Data Mining Final Project

Table of Contents:

List of Tables:

List of Figures:

1) Project Background

Today, machines are rapidly catching up to and in some cases even exceeding humans in several facets, raw computing power being the most prominent aspect. Computers can crunch numbers and solve mathematical problems much faster than human beings. Despite these technology advancements, there is still a particular skillset which humans can revel upon – Pattern Recognition. The Human Brain is the best pattern recognition machine ever created till today.

This project aims to address a small part of this pattern recognition problem by successfully creating a model which can detect and classify patterns in visual stimuli. The goal of this classification step will be to recognize an English alphabet, given some statistically significant numeric attributes about each alphabet.

2) Project Objective

Select and apply tools from this course's data mining repertoire to make a categorization decision to recognize an English alphabet, given numeric attributes about each alphabet.

3) Data Description

The dataset used for this study is called the "Letter Image Recognition Data" originally created by David J. Slate. This dataset was taken from the Machine Learning Repository at the University of California, Irvine. The dataset consists of 20000 entries. Each entry in the dataset models a randomly distorted black-and-white pixel image representing one of the 26 upper-case letters of the English alphabet derived from 20 unique fonts. Each of these entries contains 16

primitive numerical attributes (independent variables) representative of the respective English alphabet. Detailed information about each of these 16 attributes is given in the table below:

#	Name	Meaning	Type
1	Letter (Y)	Capital Letter (predicted value)	A-Z (26 factors)
2	x-box (X1)	Horizontal position of box (counting pixels from left edge of the image)	Integer
3	y-box (X2)	Vertical position of box (counting pixels from bottom edge of the image)	Integer
4	width (X3)	Width of box in pixels	Integer
5	high (X4)	Height of box in pixels	Integer
6	onpix (X5)	Total # of "on" pixels in the character image	Integer
7	x-bar (X6)	Mean x of on pixels in box	Integer
8	y-bar (X7)	Mean y of on pixels in box	Integer
9	x2bar (X8)	Mean x variance	Integer
10	y2bar (X9)	Mean y variance	Integer
11	xybar (X10)	Mean x y correlation	Integer
12	x2ybr (X11)	Mean of x * x * y	Integer
13	xy2br (X12)	Mean of x * y * y	Integer
14	x-edge (X13)	Mean edge count left to right	Integer
15	xegvy (X14)	Correlation of x-ege with y	Integer
16	y-ege (X15)	Mean edge count bottom to top	Integer
17	yegvx (X16)	Correlation of y-ege with x	Integer

Table 1: Independent and Dependent variables in the dataset

Since none of the attributes were NaN or NA, all the records given with this dataset can successfully be used for either training or testing purposes.

4) Exploratory Data Analysis

Exploratory data analysis gives a basic understanding of the dataset's characteristics with the help of summaries and visual plots. Before fitting a model to this data, it is imperative to understand properties and relationship between independent variables and dependent variables across the entire dataset. Analyzing such relationships helps determine if the data can be

modeled as is or if modifications such as normalizing the entire dataset (to adjust the variation of spread across variables) are necessary.

A) Summary

80% of the dataset was randomly selected to create the training dataset. The remaining 20% entries were chosen as the test dataset. Below are the dimensions of the dataset:

	Number of Rows	**Number of Columns**
Entire Dataset	20,000	17 (16 independent variables)
Training Dataset (80%)	16,000	17 (16 independent variables)
Test Dataset (20%)	4,000	17 (16 independent variables)

Table 2: Dataset Dimensions

Summary statistics for the independent variables across the entire dataset are given below:

Independent variable	**Min.**	**1st Qu.**	**Median**	**Mean**	**3rd Qu.**	**Max.**
x-box (X1)	0	3	4	4.024	5	15
y-box (X2)	0	5	7	7.035	9	15
width (X3)	0	4	5	5.122	6	15
high (X4)	0	4	6	5.372	7	15
onpix (X5)	0	2	3	3.506	5	15
x-bar (X6)	0	6	7	6.898	8	15
y-bar (X7)	0	6	7	7.500	9	15
x2bar (X8)	0	3	4	4.629	6	15
y2bar (X9)	0	4	5	5.179	7	15
xybar (X10)	0	7	8	8.282	10	15
x2ybr (X11)	0	5	6	6.454	8	15
xy2br (X12)	0	7	8	7.929	9	15
x-edge (X13)	0	1	3	3.046	4	15
xegvy (X14)	0	8	8	8.339	9	15
y-ege (X15)	0	2	3	3.692	5	15
yegvx (X16)	0	7	8	7.801	9	15

Table 3: Summary statistics for independent variables across the entire dataset

All the 16 independent variables are numeric in nature. Even though the minimum and maximum values are common across all the variables, the mean and median values differ. We need to study

if there are any observations across the entire dataset which are quite distant from the mean in any one direction (possible outliers). The following section evaluates this possibility.

