Stats 202 Project Report
Abhiram Madenur Venkatesha
ID: 06236990
amadenur@stanford.edu

Introduction:

The aim of this project is to return the relevant URLs for a given set of information on the queries within the search engine. The objective of the project is to return the relevance of the URLs based on the attributes in the dataset. Five different classifiers are used: Naïve Bayes, Decision Tree, K - Nearest Neighbors, Random Forest and Gradient Boosting. Amongst these Gradient boosting performs the best.

Exploratory Data Analysis:

Information on datasets:

The training dataset has 80046 observations with 13 attributes. The given test dataset has 30001 observations with 12 attributes. The first attribute is the query_id for a given search query. There are 12408 unique query_ids. The second attribute is the url_id, which is mostly unique (over 94% of records have a unique value) for the given query_ids. The attributes: query_id and url_id are categorical data. The third attribute is the length of the query, represented by query_length. This attribute presumably represents the number of terms in the search query. The query_length is a numerical attribute with values ranging from 1 to 18. The next attribute is is_homepage. The is_homepage attribute represents if a user has set the search page as a homepage or not. It is a binary attribute with values of 0 or 1. The 8 attributes following is_homepage, sig1 through sig8 are “signals” of user who has input the search query. Although their functionality is not explained, I presume that they represent the user’s search history, geo-location etc. The final attribute is a binary attribute that signifies if a url based on the search query is relevant or not. “relevance” attribute takes on the values 0 or 1. This is the attribute that has to be predicted.

The test dataset is similar to the training dataset, except that it does not have the relevance attribute. The model with the lowest misclassification error has to predict the relevance on the test dataset.

Analysis of attributes:

Each attribute has been analyzed and the basic statistical values of maximum value, minimum value, mean, median etc. has been summarized in Table 1.

Table 1

	query_length	sig1	sig2	sig3	sig4	sig5	sig6	sig7	sig8
Missing Values	0	0	0	0	0	0	0	0	0
Minimum	1	0	0	0	0	0	0	0	0
Maximum	18	1	0.86	673637	660939	46994	3645	0.88	0.94
1st Quartile	2	0.08	0.21	78	24	10	0	0.22	0.29
3rd Quartile	3	0.24	0.48	2537.75	591	336	2	0.42	0.64
Mean	2.585826	0.18324	0.346947	4857.078555	742.316256	550.527597	14.099155	0.319464	0.471846
Median	2	0.15	0.34	417	220	64	0	0.31	0.46
Variance	2.31677	0.021713	0.029772	553753762.676795	23216584.66264	3564294.667131	8112.321444	0.019224	0.053502
Standard Deviation	1.522094	0.147354	0.172545	23531.9732	4818.359126	1887.933968	90.068426	0.138651	0.231306

The queries from home pages appear to have a higher number of relevant URLs (48.6%) than the queries not from homepages (41.9%).

The proportion of the relevance of the queries seems to be higher if the is_homepage attribute is 1 (48.6%). Fig. 1 in appendix A shows the proportion of the relevant urls based on the is_homepage attribute.

The query_length, presumably representing the number of terms in the search query varies between 1 and 18. The number of relevant URLs appears to be the highest for the query_length value of 2. Fig. 2 in appendix A shows the proportion of relevant urls based the length of the search query (query_length).

Fig. 3 in appendix A shows the histogram of sig1-sig8 attributes. Four signal attributes, sig3-sig8 show a strong right skew. Prior to running classification models, logging these attributes helped achieve a normal distribution of the attribute data. Fig. 4 in appendix A shows the attributes sig3-sig6 after being logarithmically transformed. (Note: some attribute values of 0 were replaced by 0.001 in order to avoid the log. value to be undefined).

The correlation between all the attributes and the relevance is calculated. Fig. 5 in appendix A shows the correlation values in a pictorial grid representation. The correlation values are calculated before and after log. transformation is applied. sig3 and sig5 appear to have a bit of a correlation on the ranges of 0.75 to 0.8, but is not significant enough to be eliminated from consideration during building the classification.

