Final Project Report

Lu Jiankun
Stanford ID # 006321087

Abstract

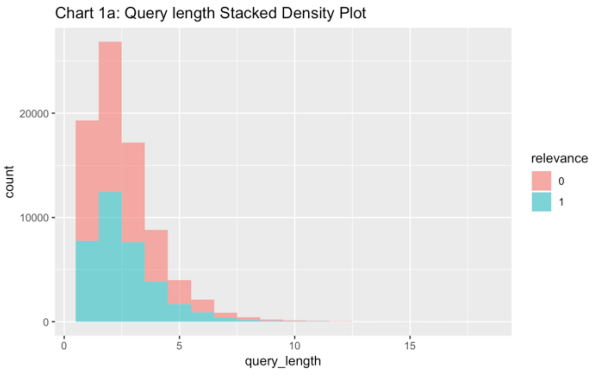
The object of this paper is to explore various data mining tools while finding the best method to make relevant predictions over a set of sample query data. I will consider five different classifiers: logistic regression, k-nearest neighbor, random forest, naïve bayes, and support vector machine. Support vector machine performs best based on my experiments.

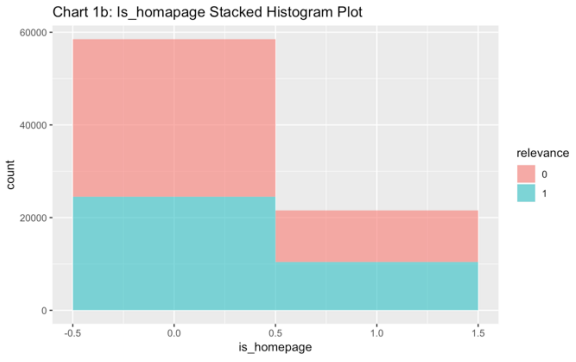
1. Introduction

The goal of this project is to build a model to predict whether a search engine query is relevant. The data provided is from a search engine query and contains URL information. The training set contains 80,046 observations and the testing set contains 30,001 observations. Ten attributes are provided on both datasets, along with query_id and url_id. Query_id and url_id are additional attributes that uniquely identify an observation but have not been deemed with any significance at the outset. The training set contains a variable relevance: 1 for being relevant, 0 for not being relevant. This is the attribute that the machine learning models are to predict.

2. Data Observation

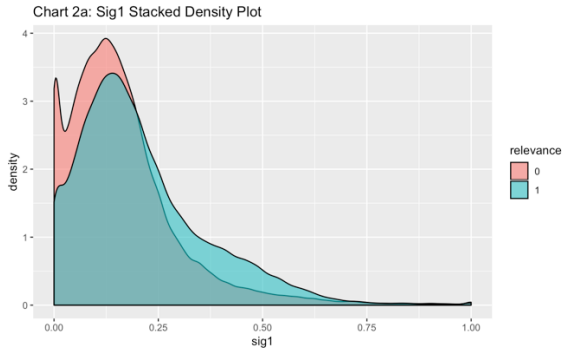
Both training and test data files have been examined and no missing data are found. All the attributes in all the observations are non-negative in both training set and testing set.
The variable query_length contains only integers, suggesting that the variable is a quantitative discrete attribute. Most of the values of query length lie in 1,2,3.
The variable is_homepage appears to be a nominal variable as its value is either 0 or 1.
Below are the stacked histograms for the two variables.