Some of the independent variables have a different unit than the rest. For example, x2bar(X8) and y2bar(X9) represent the mean value of the squared horizontal and vertical distances. This measures the correlation of the horizontal variance with the vertical position and vice versa respectively. This unit is different from x-bar(X6) and y-bar(X7) which represent the mean horizontal and vertical distance of all the "on" pixels relative to the center of the box.

The dependent variable, letter, is a factor which can take any one of 26 possible outcomes.

B) Box Plots

In order to further explore the data through visualization, boxplots were used. The below figure represents box plots for all the explanatory/predictor variables:

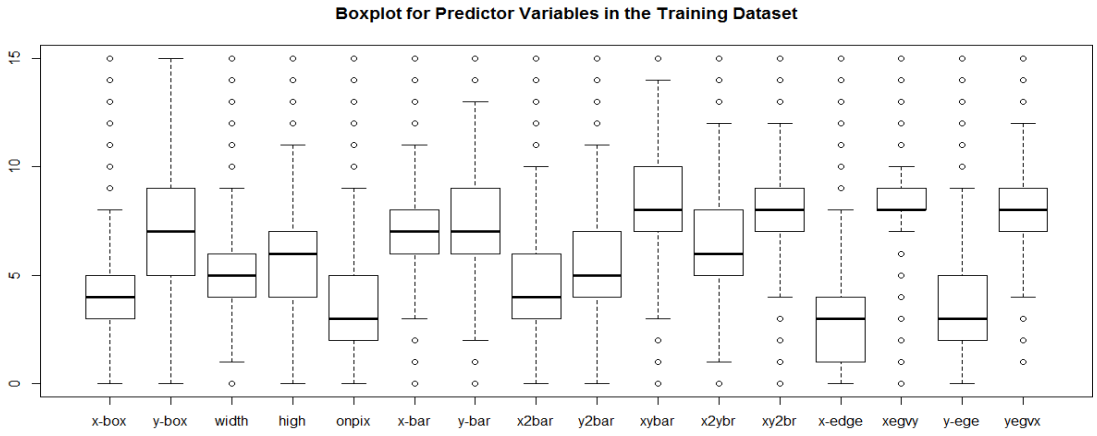

Figure 1 : Boxplots for Independent variables in the Training Dataset

The medians for independent variable "xegvy" is very close to the bottom (Q1) than to the top (Q3) of the boxplot. This is an indication of a skewed distribution. There are several possible

outliers with observed values falling beyond the whiskers on the top and bottom of each boxplot. Almost every independent variable has outliers extending in the upward direction.

5) Data-Mining Algorithms used on Letter Dataset

The Letter dataset was analyzed with three different data mining algorithms. The same random test (20%) and training (80%) datasets generated in section 4(A) are used with each of the algorithm to have a meaningful (one-one) comparison regarding the prediction accuracy of each algorithm.

A) Classification Tree (CART)

The training dataset was used to create a Classification Tree model. The Classification Tree was created using rpart() library function with the training dataset as input and "class" as the classification method. All the exploratory variables were included in the model and a symmetric cost function was used (False Positive weight = False Negative weight) for misclassification rate calculation since knowledge regarding the parameters is not known at this point.