Maximum value, minimum value, median value and mean value aggregation was performed on the signal attributes based on query_id and the correlations were re-calculated. The table below shows the summary of the correlations of the aggregated signal attributes based on the query_id.

	original value	Max per query_id	Min per query_id	Median per query_id	Mean per query_id
query_length	-0.00005431	-0.00005431	-0.00005431	-0.00005431	-0.00005431
is_homepage	0.0602881	0.0269629	0.02051457	0.00549458	0.01817045
sig1	0.16019789	-0.00570228	0.02599883	0.03311323	0.0228113
sig2	0.30581508	0.11946646	0.13694133	0.16077294	0.17057616
sig3	0.07274379	0.02448695	0.05003876	0.04491426	0.03437706
sig4	0.03425501	0.00988298	0.04032274	0.01875498	0.0180462
sig5	0.10322486	0.02844414	0.05055365	0.05915561	0.04528149
sig6	0.12451341	0.02026424	0.04411823	0.05842145	0.04081394
sig7	0.16514143	0.06472197	0.11874826	0.10455677	0.10584065
sig8	0.02873055	0.01310363	0.06413252	0.03939868	0.0445128

Looking at the values above, the correlation between aggregated minimum of signal 8 and relevance increases to 0.064 from 0.028. Therefore, sig8 attribute in the dataset is replaced by the respective aggregated minimum value.

The type of each of the attribute is shown in the table below:

Field	Data Type
query_length	integer
is_homepage	factor
url_id	integer
sig1 - sig8 (min. sig8)	numeric
relevance	factor

Classification Models:

a) Naïve Bayes Model:

The Gaussian Naive Bayes model was constructed using the sklearn.naive_bayes package in Python. Gaussian NB model was handled with default parameters. A 10 fold cross validation evaluation estimated a 40% misclassification error.

b) K-Nearest Neighbor Model:

We've used KNN from sklearn.neighbors package with KNeighborsClassifier class in python.We've used 10 fold cross validation. For each row of the test set, the k nearest training set vectors are found (in Euclidean distance), with ties broken at random and the classification is decided by majority vote. For this data set we did not expect optimal results as nearest neighbour works poorly with high dimensionality. The results obtained validated this.

To construct the K-Nearest Neighbors we used the KNeighborsClassifier class imported from the sklearn.neighbors library in python. For each row of the test set, the k nearest training set vectors are found (in Euclidean distance), with ties broken at random and the classification is decided by majority vote. For this data set we did not expect optimal results as nearest neighbor works poorly with high dimensionality. A 10 fold cross validation was used for evaluation. Without scaling, and the K value being 207 the misclassification rate is 0.509.The figure below shows the misclassification rate without scaling.

The K value for the lowest result of a misclassification of 0.509 was 207. Fig.1 in appendix B shows how the misclassification error varies as K increases from 200 to 250.

The model performed similar with scaling as well. The K value for the lowest result of a misclassification of 0.509 was 207 even with scaling. Fig. 2 in appendix B shows how the misclassification error varies as K increases from 200 to 250, with scaled data.

c) Decision Trees

The tree package from the sklearn library was used to implement Decision Trees. 'gini' criterion was chosen. A 10 fold cross validation evaluation was performed. Fig. 3 in appendix B shows how the misclassification error varies as depth of the tree increases. The lowest misclassification error was found to be 0.346 with a depth of 6.

d) Random Forest

Random Forest is an ensemble method taking a combination of tree predictors, such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest.

The ensemble library in sklearn package is used, in which the RandomForestClassifier class is present. The evaluation of the model was done using 10 fold cross validation. The table below shows the misclassification rates for the given range of estimators. The lowest misclassification error was found to be 0.345 for a random forest classification model of 2500 trees.

