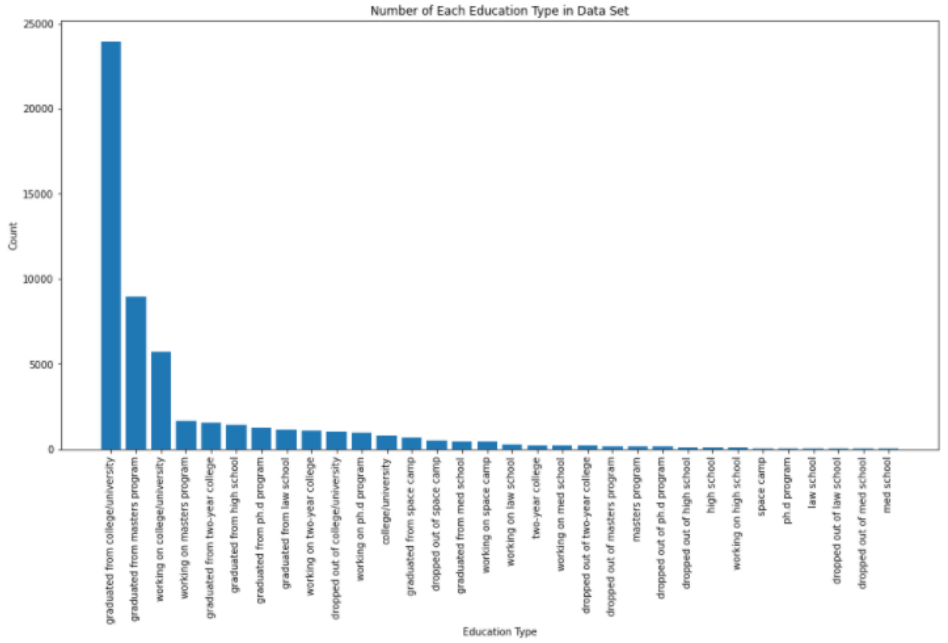
Predict Someone Who Uses Big Words in OkCupid Data

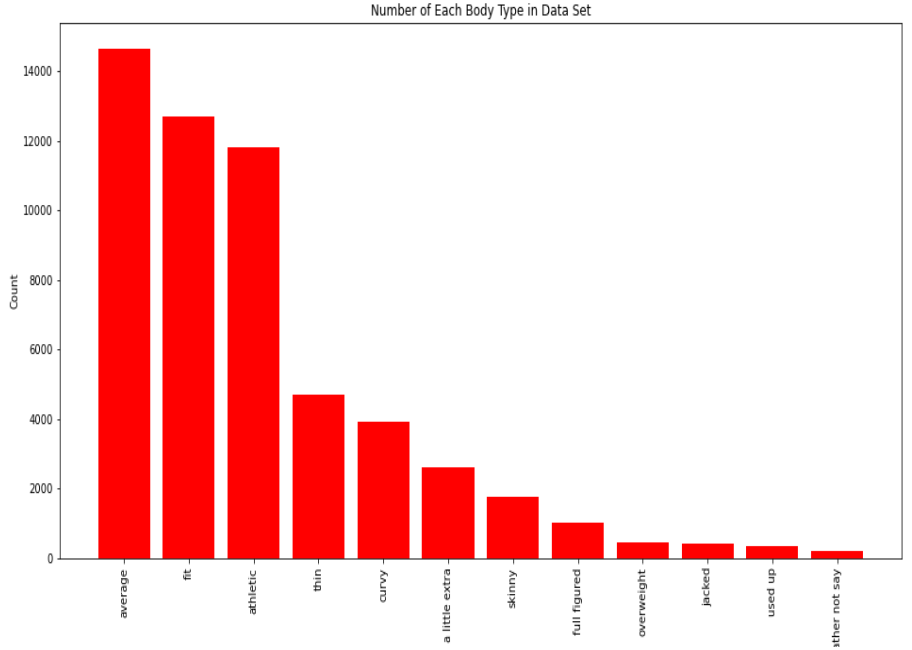
Other than finding who uses big words you might be able to ascertain individuals who are well-read but also even a bit pretentious in the data set! Those individuals could be matched!

Data Exploration

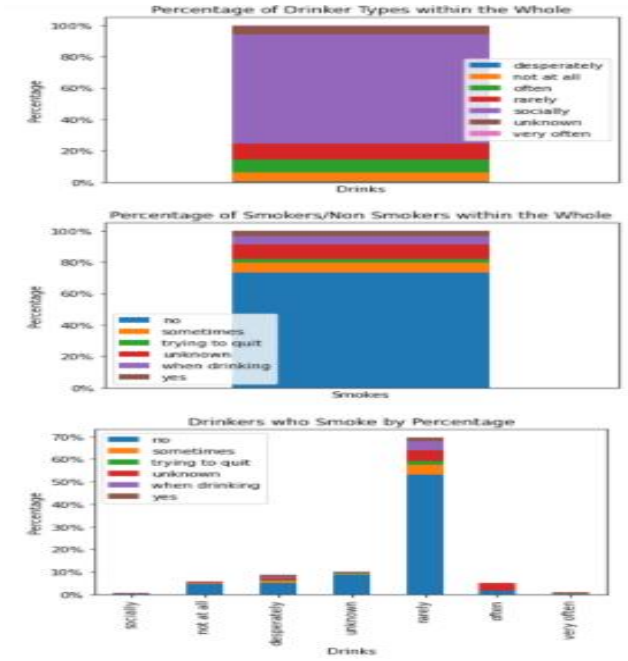
- During this phase of analysis, I analyzed several trends within the data set
  - Counts of Various Categorical-type Variables and the percentage of those categories within their sums
  - The correlation relationship between variables

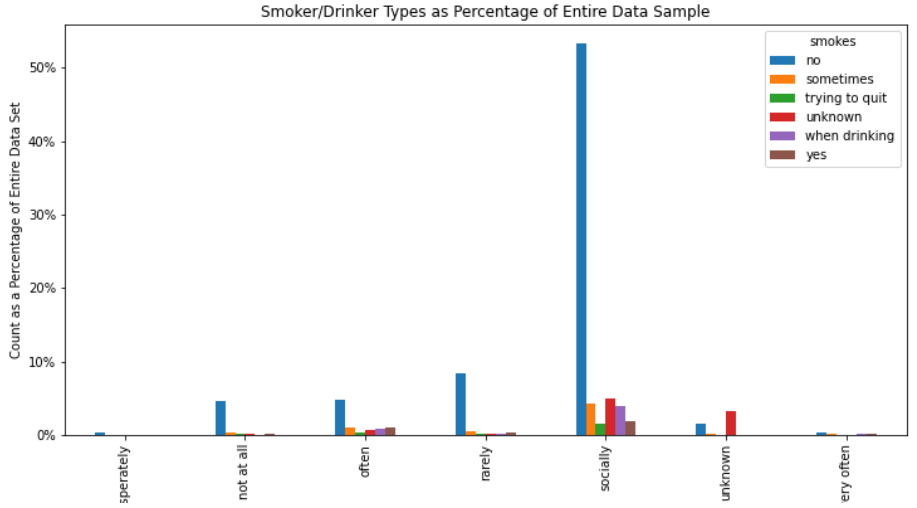
Count of Each Sample Feature



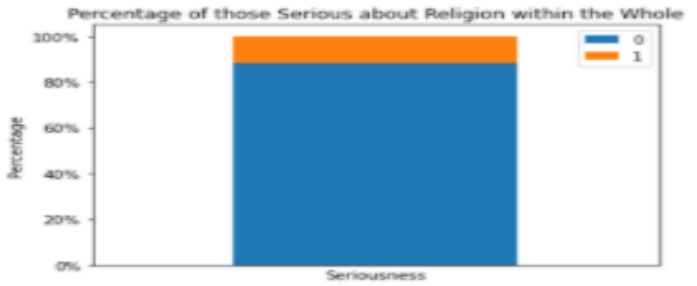


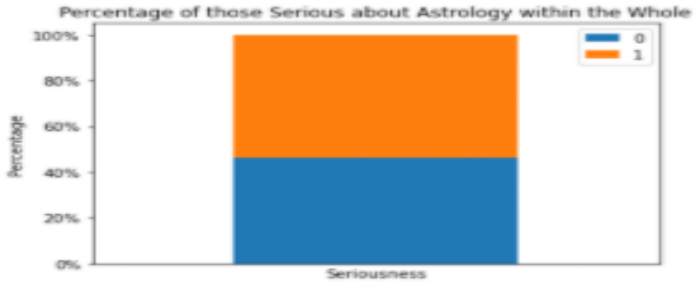
Relationship Between Smoking and Drinking

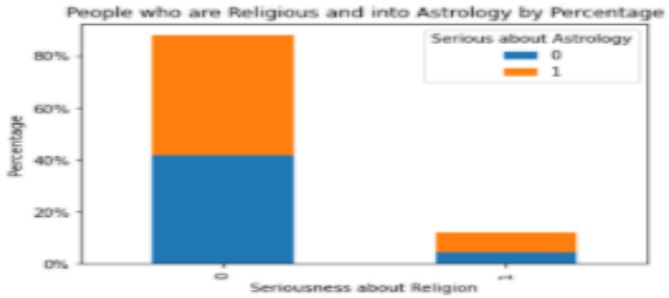




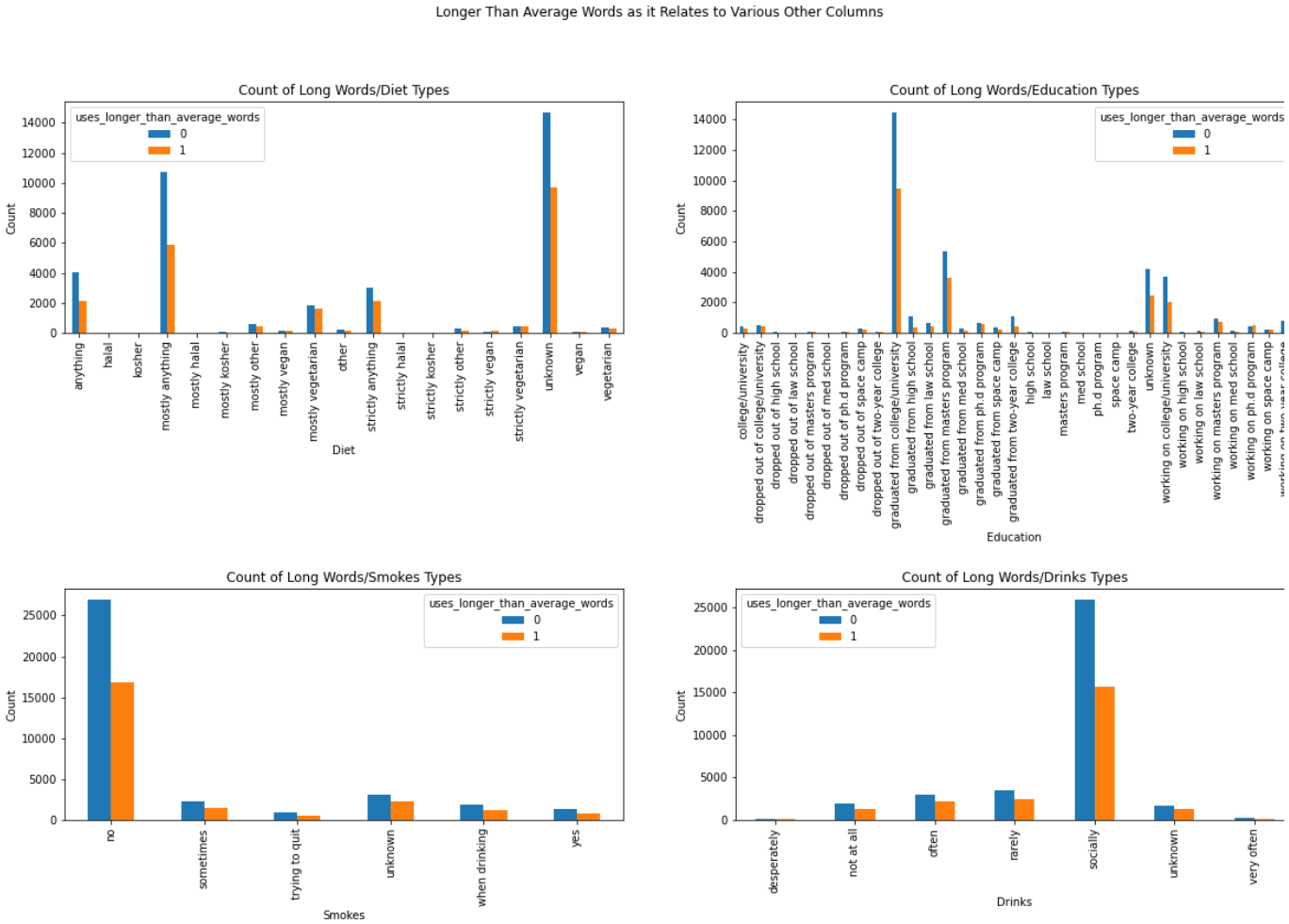
Other Relationships







What types of people use longer than average words?



Question

- Can I use various feature sets and machine learning algorithms to predict someone that uses longer than average words?
  - I thought predicting someone's vocabulary choice might be interesting but also this might reveal someone who is a bit pretentious.

Augment Data

- Created two binary true/false columns called “serious about religion” and “serious about astrology”.
  - These columns were created by looking for people that indicated they a religion or astrology and that they were serious about it
- Categorized several columns including education, religion, astrology, diet, drinks, smokes
  - This just provides a numerical category variable for each unique type within the column
- Created an indicator for whether the user uses “longer than average words.” How did we arrive here?
  - Each individual had eight essays. We merged those essays together, counted length of the entire essays in characters, calculated the length of each word within the essay, and then calculated the average length of each word
  - Then set to True if the individual’s average length of words in his/her essay is longer than the average length for all individuals in the entire OkCupid data set. Else False.
- Min/max normalization of data

Define Feature Columns to Predict an Individual who Uses Big Words

Feature Columns

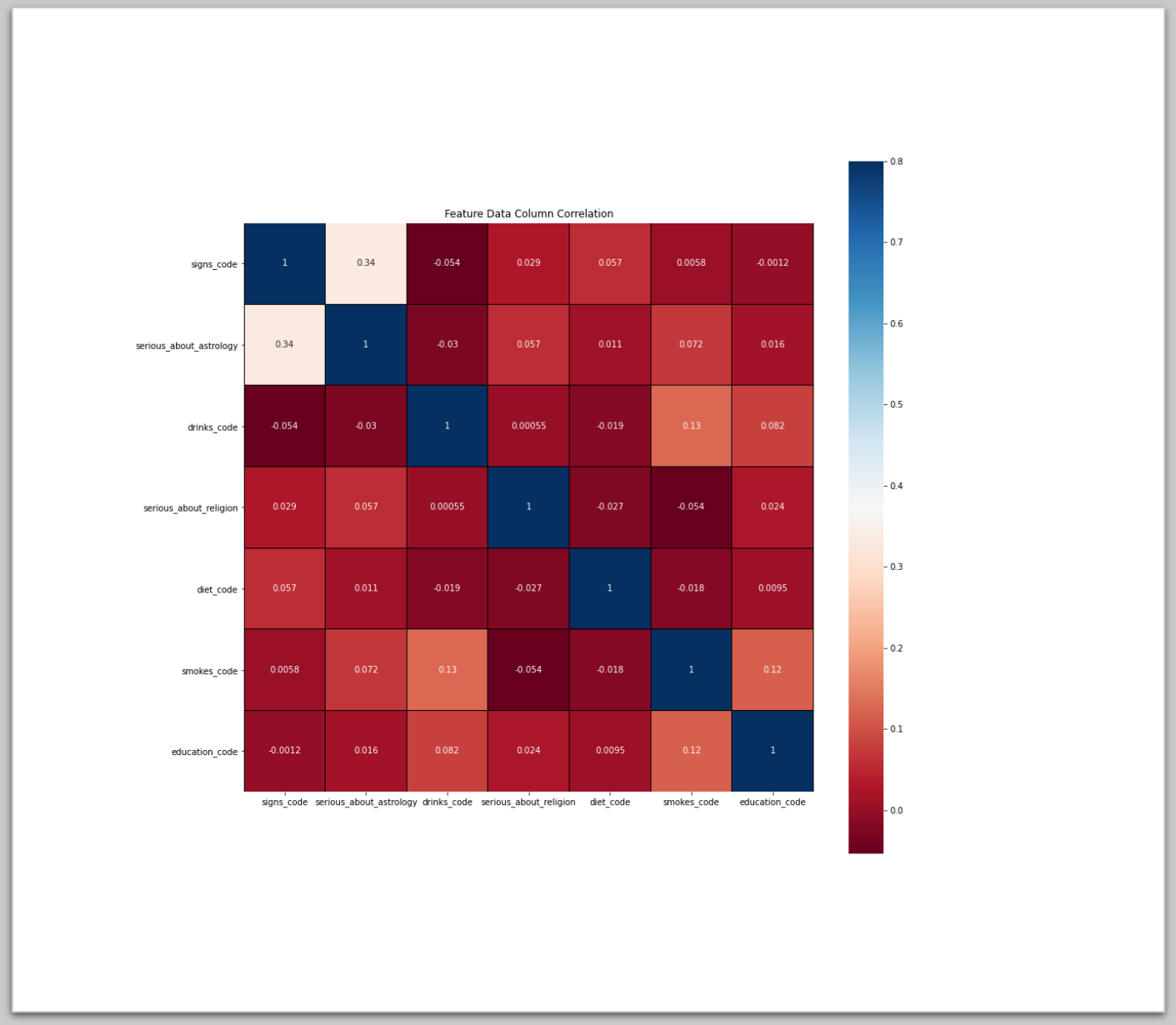
- Binary True/False Features
  - Serious about religion
  - Serious about astrology
- Categorically mapped Features
  - Education code
  - Religion code
  - Astrology code
  - Diet Code
  - Drinks code
  - Smoke code

Define Target Column

- Uses Longer than average words

Relationship Between Possible Feature Columns and Target

- Correlation matrices do not make a lot of sense for the data set because I just categorized types within the variables. The three columns that could be evaluated this way are serious about religion, serious about astrology, and uses longer than average words
- Unfortunately, none of these columns are correlated positively or negatively. Meaning one does not go up as the other goes down and both do not go up together. This is indicated by the values being close to zero.



Employ Machine Learning Algorithms to Predict Target

- Supervised ML algorithms
  - K Nearest Neighbors (KNN)
  - SVM – did not work because I think the data set is too large and diverse
  - Decision Tree
  - Random Forest
- Regression
  - Logarithmic
  - Linear – did not work for the problem set

Methodologies

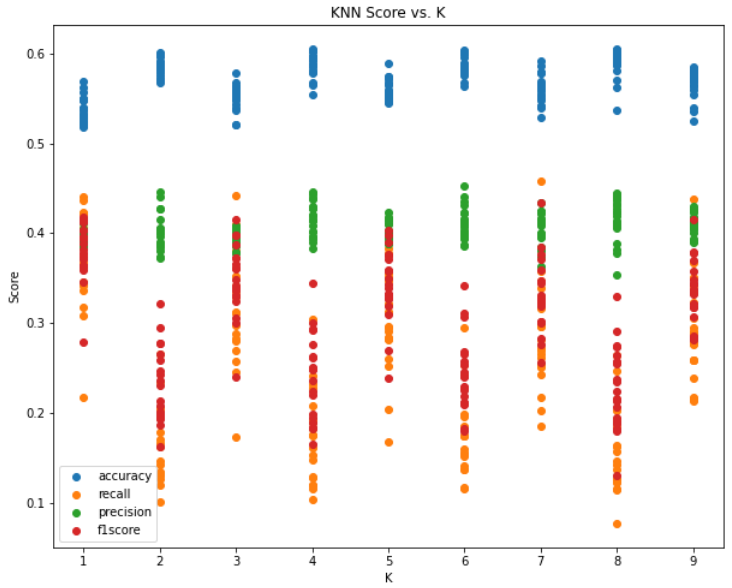
- For each ML pipeline I chose to deploy a model many times with any combination of four or five column choices within the feature set. I also varied the model's deployment with different model determinations. For instance, for KNN I tested models with many feature columns and different K numbers.
- The train_data, test_data, train_labels, test_labels split is the same every time and set using a random seed
- After testing a full range of options, I tracked which variation of the model's deployment provided the highest accuracy score.
- I also tested for recall, precision, and f1 score but these were only analyzed most accurate model was chosen. I weighed these other analyses for the model's effectiveness in comparison to the use of other algorithms.
- Finally I timed the execution of the tests with each algorithm and compared the times of the most accurate models from each algorithm against each other

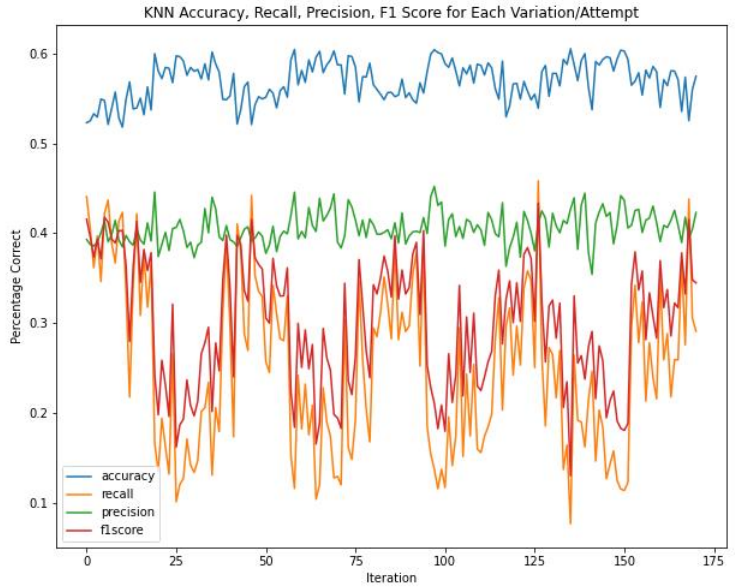
K Nearest Neighbors (KNN)

- Parameters
  - Try K (number of centroids) 1 to 100
  - In 20 loops, try random assortment of 4 or 5 of the feature columns to predict target column
- Indicate model parameters with the highest accuracy, but also create graphs depicting recall, precision, f1 score
- Compare metrics of models in relation to various Ks too
- Track time of execution of model with highest accuracy

Accuracy, Recall, Precision, F1 Score for all Models

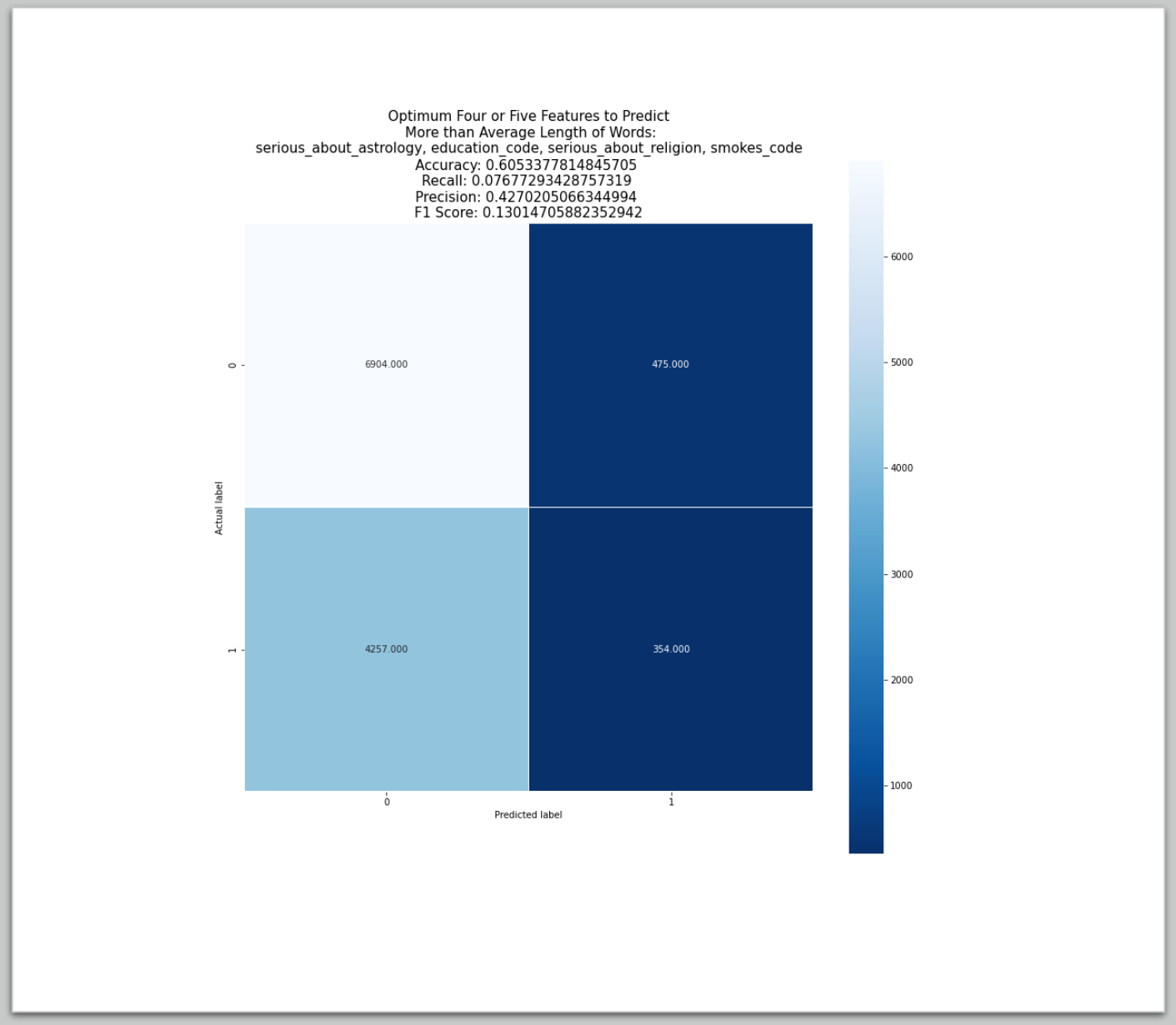
- Accuracy remained over 55% for the models under different parameters but other metrics were low.
- Precision fluctuated around 40%
- Recall and f1 score were highest at around the 5th iteration and 128th iteration while at their lowest around 25th iteration
- Accuracy was highest with a K of 8
- Even though at a K of 8 there was the most accurate findings, the recall was at its lowest





Metrics, Feature Columns for the Optimum Model in Terms of Accuracy, which had a K of 8

- Does it make sense?
- The model tends to fairly evenly correctly or incorrectly predict those who do and those who do not use big words
- Optimum Columns: serious_about_astrology, education_code, serious_about_religion, smokes_code



Further Analysis

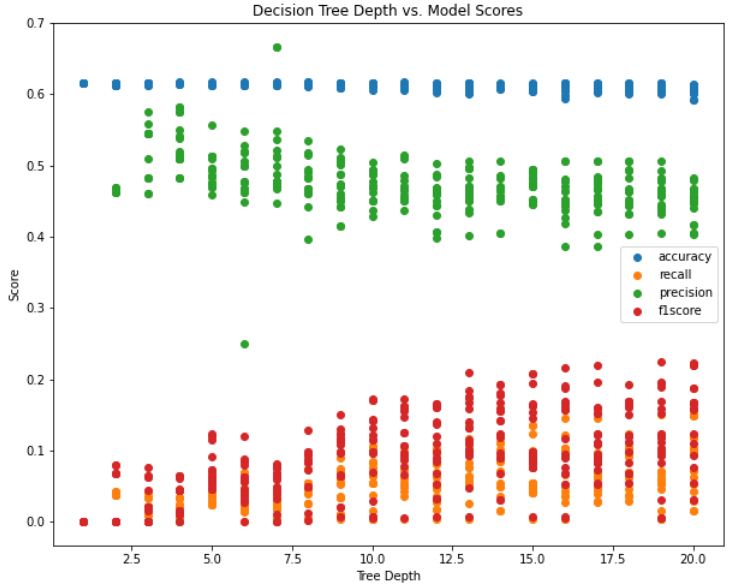
- Time of execution of the most accurate model - 5.267 seconds
  - This may not seem like very long but when you are running models with different parameters and feature column selections this can add up.
  - My complete routine took several minutes
- Accuracy remained high for KNN but other metrics varied

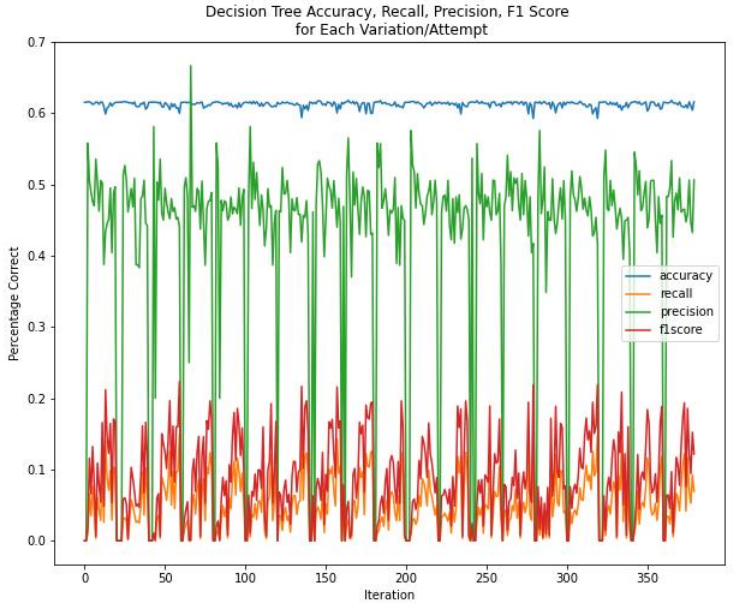
Decision Tree

- Parameters
  - In 20 loops, try tree depths 1 to 20
  - And try random assortment of 4 or 5 of the feature columns to predict target column
- Indicate model parameters with the highest accuracy, but also create graphs depicting recall, precision, f1 score
- Track time of execution of model with highest accuracy

Accuracy, Recall, Precision, F1 Score for all Models

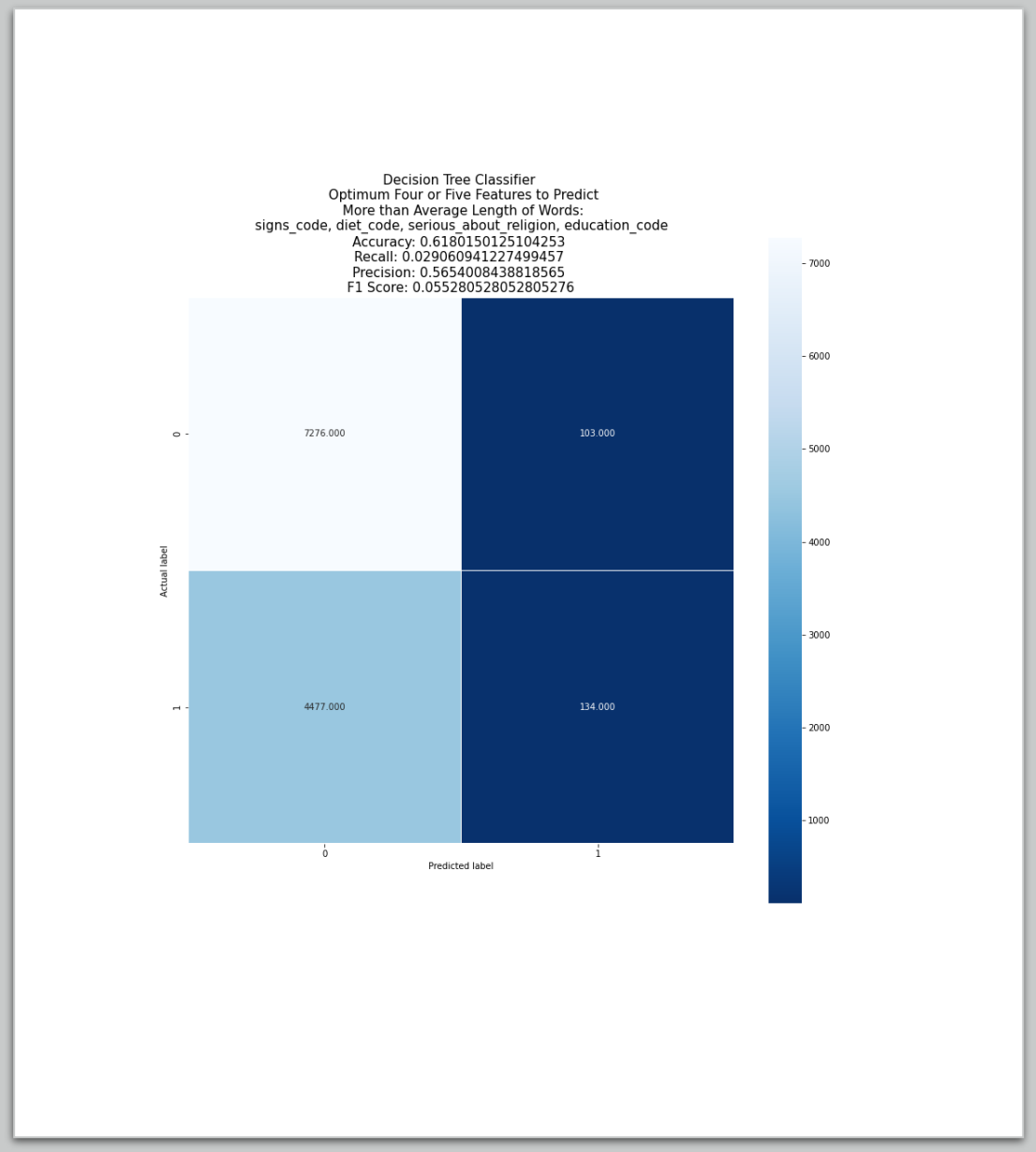
- Accuracy generally is over 60%
- Precision is at one point extremely close to 70% when tree depth was around 7
- Recall and f1 score are extremely low and generally lower than the metrics for KNN
- Tree Depth does not greatly affect accuracy but it has a bearing over recall, precision, and f1 score





Optimum Model in Terms of Accuracy

- Model with the highest accuracy includes a tree depth of 8
- Optimum Columns: signs_code, diet_code, serious_about_religion, education_code
- There are fewer numbers in terms of those who use longer than average words than those actual and predicted with KNN
  - Meaning that there are probably limitations in looking at accuracy for the effectiveness of the model
- Precision is higher for the most accurate model than what we achieved under KNN



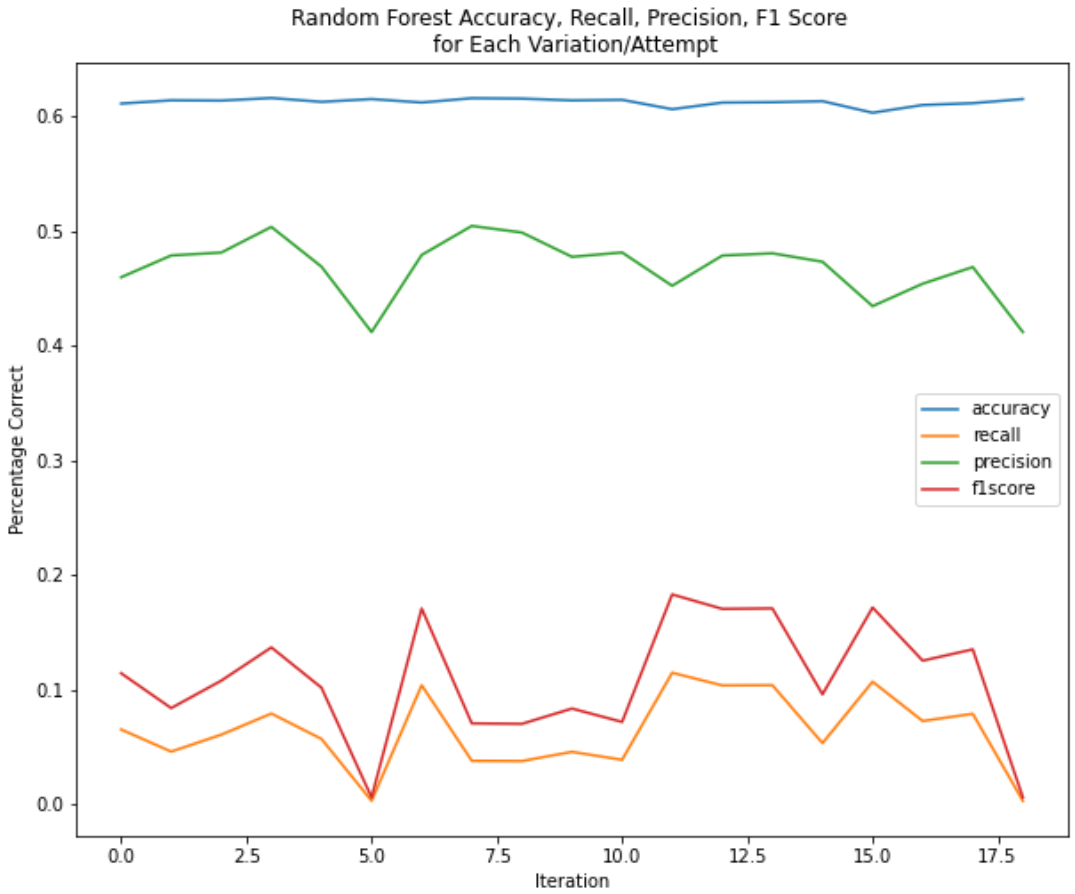
Further Analysis

- Time of execution – 0.111 s
  - Fast time in comparison to KNN

Random Forest

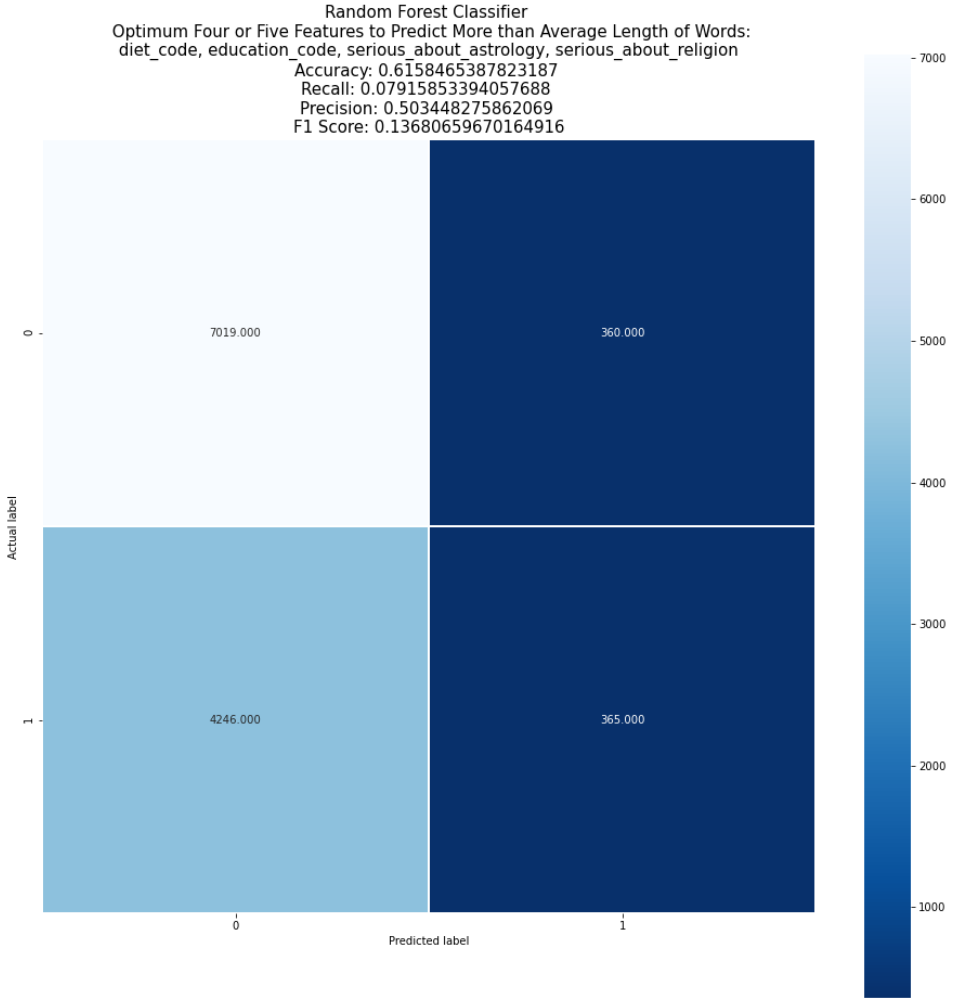
- Parameters
  - In 20 loops, try random assortment of 4 or 5 of the feature columns to predict target column
- Indicate model parameters with the highest accuracy, but also create graphs depicting recall, precision, f1 score
- Identify features' coefficients
  - Importance of random forest features in terms of model
- Track time of execution of model with highest accuracy

Accuracy, Recall, Precision, F1 Score for all Models



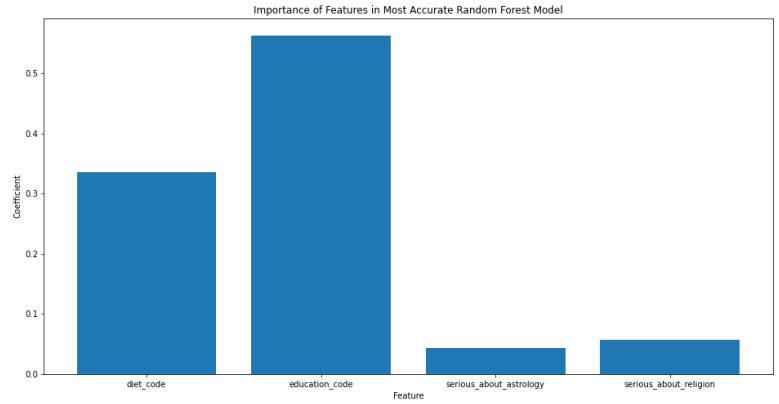
Optimum Model in Terms of Accuracy

- Optimum feature columns:
  - Diet_code, education_code, serious_about_astrology_ serious_about_religion
- Seems to predict as well as KNN in terms of the amount of predicted being close to actual for the uses longer than average words



Further Analysis

- Time of execution – 2.53 seconds
  - Fast time in comparison to KNN but longer time than optimum decision tree model
- Education Code the most important feature to predict target

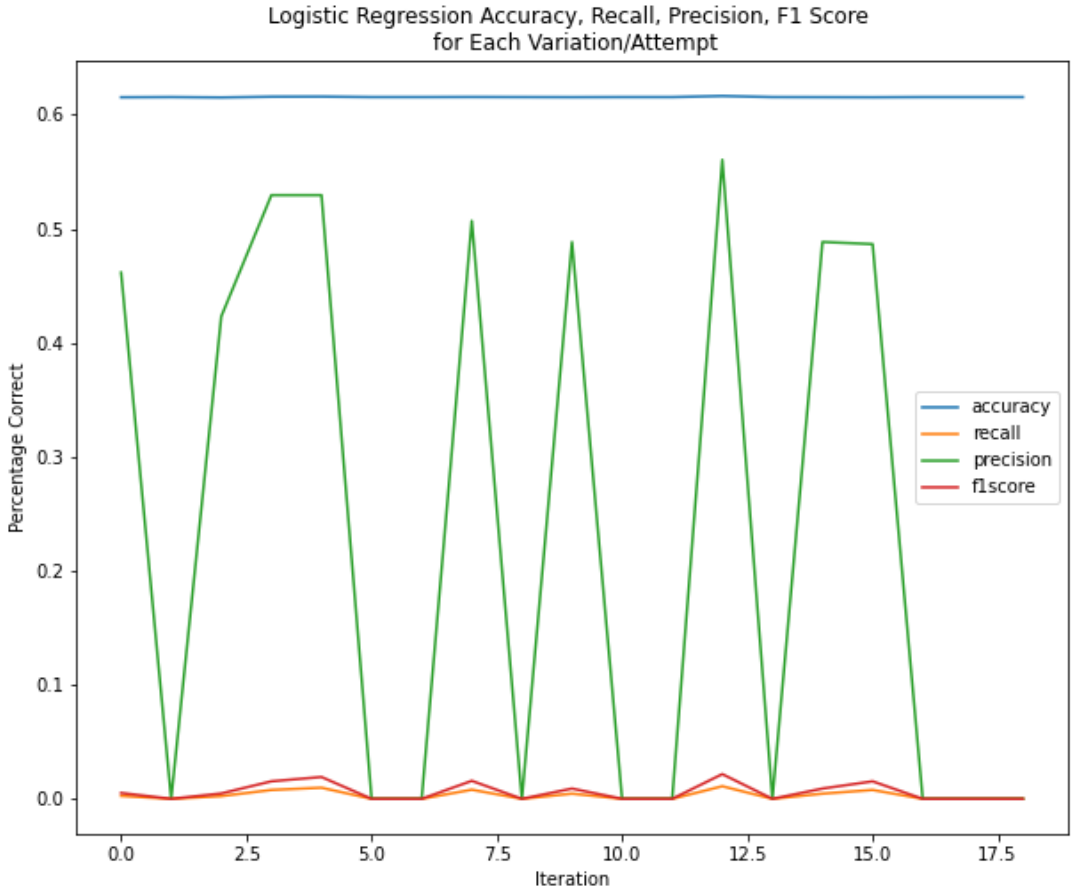


Logarithmic Regression

- Parameters
  - In 20 loops, try random assortment of 4 or 5 of the feature columns to predict target column
- Indicate model parameters with the highest accuracy, but also create graphs depicting recall, precision, f1 score
- Track regression coefficients to determine feature importance in predicting target
  - Regression affords us this technique
- Track time of execution of model with highest accuracy

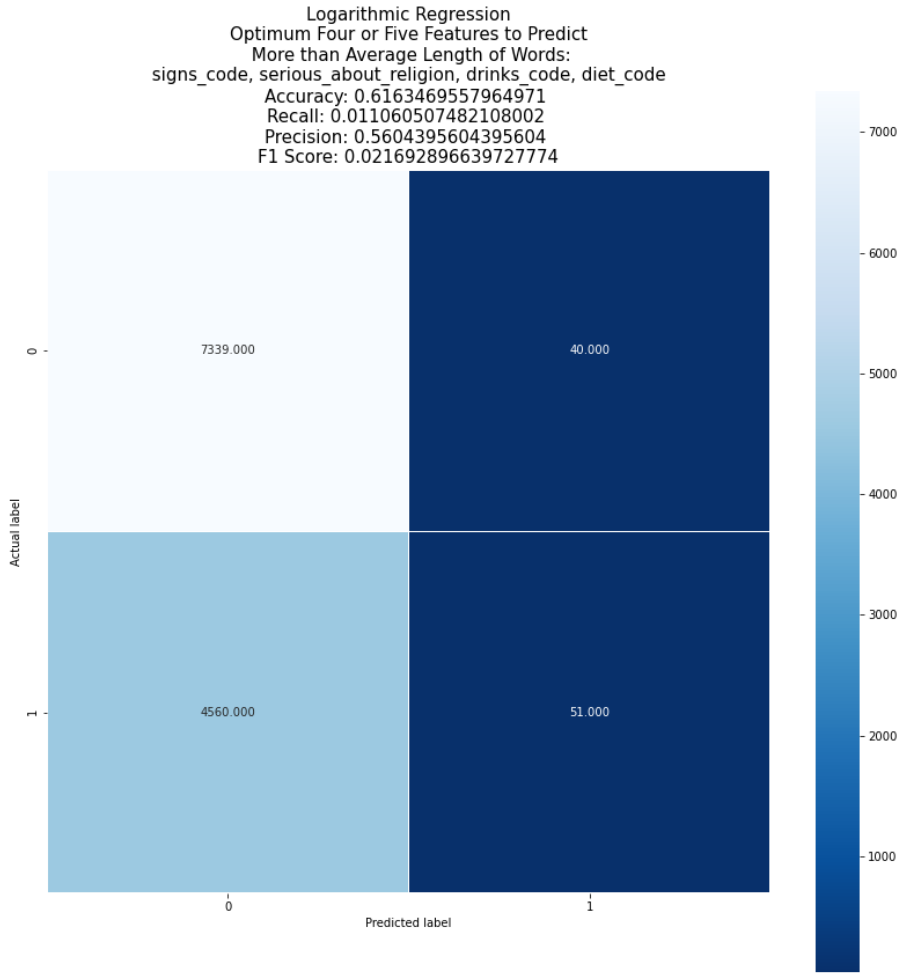
Accuracy, Recall, Precision, F1 Score for all Models

- Accuracy stagnant at about 60%
- While precision fluctuates greatly
- Recall and f1 score are terrible at just over 1%



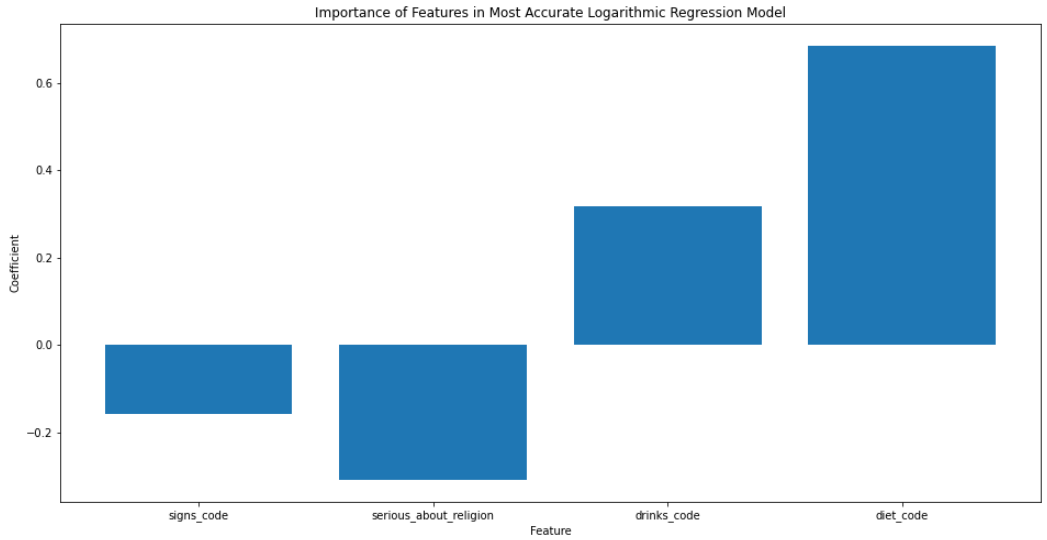
Optimum Model in Terms of Accuracy

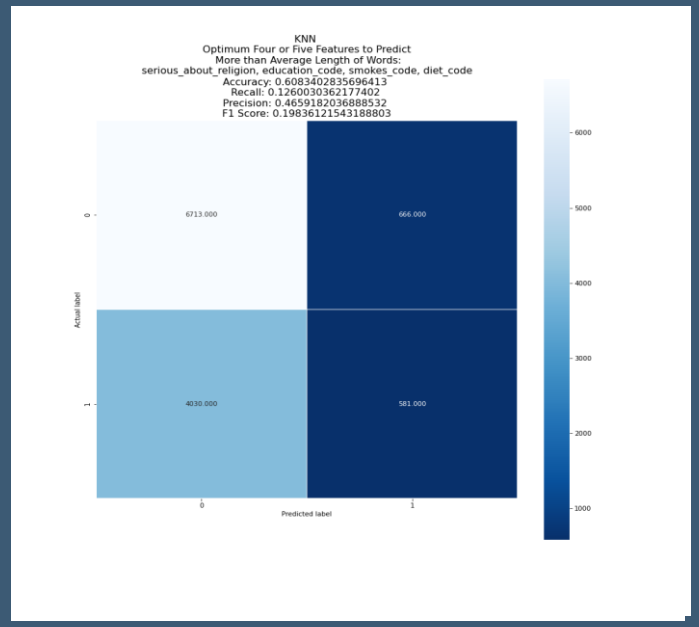
- Optimum feature columns:
  - Signs_code, serious_about_religion, drinks_code, diet_code
- The amount of actual and predicted people who use longer than average words does not make sense which could be contributing to the very low recall and f1 scores

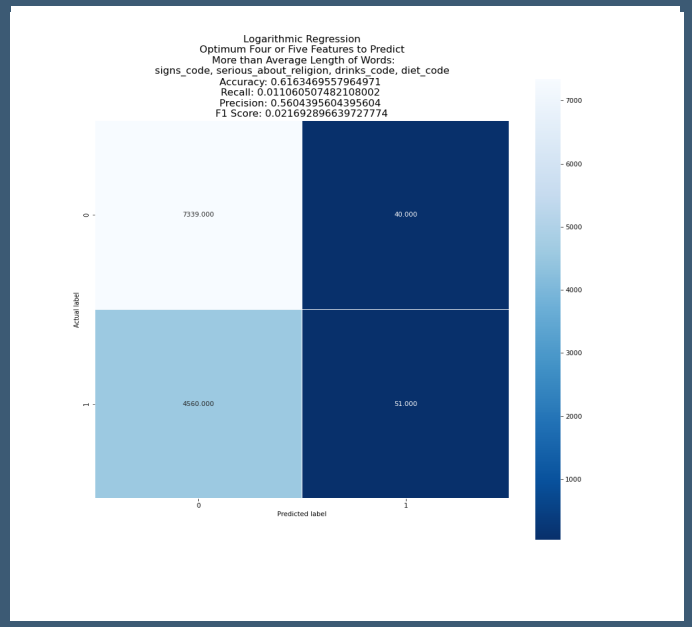


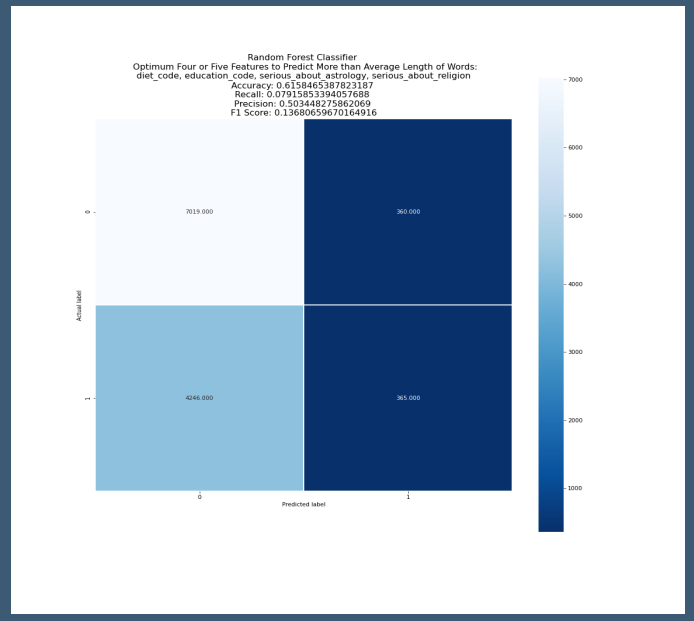
Further Analysis

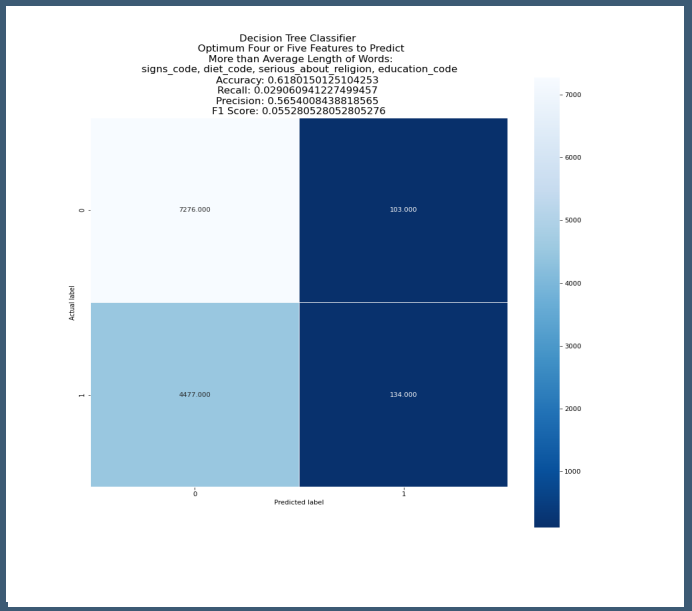
- Time of execution – 0.04 seconds
  - Fastest time in comparison to all other optimum models and ML tests
- Education Code the most important feature to predict target











Compare all Four “Optimum” Models

Conclusions

- None of the models were very effective but the algorithm with the highest recall and f1 scores was KNN
- All algorithms hovered around 60% accuracy but had diminishing returns when evaluating other scores like precision, recall, and f1 score. Most had 50% or so precision and all showed less than 15% recall and f1 score
- KNN took the longest to execute but was the most effective
- The optimum feature column set idea seemed to show that various algorithms disagreed which columns were most important to classify the target column

Conclusions Continued

- I need to make new feature columns to obtain an effective model and redeploy the same algorithms
- I tried SVM for the problem set but it did not execute in a reasonable amount of time. I read that SVM may be limited with feature sets that are as large as this one in rows and columns
- I also tried linear regression but realized that this algorithm would not work for my problem which is classify a binary target. Linear works well for continuous problem sets (predicting a range of numbers). I also tried Bayes classification but ran into issues