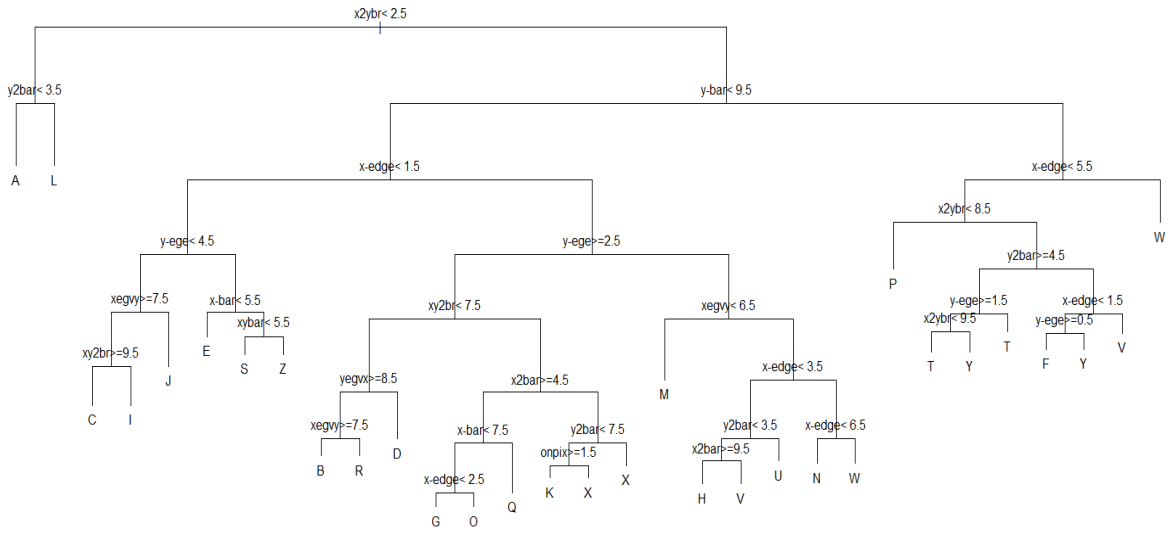

Figure 2: Classification Tree with Complexity Parameter (cp = 0.056)

The default Classification Tree complexity parameter (cp) of 0.01 was not sufficient to create a classification tree which can produce all the 26 possible outcomes. Smaller the complexity parameter (cp), the larger (complex) the tree rpart will attempt to fit.

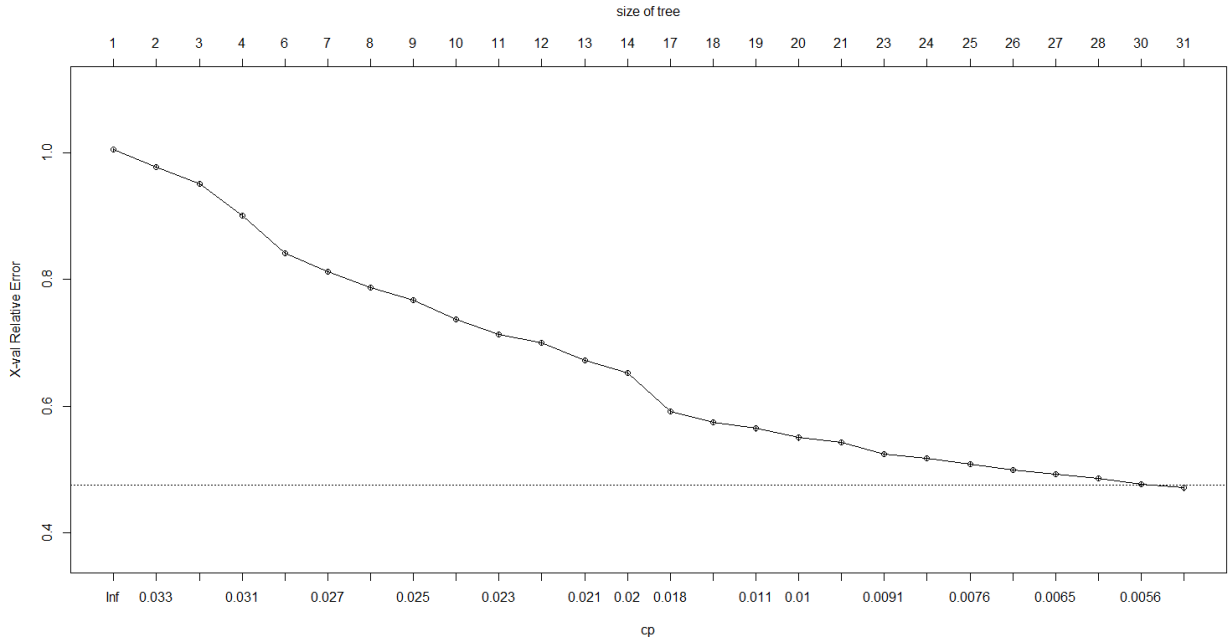

Figure 3 : Classification Tree - Relationship between 10-fold CV error in Training Dataset and Size of the Tree

Plotcp() function was used to find the relationship between 10-fold cross validation error in the training set and the size of the tree. Looking at this plot the error rate lies at 1 Standard Error above the minimum for a cp value of 0.0056. Hence, this cp value was used to generate the above Classification Tree.

In-sample prediction and out-of-sample prediction was done using the above generate classification tree. Table 4 shown below contains the confusion matrix for out-of-sample prediction results. The in-sample confusion matrix is skipped, as it looks quite similar to the out-of-sample matrix.