Number of Estimators	**Misclassification Rates**
500	0.35630789
1000	0.35339685
1500	0.35097318
2000	0.34944899
2500	0.34891191

e) Gradient Boosting

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. The GradientBoostingClassifier class was used from the sklearn.ensemble package.

This was the best classifier amongst all the other classifiers used above with a misclassification error rate as 0.342 with 41 estimators for gradient boosting. Fig. 4 in appendix B shows how the misclassification error varies as the n_estimators increases.

Conclusion:

The table below summarizes the misclassification errors for different classification models.

Classifier	**Parameters**	**Misclassification**
Naive Bayes	Default parameters	0.40
Nearest Neighbour	250 neighbours with scaling	0.509
Decision Tree	Max depth = 6, Criterion= 'gini'	0.346
Random Forest	2500 trees	0.3489
Gradient Boosting	41 estimators	0.342

Due to computational limitations the SVM models (linear and gaussian) could not be run. Also due to time constraints generated by each of the 5 models taking a long time to run, a good amount of tuning could not be achieved to improve the model's accuracies. The best classification model was gradient boosting and it had an error rate of 0.342. The predictions are submitted in a .txt and a .csv file separately. Both files have the same content.

Acknowledgement:

I would like acknowledge working with Manisha Sudhir (manisha4@stanford.edu, ID: 06236246) throughout the project.

References:

[1]: "Relevance Ranking for Predicting Web Search Results" by Pedro M. Teixeira (Doctoral Program in Informatics Engineering, Faculdade de Engenharia da Universidade do Porto pro11007@fe.up.pt)

[2]: Pandas package Documentation - https://pandas.pydata.org/pandas-docs/stable/

[3]: Numpy & Scipy Documentation - https://docs.scipy.org/doc/

[4]: scikit-learn Documentation - http://scikit-learn.org/stable/documentation.html

[5]: Matplotlib Documentation - https://matplotlib.org/contents.html

Appendix A

Fig. 1 (below) shows the proportion of the relevant urls based on the is_homepage attribute:

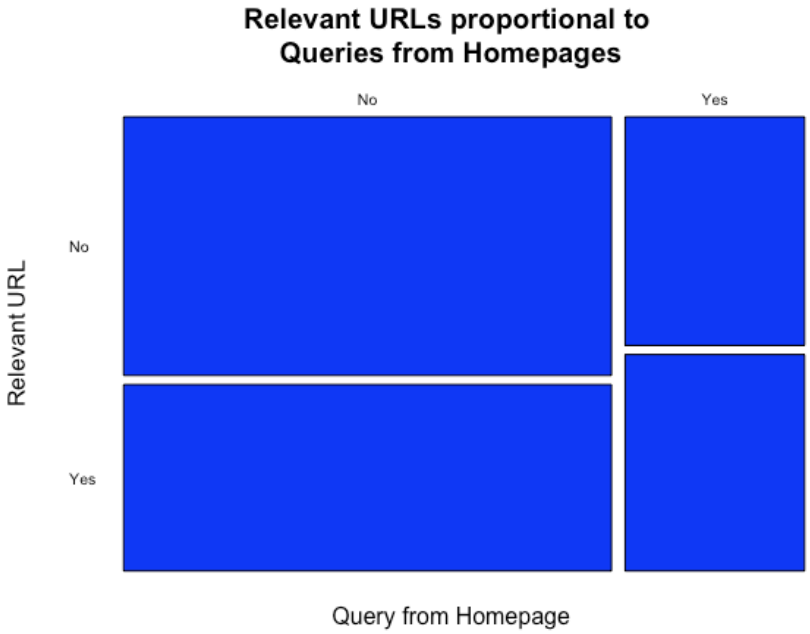


Fig. 2 (below) shows the proportion of relevant urls based the length of the search query (query_length):

