Forest Cover Type

Predict 454 DL SEC-55

Final Report

Thomas Callan
Ying Cheng
Sivakumar Desai
Crystal Mosley
Ryan Ruiz

1. Introduction

The purpose of this project is to develop multiple machine learning models to predict the seven forest cover types, based on the associated attributes provided. The observations are taken from 30m by 30m patches of forest that are classified as one of seven cover types: Spruce/Fir, Cottonwood/Willow, Krummholz, Lodgepole, Aspen, Ponderosa and Douglas Fir. Being able to classify forest cover types are useful for anyone in the natural resource industry in that they may need information about the land to make decisions regarding ecosystem management strategies and reviewing the overall health of the world's wooded areas. Being that this is a classification type of problem, we were trying to identify which set of categories a new observation belongs, based on a training set of data containing observations whose category membership is known. The classification algorithms that we decided to use were: Naive Bayes, Random Forest, Linear Discriminant Analysis, K-Nearest Neighbors, Conditional Inference Trees, SVM and Gradient Boosting. The best results by AUC (area under the curve) were found by the Random Forest algorithm, closely followed by the K-Nearest Neighbors algorithm by 1/10th of a percent.

2. The Modeling Problem

The Forest Cover Type challenge is a classification problem set trying to utilize data from four wilderness areas of Roosevelt National Forest, namely: Rawah, Comanche Peak, Neota and Cache la Poudre, to be able to identify and predict the ecological balance of an area using cartographic features. These forests have gone untouched by humans; therefore, scientists are at an advantage in utilizing this data. Usually forest cover types are identified using a field task force or by remote sensing technologies which can be time consuming and costly. If data are needed in remote regions or other parts of the world, these outdated techniques are not practical. Utilizing predictive models that are trained on existing data sets can be useful to approximate cover types for unknown areas. We have seven cover types to predict. Since this is a multi-classification problem, we will explore Naive Bayes, Random Forest, Linear Discriminant Analysis, K-Nearest Neighbors, Conditional Inference Trees, SVM and Gradient Boosting to predict forest types. As go we through the modeling process, we will explore the below modeling techniques for their respective reasons below:

- *Naïve Bayes (NB)* – This technique is a classifier based on Bayes' Theorem with independence assumptions between predictors. It is easy to build with no complicated iterative parameter estimation which makes it useful for very large datasets.
    - Advantages –
        - Fast, highly scalable, and used for binary and multi-class classification.
    - Disadvantages –
        - Considers that the features are independent of each other.
- *Random Forest (RF)* – This technique is an ensemble classifier that consists of many decision trees that output the class that is the mode of class output, by individual trees.
    - Advantages –
        - It runs efficiently on large datasets.
        - It can handle thousands of input variables without variable deletion.
        - It gives estimates of what variables are important in the classification.
        - It generates an internal unbiased estimate of the generalization error as the forest building progresses.
    - Disadvantages –
        - Have been observed to overfit for some datasets with noisy classification/regression tasks.
        - For data including categorical variables with different number of levels, random forests are biased in favor of those attributes with more levels. Therefore, the variable importance scores from random forest are not reliable for this type of data.
- *Linear Discriminant Analysis (LDA)* – This technique finds a linear combination of features that characterizes or separates two or more classes of objects or events, and seeks to reduce dimensionality while preserving as much of the class discriminatory information as possible.
    - Advantages –
        - Helps reduce computational costs for a given classification task, but it can also be helpful to avoid overfitting by minimizing the error in parameter estimation.
    - Disadvantages –
        - If the classification error estimates establish that more features are needed, some other method must be employed to provide those additional features.
        - If the distributions are significantly non-Gaussian, the projections may not preserve complex structure in the data needed for classification.
        - If discriminatory information is not in the mean but in the variance of the data, this technique will fail.
- *K-Nearest Neighbors (K-NN)* – This technique's objective is to classify an object by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.
    - Advantages –
        - Robust to noisy training data.
        - Effective on large training datasets.
    - Disadvantages –

        - Manually need to determine the number of nearest neighbors.
        - Distance based learning is not clear which type of distance to use and which attribute to use to produce the best results.
- *Conditional Inference Trees (CTree)* – This technique is a non-parametric class of regression trees embedding tree-structured regression models into a well-defined theory of conditional inference procedures.
    - Advantages –
        - Help to avoid overfitting of trees.
        - Help to avoid selection bias towards covariables with many possible splits.
        - Help to avoid difficult interpretation due to selection bias.
    - Disadvantages –
        - Trees can be very non-robust. A small change in the training data can result in a large change in the tree and consequently the final predictions.
- *Support Vector Machine (SVM)* – This technique is a discriminative classifier where the algorithm outputs an optimal hyperplane which categorizes new examples.
    - Advantages –
        - Works well with even unstructured and semi structured data like text, Images and trees.
        - It scales relatively well to high dimensional data.
    - Disadvantages –
        - Choosing an appropriate kernel function is not easy.
        - Long training time for large datasets.
        - Difficult to understand and interpret the final model, variable weights and individual impact.
- *Gradient Boosting Machine (GBM)* – This technique produces a prediction model in the form of an ensemble of weak prediction models. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.
    - Advantages –
        - Often provides predictive accuracy that cannot be beat.
        - Lots of flexibility - can optimize on different loss functions and provides several hyperparameter tuning options that make the function fit very flexible.
        - No data pre-processing required - often works great with categorical and numerical values as is.
        - Handles missing data - imputation not required.
    - Disadvantages –
        - Will continue improving to minimize all errors. This can overemphasize outliers and cause overfitting. Must use cross-validation to neutralize.
        - Computationally expensive - GBMs often require many trees (>1000) which can be time and memory exhaustive.
        - The high flexibility results in many parameters that interact and influence heavily the behavior of the approach. This requires a large grid search during tuning.

3. The Data

The data were taken from 30m by 30m areas of forest that were classified into seven cover types: Spruce/Fir, Lodgepole Pine, Cottonwood/Willow, Aspen, Douglas-fir, and Krummholz. We have in total 581,012 observations with 55 attributes. There are 10 numeric attributes, and 45 categorical attributes. All variables are classified as integers with soil types 1-

40 being binary (0=absence, 1=presence) and wilderness areas 1-4 also being binary (0=absence, 1=presence). **Table 1** displays the Variable, Type, and Descriptions of each.

Table 1: Forest Cover Data Dictionary

	Attributes	Type	Description
1	elevation	Numeric	Elevation, measured in meters (m)
2	aspect	Numeric	Azimuth from true north
3	slope	Numeric	Slope measured in º (degrees)
4	horizontalDistToWater	Numeric	Horizontal distance to nearest surface water feature (m)
5	verticalDistToWater	Numeric	Vertical distance to nearest surface water feature (m)
6	horizontalDistToRoad	Numeric	Horizontal distance to nearest roadway (m)
7	hillshade9Am	Numeric	Relative measures of incident sunlight at 9am on the summer solstice (0 to 255 index)
8	hillshade12Pm	Numeric	Relative measures of incident sunlight at 12pm on the summer solstice (0 to 255 index)
9	hillshade3Pm	Numeric	Relative measures of incident sunlight at 3pm on the summer solstice (0 to 255 index)
10	horizontalDistToFirePoints	Numeric	Horizontal distance to nearest historic wildfire ignition point (m)
11	rawahWildArea	Categorical	Wilderness designation area, 0-absence, 1-presence
12	neotaWildArea	Categorical	Wilderness designation area,0-absence, 1-presence
13	comancheWildArea	Categorical	Wilderness designation area,0-absence, 1-presence
14	cacheWildArea	Categorical	Wilderness designation area,0-absence, 1-presence
15	soilType1 - soilType40	Categorical	Soil type designation,0-absence, 1-presence
16	coverType	Categorical	Seven different cover types: Spruce/Fir, Cottonwood/Willow, Krummholz, Lodgepole, Aspen, Ponderosa, Douglas Fir

Before we move to exploratory data analysis and modeling, we need to check the data quality. **Table 2** is a summary of the ten numeric attributes. It gives us an overall view of the minimum value, mean, median, and maximum values. There are no missing values. Taken standard deviation into consideration, by comparing minimum values with 1% percentile to find potential outliers from lower values, comparing maximum values with 99% percentile to find large outliers. We have highlighted the potential outliers in red below. Except for slope, all the numeric attributes might have outliers.

Table 2: Forest Cover Numeric Attributes Summary

	min	Quantile.1%	Quantile.25%	Mean	Median	Quantile.75%	Quantile.99%	Max	sd	Zeros	missing
elevation	1,859	2,122	2,809	2,959	2,996	3,163	3,438	3,858	279.98	-	0
aspect	0	1	58	156	127	260	356	360	111.91	4,914	0
slope	0	2	9	14	13	18	35	66	7.49	656	0
horizontalDistToWater	0	-	108	269	218	384	937	1,397	212.55	24,603	0
verticalDistToWater	-173	(42)	7	46	30	69	251	601	58.30	38,665	0
horizontalDistToRoad	0	150	1,106	2,350	1,997	3,328	6,112	7,117	1559.25	124	0
hillshade9Am	0	127	198	212	218	231	252	254	26.77	13	0
hillshade12Pm	0	162	213	223	226	237	254	254	19.77	5	0
hillshade3Pm	0	41	119	143	143	168	226	254	38.27	1,338	0
horizontalDistToFirePoints	0	182	1,024	1,980	1,710	2,550	6,262	7,173	1,324.20	51	0

Figure 1: Boxplot of All Numeric Variables

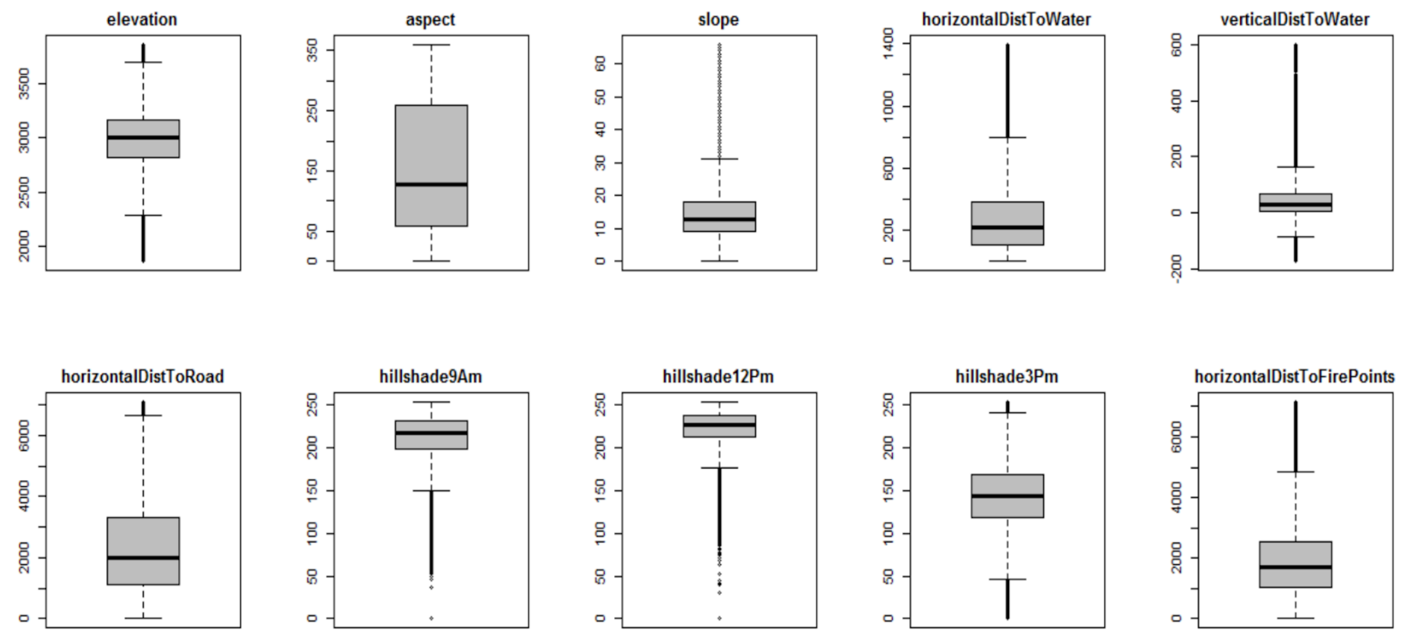


Figure 1 above shows the boxplots containing all the numeric variables. We do see there are many outliers present in the plots, which confirms what we observed before. The only numeric do not have outliers is aspect.

Table 3: Forest Cover Categorical Attributes Summary

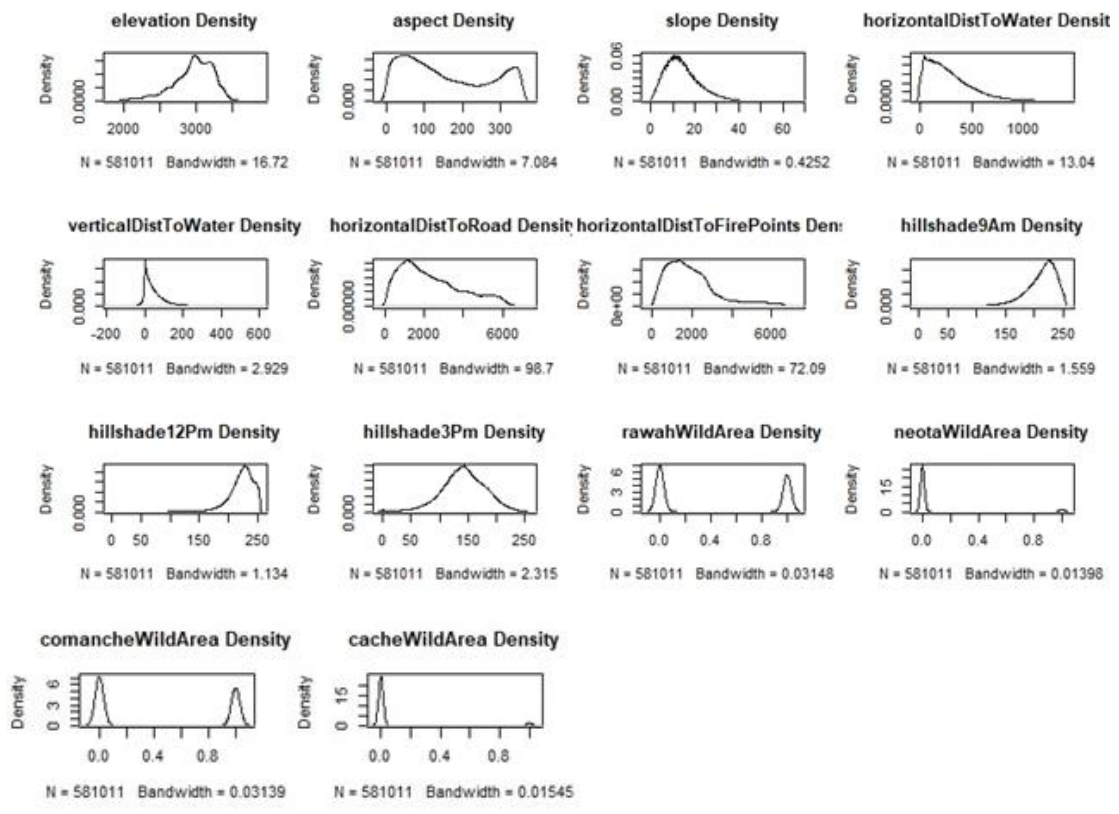
Attributes	missing	Count 1	Percentage
rawahWildArea	0	260,796	44.9%
neotaWildArea	0	29,884	5.1%
comancheWildArea	0	253,364	43.6%
cacheWildArea	0	36,968	6.4%
soilType1	0	3,031	0.5%
soilType2	0	7,525	1.3%
soilType3	0	4,823	0.8%
soilType4	0	12,396	2.1%
soilType5	0	1,597	0.3%
soilType6	0	6,575	1.1%
soilType7	0	105	0.0%
soilType8	0	179	0.0%
soilType9	0	1,147	0.2%
soilType10	0	32,634	5.6%
soilType11	0	12,410	2.1%
soilType12	0	29,971	5.2%
soilType13	0	17,431	3.0%
soilType14	0	599	0.1%
soilType15	0	3	0.0%
soilType16	0	2,845	0.5%
soilType17	0	3,422	0.6%
soilType18	0	1,899	0.3%
soilType19	0	4,021	0.7%
soilType20	0	9,259	1.6%
soilType21	0	838	0.1%
soilType22	0	33,373	5.7%
soilType23	0	57,752	9.9%
soilType24	0	21,278	3.7%
soilType25	0	474	0.1%
soilType26	0	2,589	0.4%
soilType27	0	1,086	0.2%
soilType28	0	946	0.2%
soilType29	0	115,247	19.8%
soilType30	0	30,170	5.2%
soilType31	0	25,666	4.4%
soilType32	0	52,519	9.0%
soilType33	0	45,154	7.8%
soilType34	0	1,611	0.3%
soilType35	0	1,891	0.3%
soilType36	0	119	0.0%
soilType37	0	298	0.1%
soilType38	0	15,573	2.7%
soilType39	0	13,806	2.4%
soilType40	0	8,750	1.5%

🟥 Below 5%

Table 3 summarizes all the categorical attributes, except the dependent variable cover type. There are no missing values. Only three variables (highlighted in green) have “presence=1” take over 10%. Most of the variables (highlighted in red) have “1” less than 5%. There are so many rare cases in the forest cover study.

Figure 2 below contains density plots for the 10 integer predictor variables which provide insight into their distributions. Things to look for within these plots are long tails, skewness, high frequencies, and potential anomalies. Elevation has a relatively normal distribution and Aspect contains two normal distributions. *Slope*, *horizontalDistToFirePoints*, and *horizontalDistToRoad* all have similar distributions. *HorizontalDistToWater* and *verticalDistToWater* seem to spike around 0. *Hillshade9Am*, *hillshade12Pm*, and *hillshade3Pm* distributions display left skew, left skew, and normal, as expected.

Figure 2: Density plots of 10 predictor variables



To see if any variables showed correlation between one another, we reviewed a scatterplot matrix of all the predictor variables, as shown in **Figure 3**. Seeing as how the *hillshade* variables all seemed most correlated with each other, we looked deeper into their specific correlations, **Figure 4**. The outcome, is there are four pairwise correlations that have a value higher than absolute 0.5:

Hillshade_Noon, Hillshade_3pm	(0.61)
Hillshade_9am, Hillshade_3pm	(-0.78)
Horizontal_Distance_To_Hydrology, Vertical_Distance_To_Hydrology	(0.65)
Slope, Hillshade_Noon	(-0.61)
Aspect, Hillshade_9am	(-0.59)
Aspect, Hillshade_3pm	(0.64)
Elevation, Horizontal_Distance_To_Roadways	(0.58)

Figure 3: Correlation matrix of all predictor variables

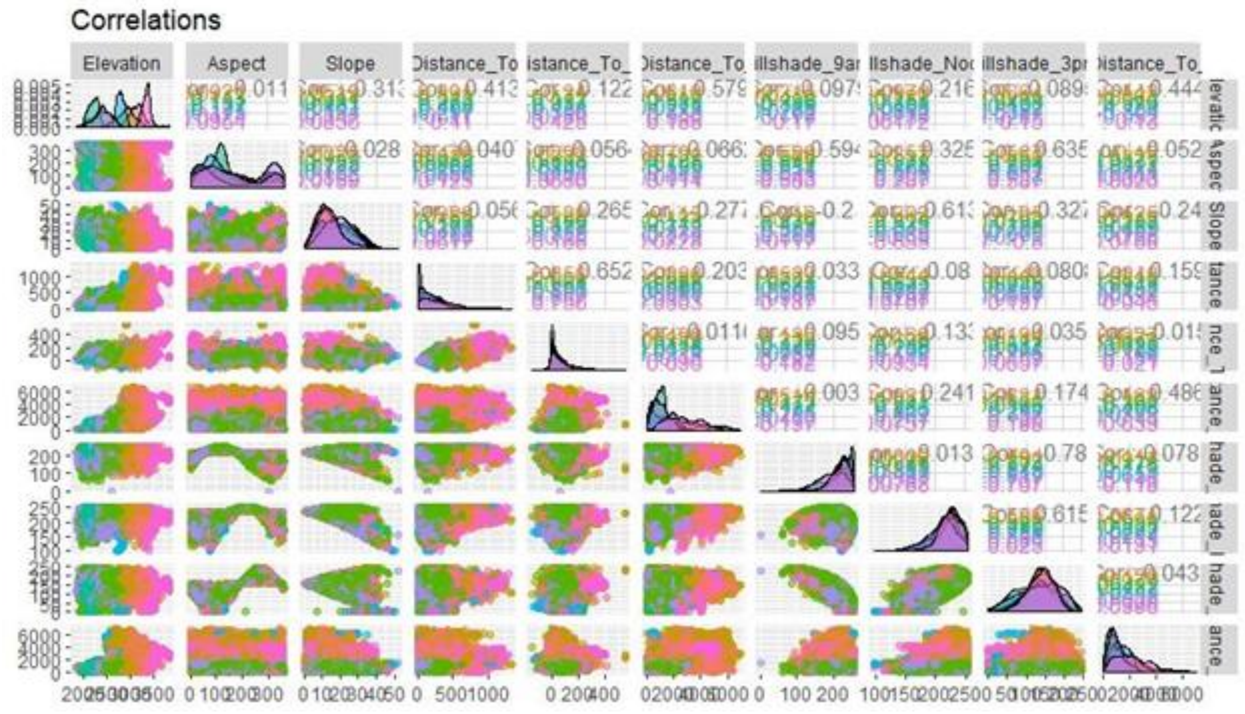
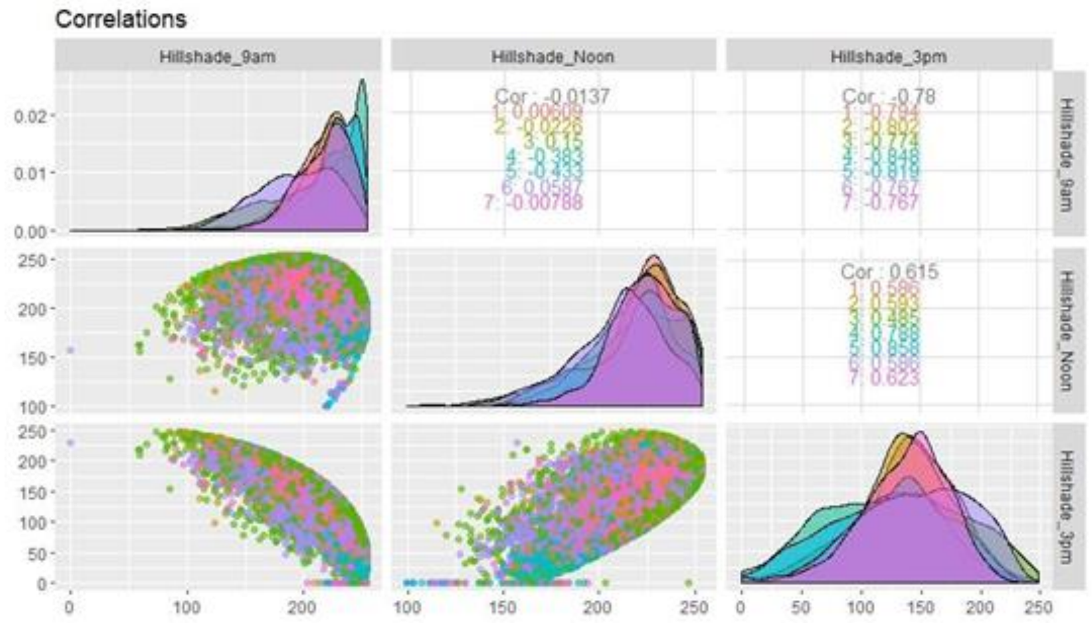


Figure 4: Correlation matrix of only Hillshade variables



4. Exploratory Data Analysis

Now that we have a firm understanding of our data, and how we could potentially use it to predict the type of forest cover for a given 30m x 30m cell we are ready to begin doing our initial exploratory analysis.

We will begin with some basic data cleanliness checks where we will look for missing or erroneous values and will confirm that data types are what we expect them to be. We will use R's 'summary' function to quickly give us some key metrics for each column in our data set. We inspect each variable in search of a few things initially; do we have values for every observation? Do the values we have for each observation make sense based on the expected values we've outlined in Section 3? Do any of our basic summary statistics show evidence of majorly skewed data or highly influential outliers?

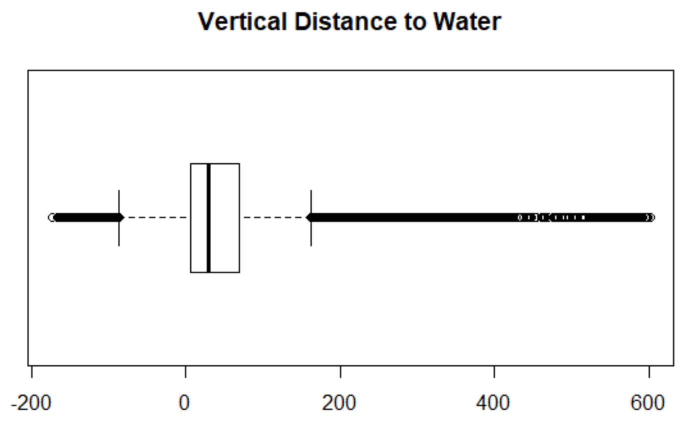
Fortunately, we first see that we are not missing any data across any of our 55 variables. If we were to find missing variables anywhere, we would have some choices as to how to proceed. We could drop the observations with missing variables, we could drop those variables entirely, or we could populate the missing data points with a reasonable replacement such as the mean or median of that variable. That decision would require additional analysis based on how much data we have, how many data elements are missing, and numerous other factors. In this case, we don't need to worry about this, so we move on.

We now move onto investigating the reasonability of our variable ranges and summary statistics for each variable. Starting with elevation, we see a range of values between 1,859 and 3,858 with mean and median values that fall almost perfectly in-between this range at ~ 3,000. This seems completely reasonable based on our understanding of this variable and there is no evidence of invalid observations. We see a minimum aspect value of 0 and a max of 360. As aspect is measured as 'degrees azimuth', we do expect to see values from 0 to 360 so we are looking good here as well.

For slope we see values ranging 0 to 66 and we know that slope is also measured in degrees. When thinking about slope, a value of 90 would indicate straight down, so a max of 66 seems quite steep, but reasonable. We also see that our mean and median values are on the lower end of this range, around 13-14 degrees. This also seems logical as we would expect most land to be relatively flat with occasional areas of higher slope.

We have 3 variables that measure horizontal distance. These variables represent distance in meters to the nearest water, road, and wildfire ignition points. We would expect similar looking results for all 3 of these variables as they should all have minimum values at or around zero and should have some reasonable maximum value. We see that the max values for distance to water and fire ignition points are much greater than distance to water which seems reasonable since we are dealing with primarily wilderness area. We also have a variable that measures vertical distance to water. This variable is unique from the horizontal measurements in that it has a minimum value of -173. When we think about this logically, it makes sense as something can only be a positive distance away in horizontal measurement, but something can be either above or below you vertically, so as such we are not surprised to see negative values in this column. To confirm that this is not an anomaly we observe a boxplot of all values of vertical distance to water, **Figure 5**, and see that there are a significant count of negative figures and our mean and median values are relatively close to zero as a result.

Figure 5: Boxplot of Vertical Dist. to Water



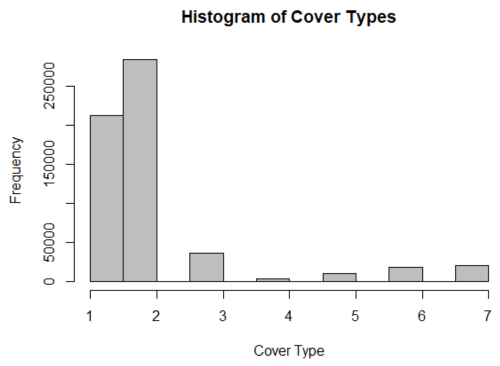
There are 3 variables that measure hill shade levels at various times of the day (9 AM, 12 PM, and 3 PM). Hill shade intensity is measured on a scale of 0 to 255, and we see that all 3 variables have minimum values of 0 and maximum values of 254 so we are confident that there are no out of line values here.

We have 44 binary variables as well; 40 representing soil types of 1-40 and 4 indicating whether the cell is part of the corresponding wilderness area. For all these binary columns we observe minimum values of 0 and max values of 1. We also find that our mean values of each value for these two groups add up to 1 which tells us that each observation most likely belongs to 1 and only 1 category. Although that observation is not a necessity for our modeling, it’s something useful to understand as we continue our analysis and modeling. We also observe value counts of each binary variable to ensure that we only have values of 0 and 1, and we don’t have any variables that are either all 0 or 1. We don’t, so we can move on.

Finally, we investigate our target variable and as expected we see a minimum value of 1 and a maximum value of 7 which is what we expect to see. Because this is a categorical variable, the mean value is meaningless, but we do see that we have a median value of 2 so we will expect to see 2 be the most frequently occurring forest cover type.

As we begin trying to understand how these 54 predictor variables relate to our dependent variable, a good place to start is first understanding the distribution of our dependent variable. We begin by observing a histogram of counts of each of our 7 forest cover types in **Figure 6**.

Figure 6: Histogram of Cover Type

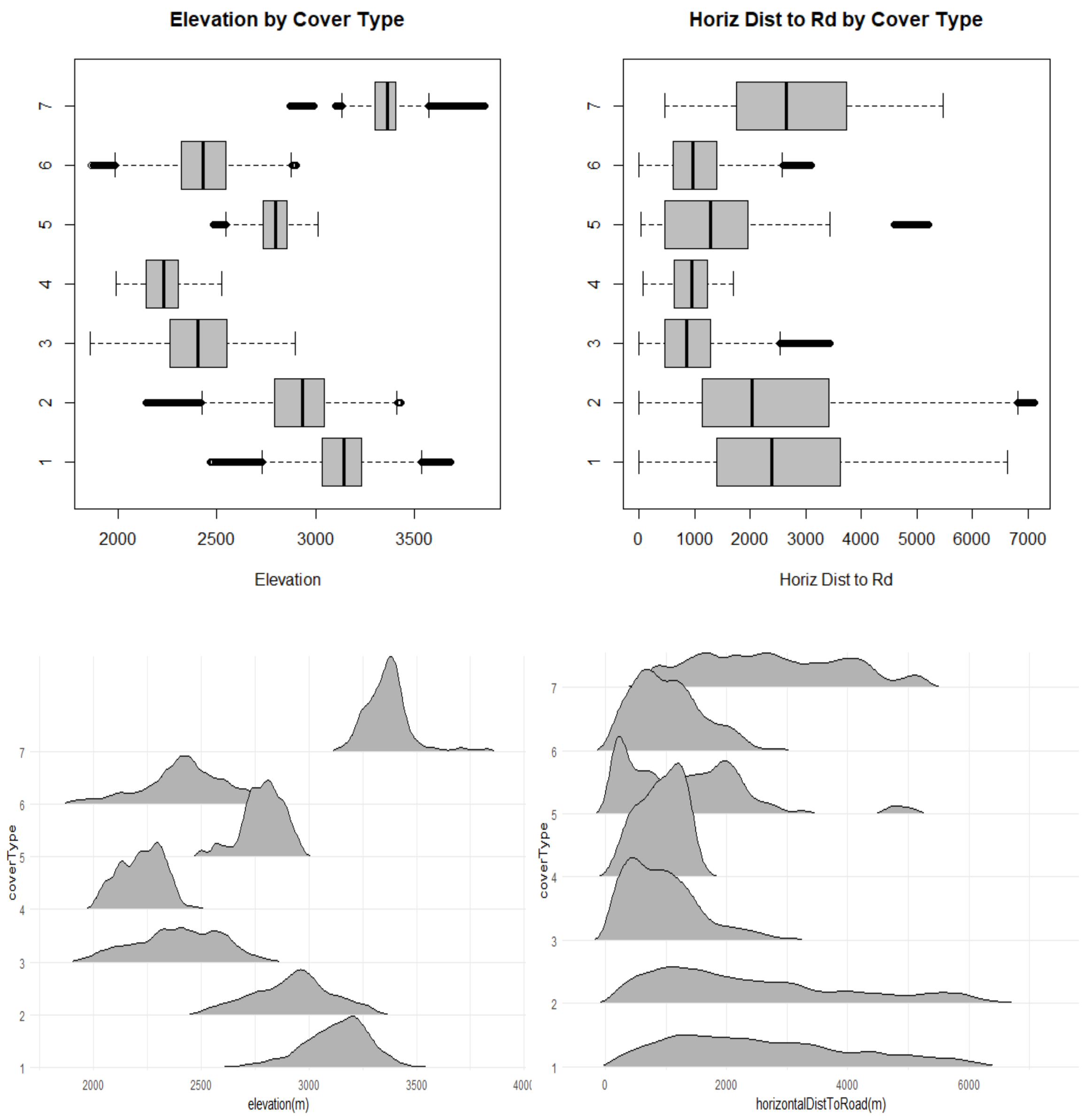


This graphic provides us with several important takeaways. First, we see that forest cover values of 1 and 2 are much more common than any other value: *coverType* 1 = 36.46%, *coverType* 2 = 48.76% of the full dataset. We also see that 2 is indeed our most common cover type, which confirms our observation based on the results of the summary function. We also see that the count of '4' values is extremely low compared to the others. Broadly speaking, this distribution is quite unique in shape, and we have a massive amount of data to train and test on, so we would expect that any outputs from our models would fit a similar distribution.

Our next step is to evaluate each of our 10 continuous variables to try to identify any patterns in differences in distribution between the 7 categorical variables. We do this by observing box plots of each continuous variable split by the values of *coverType*. The most interesting observations are explored below in **Figure 7**.

When observing elevation by cover type we see significant differences between each cover type value which lets us know that elevation is most likely going to be a very strong predictor of cover type. We see similar strong identifying characteristics in our horizontal distance to road data. Furthermore, amongst these two attributes, we see that they complement each other well as predictors. For example, the horizontal distance to road data sees cover types 1, 2, and 7 as similar within that context, but those 3 elements are significantly different if you look by cover type. When used together in a tree-based model these predictors will be very powerful and we'd expect these to be some of our most important variables in random forest models.

Figure 7: Boxplots of elevation and horizontal distance to road by cover type

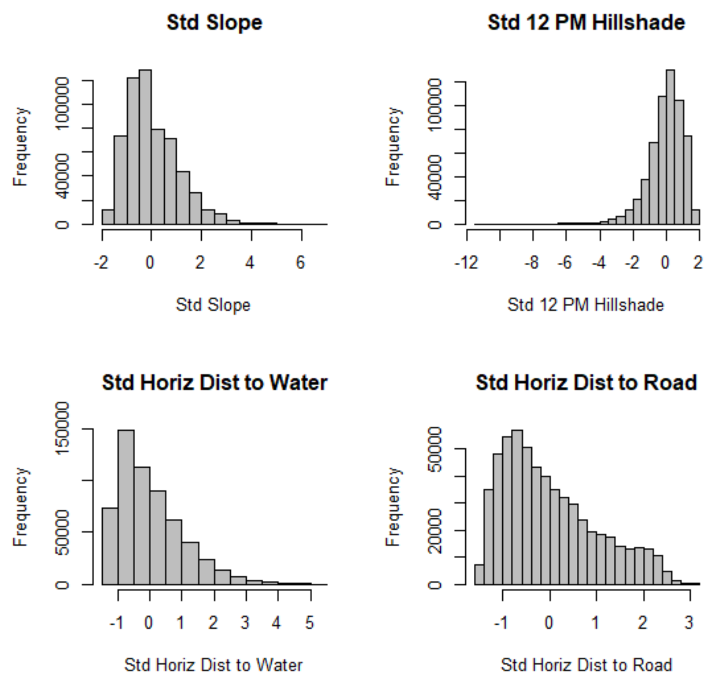


Based on the above, we've got some very rich data available and we would expect to be able to come up with some very accurate predictive models as a result. The next major item we need to tackle is scaling of our data. While some regression methods can obtain accurate results regardless of whether the data is in the same scale, other methods such as K-nearest neighbors need continuous variables to be on the same scale. Rather than deal with scaling later, we should just put all 10 continuous variables on the same scale for all models. We will zero center our variables around 0 and adjust values to standard deviations from 0 by applying the following formula: *x.std = (x-mean(x)) /std dev (x)*

Once our variables are normalized, we observe the distributions of each continuous variable. Here we are looking for potentially skewed distributions as well as any other interesting characteristics that may indicate that we should apply a transform to the data or exclude the variable entirely from our modeling.

Upon investigation of histograms we see that we do have some extremely skewed data, **Figure 8**. Some of the worst offenders are shown below:

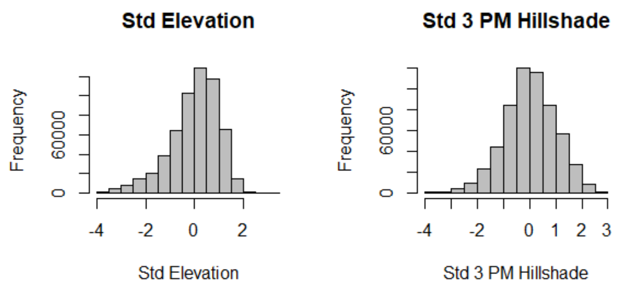
Figure 8: Histograms of heavily skewed variables



Because of the high levels of skewness of some of these predictor variables, we will research potential transforms that can handle both zeros and negative numbers but also will force this data into a distribution that is closer to normal in shape. We will also expect to see better performance from models that do not require data to be normally distributed to be accurate.

Figure 9 shows a couple of our continuous variables showing shapes that are at least closer to normal than the above examples, such as elevation and *hillshade* at 3 PM:

Figure 9: Histogram of elevation and 3 PM Hillshade



5. Predictive Modeling: Methods and Results

5.1 Model Evaluation Criteria

Before we begin modeling, particularly because this is a group project, we must define a consistent set of evaluation metrics that each model can be evaluated on so that we are comparing models in a consistent manner from team member to team member. Because we are dealing with a classification problem, that distinction is relatively straightforward. Our primary model evaluation criteria will be predictive accuracy on our test data. Our secondary evaluation criteria will be predictive accuracy on training data and area under the curve (AUC) on both test and training data sets.

Each of the models fitted and evaluated below will be evaluated on these same metrics and results will be presented in a consolidated grid at the end for comparison and discussion.

5.2 Train/Test Split Methodology

For purposes of training and evaluating our models, we will be utilizing a 70/30 split where 70% of our data will be used to train the model and the remaining 30% will be used to evaluate the performance of our trained models. The training set contains 406,708 observations with 56 variables, while the test set contains 174,303 observations with 55 variables. We will evaluate model accuracy on both the train and test data sets which will also allow us to identify any signs of over or under-fitting models.

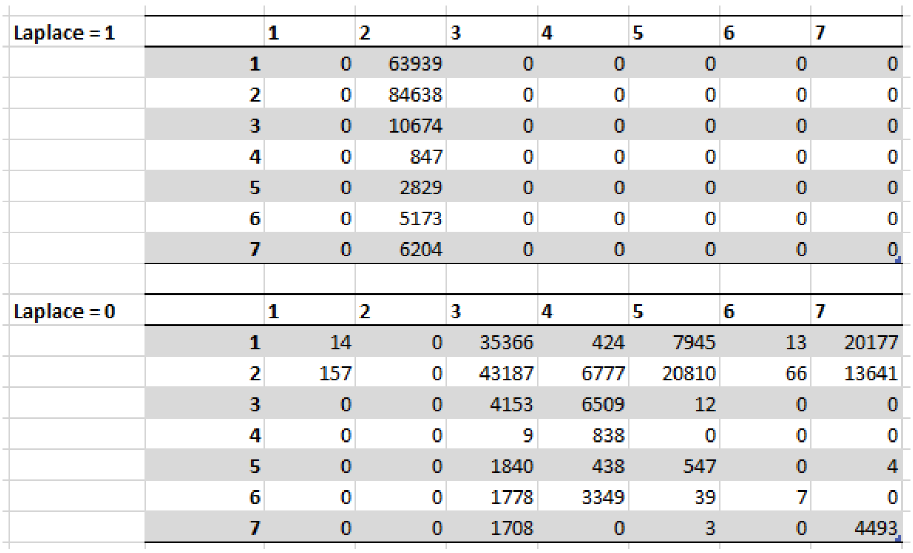
5.3 Models

Model 1: Naive Bayes

To begin the modeling process, we began by trying out the Naive Bayes prediction model. As this is a modeling method that we have limited experience with compared to the others, we didn't know what to expect here. The two main hyperparameters that we had to tinker with were *usekernel* which engages kernel smoothing in the model and *laplace* which allows you to add a constant to each value to avoid issues encountered with zeros in the data (which we have a lot of). The most common use of the *laplace* variable is 1 and *usekernel* is binary so we ended up fitting a total of 4 Naive Bayes models. One with *usekernel* = FALSE and *laplace* = 0, one with *usekernel* = TRUE and *laplace* = 0, one with *usekernel* = FALSE and *laplace* = 1, and finally one with *usekernel* = TRUE and *laplace* = 1.

While it seems that our model fitted appropriately given the feedback when we printed the fit of each model, none of the models did a good job predicting cover type. As a matter of fact, if we fitted the model using *laplace* = 0 we got 2.4% accuracy on both the train and test sets. Either model where *laplace* = 1 predicted every observation as a 2 which gave a train accuracy of 48.8% and a test accuracy of 48.6%. The confusion matrices of these two models are provide below in **Table 4** for reference.

Table 4: Confusion Matrix of Laplace 1 and 0



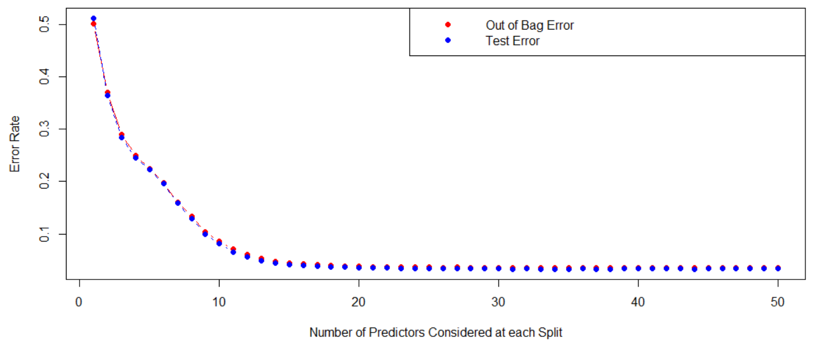
Laplace = 1		1	2	3	4	5	6	7
	1	0	63939	0	0	0	0	0
	2	0	84638	0	0	0	0	0
	3	0	10674	0	0	0	0	0
	4	0	847	0	0	0	0	0
	5	0	2829	0	0	0	0	0
	6	0	5173	0	0	0	0	0
	7	0	6204	0	0	0	0	0
Laplace = 0		**1**	**2**	**3**	**4**	**5**	**6**	**7**
	1	14	0	35366	424	7945	13	20177
	2	157	0	43187	6777	20810	66	13641
	3	0	0	4153	6509	12	0	0
	4	0	0	9	838	0	0	0
	5	0	0	1840	438	547	0	4
	6	0	0	1778	3349	39	7	0
	7	0	0	1708	0	3	0	4493

As these results are considerably worse than expected, we used the same code to run models on different data sets with less output classes and got very good results so our conclusion here is that this is just not the right model for this data. It seems that once we deal with the zeros in our data, we end up just picking the most common class for every row since the odds, on a per row basis, are always the highest that it’s a 2, but of course that’s still only correct less than half the time.

Model 2: Random Forest

We predicted forest cover type using a random forest. We tuned the model to determine the number of predictors used at each split by repeated running a 75-tree model. **Figure 10** displays the out of bag error and test error for each number of predictors considered. The error rate flattens around 20 predictors, but the actual minimum is at 33 predictors considered at each split.

Figure 10: Error Rates for Number of Predictors Considered



We considered that there were not enough trees used in the random forest model, so we compared the 75-tree and 33-predictor model against a 250-tree and 20-predictor model and found the 33-predictor random forest still had a smaller test error. **Figure 11** displays the variable importance for the 33-predictor random forest model. Elevation has the highest importance followed by horizontal distance to road and horizontal distance to fire points.

Figure 11: Variable importance for 33-predictor RF model

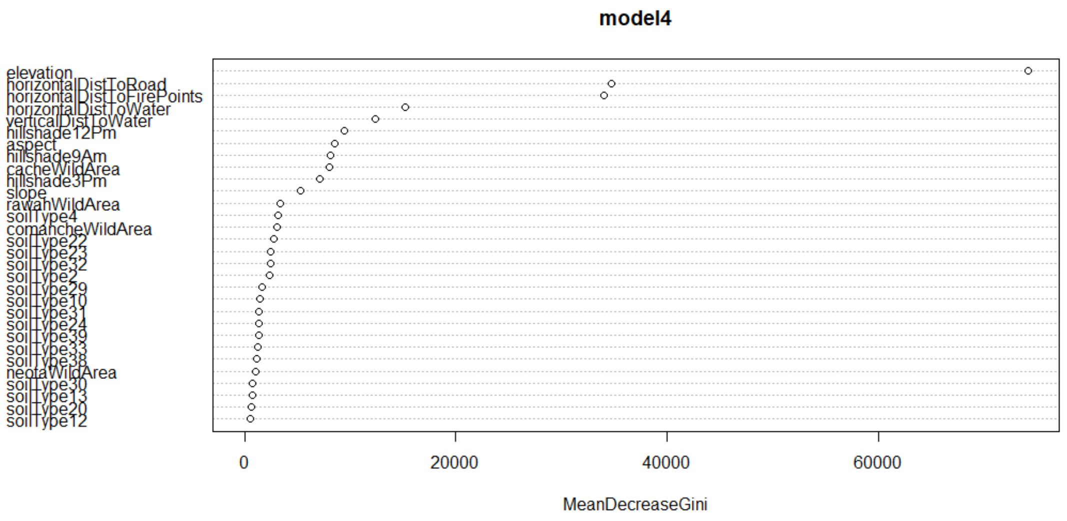


Table 5 shows the training results. Overall accuracy is 100%.

Table 5: RF model training results

		Actual Type							
		1	2	3	4	5	6	7	
Predicted Type	1	147901	0	0	0	0	0	0	100.0%
	2	0	198663	0	0	0	0	0	100.0%
	3	0	0	25080	0	0	0	0	100.0%
	4	0	0	0	1900	0	0	0	100.0%
	5	0	0	0	0	6664	0	0	100.0%
	6	0	0	0	0	0	12194	0	100.0%
	7	0	0	0	0	0	0	14306	100.0%
		100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%

Table 6 shows the test results. Overall accuracy is 96.7%. The highest accuracy is type “7” with an accuracy of 97.8%. The lowest accuracy found is type “4” with an accuracy of 90.9%.

Table 6: RF model test results

		Actual Type							
		1	2	3	4	5	6	7	
Predicted Type	1	61409	1484	1	0	25	3	203	97.3%
	2	2383	82818	114	0	380	93	32	96.5%
	3	1	119	10346	72	25	247	0	95.7%
	4	0	0	51	745	0	24	0	90.9%
	5	18	122	9	0	2391	6	2	93.8%
	6	9	78	153	30	8	4800	0	94.5%
	7	119	17	0	0	0	0	5967	97.8%
		96.0%	97.8%	96.9%	88.0%	84.5%	92.8%	96.2%	96.7%

Overall, the random forest models produced a high level of accuracy. This is likely due to the large number of variables included in the model which were able to accurately classify the cover type perfectly in the training data.

Model 3: Linear Discriminant Analysis

We also attempted to predict cover type using a linear discriminant model. The model reported high collinearity which can be expected as there often wasn't a lot of variation across the soil type variables. This could affect interpretation of the variables, but we are less concerned with interpretation for this analysis and more interested in the quality of prediction. **Table 7** shows the test results. Overall accuracy is 67.9%. The highest accuracy is type "2" with an accuracy of 75.7%. The lowest accuracy found is type "4" with an accuracy of 25.6%.

Table 7: LDA model training results

		Actual Type							
		1	2	3	4	5	6	7	
Predicted Type	1	91226	36159	0	0	55	0	2657	70.1%
	2	40691	151195	1237	0	4509	2034	58	75.7%
	3	125	3513	13501	585	613	3091	54	62.8%
	4	0	828	1957	1100	0	416	0	25.6%
	5	182	2811	188	0	1483	449	0	29.0%
	6	71	3109	8197	215	4	6204	0	34.9%
	7	15606	1048	0	0	0	0	11537	40.9%
		61.7%	76.1%	53.8%	57.9%	22.3%	50.9%	80.6%	67.9%

Table 8 shows the test results. Overall accuracy is 68%. The highest accuracy is type “2” with an accuracy of 75.5%. The lowest accuracy found is type “4” with an accuracy of 26.6%.

Table 8: LDA model test results

		Actual Type							
		1	2	3	4	5	6	7	
Predicted Type	1	39466	15267	0	0	20	0	1122	70.6%
	2	17546	64543	525	0	1932	878	24	75.5%
	3	57	1548	5768	236	236	1310	24	62.8%
	4	0	372	851	509	0	182	0	26.6%
	5	69	1168	78	0	637	196	0	29.7%
	6	25	1303	3452	102	4	2607	0	34.8%
	7	6776	437	0	0	0	0	5034	41.1%
		61.7%	76.3%	54.0%	60.1%	22.5%	50.4%	81.1%	68.0%

Model 4: K-Nearest Neighbors

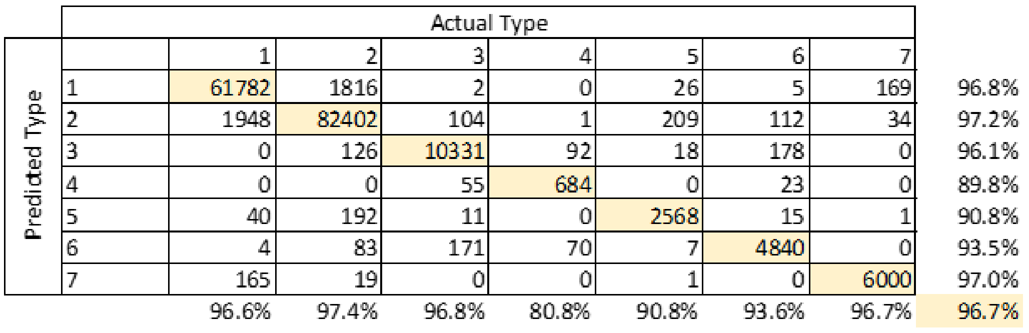
For the K-nearest neighbors model we standardized the numeric variables to provide better distance values. For models that included all variables in the sample, we compared 3 variations, a model using 1 neighbor, a model using 3 neighbors, and a model using 9 neighbors. **Table 9** displays the accuracy on the test dataset of each variation. We found that 3 nearest neighbors provided the most accurate model at 96.7%.

Table 9: KNN Accuracy Comparison

Model	Accuracy
KNN 1	96.4%
KNN 3	96.7%
KNN 9	95.9%

Table 10 shows the training results. Overall accuracy is 96.7%. The highest accuracy is type “2” with an accuracy of 97.2%. The lowest accuracy found is type “4” with an accuracy of 89.8%.

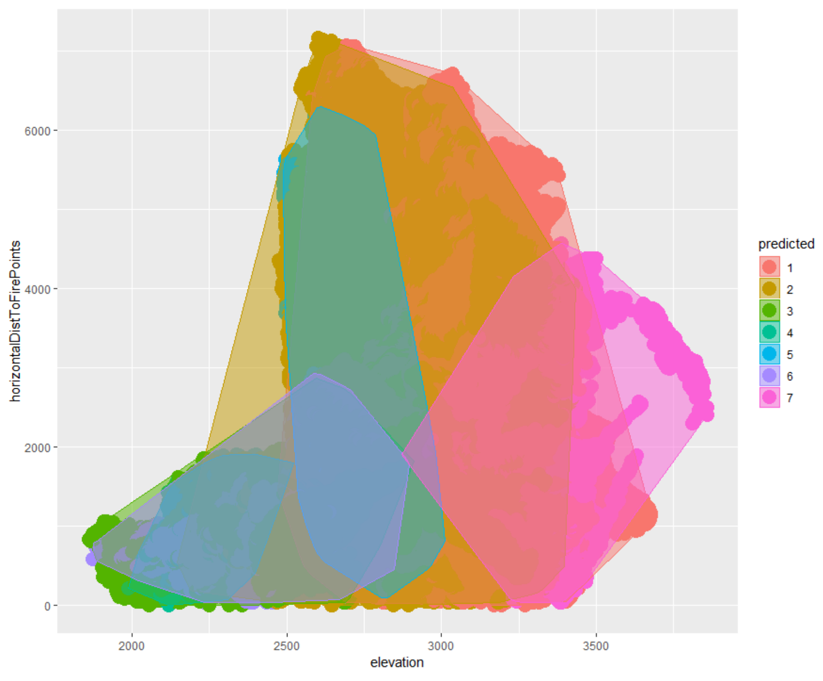
Table 10: KNN model training results



		Actual Type							
		1	2	3	4	5	6	7	
Predicted Type	1	61782	1816	2	0	26	5	169	96.8%
	2	1948	82402	104	1	209	112	34	97.2%
	3	0	126	10331	92	18	178	0	96.1%
	4	0	0	55	684	0	23	0	89.8%
	5	40	192	11	0	2568	15	1	90.8%
	6	4	83	171	70	7	4840	0	93.5%
	7	165	19	0	0	1	0	6000	97.0%
		96.6%	97.4%	96.8%	80.8%	90.8%	93.6%	96.7%	96.7%

The AUC for the KNN model is 0.9653. **Figure 12** displays the boundaries for the cover type classifications from the KNN model using elevation and horizontal distance to fire points. There is a lot of overlap in the boundaries, but the plot gives you an idea of how the model has classified observations.

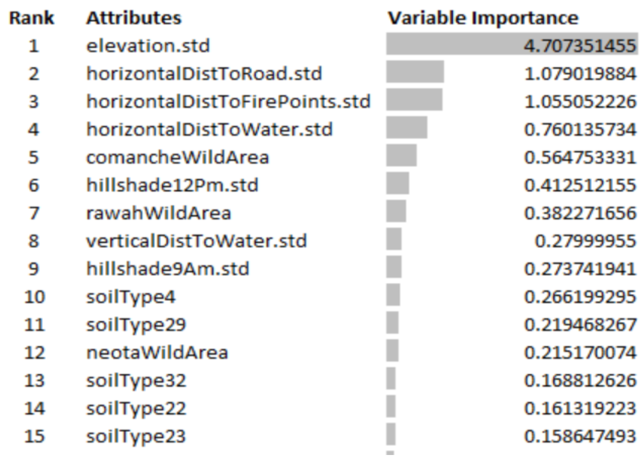
Figure 12: Boundaries for cover type classifications from the KNN model



Model 5: Conditional Inference Trees

We have also tried to predict forest cover type with conditional inference trees. **Table 11** shows the top 15 predictors. A we can see standardized elevation is the most predictive attribute to predict forest cover.

Table 11: Top 15 predictors for forest cover



Rank	Attributes	Variable Importance
1	elevation.std	4.707351455
2	horizontalDistToRoad.std	1.079019884
3	horizontalDistToFirePoints.std	1.055052226
4	horizontalDistToWater.std	0.760135734
5	comancheWildArea	0.564753331
6	hillshade12Pm.std	0.412512155
7	rawahWildArea	0.382271656
8	verticalDistToWater.std	0.27999955
9	hillshade9Am.std	0.273741941
10	soilType4	0.266199295
11	soilType29	0.219468267
12	neotaWildArea	0.215170074
13	soilType32	0.168812626
14	soilType22	0.161319223
15	soilType23	0.158647493

The tree is too big to print out. **Figure 13** is a simple decision tree with only two splits. We can see both splits determined by elevation. When standardized elevation is more than 0.3, the cover type will more likely be spruce/fir. When standardized elevation is less than -1.6, the cover type will more likely be ponderosa pine.

Figure 13: Decision Tree with two splits

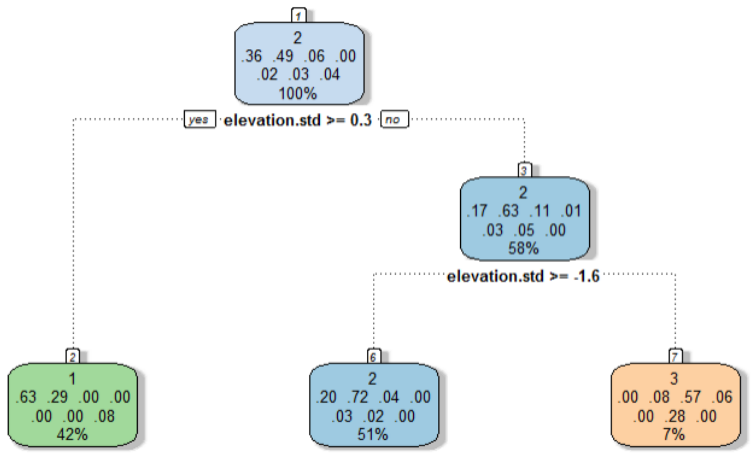
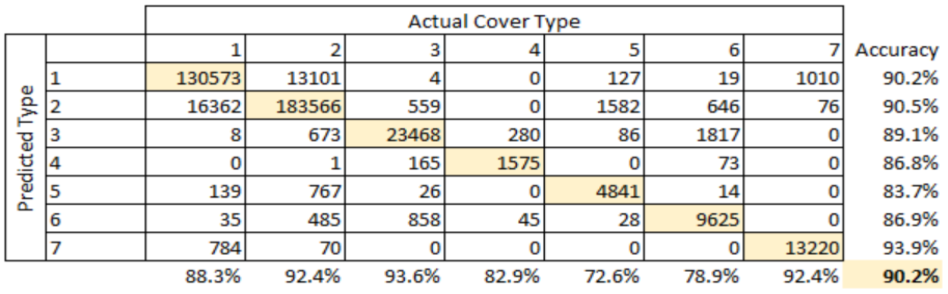


Table 12 shows the training results. Overall accuracy is 90.2%. The highest accuracy is type “7”, with accuracy 93.9%. The lowest accuracy found is type “5” with an accuracy of 83.7%.

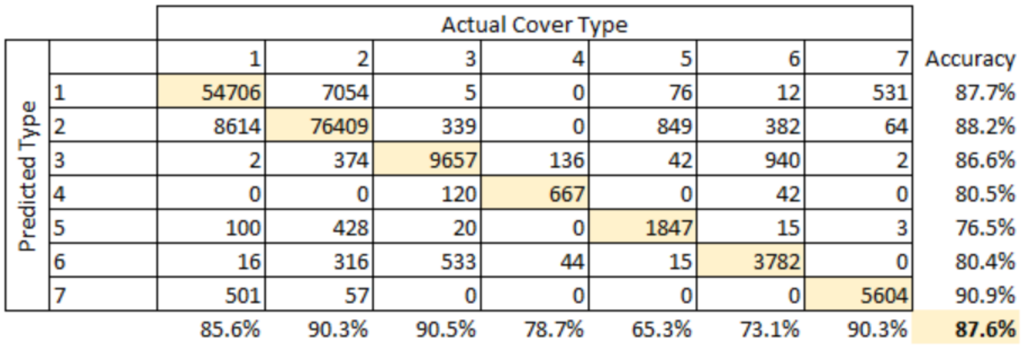
Table 12: CTree model training results



		Actual Cover Type							
		1	2	3	4	5	6	7	Accuracy
Predicted Type	1	130573	13101	4	0	127	19	1010	90.2%
	2	16362	183566	559	0	1582	646	76	90.5%
	3	8	673	23468	280	86	1817	0	89.1%
	4	0	1	165	1575	0	73	0	86.8%
	5	139	767	26	0	4841	14	0	83.7%
	6	35	485	858	45	28	9625	0	86.9%
	7	784	70	0	0	0	0	13220	93.9%
		88.3%	92.4%	93.6%	82.9%	72.6%	78.9%	92.4%	**90.2%**

Table 13 shows the test results. Overall accuracy is 87.6%. The highest accuracy is also found in type “7”, with accuracy 90.9%. The lowest accuracy found is type “5” with an accuracy of 76.5%. Compared to the training data results, the test data accuracy was lower.

Table 13: CTree model test results



		Actual Cover Type							
		1	2	3	4	5	6	7	Accuracy
Predicted Type	1	54706	7054	5	0	76	12	531	87.7%
	2	8614	76409	339	0	849	382	64	88.2%
	3	2	374	9657	136	42	940	2	86.6%
	4	0	0	120	667	0	42	0	80.5%
	5	100	428	20	0	1847	15	3	76.5%
	6	16	316	533	44	15	3782	0	80.4%
	7	501	57	0	0	0	0	5604	90.9%
		85.6%	90.3%	90.5%	78.7%	65.3%	73.1%	90.3%	**87.6%**

Model 6: SVM

The next model used were Support Vector Machines to predict forest cover types. For this method we fit the SVM with a radial kernel and a gamma of 0.0185. For this method we haven’t changed the value of cost. Increasing the value of cost can increase the accuracy of the training data but this may result in very irregular decision boundaries and risk overfitting the data. **Table 14** shows the training results. Overall accuracy is 79.2%. The highest accuracy is type “7” with an accuracy of 84.9%. The lowest accuracy found is type “6” with an accuracy of 66%.

Table 14: SVM model training results

		Actual Cover Type							
		1	2	3	4	5	6	7	Accuracy
Predicted Cover Type	1	111,182	23,856	-	-	218	12	3,533	80.1%
	2	35,245	171,439	1,858	3	5,183	2,254	86	79.3%
	3	28	1,535	21,765	914	174	4,845	-	74.4%
	4	-	4	192	847	-	70	-	76.1%
	5	14	212	10	-	1,043	7	-	81.1%
	6	86	1,085	1,203	159	28	4,969	-	66.0%
	7	1,733	180	-	-	-	-	10,738	84.9%
		75.0%	86.4%	87.0%	44.0%	15.7%	40.9%	74.8%	***79.2%***

Table 15 shows the test results. Overall accuracy is 79.3%. The highest accuracy is type “7” with an accuracy of 85.6%. The lowest accuracy found is type “6” with an accuracy of 66.2%.

Table 15: SVM model test results

		Actual Cover Type							
		1	2	3	4	5	6	7	Accuracy
Predicted Cover Type	1	47,707	10,003	-	-	115	1	1,463	80.5%
	2	15,068	73,723	844	1	2,224	1,000	30	79.4%
	3	13	617	9,332	417	87	2,092	-	74.3%
	4	-	1	74	341	-	29	-	76.6%
	5	5	99	9	-	415	2	-	78.3%
	6	43	482	467	65	6	2,086	-	66.2%
	7	716	65	-	-	-	-	4,660	85.6%
		75.1%	86.7%	87.0%	41.4%	14.6%	40.0%	75.7%	***79.3%***

Model 7: Gradient Boosting

The final model used to predict the forest cover type was the Gradient Boosting Machine. For this method, we used 600 trees, limiting the depth of each tree to 7 and used a value of 0.01 for the shrinkage parameter. **Table 16** shows the training results.

Overall accuracy is 77.9%. The highest accuracy is type “7” with an accuracy of 88.6%. The lowest accuracy found is type “6” with an accuracy of 69.7%.

Table 16: GBM model training results

		Actual Cover Type							
		1	2	3	4	5	6	7	**Accuracy**
Predicted Cover Type	1	111,215	30,032	-	-	34	-	4,290	76.4%
	2	35,804	165,316	2,400	1	4,595	2,527	88	78.4%
	3	28	1,443	21,276	389	157	3,892	-	78.3%
	4	-	7	136	1,480	-	51	-	88.4%
	5	34	246	-	-	1,854	-	-	86.9%
	6	21	1,172	1,216	53	6	5,687	-	69.7%
	7	1,186	95	-	-	-	-	9,979	88.6%
		75.0%	83.4%	85.0%	77.0%	27.9%	46.8%	69.5%	***77.9%***

Table 17 shows the test results. Overall accuracy is 77.8%. The highest accuracy is type “7” with an accuracy of 89.4%. The lowest accuracy found is type “6” with an accuracy of 68%.

Table 17: GBM model test results

		Actual Cover Type							
		1	2	3	4	5	6	7	**Accuracy**
Predicted Cover Type	1	47,728	12,891	-	-	8	-	1,834	76.4%
	2	15,315	70,847	1,043	-	1,996	1,111	21	78.4%
	3	13	607	9,029	185	82	1,746	-	77.4%
	4	-	2	84	613	-	25	-	84.7%
	5	20	106	-	-	760	-	-	85.8%
	6	7	494	570	26	1	2,328	-	68.0%
	7	469	43	-	-	-	-	4,298	89.4%
		75.1%	83.4%	84.2%	74.4%	26.7%	44.7%	69.9%	***77.8%***

5.4 Model Results:

Table 18: Training data model evaluation criteria results

		Accuracy	**AUC**
Model 1	Naïve Bayes	0.488	0.500

Model 2	Random Forest	1	1
Model 3	Linear Discriminant Analysis	0.6792	0.7101
Model 4	K-Nearest Neighbors	NA	NA
Model 5	Conditional Inference Trees	0.9020	0.9271
Model 6	SVM	0.7917	0.8492
Model 7	Gradient Boosting	0.7790	0.8709

Table 19: Test data model evaluation criteria results

		Accuracy	**AUC**
Model 1	Naïve Bayes	0.486	0.500
Model 2	Random Forest	0.9668	0.9707
Model 3	Linear Discriminant Analysis	0.6802	0.7097
Model 4	K-Nearest Neighbors	0.9673	0.9653
Model 5	Conditional Inference Trees	0.8759	0.8985
Model 6	SVM	0.7932	0.8482
Model 7	Gradient Boosting	0.7780	0.8681

6. Conclusion

Throughout this report, we developed and utilized multiple machine learning models to predict whether a cover type is Spruce/Fir, Cottonwood/Willow, Krummholz, Lodgepole, Aspen, Ponderosa and Douglas Fir. The machine learning methods utilized were Naïve Bayes, Random Forest, Linear Discriminant Analysis, K-Nearest Neighbors, Conditional Inference Trees, SVM and Gradient Boosting. Of each method used, Random Forest and K-Nearest Neighbors were within 1/10th of a percent of each other and did very well in comparison to the other models with at least 97% accuracy; Conditional Inference Trees also did very well with about 88% accuracy.

Our team has limited experience with the Naïve Bayes method which is more than likely why the model did not show good results. Some improvements to this model could be to use log probabilities, segment the data, use a smaller sample size, or remove redundant features.

Our SVM and GBM models were between 78-79% accurate, which is not too bad. The SVM mode can be improved by pre-processing of features (i.e.: normalization and various transformations), decision of the choice of kernel, and proper tuning of the hyperparameters. For the GBM, there are a lot of tuning options for improving the accuracy. Finagling with the tree-specific and boosting parameters can change these models significantly.

R Code Repository

```
library(ggplot2)
library(dplyr)
library(GGally)
library(clusterSim)
library(caret)
library(glmnet)
library(tree)
library(randomForest)
library(gbm)
library(psych)
library(caret)
library(tidyr)
library(dplyr)

#------------------------------------------
#This section needs to use 100% of the data
#------------------------------------------

# read in our data
covtype <- read.csv("covtype.csv", header = TRUE)

# name the columns
names(covtype) = c("elevation", "aspect", "slope", "horizontalDistToWater", "verticalDistToWater",
                  "horizontalDistToRoad", "hillshade9Am", "hillshade12Pm", "hillshade3Pm",
             "horizontalDistToFirePoints","rawahWildArea", "neotaWildArea","comancheWildArea",
             "cacheWildArea","soilType1","soilType2","soilType3","soilType4","soilType5",
             "soilType6","soilType7","soilType8","soilType9","soilType10","soilType11",
                  "soilType12","soilType13","soilType14","soilType15","soilType16","soilType17",
             "soilType18","soilType19","soilType20","soilType21","soilType22","soilType23",
             "soilType24","soilType25","soilType26","soilType27","soilType28","soilType29",
             "soilType30","soilType31","soilType32","soilType33","soilType34","soilType35",
                  "soilType36","soilType37","soilType38","soilType39","soilType40","coverType")

str(covtype)
#581,011 observations, 55 variables

summary(covtype)
describe(covtype)

apply(is.na(covtype),2,sum)
#no missing values; therefore, no imputation needed

#-------------
#Density plots
#-------------

par(mfrow=c(3,4))
dp_elevation <- density(covtype$elevation)
dp_aspect <- density(covtype$aspect)
dp_slope <- density(covtype$slope)
dp_horizontalDistToWater <- density(covtype$horizontalDistToWater)
dp_verticalDistToWater <- density(covtype$verticalDistToWater)
```

```
dp_horizontalDistToRoad <- density(covtype$horizontalDistToRoad)
dp_horizontalDistToFirePoints <- density(covtype$horizontalDistToFirePoints)
dp_hillshade9Am <- density(covtype$hillshade9Am)
dp_hillshade12Pm <- density(covtype$hillshade12Pm)
dp_hillshade3Pm <- density(covtype$hillshade3Pm)
dp_rawahWildArea <- density(covtype$rawahWildArea)
dp_neotaWildArea <- density(covtype$neotaWildArea)
dp_comancheWildArea <- density(covtype$comancheWildArea)
dp_cacheWildArea <- density(covtype$cacheWildArea)
dp_soilType1 <- density(covtype$soilType1)
dp_soilType2 <- density(covtype$soilType2)
dp_soilType3 <- density(covtype$soilType3)
dp_soilType4 <- density(covtype$soilType4)
dp_soilType5 <- density(covtype$soilType5)
dp_soilType6 <- density(covtype$soilType6)
dp_soilType7 <- density(covtype$soilType7)
dp_soilType8 <- density(covtype$soilType8)
dp_soilType9 <- density(covtype$soilType9)
dp_soilType10 <- density(covtype$soilType10)
dp_soilType11 <- density(covtype$soilType11)
dp_soilType12 <- density(covtype$soilType12)
dp_soilType13 <- density(covtype$soilType13)
dp_soilType14 <- density(covtype$soilType14)
dp_soilType15 <- density(covtype$soilType15)
dp_soilType16 <- density(covtype$soilType16)
dp_soilType17 <- density(covtype$soilType17)
dp_soilType18 <- density(covtype$soilType18)
dp_soilType19 <- density(covtype$soilType19)
dp_soilType20 <- density(covtype$soilType20)
dp_soilType21 <- density(covtype$soilType21)
dp_soilType22 <- density(covtype$soilType22)
dp_soilType23 <- density(covtype$soilType23)
dp_soilType24 <- density(covtype$soilType24)
dp_soilType25 <- density(covtype$soilType25)
dp_soilType26 <- density(covtype$soilType26)
dp_soilType27 <- density(covtype$soilType27)
dp_soilType28 <- density(covtype$soilType28)
dp_soilType29 <- density(covtype$soilType29)
dp_soilType30 <- density(covtype$soilType30)
dp_soilType31 <- density(covtype$soilType31)
dp_soilType32 <- density(covtype$soilType32)
dp_soilType33 <- density(covtype$soilType33)
dp_soilType34 <- density(covtype$soilType34)
dp_soilType35 <- density(covtype$soilType35)
dp_soilType36 <- density(covtype$soilType36)
dp_soilType37 <- density(covtype$soilType37)
dp_soilType38 <- density(covtype$soilType38)
dp_soilType39 <- density(covtype$soilType39)
dp_soilType40 <- density(covtype$soilType40)
dp_coverType <- density(covtype$coverType)

plot(dp_elevation, main = "elevation Density")
plot(dp_aspect, main = "aspect Density")
plot(dp_slope, main = "slope Density")
plot(dp_horizontalDistToWater, main = "horizontalDistToWater Density")
plot(dp_verticalDistToWater, main = "verticalDistToWater Density")
plot(dp_horizontalDistToRoad, main = "horizontalDistToRoad Density")
plot(dp_horizontalDistToFirePoints, main = "horizontalDistToFirePoints Density")
plot(dp_hillshade9Am, main = "hillshade9Am Density")
plot(dp_hillshade12Pm, main = "hillshade12Pm Density")
plot(dp_hillshade3Pm, main = "hillshade3Pm Density")
plot(dp_rawahWildArea, main = "rawahWildArea Density")
plot(dp_neotaWildArea, main = "neotaWildArea Density")
plot(dp_comancheWildArea, main = "comancheWildArea Density")
plot(dp_cacheWildArea, main = "cacheWildArea Density")
plot(dp_soilType1, main = "soilType1 Density")
plot(dp_soilType2, main = "soilType2 Density")
plot(dp_soilType3, main = "soilType3 Density")
plot(dp_soilType4, main = "soilType4 Density")
plot(dp_soilType5, main = "soilType5 Density")
plot(dp_soilType6, main = "soilType6 Density")
plot(dp_soilType7, main = "soilType7 Density")
plot(dp_soilType8, main = "soilType8 Density")
plot(dp_soilType9, main = "soilType9 Density")
plot(dp_soilType10, main = "soilType10 Density")
plot(dp_soilType11, main = "soilType11 Density")
plot(dp_soilType12, main = "soilType12 Density")
plot(dp_soilType13, main = "soilType13 Density")
plot(dp_soilType14, main = "soilType14 Density")
plot(dp_soilType15, main = "soilType15 Density")
plot(dp_soilType16, main = "soilType16 Density")
plot(dp_soilType17, main = "soilType17 Density")
plot(dp_soilType18, main = "soilType18 Density")
plot(dp_soilType19, main = "soilType19 Density")
plot(dp_soilType20, main = "soilType20 Density")
plot(dp_soilType21, main = "soilType21 Density")
plot(dp_soilType22, main = "soilType22 Density")
plot(dp_soilType23, main = "soilType23 Density")
plot(dp_soilType24, main = "soilType24 Density")
plot(dp_soilType25, main = "soilType25 Density")
plot(dp_soilType26, main = "soilType26 Density")
plot(dp_soilType27, main = "soilType27 Density")
plot(dp_soilType28, main = "soilType28 Density")
plot(dp_soilType29, main = "soilType29 Density")
plot(dp_soilType30, main = "soilType30 Density")
plot(dp_soilType31, main = "soilType31 Density")
plot(dp_soilType32, main = "soilType32 Density")
plot(dp_soilType33, main = "soilType33 Density")
plot(dp_soilType34, main = "soilType34 Density")
plot(dp_soilType35, main = "soilType35 Density")
plot(dp_soilType36, main = "soilType36 Density")
plot(dp_soilType37, main = "soilType37 Density")
plot(dp_soilType38, main = "soilType38 Density")
plot(dp_soilType39, main = "soilType39 Density")
plot(dp_soilType40, main = "soilType40 Density")

dev.off()

#---------------------
#Boxplots of variables
#---------------------
```

```
#All variables
boxplot(covtype, main = "Boxplot of all variables")

#BP of horizontalDistToWater and verticalDistToWater
boxplot(covtype$horizontalDistToWater, covtype$verticalDistToWater,
        main = "Boxplot of horizontalDistToWater and verticalDistToWater")

#BP of horizontalDistToRoad and horizontalDistToFirePoints
boxplot(covtype$horizontalDistToRoad, covtype$horizontalDistToFirePoints,
        main = "Boxplots of horizontalDistToRoad and horizontalDistToFirePoints")

#------------------------------------------------------
#Review a quick scatterplot of the data, minus soil types
#------------------------------------------------------

pairs(~elevation+aspect+slope+horizontalDistToWater+verticalDistToWater+horizontalDistToRoad+
      hillshade9Am+hillshade12Pm+hillshade3Pm+horizontalDistToFirePoints+rawahWildArea+neotaWild
Area+
        comancheWildArea+cacheWildArea+coverType, data = covtype, main="Forest Cover Data
Scatterplot")



#------------------------------------------------------------------------
#Review a quick scatterplot of the soil types, wilderness areas, hillshade
#------------------------------------------------------------------------
pairs(~hillshade9Am+hillshade12Pm+hillshade3Pm+rawahWildArea+neotaWildArea+
      comancheWildArea+cacheWildArea+soilType1+soilType2+soilType3+soilType4+soilType5+soilTyp
e6+soilType7+soilType8+
      soilType9+soilType10+soilType11+soilType12+soilType13+soilType14+soilType15+soilType16+soil
Type17+soilType18+

        soilType19+soilType20+soilType21+soilType22+soilType23+soilType24+soilType25+soilType26
+soilType27+soilType28+

        soilType29+soilType30+soilType31+soilType32+soilType33+soilType34+soilType35+soilType36
+soilType37+soilType38+
        soilType39+soilType40+coverType, data = covtype, main="Forest Cover Data Scatterplot")

#----------------------------------
#See if any features are correlated
#----------------------------------

require(corrgram)
corrgram.data <- covtype[, c("elevation", "aspect", "slope", "horizontalDistToWater",
"verticalDistToWater",
                        "horizontalDistToRoad", "hillshade9Am", "hillshade12Pm", "hillshade3Pm",
                        "horizontalDistToFirePoints","rawahWildArea",
"neotaWildArea","comancheWildArea",
                        "cacheWildArea","soilType1","soilType2","soilType3","soilType4","soilType5",
                        "soilType6","soilType7","soilType8","soilType9","soilType10","soilType11",
                          "soilType12","soilType13","soilType14","soilType15","soilType16","soilType17",
                         "soilType18","soilType19","soilType20","soilType21","soilType22","soilType23",
                        "soilType24","soilType25","soilType26","soilType27","soilType28","soilType29",
                        "soilType30","soilType31","soilType32","soilType33","soilType34","soilType35",
                          "soilType36","soilType37","soilType38","soilType39","soilType40","coverType")]

corrgram.vars <- c("elevation", "aspect", "slope", "horizontalDistToWater", "verticalDistToWater",
              "horizontalDistToRoad", "hillshade9Am", "hillshade12Pm", "hillshade3Pm",
              "horizontalDistToFirePoints","rawahWildArea", "neotaWildArea","comancheWildArea",
              "cacheWildArea","soilType1","soilType2","soilType3","soilType4","soilType5",
              "soilType6","soilType7","soilType8","soilType9","soilType10","soilType11",
                 "soilType12","soilType13","soilType14","soilType15","soilType16","soilType17",
              "soilType18","soilType19","soilType20","soilType21","soilType22","soilType23",
              "soilType24","soilType25","soilType26","soilType27","soilType28","soilType29",
              "soilType30","soilType31","soilType32","soilType33","soilType34","soilType35",
                 "soilType36","soilType37","soilType38","soilType39","soilType40","coverType")

corrgram(corrgram.data[,corrgram.vars], order=FALSE, lower.panel=panel.shade, upper.panel=panel.pie,
         diag.panel=panel.minmax, text.panel=panel.txt, main="Forest Cover Data")



forestTrainCor=cor(covtype[,1:10])
forestTrainCor

ggpairs(data = covtype, columns = 1:10, title = "Correlations of All Except Binary",
         mapping = aes(colour = Cover_Type, alpha = .3))
ggpairs(data = covtype, columns = 7:9, title = "Correlations of Hillshades",
         mapping = aes(colour = Cover_Type, alpha = .3))
#-----------------------------------------------------------------------------------------------------------------------

setwd("G:/My Drive/MSPA/MSDS 454/Project")
library(readr)
library(lessR)
library(plyr)
library(randomForest)
library(e1071)
library(naivebayes)

# read in our data
covtype <- read_csv("Forest_Cover/covtype.data/covtype.csv", col_names = FALSE)

# name the columns
names(covtype) = c("elevation", "aspect", "slope", "horizontalDistToWater",
"verticalDistToWater",
            "horizontalDistToRoad", "hillshade9Am", "hillshade12Pm", "hillshade3Pm",
            "horizontalDistToFirePoints","rawahWildArea",
"neotaWildArea","comancheWildArea",
            "cacheWildArea","soilType1","soilType2","soilType3","soilType4","soilType5",
            "soilType6","soilType7","soilType8","soilType9","soilType10","soilType11",
            "soilType12","soilType13","soilType14","soilType15","soilType16","soilType17",
            "soilType18","soilType19","soilType20","soilType21","soilType22","soilType23",
            "soilType24","soilType25","soilType26","soilType27","soilType28","soilType29",
            "soilType30","soilType31","soilType32","soilType33","soilType34","soilType35",
            "soilType36","soilType37","soilType38","soilType39","soilType40","coverType")
```

```
# start with some basics
summary(covtype)

# now let's check out our datatypes
str(covtype)

# look at our variable counts for all of our categorical variables
catVars<-
c('soilType1','soilType2','soilType3','soilType4','soilType5','soilType6','soilType7','soilType8',
          'soilType9','soilType10','soilType11','soilType12','soilType13','soilType14','soilType15','soi
lType16',
          'soilType17','soilType18','soilType19','soilType20','soilType21','soilType22','soilType23','s
oilType24',
          'soilType25','soilType26','soilType27','soilType28','soilType29','soilType30','soilType31','s
oilType32',
          'soilType33','soilType34','soilType35','soilType36','soilType37','soilType38','soilType39','s
oilType40',
          'rawahWildArea','neotaWildArea','comancheWildArea','cacheWildArea')

for (i in catVars)
  {
  print(count(covtype,i))
  }

# at this point our data is clean, let's look at the distribution of our target variable
hist(covtype$coverType, main = "Histogram of Cover Types", xlab = "Cover Type",col="grey")

# important takeaway here is the fact that we see a ton of 1's and 2's and then much less 3-7
# we should be cautious to validate that our predictions follow a similar distribution

# also saw that our vertical distance to water had at least 1 negative value in there, let's
# make sure that looks legit
boxplot(covtype$verticalDistToWater, main = "Vertical Distance to Water", horizontal = TRUE)

# now let's see if there are any patterns between our continuous variables and our target
contVars<- c("elevation","aspect","slope","horizontalDistToWater","verticalDistToWater",
            "horizontalDistToRoad","hillshade9Am","hillshade12Pm","hillshade3Pm",
            "horizontalDistToFirePoints")

#for(i in contVars)
#  {
#  boxplot( i ~ coverType, data=covtype,main=paste(v,"by coverType"), xlab = v)
#  }

boxplot(elevation ~ coverType,data=covtype,main=paste("Elevation by Cover Type"),
       col="gray",xlab = "Elevation",horizontal = TRUE)

boxplot(horizontalDistToRoad ~ coverType,data=covtype,main=paste("Horiz Dist to Rd by
Cover Type"),
	col="gray",xlab = "Horiz Dist to Rd",horizontal = TRUE)

covtype$elevation.std<-(covtype$elevation-mean(covtype$elevation))/sd(covtype$elevation)
covtype$aspect.std<-(covtype$aspect-mean(covtype$aspect))/sd(covtype$aspect)
covtype$slope.std<-(covtype$slope-mean(covtype$slope))/sd(covtype$slope)
covtype$horizontalDistToWater.std<-(covtype$horizontalDistToWater-
mean(covtype$horizontalDistToWater))/
  sd(covtype$horizontalDistToWater)
covtype$verticalDistToWater.std<-(covtype$verticalDistToWater-
mean(covtype$verticalDistToWater))/
  sd(covtype$verticalDistToWater)
covtype$horizontalDistToRoad.std<-(covtype$horizontalDistToRoad-
mean(covtype$horizontalDistToRoad))/
  sd(covtype$horizontalDistToRoad)
covtype$hillshade9Am.std<-(covtype$hillshade9Am-
mean(covtype$hillshade9Am))/sd(covtype$hillshade9Am)
covtype$hillshade12Pm.std<-(covtype$hillshade12Pm-
mean(covtype$hillshade12Pm))/sd(covtype$hillshade12Pm)
covtype$hillshade3Pm.std<-(covtype$hillshade3Pm-
mean(covtype$hillshade3Pm))/sd(covtype$hillshade3Pm)
covtype$horizontalDistToFirePoints.std<-(covtype$horizontalDistToFirePoints-
mean(covtype$horizontalDistToFirePoints))/
  sd(covtype$horizontalDistToFirePoints)

# seeing some interesting stuff there, let's now take a look at the distributions of each of our
continuous variables
# will also show Q-Q plots for each
par(mfrow = c(1,1))
hist(covtype$elevation.std,main="Std Elevation",  xlab ="Std Elevation",col="gray" )
#qqnorm(covtype$elevation.std,main = "Q-Q Plot of std elevation", col = "red")
#qqline(covtype$elevation.std, col = "black")
plot()

hist(covtype$aspect.std, main = "Std Aspect", xlab ="Std Aspect",col="gray" )
#qqnorm(covtype$aspect.std,main = "Q-Q Plot of std aspect", col = "red")
#qqline(covtype$aspect.std, col = "black")
plot()

hist(covtype$slope.std, main = "Std Slope", xlab ="Std Slope",col="gray" )
#qqnorm(covtype$slope.std,main = "Q-Q Plot of std slope", col = "red")
#qqline(covtype$slope.std, col = "black")
plot()

hist(covtype$horizontalDistToWater.std, main = "Std Horiz Dist Water", xlab ="Std Horiz Dist
Water",col="gray" )
#qqnorm(covtype$horizontalDistToWater.std,main = "Q-Q Plot of std horizontalDistToWater",
col = "red")
#qqline(covtype$horizontalDistToWater.std, col = "black")
plot()
```

```
hist(covtype$verticalDistToWater.std,main="std verticalDistToWater")
#qqnorm(covtype$verticalDistToWater.std,main = "Q-Q Plot of std verticalDistToWater", col =
"red")
#qqline(covtype$verticalDistToWater.std, col = "black")
plot()

hist(covtype$horizontalDistToRoad.std,main="std horizontalDistToRoad")
#qqnorm(covtype$horizontalDistToRoad.std,main = "Q-Q Plot of std horizontalDistToRoad", col
= "red")
#qqline(covtype$horizontalDistToRoad.std, col = "black")
plot()

hist(covtype$hillshade9Am.std,main="std hillshade9Am")
#qqnorm(covtype$hillshade9Am.std,main = "Q-Q Plot of std hillshade9Am", col = "red")
#qqline(covtype$hillshade9Am.std, col = "black")
plot()

hist(covtype$hillshade12Pm.std,main="std hillshade12Pm")
#qqnorm(covtype$hillshade12Pm.std,main = "Q-Q Plot of std hillshade12Pm", col = "red")
#qqline(covtype$hillshade12Pm.std, col = "black")
plot()

hist(covtype$hillshade3Pm.std,main="std hillshade3Pm")
#qqnorm(covtype$hillshade3Pm.std,main = "Q-Q Plot of std hillshade3Pm", col = "red")
#qqline(covtype$hillshade3Pm.std, col = "black")
plot()

hist(covtype$horizontalDistToFirePoints.std,main="std horizontalDistToFirePoints")
#qqnorm(covtype$horizontalDistToFirePoints.std,main = "Q-Q Plot of std
horizontalDistToFirePoints", col = "red")
#qqline(covtype$horizontalDistToFirePoints.std, col = "black")
plot()

# let's display 2x2 for the doc
par(mfrow = c(2,2))
hist(covtype$slope.std, main = "Std Slope", xlab ="Std Slope",col="gray" )
hist(covtype$hillshade12Pm.std, main = "Std 12 PM Hillshade", xlab ="Std 12 PM
Hillshade",col="gray" )
hist(covtype$horizontalDistToWater.std, main = "Std Horiz Dist to Water", xlab ="Std Horiz Dist
to Water",col="gray" )
hist(covtype$horizontalDistToRoad.std, main = "Std Horiz Dist to Road", xlab ="Std Horiz Dist to
Road",col="gray" )
plot()

# and the good ones
par(mfrow = c(1,2))
hist(covtype$elevation.std, main = "Std Elevation", xlab ="Std Elevation",col="gray" )
hist(covtype$hillshade3Pm.std, main = "Std 3 PM Hillshade", xlab ="Std 3 PM
Hillshade",col="gray" )
plot()
```

```
# we will keep our standardized continuous data and drop the raw values now
covtype<-subset(covtype, select=-
c(elevation,aspect,slope,horizontalDistToWater,verticalDistToWater,
                                horizontalDistToRoad,hillshade9Am,hillshade12Pm,hillshade3Pm,
                                horizontalDistToFirePoints))

# now we will split into 70% train and 30% test data
set.seed(1)
train_index <- sample(1:nrow(covtype), 0.7 * nrow(covtype))
test_index <- setdiff(1:nrow(covtype), train_index)
covtype_train <- covtype[train_index,]
covtype_test <- covtype[test_index,]

# function that spits out test and train confusion matrices and accuracy
printALL=function(model){
  trainPred=predict(model, newdata = covtype_train, type = "class")
  trainTable=table(covtype_train$coverType, trainPred)
  testPred=predict(model, newdata=covtype_test, type="class")
  testTable=table(covtype_test$coverType, testPred)
  trainAcc=(trainTable[1,1]+trainTable[2,2]+trainTable[3,3])/sum(trainTable)
  testAcc=(testTable[1,1]+testTable[2,2]+testTable[3,3])/sum(testTable)
  message("Contingency Table for Training Data")
  print(trainTable)
  message("Contingency Table for Test Data")
  print(testTable)
  message("Accuracy")
  print(round(cbind(trainAccuracy=trainAcc, testAccuracy=testAcc),3))
}

# now we try our first random forest model
rf.model<-randomForest(coverType ~ ., data=covtype_train, ntree=50)
printALL(rf.model)

# Naive Bayes model setup and execution, doesn't perform very well
nb_1=naive_bayes(x = covtype_train[,!names(covtype_train) =="coverType"],
                         y = covtype_train$coverType,usekernel=FALSE)
printALL(nb_1)

nb_2=naive_bayes(x = covtype_train[,!names(covtype_train) =="coverType"],
               y = covtype_train$coverType,usekernel=TRUE)
printALL(nb_2)

nb_3=naive_bayes(x = covtype_train[,!names(covtype_train) =="coverType"],
               y = covtype_train$coverType,usekernel=TRUE,laplace = 1)
printALL(nb_3)

nb_4=naive_bayes(x = covtype_train[,!names(covtype_train) =="coverType"],
               y = covtype_train$coverType,usekernel=FALSE,laplace = 10)
printALL(nb_4)
```

```
#Random Forest

# Using For loop to identify the right mtry for model
oob.err=double(13)
test.err=double(13)

for (mtry in 1:50) {
  model3 <- randomForest(coverType ~ ., data =covtype_train, ntree = 75, mtry=mtry)
  oob.err[mtry] = model3$err.rate[75]

  pred <- predict(model3, covtype_test, type = "class")
  test.err[mtry] = mean(pred == covtype_test$coverType)

  cat(mtry, " ")
}

matplot(1:mtry, cbind(oob.err,1-test.err), pch=19,
        col = c("red", "blue"), type = "b", ylab = "Mean Squared Error",
        xlab = "Number of Predictors Considered at each Split")
legend("topright", legend = c("Out of Bag Error", "Test Error"), pch = 19,
        col = c("red", "blue"))

#LDA

# Fit the model
model_lda <- lda(coverType ~ ., data = covtype_train)
model_lda
# Make predictions
predicted.lda <- predict (model_lda , covtype_test)
lda.class = predicted.lda$class
table(lda.class, covtype_test$coverType)
# Model accuracy
mean(lda.class == covtype_test$coverType)


#models with standardized variables
covtype2 = covtype
covtype2$elevation <- (covtype2$elevation - mean(covtype2$elevation)) /
sd(covtype2$elevation)
covtype2$aspect <- (covtype2$aspect - mean(covtype2$aspect)) / sd(covtype2$aspect)
covtype2$slope <- (covtype2$slope - mean(covtype2$slope)) / sd(covtype2$slope)
covtype2$horizontalDistToWater <- (covtype2$horizontalDistToWater -
mean(covtype2$horizontalDistToWater)) / sd(covtype2$horizontalDistToWater)
covtype2$verticalDistToWater <- (covtype2$verticalDistToWater -
mean(covtype2$verticalDistToWater)) / sd(covtype2$verticalDistToWater)
covtype2$horizontalDistToRoad <- (covtype2$horizontalDistToRoad -
mean(covtype2$horizontalDistToRoad)) / sd(covtype2$horizontalDistToRoad)
covtype2$hillshade9Am <- (covtype2$hillshade9Am - mean(covtype2$hillshade9Am)) /
sd(covtype2$hillshade9Am)
```

```
covtype2$hillshade12Pm <- (covtype2$hillshade12Pm - mean(covtype2$hillshade12Pm)) /
sd(covtype2$hillshade12Pm)
covtype2$hillshade3Pm <- (covtype2$hillshade3Pm - mean(covtype2$hillshade3Pm)) /
sd(covtype2$hillshade3Pm)
covtype2$horizontalDistToFirePoints <- (covtype2$horizontalDistToFirePoints -
mean(covtype2$horizontalDistToFirePoints)) / sd(covtype2$horizontalDistToFirePoints)


set.seed(1)
train_index2 <- sample(1:nrow(covtype2), 0.7 * nrow(covtype2), replace = FALSE)
test_index2 <- setdiff(1:nrow(covtype2), train_index2)
covtype_train_std <- covtype[train_index2,]
covtype_test_std <- covtype[test_index2,]

#KNN
library(class)
attach (covtype_train_std)
train.x <- cbind(elevation, aspect, slope, horizontalDistToWater, verticalDistToWater,
                 horizontalDistToRoad, hillshade9Am, hillshade12Pm, hillshade3Pm,
                 horizontalDistToFirePoints,rawahWildArea, neotaWildArea,comancheWildArea,
                 cacheWildArea,soilType1,soilType2,soilType3,soilType4,soilType5,
                 soilType6,soilType7,soilType8,soilType9,soilType10,soilType11,
                 soilType12,soilType13,soilType14,soilType15,soilType16,soilType17,
                 soilType18,soilType19,soilType20,soilType21,soilType22,soilType23,
                 soilType24,soilType25,soilType26,soilType27,soilType28,soilType29,
                 soilType30,soilType31,soilType32,soilType33,soilType34,soilType35,
                 soilType36,soilType37,soilType38,soilType39,soilType40)
detach(covtype_train_std)
attach (covtype_test_std)
test.x <- cbind(elevation, aspect, slope, horizontalDistToWater, verticalDistToWater,
                 horizontalDistToRoad, hillshade9Am, hillshade12Pm, hillshade3Pm,
                 horizontalDistToFirePoints,rawahWildArea, neotaWildArea,comancheWildArea,
                 cacheWildArea,soilType1,soilType2,soilType3,soilType4,soilType5,
                 soilType6,soilType7,soilType8,soilType9,soilType10,soilType11,
                 soilType12,soilType13,soilType14,soilType15,soilType16,soilType17,
                 soilType18,soilType19,soilType20,soilType21,soilType22,soilType23,
                 soilType24,soilType25,soilType26,soilType27,soilType28,soilType29,
                 soilType30,soilType31,soilType32,soilType33,soilType34,soilType35,
                 soilType36,soilType37,soilType38,soilType39,soilType40)
detach(covtype_test_std)
train.covertype =covtype_train_std$coverType

knn.pred=knn(train.x,test.x,train.covertype,k=1)
table(knn.pred ,covtype_test_std$coverType)
mean(knn.pred == covtype_test_std$coverType)

knn.pred3=knn(train.x,test.x,train.covertype,k=3)
table(knn.pred3 ,covtype_test_std$coverType)
mean(knn.pred3 == covtype_test_std$coverType)
```

```
knn.pred9=knn(train.x,test.x,train.covertype,k=9)
table(knn.pred9 ,covtype_test_std$coverType)
mean(knn.pred9 == covtype_test_std$coverType)


#SVM

for_cov = read.csv('C:/Users/Sivakumar/Documents/covtype.data.csv')
library(caret)
for_cov$Cover_Type<-as.factor(for_cov$Cover_Type)
set.seed(42)
index <- createDataPartition(for_cov$Cover_Type, p = 0.7, list = FALSE)
train_data <- for_cov[index, ]
test_data  <- for_cov[-index, ]
dim(train_data)
scl <- function(x){ (x - min(x))/(max(x) - min(x)) }
train_data[, 1:10] <- data.frame(lapply(train_data[, 1:10], scl))
x_train = train_data[,1:54]
y_train = train_data[,55]

test_data[,1:10] <- data.frame(lapply(test_data[, 1:10], scl))
x_test = test_data[,1:54]
y_test = test_data[,55]

#Support Vector Machines
library(e1071)
svm_model1 <- svm(x_train,y_train)
summary(svm_model1)

pred<-predict(svm_model1,x_test)
table(pred,y_test)

pred_tr<-predict(svm_model1,x_train)
table(pred_tr,y_train)

library(pROC)
library(multiROC)
multiclass.roc(as.numeric(pred),as.numeric(test_data$Cover_Type))
multiclass.roc(as.numeric(pred_tr),as.numeric(train_data$Cover_Type))
```

<u>Output:</u>

```
Call:
svm.default(x = x_train, y = y_train)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.01851852
```

```
Number of Support Vectors:  211925

 ( 6359 89130 83145 7823 12660 11097 1711 )


Number of Classes:  7

Levels:
 1 2 3 4 5 6 7
```

GBM:

```
library(gbm)

BST = gbm(Cover_Type~.,data=train_data,
        distribution='multinomial',
        n.trees=600,
        interaction.depth=7,
        #cv.folds=5,
        shrinkage=0.01)

predBST = predict(BST,n.trees=600, newdata=test_data,type='response')
dim(predBST)
p.predBST <- apply(predBST, 1, which.max)
table(p.predBST,test_data$Cover_Type)

predBST_train = predict(BST,n.trees=600, newdata=train_data,type='response')
dim(predBST_train)
p.predBST_train <- apply(predBST_train, 1, which.max)
table(p.predBST_train,train_data$Cover_Type)

library(pROC)
library(multiROC)
multiclass.roc(as.numeric(p.predBST),as.numeric(test_data$Cover_Type))
multiclass.roc(as.numeric(p.predBST_train),as.numeric(train_data$Cover_Type))
```