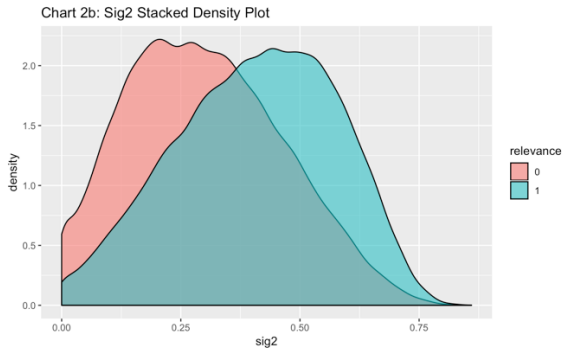


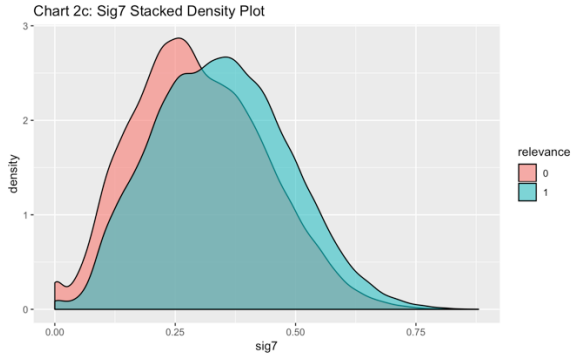


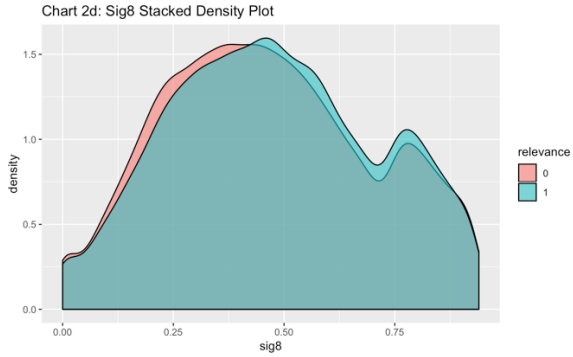
The variables sig1, sig2, sig7, and sig8 take values between 0 and 1. These variables are likely to be quantitative continuous variables. I made stacked density plots for these variables using ggplot2 package, the diagrams are put below. As shown in the diagrams, the four variables are roughly normal distributed. The stacked density plots also tell us the significance of the variables. For example, the distribution of sig2 for observations with

relevance 0 (the red area in chart 2b) is significantly different from that for observations with relevance 1 (the blue area in chart 2b), showing that sig2 is likely to be a strong predictor.

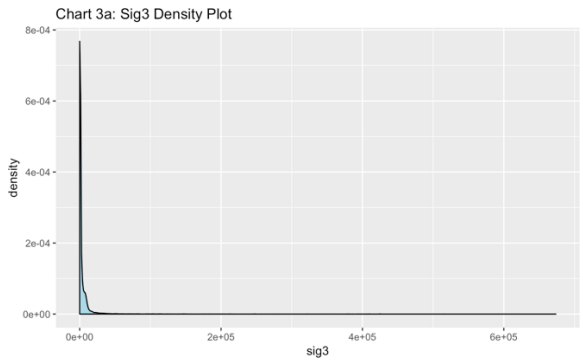


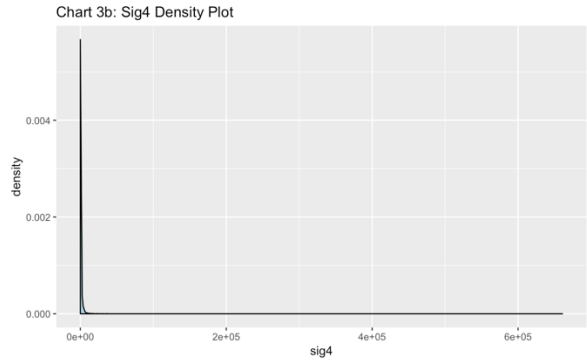


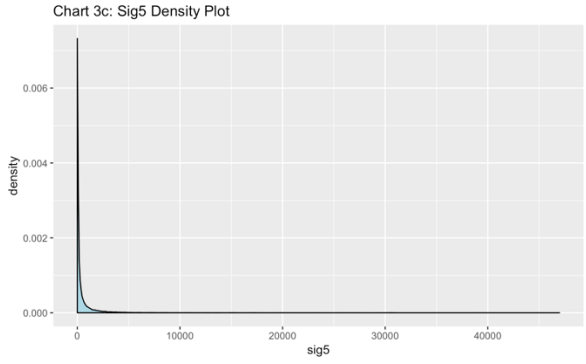


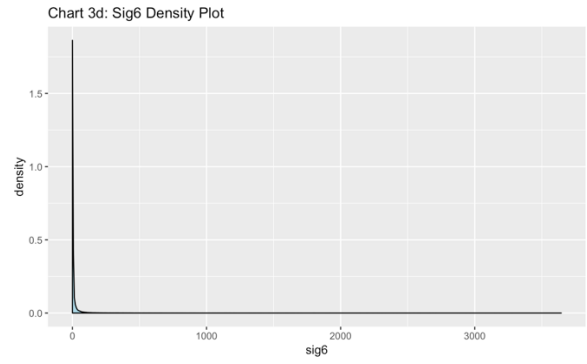


Below are the density plots for sig3, sig4, sig5, sig6. As shown in the diagrams, the distributions for these variables are all concentrated on the far left, but extend a long distance out, making the data unmanageable. In order to achieve accurate prediction, the values of these attributes are transformed using the formula f(x) = ln(1+x). The details are shown in part 3 Data Selection, Preprocessing & Transformation of the report.



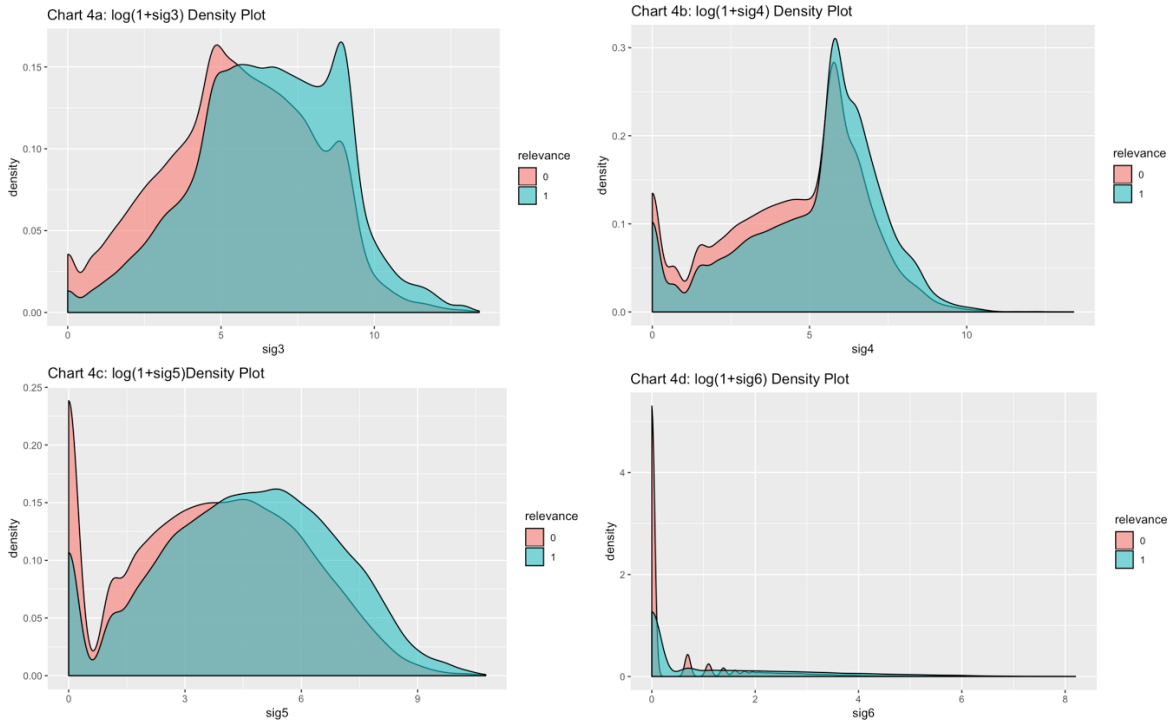




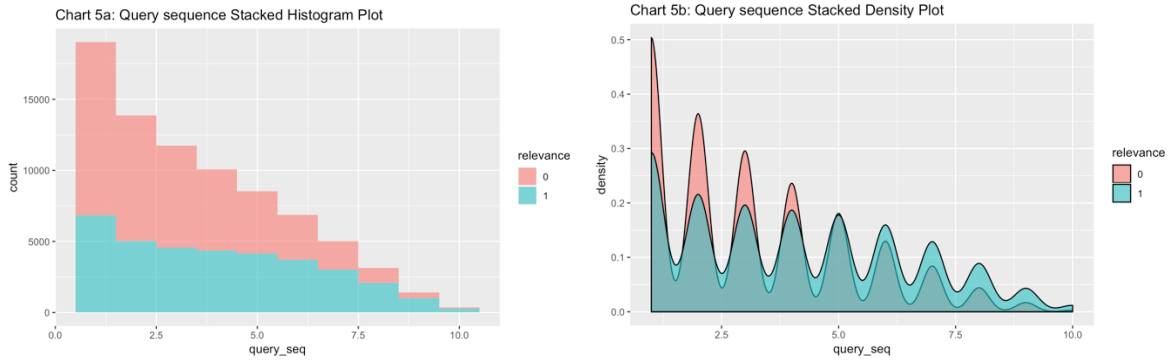


3. Data Selection, Preprocessing & Transformation

As mentioned in part 2, I transformed sig3-sig6 using the formula f(x) = ln(1+x). By doing so, the new distribution is much closer to normal distribution for sig3, sig4 and sig5. For sig6, however, the distribution is still not very ideal. Below are the stacked density plots after transformation。



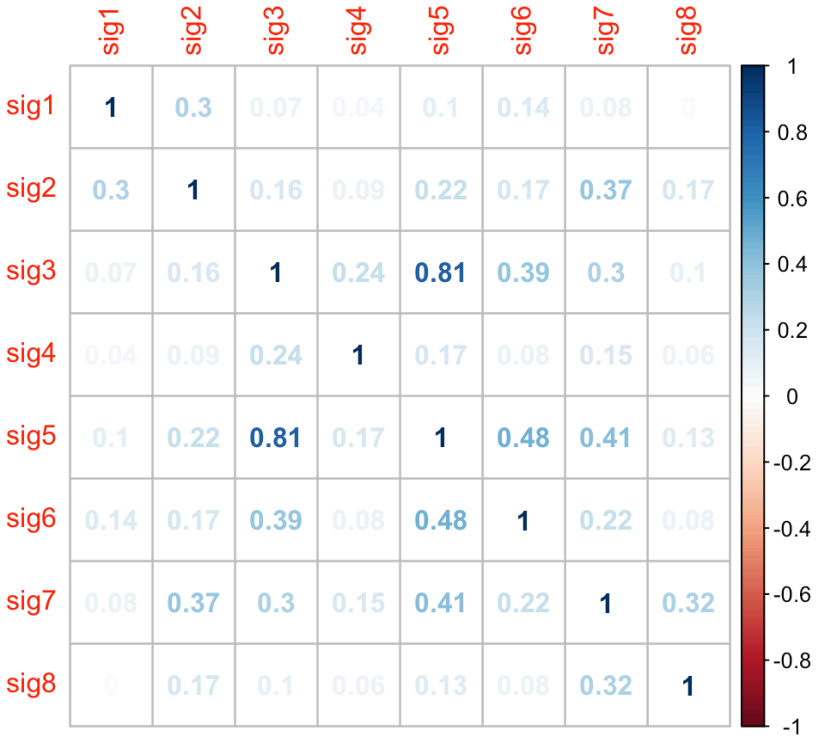
The url_id and query_id are additional information given in the data. I noticed that the url_id have sequential sequences within a query_id, indicating that those are sequential queries, thus may have some predictive value. In order to utilize the hidden information, I created a new attribute named query_seq based on the relevant sequence of url_id within the corresponding query_id. Below are the stacked histogram and density plots for query_seq. As shown in the diagrams, it is obvious that there is a relationship between quert_seq and relevance: relevance is more likely to be 1 when query_seq is large.



Lu Jiankun 006321087

I checked the correlations between variables both before and after the transformation of sig3-sig6 to determine which variables to be used in classifications. I visualize the correlation matrix using the corrplot package as shown below.
Before transformation, sig3 and sig5 appear to have strong correlation. After transformation, the correlations between different attributes become stronger.

Before transformation:



After transformation:

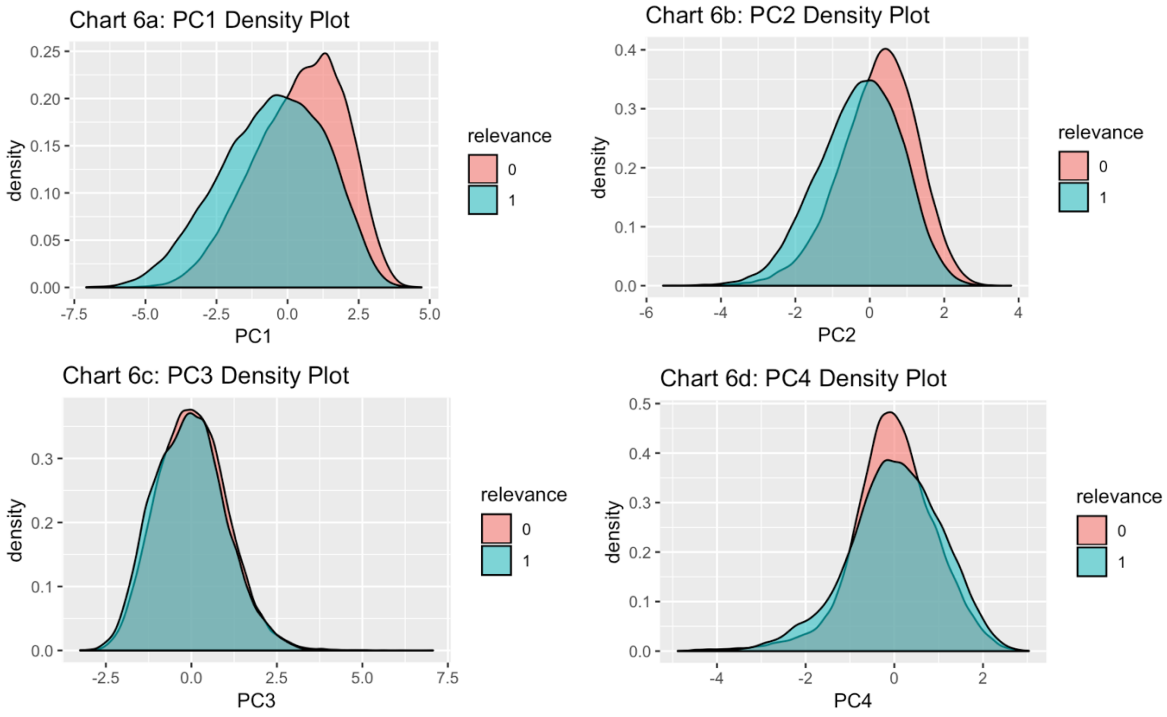
Lu Jiankun 006321087

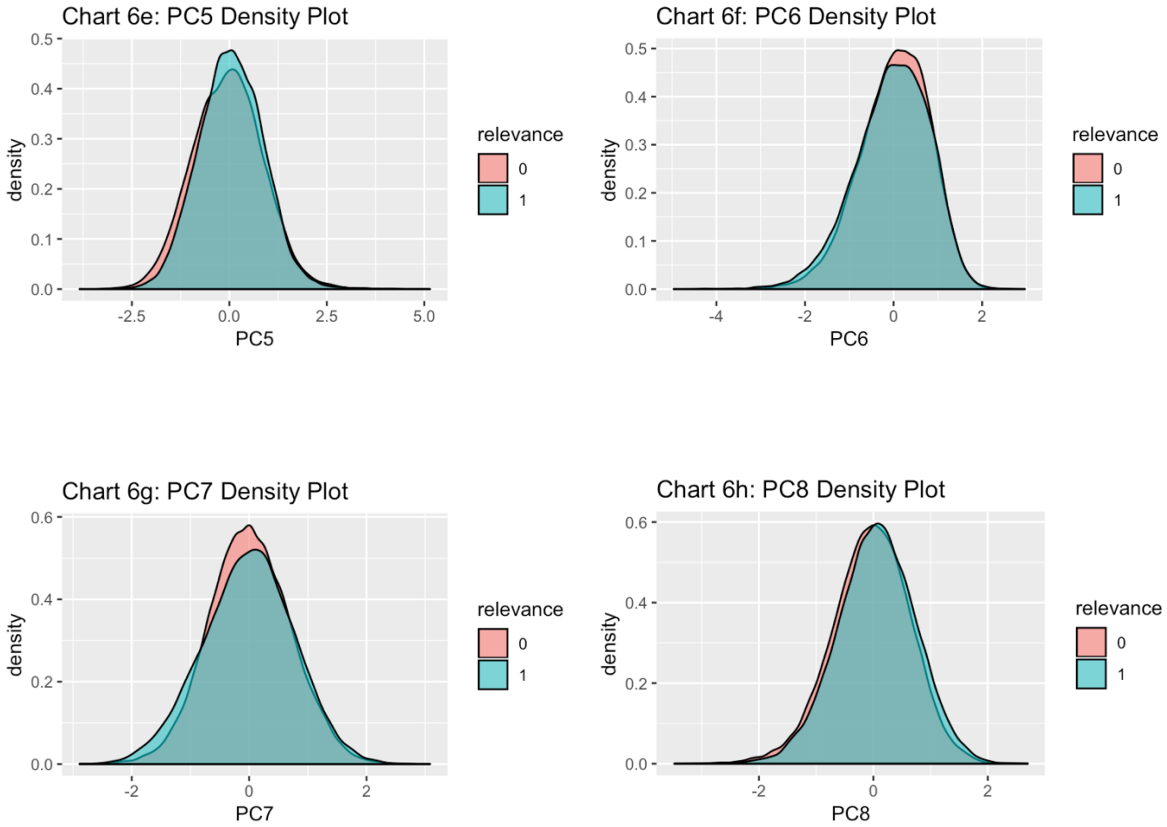
Besides the strong correlations between several attributes, another problem I notice is that the dimensions of different variables are different. These problems can potentially result in inaccurate prediction results. Therefore, I select some of the variables to use and preprocess the data before building the machine learning models.

As sig3 and sig5 are strongly correlated both before and after transformation, I choose to discard one of them. The distribution of sig5 is closer to normal distribution compared with that of sig3 as shown in Chart4 series. Also, sig3 is strongly correlated to sig4 and sig 7 after transformation. Therefore, I decide to discard sig3.
The attributes I choose to use are: is_homepage, sig1, sig2, sig4, sig5, sig6, sig7, sig8 and query_seq (the new attribute I create).

After selecting the attributes, I preprocessed the data. First, I normalized the data so that all the variables have the same dimension. Then I performed principle component analysis to the normalized data to convert the set of correlated variables into a set of values of linearly uncorrelated variables, eliminating the correlation between different variables. Eight new attributes PC1-PC8 are created from the existing attributes after performing principle component analysis. The new created features along with is_homepage, are the nine final features I use to build the models.

Below are the stacked density plots for the new attributes PC1-PC8. As shown in the graphs, they are all very close to normal distribution. I also checked if there is any correlation between the new variables, and it turns out there is no correlation between any of the variables.





4. Solution Evaluations

I tried five models to predict whether a query is relevant. In order to evaluate different models, I split the training data (training.csv) into two sets, one training set and one test set. The training set and test set are spitted randomly using function sample(). 70% data belongs to training set and 30% belongs to test set. Table 1 shows the partitioning of the training set into the working training and testing set. The make-up of relevance is similar in both sets. The evaluations of the models are based on cross-validation error rate (specifically, 5-Fold Cross-validation and 10-Fold Cross-validation).

Table1: Make up of Relevance

Relevance	Total		Train		Test	
	Count	Percent	Count	Percent	Count	Percent
Total	80,046	100.0%	56,032	100.0%	24,014	100.0%
1	34,987	43.7%	24,602	43.9%	10,385	43.2%
0	45,059	56.3%	31,430	56.1%	13,629	56.8%

5. Model Selection

I tested five models to determine which one had the best fit. I consider nine of the ten attributes provided (all but sig3) as well as the constructed variable query_seq. Table 2 gives the error rates for each of the models after determining the best fit for that model.

Table 2: Model Errors

Model	Training Error	Test Error
Naïve Bayes	35.33%	35.40%
Logistic Regression	33.85%	34.11%
Random Forest	0.00%	33.99%
K-Nearest Neighbor (K = 105)	32.54%	33.66%
Support Vector Machine	32.45%	33.14%

(a) Naïve Bayes

Naïve Bayes classifier is a simple and straightforward classifier that can deal with both numerical and categorical attributes. In this project, I used the function naïve Bayes in e1071 package. As Naïve Bayes classifier perform well when the features are independent of each other, I performed principle component analysis to the data, decreasing the error rate by 1.2% to 35.4%. However, the test error is still higher than the other models.

(b) Logistic Regression

The logistic regression model is used to estimate the probability of a binary response based on one or more predictor independent variables. The model itself simply models probability of output in terms of input and does not perform statistical classification. It can be used to make a classifier by choosing a cutoff value and classifying inputs with probability greater than the cutoff as one class, below the cutoff as the other. In this project, I chose 0.5 as the cutoff value and apply 5-Fold cross validation to the training data to build the best model. The best logistic regression model gives test error rate of 34.11%.

(c) Random Forest

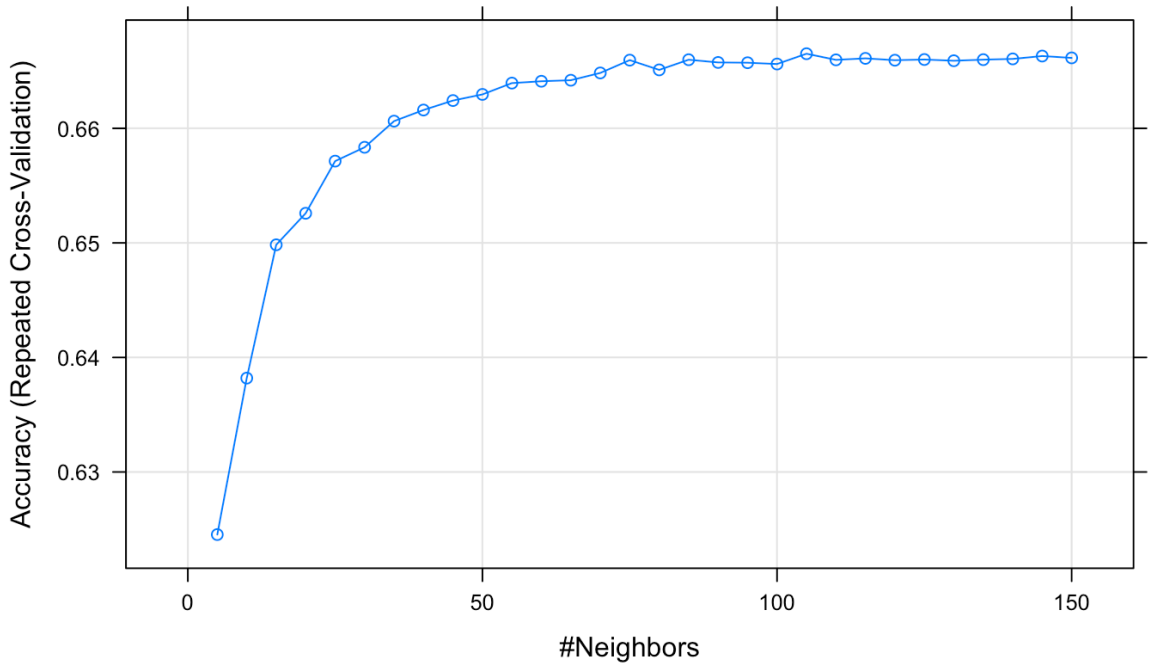
Random Forest classifier is a very useful ensemble method based on Decision Tree classifier. It randomly chooses a subset of the available attributes and runs a decision tree classifier. This is done over some predetermined set of iterations, and the combinations of all the decision tree classifiers are considered when determining the predicted value. I used the randomForest() model from the randomForest package.

As I mentioned before, I use 9 features to build the model. Therefore, I set mtry = $\sqrt{9} = 3$ and ntrees = 1000, which means I use 1000 decision trees and consider 3 variables at each split. 5-Fold cross validation is applied to build the best model. The training error is 0, and the testing error is 33.99%.

(d) K-Nearest Neighbors

K-nearest neighbors is another useful classifier. It counts the number of cases where relevance is 1 and the cases where relevance is 0 at a given point, and then assigns the value with a higher frequency to that point. I use knn3() from package caret to implement K-Nearest Neighbors classifier.

The key issue is to define distance and scale variables. If the features have different dimension, the result of K-Nearest Neighbors classifier could be inaccurate. Therefore, I normalized all the data when preprocessing the data to improve the results.
As it is not clear what value k should take before running the model, I test all the values of multiple of 5 from 0 to 150. The accuracy is evaluated using 10-Fold cross-validation. The minimum cross-validation error occurs at k=105, with an error rate of 33.66%. Below is a graph showing the trends of the accuracy as the value of k increases.



(e) Support Vector Machine

Support vector machine (SVM) tries to find a hyperplane with the largest margin between the prediction classifiers in the training set. Margin is the distance between two parallel lines where each line partitions off one set of the prediction value. It faces the same problem as K-nearest Neighbor: defining distance and scaling variables. Therefore, the data is normalized when preprocessing for the same reason.
I use svm() in e1071 package to implement support vector machine classifier. First, I tried both linear kernel and radial kernel with cost = 0.01. The radial kernel gives better performance as shown in table3. Therefore, I choose to use radial kernel.

Table3: SVM error for different kernals

Model	Error
Radial Kernal	33.75%
Linear Kernal	33.91%

As it is not clear what value cost should take before running the model, I test several different costs from 0.01 to 10. The error is evaluated using 10-Fold cross-validation. The results are shown in table4 and the diagram below. The minimum cross-validation error occurs when cost is 3.162. The best model gives a test error of 33.14%. This is also the final model I choose to predict the test set (the given test.csv).

Table4: SVM error for different costs.

Cost	Error
0.01	33.73%
0.0316	33.54%
0.1	33.40%
0.3162	33.23%
1	33.11%
3.162	33.03%
10	33.12%