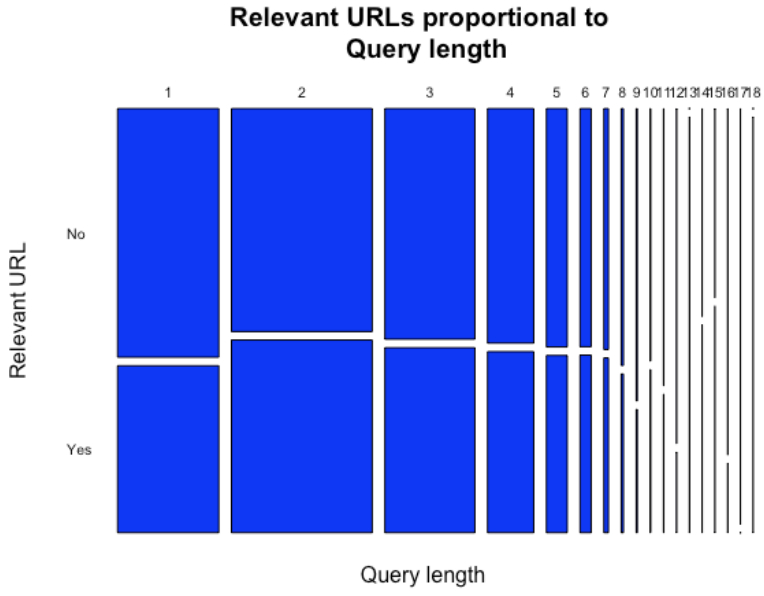


Fig. 3 (below) shows the histogram of sig1-sig8 attributes:

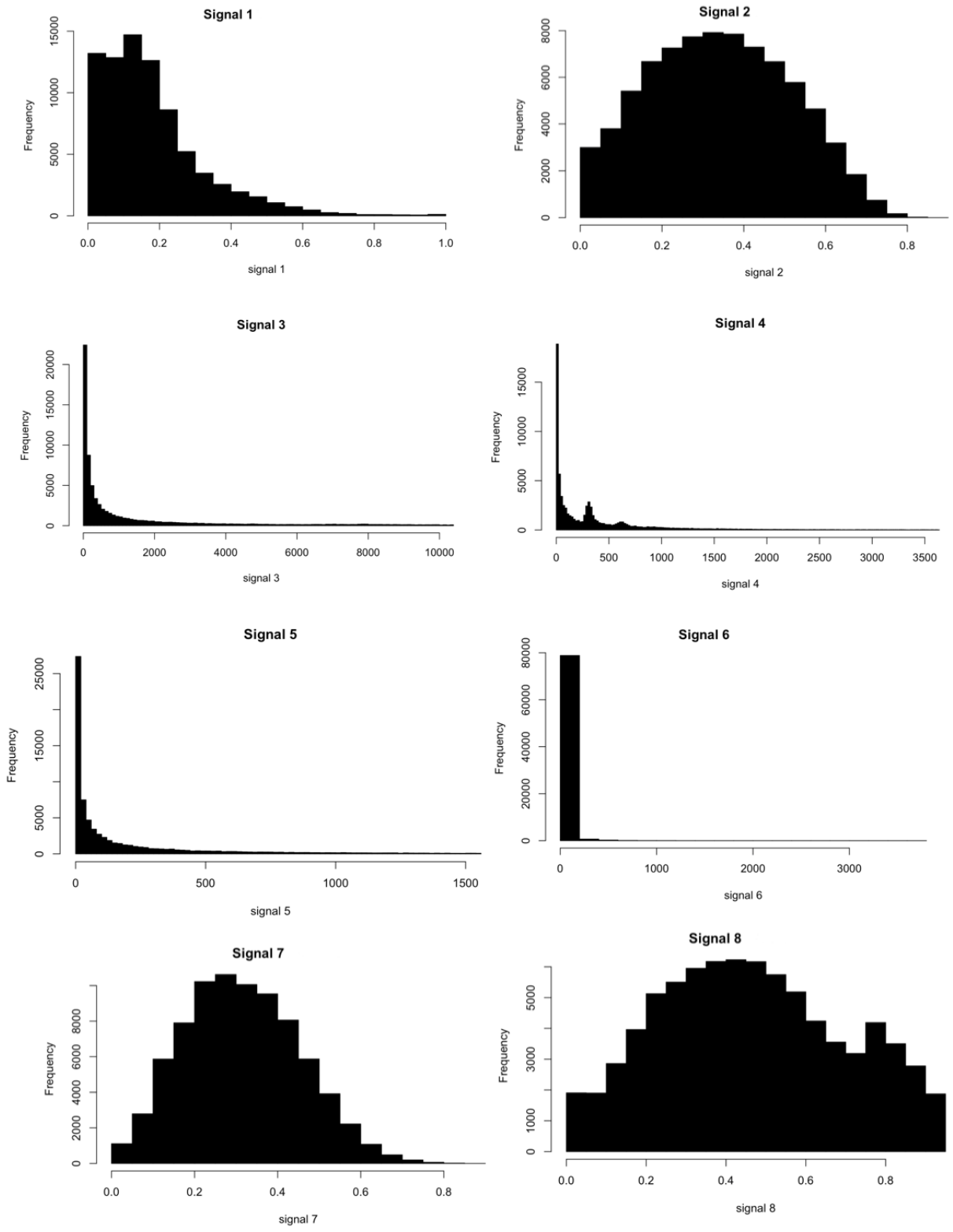


Fig. 4 (below) shows the attributes sig3-sig6 after being logarithmically transformed:

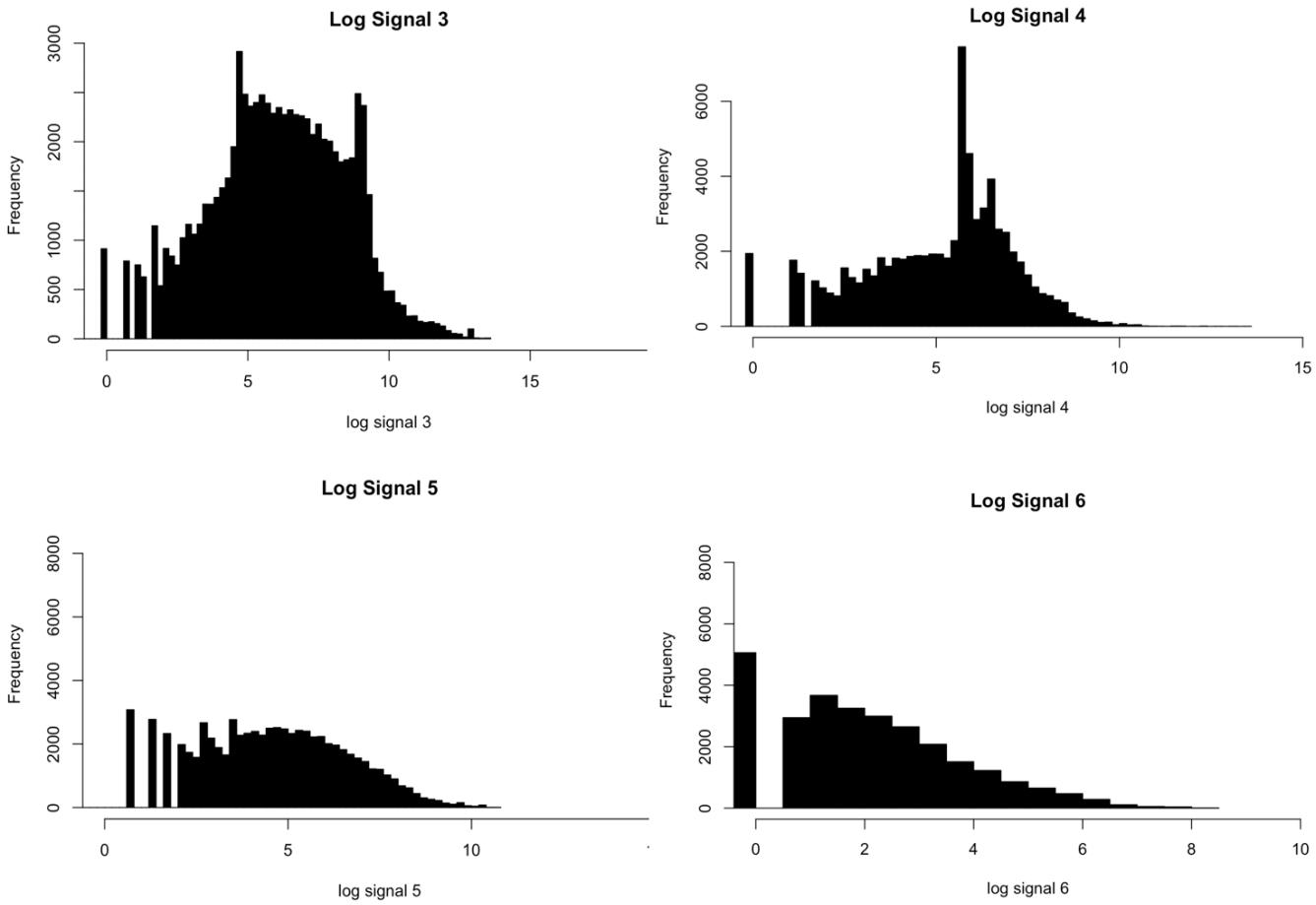
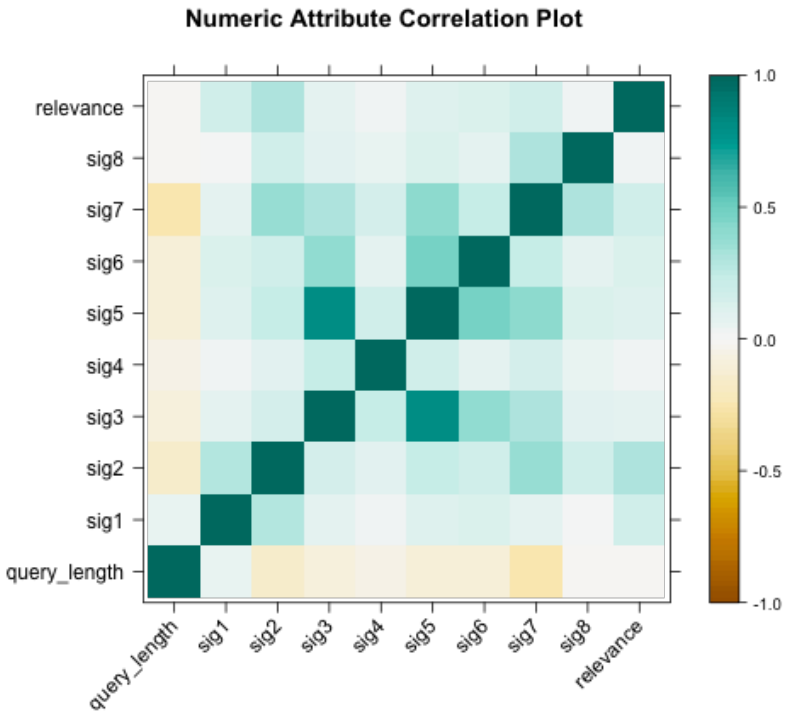
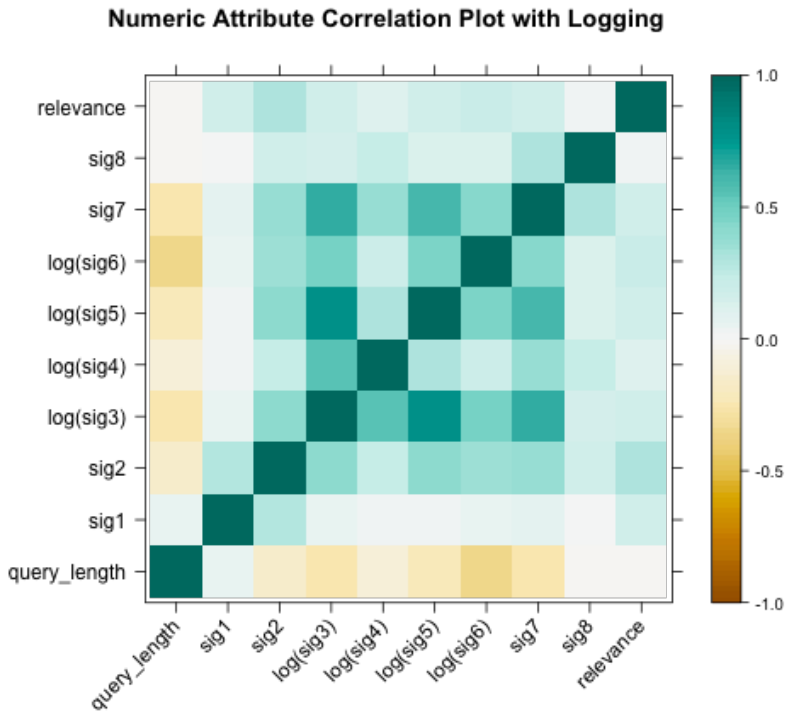


Fig. 5 (below) shows the correlation values in a pictorial grid representation:





Appendix B

Fig.1 (below) shows how the misclassification error varies as K increases from 200 to 250:

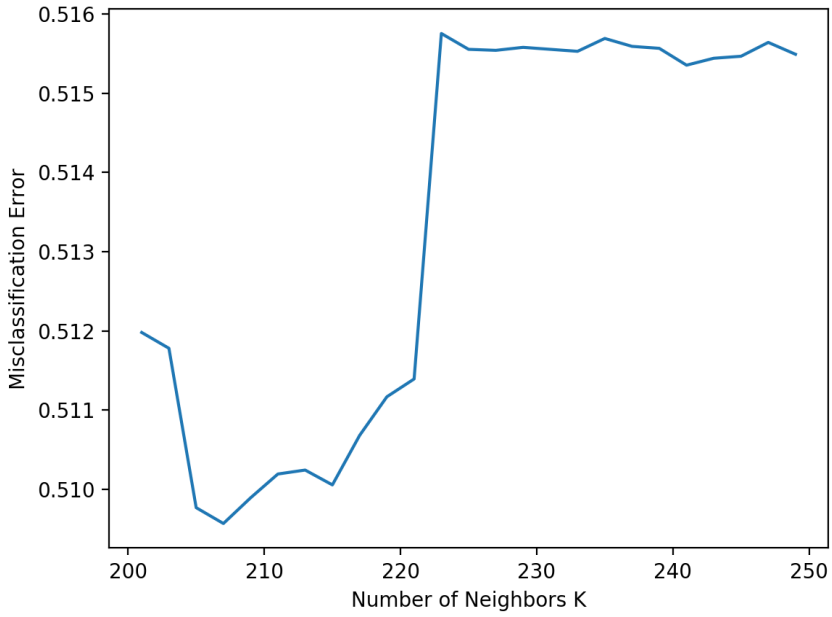


Fig. 2 (below) shows how the misclassification error varies as K increases from 200 to 250, with scaled data:

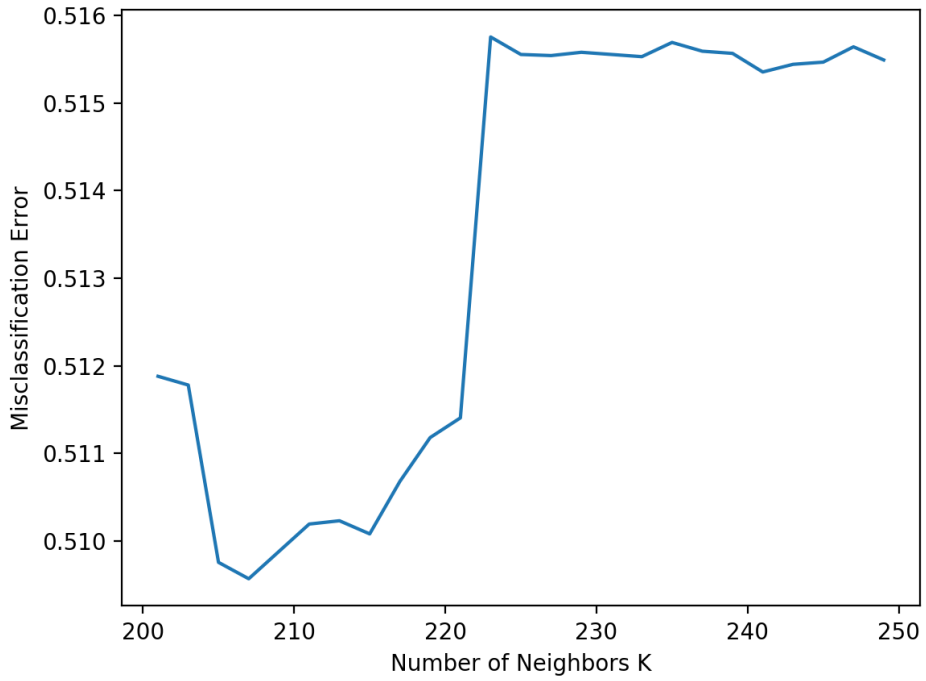


Fig. 3 (below) shows how the misclassification error varies as depth of the tree increases:

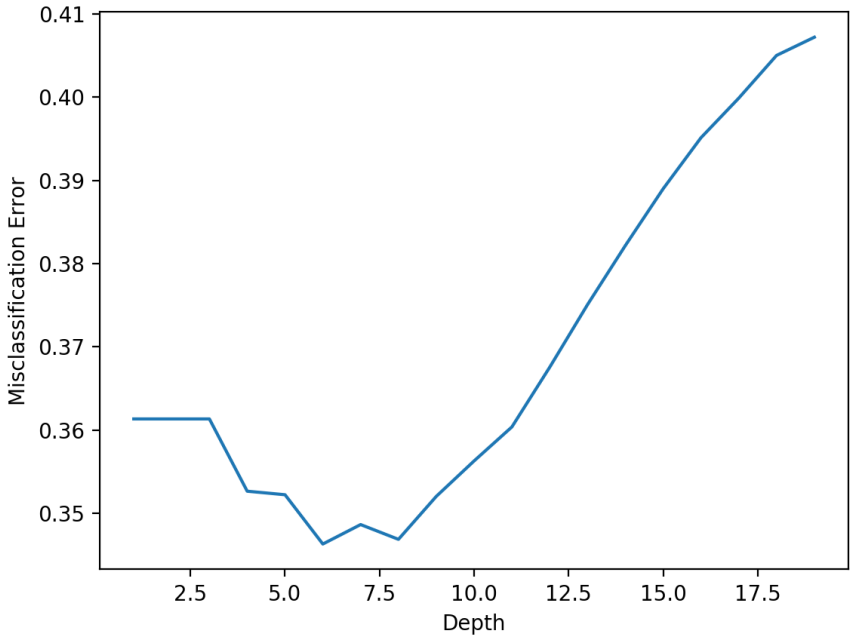


Fig. 4 (below) shows how the misclassification error varies as the n_estimators increases: