_events               3.845065e-04
total_nonproduction_events -1.903365e-04
total_remove_cut_events    -4.182445e-04
total_paste_events         -1.161791e-03
total_replace_events        1.840437e-02
avg_action_time            -1.252771e-04
max_action_time             1.594265e-06
min_action_time            -4.849160e-02
sd_action_time              .
final_word_count            1.049640e-03
avg_text_change_length     -9.385741e-02
total_text_removed         -1.689733e-04
max_cursor_movement         .
avg_cursor_movement        -7.187129e-02
sd_cursor_movement          5.070199e-03
```

```
 [1] "(Intercept)"                "avg_text_change_length"     "avg_cursor_movement"
 [4] "min_action_time"            "total_replace_events"       "sd_cursor_movement"
 [7] "total_paste_events"         "final_word_count"           "total_remove_cut_events"
[10] "total_events"               "total_nonproduction_events" "total_text_removed"
[13] "avg_action_time"            "max_action_time"            "sd_action_time"
[16] "max_cursor_movement"
```

Random forest variable importance

```
                          IncNodePurity
total_events                 17.4698577
total_nonproduction_events    3.7838847
total_remove_cut_events       3.9744701
total_paste_events            0.4763130
total_replace_events          1.8217903
avg_action_time               4.4580144
max_action_time               4.0362660
min_action_time               0.3842305
sd_action_time                3.6217054
final_word_count             51.0266429
avg_text_change_length        4.1440921
total_text_removed            3.9363097
max_cursor_movement           5.3090077
avg_cursor_movement           4.5206313
sd_cursor_movement            4.7586477
```

```
 [1] "final_word_count"          "total_events"               "max_cursor_movement"
 [4] "sd_cursor_movement"        "avg_cursor_movement"        "avg_action_time"
 [7] "avg_text_change_length"    "max_action_time"            "total_remove_cut_events"
[10] "total_text_removed"        "total_nonproduction_events" "sd_action_time"
[13] "total_replace_events"      "total_paste_events"         "min_action_time"
```

If we compare these two lists of variable importance and identify which variables are important in both models, we find that the most important variables are “avg_cursor_movement”, “avg_text_change_length”, “final_word_count”. More active cursor movement could suggest more editing and refinement of the essay. Likewise, higher active text change length could mean there were multiple revisions of an essay, which could lead to a better essay. Finally, a higher final word count could mean the author has been very detailed in their explanations, which could lead to a better essay.