Truth \ Predicted	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	111	0	0	1	0	0	1	0	0	1	22	2	1	0	1	0	9	0	0	0	0	0	0	0	0	0
B	0	93	0	24	0	0	0	0	2	0	4	0	0	0	6	1	8	14	0	0	0	0	0	0	0	10
C	0	2	71	3	1	0	29	0	0	7	20	0	0	0	11	3	1	0	0	0	7	0	0	0	0	0
D	0	14	0	117	0	0	0	0	0	0	2	6	0	0	4	0	1	8	0	0	13	0	0	0	0	0
E	0	0	1	0	37	0	8	0	4	0	35	0	0	0	19	0	6	0	0	0	0	0	0	34	0	5
F	0	4	0	21	0	35	0	0	3	0	1	0	0	0	3	39	1	1	0	16	2	1	0	0	17	0
G	0	1	0	8	0	0	75	0	0	1	3	1	0	0	37	0	18	3	0	0	0	0	2	0	0	23
H	0	2	0	25	0	0	0	33	0	0	34	5	0	0	22	3	7	9	0	0	4	0	0	0	0	0
I	0	9	0	11	0	0	0	0	116	5	3	1	0	0	0	2	4	1	0	0	0	0	0	0	0	1
J	0	10	0	14	0	0	0	0	3	108	2	18	0	0	0	1	2	0	0	0	6	0	0	0	0	0
K	0	0	0	3	0	0	0	0	0	0	75	5	0	7	24	0	0	1	0	0	8	7	0	8	0	0
L	0	2	1	0	0	0	1	0	2	8	30	97	0	0	8	0	1	0	0	0	1	0	0	0	0	0
M	3	0	0	8	0	0	0	0	0	0	18	4	101	4	6	0	6	0	0	0	0	0	14	0	0	0
N	0	1	0	11	0	0	0	0	0	0	4	6	1	95	2	4	0	0	0	0	0	1	9	0	0	0
O	0	5	0	28	0	0	15	0	0	0	0	0	0	0	39	0	41	0	0	0	15	3	1	0	0	0
P	0	10	0	38	0	0	1	0	12	0	0	0	0	0	7	94	0	0	0	4	0	2	1	0	0	0
Q	0	4	0	4	0	0	3	0	0	0	23	1	0	0	13	5	98	1	0	0	0	4	1	2	0	0
R	1	1	0	28	0	0	5	0	0	0	24	9	0	0	11	35	9	33	0	0	0	0	0	0	0	0
S	0	5	0	41	0	0	3	0	2	0	17	5	0	0	0	2	22	2	42	0	2	0	0	6	0	8
T	0	1	0	7	0	0	0	0	1	0	10	0	0	0	7	9	1	0	0	106	0	1	0	7	6	0
U	0	0	0	7	0	0	0	0	0	0	9	0	1	17	15	0	8	0	0	0	105	0	1	0	0	0
V	0	1	0	2	0	0	0	0	0	0	9	0	0	1	5	1	0	0	0	1	1	112	3	0	2	0
W	0	0	0	3	0	0	0	0	0	0	0	0	0	4	4	1	8	1	0	0	0	4	106	0	0	0
X	0	8	0	26	0	0	0	0	0	0	44	9	0	0	10	0	0	1	0	0	1	0	0	72	1	0
Y	0	0	1	19	0	0	0	0	2	0	6	0	0	0	0	4	6	0	0	13	4	12	0	0	82	0
Z	0	1	0	16	0	0	14	0	0	0	5	0	0	0	0	7	2	0	0	0	0	0	0	13	0	101

Table 4: Classification Tree – Out-of-sample Confusion matrix (Truth Table)

As can be seen, the default model has made a lot of classification errors, which has resulted in poor classification accuracy with both the training and testing datasets. The prediction accuracy values are given in the Table 5 below:

Dataset	**Prediction Accuracy**
Training Dataset	54.45%
Testing Dataset	53.85%

Table 5: Classification Tree - Prediction Accuracy

The prediction accuracy could be improved by removing parameters which affect the prediction accuracy and adjusting the complexity parameter. Random Forest analysis, included in this report (section 5(C)) does some of this tuning automatically. Instead of improving the Classification Tree accuracy, fine tuning Random Forest model was chosen since Random Forest has much better prediction accuracy.

B) k Nearest Neighbors (kNN)

k Nearest Neighbor (kNN) algorithm is based on a simple underlying idea – classify the value of a new point based on k points(values) closest to it. Unlike Logistic Regression which uses a best-fit line and CART which uses a Tree Model, kNN does not use a virtual model on top of the training dataset. Therefore, a non-parametric classification technique such as kNN is very useful for this project since there is no prior information about the dataset or about which variables are more important to make predictions.

Since kNN does not use a logical model on top of the training distance, the scale of different attributes is very important. If a single numeric attribute has a large spread, it can dominate the distance calculation. Therefore all the independent variables were normalized into their respective Z-scores. As can be expected, after normalization, the mean of each column tends towards zero (very small number) and the standard deviation of each column tends towards 1 (equal variance).

The value of ‘k’ in kNN determines how many neighbors can influence the classification decision. Cross validation was employed to determine the optimum value of ‘k’. After testing a range of ‘k’ values between 1 and 20, it can be found from the graph below that increasing ‘k’ decreases the accuracy. Therefore k=1 is the optimum value of ‘k’ for this dataset. This is shown graphically in the figure below.

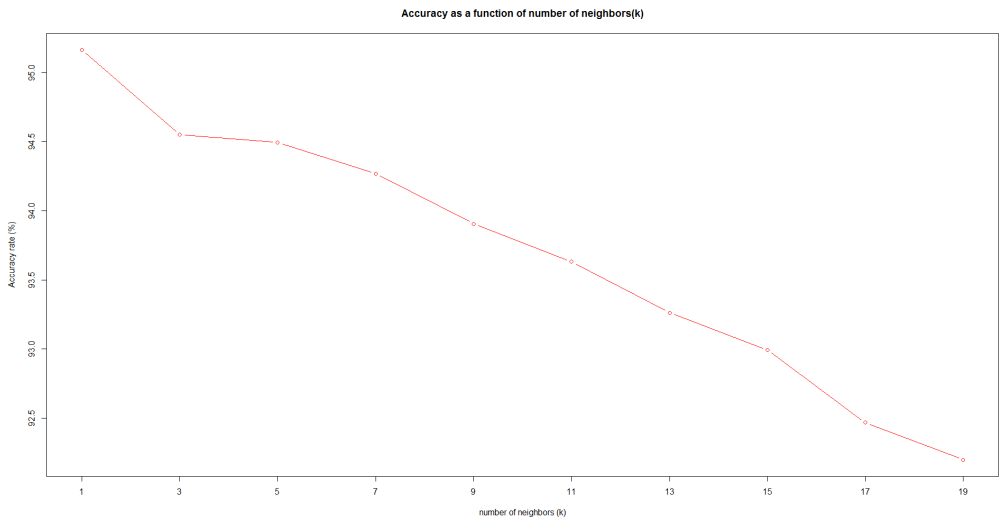

Figure 4: kNN – Prediction Accuracy as a function of number of neighbors (k)

With the optimal value of 'k', the prediction accuracy of the test and training set was determined using the 'predict' function as shown below in Table 6:

#Neighbors	Training Dataset Accuracy	Testing Dataset Accuracy
k = 1	1.00	0.94975

Table 6 : kNN – Prediction Accuracy with optimal 'k'

The optimal value of 'k' computed through cross-validation, was used to calculate the prediction accuracy with the test dataset. As can be expected, the training dataset which was used to create the kNN model is 100% accurate (since k = 1). The prediction accuracy for the test dataset is much better when compared to the Classification Tree model (section 5(A)). The confusion matrix for in-sample prediction is skipped as it is 100% accurate. Instead, the confusion matrix for the out-of-sample dataset is shown below in Table 7:

Truth	Predicted A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	148	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
B	0	146	0	3	1	0	1	0	0	0	1	0	0	0	0	0	0	2	0	0	0	8	0	0	0	0
C	0	0	149	0	0	0	3	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	0
D	0	0	0	158	0	0	1	2	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0
E	0	0	1	0	138	0	4	0	0	0	0	2	0	0	0	0	1	0	0	0	0	0	0	1	0	2
F	0	0	0	0	0	135	0	0	0	0	0	0	0	1	0	7	0	0	0	1	0	0	0	0	0	0
G	0	0	1	1	2	0	157	1	0	0	0	0	0	0	6	0	0	0	0	0	0	3	1	0	0	0
H	0	3	0	4	1	0	1	126	0	0	2	0	0	0	3	0	0	4	0	0	0	0	0	0	0	0
I	0	0	0	0	0	1	0	0	142	9	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	1	4	159	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	3	0	0	7	0	0	122	0	0	0	0	0	0	2	0	0	0	0	0	4	0	0
L	0	0	0	0	0	0	1	2	0	2	0	144	0	0	0	2	0	0	0	0	0	0	0	0	0	0
M	0	2	0	0	0	0	0	0	0	0	0	0	160	1	0	0	0	0	0	0	0	1	0	0	0	0
N	0	0	0	0	0	0	0	1	0	0	0	0	0	128	1	0	0	2	0	0	0	1	1	0	0	0
O	0	0	0	5	0	0	0	1	0	0	0	0	0	0	139	0	2	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	2	0	0	0	0	0	1	0	0	0	164	1	1	0	0	0	0	0	0	0	0
Q	0	0	0	0	1	0	0	0	0	0	0	0	0	0	6	1	149	2	0	0	0	0	0	0	0	0
R	0	2	0	0	0	1	1	1	0	0	1	0	0	3	0	0	0	145	0	0	0	2	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	155	0	0	0	0	0	0	2
T	1	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0	1	0
U	0	0	0	0	0	0	0	2	0	0	0	0	1	0	0	0	0	0	0	0	159	0	0	0	1	0
V	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	136	0	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	130	0	0	0
X	0	0	0	0	1	0	0	0	0	0	5	1	0	0	0	0	0	0	0	2	1	0	0	161	0	1
Y	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	1	1	0	144	0
Z	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	1	0	155

Table 7: kNN – Out-of-sample Confusion matrix (Truth Table)

In order to get a better understanding of how good this test-prediction is, the prediction accuracy is contrasted across the entire range of 'k' used for cross-validation. The figure below shows how the test and training dataset perform for a variety of 'k' values. As can be seen, the nearest neighbor kNN algorithm (k=1) offers the best accuracy for both training and test-dataset.

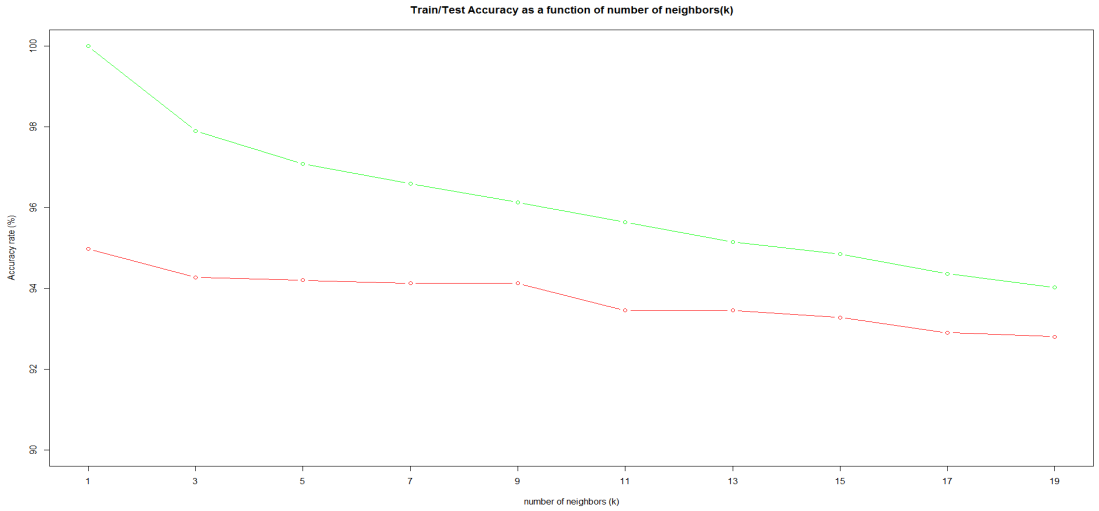

Figure 5: kNN – Train/Test Accuracy as a function of number of neighbors (k)

Since a uniform scale (z-scores) was adopted for all the independent variables, we have successfully avoided variation of spread across variables. Moreover, with a large training dataset of 16,000 entries, the curse of dimensionality should also be alleviated. Both of these observations are validated by performing kNN on the test-data set and showing that the prediction accuracy of the test dataset is quite close to the training dataset.

C) Random Forest

Random Forests are an ensemble learning algorithm built from aggregating trees. They can be used for regression and classification problems. Random Forest algorithm works as a large collection of decorrelated decision trees. The main idea is bootstrap aggregation (bagging) - using a combination of learning models increases the classification accuracy and tends to decrease the variance of the model without increasing the bias [2]. Random Forest uses random subsets of prediction variables and a random subset of training data to build each of the regression/classification trees in the forest (ensemble). After building the model, the entire Random Forest is used to predict each of the test-samples. Typically, the most common response (highest mode) is chosen for classification problems and the average is chosen for regression problems.

The below plot is an estimate of which variables are important for the model. The mean decrease in accuracy plot shows how each variable affects the accuracy of the random forest. The more the accuracy of the random forest decrease due to addition of a single variable, more the importance of the variable. Mean decrease in Gini Co-efficient is a measure of how each variable contributes to the homogeneity of the nodes. Therefore, important variables have a higher mean decrease in accuracy and higher mean decrease in Gini-Coefficient. As can be gleaned, x-edge,

y-ege and y2bar are the 3 most important independent variables and width, high, y-bar are the 3 least important independent variables for predicting the entire dataset.

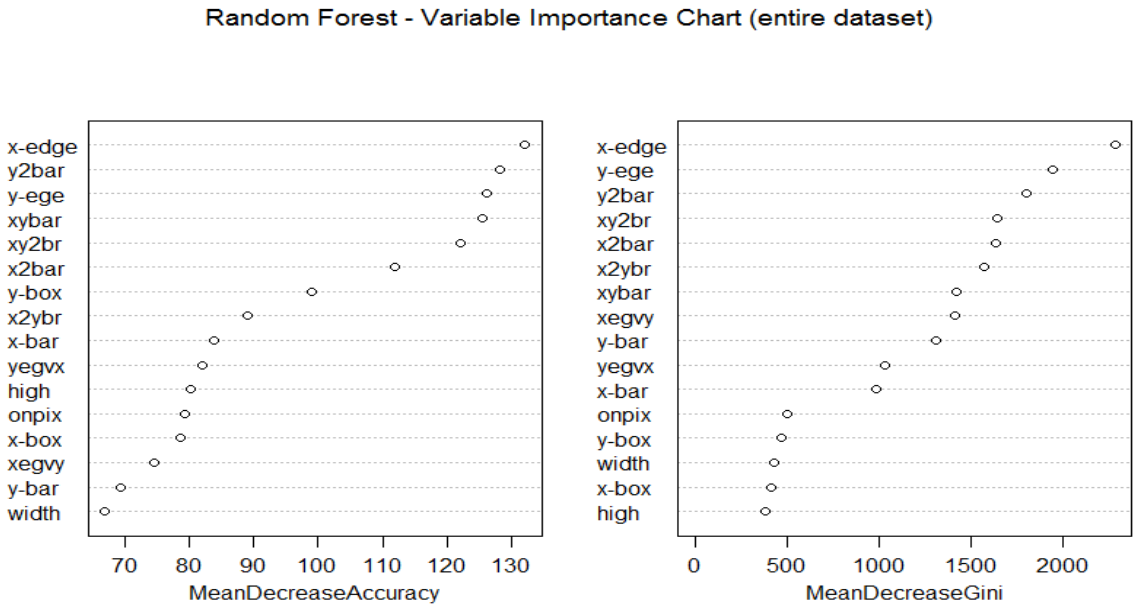

Figure 6: Random Forests – Variable Importance Chart (entire Dataset)

Random Forests have great accuracy, do not overfit, can handle a large amount of features and do not require cross-validation (due to bagging property). Therefore, two possible analyses on this dataset with Random Forest algorithm are shown in the following sections.

a) <u>Varying the number of trees used in the Random Forest model</u>

As can be seen in the below figure (Fig 7), the error rate is high for Random Forests with less than 100 trees. Above 100 trees, the error rate is constant. Therefore, the default number of trees (ntree = 500) is a good choice for this dataset.

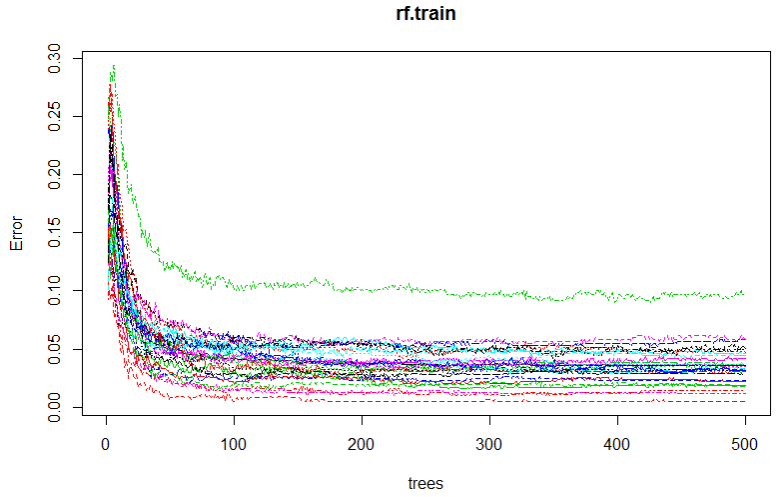

Figure 7: Random Forest – Prediction Error as a function of number of trees

b) Varying the number of variables randomly sampled as candidates at each split

As shown in the below figure (Fig 8), the error rate increases as the number of random variables is increased at each split. Therefore, for this dataset, choosing 2 random variables for each split gives the lowest OOB error estimate.

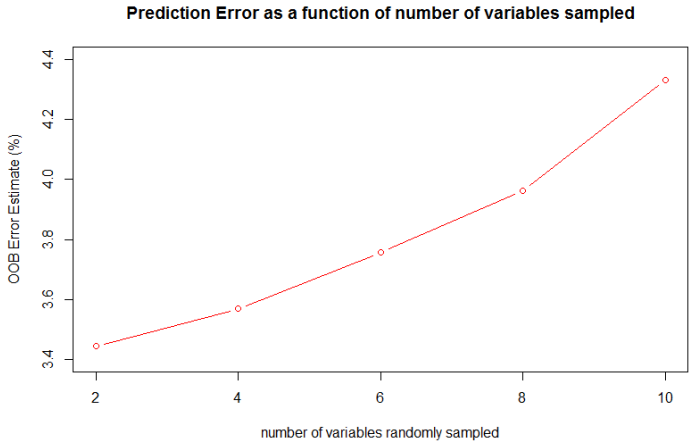

Figure 8: Random Forests – Prediction Error as a function of number of variables sampled

The above two optimizations show that a Random Forest with 500 trees and 2 random variables for each split is the best choice for this dataset. This optimized Random Forest is used to predict the training and testing dataset. The confusion matrix for in-sample prediction is skipped as it is 99.99% accurate. Instead, the confusion matrix for the out-of-sample dataset is shown below in Table 8:

	Predicted																								
Truth	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	148	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
B	0	153	0	0	1	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	4	0	0	0	0
C	0	0	150	0	1	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0
D	0	0	0	162	0	0	0	2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
E	0	0	1	0	140	0	3	0	0	0	0	2	0	0	0	0	1	0	0	0	0	0	0	1	0	1
F	0	1	0	1	0	139	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	1	0	0
G	0	0	0	3	2	0	162	1	0	0	0	0	0	0	2	0	0	1	0	0	0	1	0	0	0	0
H	0	1	0	1	0	0	1	134	0	0	0	0	0	0	2	0	0	5	0	0	0	0	0	0	0	0
I	0	0	0	0	0	1	0	0	143	8	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	1	0	1	3	159	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	1	0	0	132	0	0	0	0	0	0	3	0	0	0	0	0	2	0	0
L	0	0	1	0	0	0	1	0	0	1	0	146	0	0	0	0	0	1	1	0	0	0	0	0	0	0
M	0	0	0	0	0	0	1	0	0	0	0	0	161	1	0	0	0	0	0	0	0	1	0	0	0	0
N	0	0	0	1	0	0	0	1	0	0	0	0	1	127	0	0	0	2	0	0	0	2	0	0	0	0
O	0	1	0	2	0	0	0	0	0	0	0	0	0	0	143	0	0	1	0	0	0	0	0	0	0	0
P	0	1	0	1	0	2	0	0	0	0	0	0	0	0	1	164	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	1	0	0	0	0	0	0	0	0	0	4	0	152	2	0	0	0	0	0	0	0	0
R	0	3	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	150	0	0	0	0	0	0	0	0
S	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	155	0	0	0	0	0	0	0
T	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	152	0	0	0	0	0	0
U	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	161	0	0	0	0	0
V	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	135	0	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	131	0	0	0
X	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	170	0	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	148	0
Z	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	155

Table 8 : Random Forest – Out-of-sample Confusion matrix (Truth Table)

Table 9 shows in-sample and out-of-sample prediction accuracies for the optimized Random Forest model. As can be seen, the optimized Random Forest has outperformed the Classification Tree and kNN models.

	Training Dataset Accuracy	**Testing Dataset Accuracy**
Optimized Random Forest	0.9999375	0.968

Table 9: Random Forest – Prediction Accuracy with optimal Random Forest

6) Conclusion

Classification Tree was chosen instead of Multinomial Logistic Regression for this dataset because Classification Trees are better suited for classification problems with more than two outcomes and because Logistic Regression may tend to break when there are many independent variables. Prediction accuracy was employed as the common metric to compare the three algorithms chosen to model this dataset. The below table compares the in-sample and out-of-sample prediction accuracies across the three models:

	Classification Tree	**kNN (k=1)**	**Random Forest**
In-sample Prediction Accuracy	54.45%	100%	99.99%
Out-of-sample Prediction Accuracy	53.85%	94.98%	96.80%

Table 10: Comparing Prediction Accuracy across Models

Analyzing the prediction accuracies of the above three models, it is evident that Random Forest outperforms both the Classification Tree and kNN models for the Out-of-sample dataset. kNN is comparable to the Random Forest model for in-sample prediction because of heavy over-fitting (each training dataset value is used to model itself). This project has successfully contrasted three data mining models on the UCI Letter dataset and has shown the classification performance for each of these models.

References:

[1] P.M. Murphy, UCI repository of machine learning databases. Department of Information and Computer Science, UCI, 1994. http://archive.ics.uci.edu/ml/datasets/Letter+Recognition.

[2] Leo Breiman and Adele Cutler, Random Forests. Department of Statistics, UC Berkeley. http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm.