Forest Cover Type Classification Study

Thomas Kolasa and Aravind Kolumum Raja

Introduction

We found the Forest Cover type dataset in the UCI Machine Learning Repository that takes forestry data from four wilderness areas in Roosevelt National Forest in northern Colorado (Click here (https://tourbuilder.withgoogle.com/tour/ahJzfmd3ZWItdG91cmJ1aWxkZXJyEQsSBFRvdXIYgICAk5uU5AoM) for a tour of the area). The observations are taken from 30m by 30m patches of forest that are classified as one of seven cover types:

1. Spruce/Fir
2. Lodgepole Pine
3. Ponderosa Pine
4. Cottonwood/Willow
5. Aspen
6. Douglas-fir
7. Krummholz

The actual forest cover type for a given observation was determined from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. Kaggle hosted the dataset in a competition with a training set of 15,120 observations and a test set of 565,892 observations. The relative sizes of the train and test sets makes classification of cover type a challenging problem. We decided to use the machine learning and visualization packages available in R for this project.

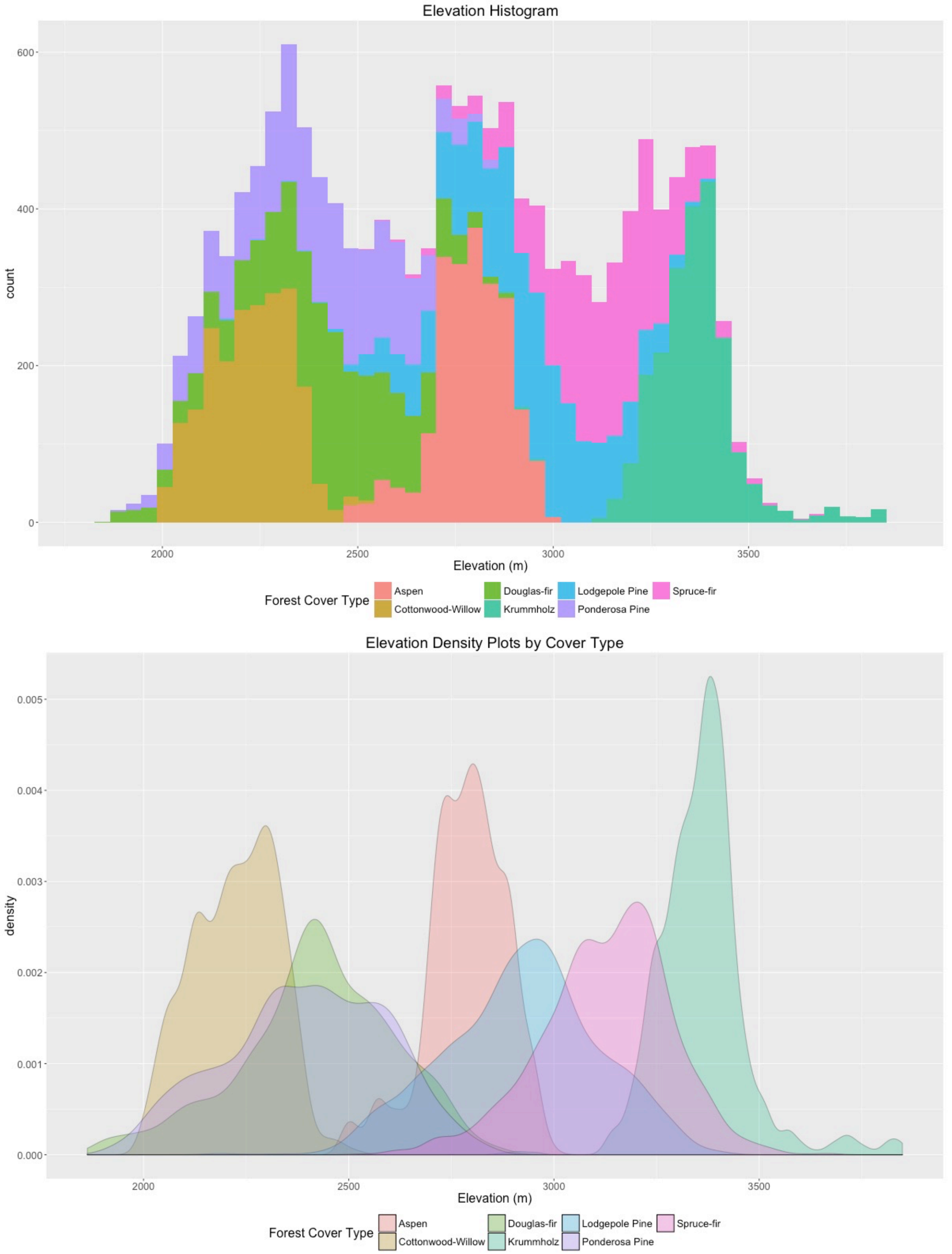
Data Exploration

Name	Measurement	Description
Elevation	meters	Elevation in meters
Aspect	azimuth	Aspect in degrees azimuth
Slope	degrees	Slope in degrees
Horizontal Distance To Hydrology	meters	Horz Dist to nearest surface water features
Vertical Distance To Hydrology	meters	Vert Dist to nearest surface water features

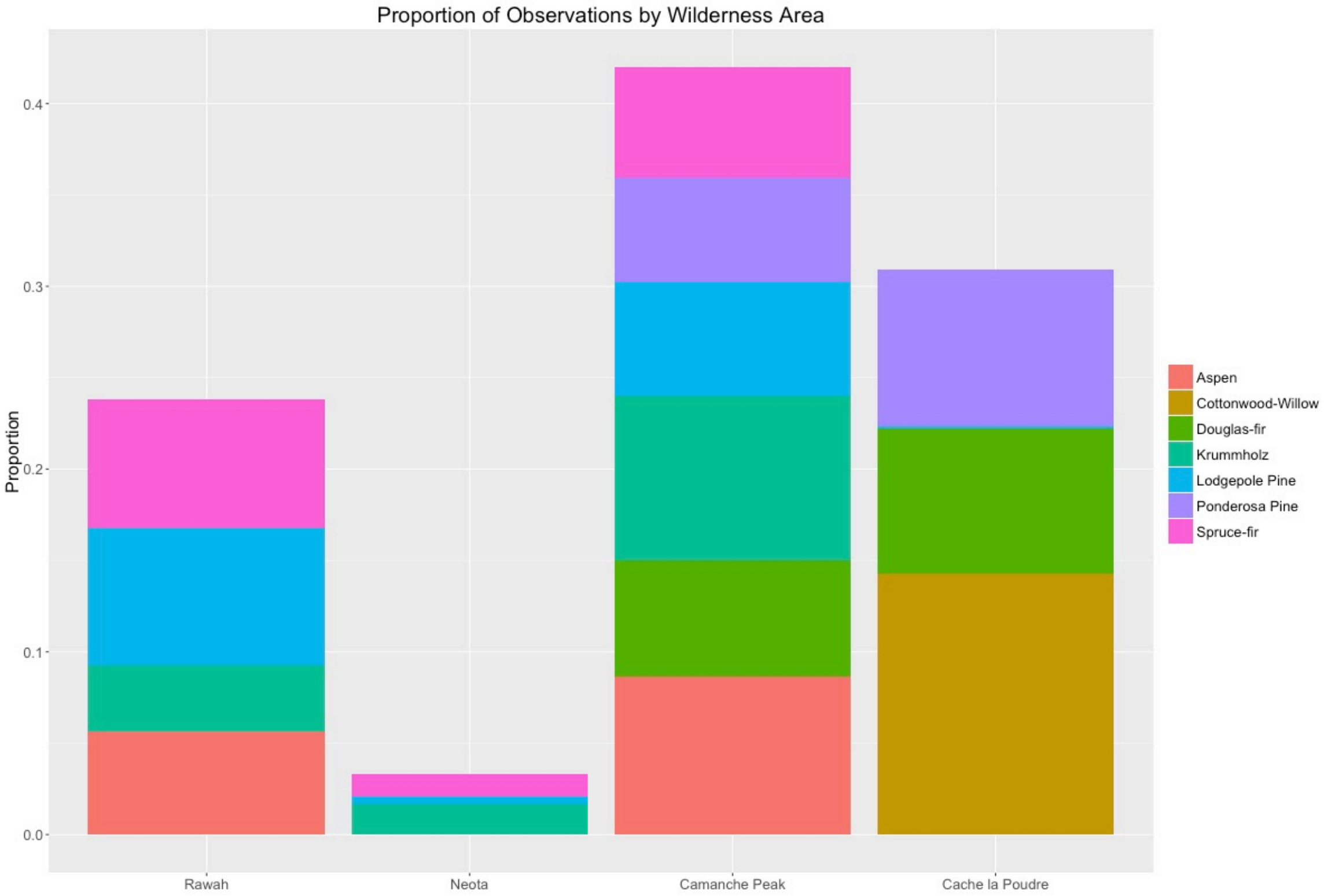
Horizontal Distance To Roadways	meters	Horz Dist to nearest roadway
Hillshade 9am	0 to 255 index	Hillshade index at 9am, summer solstice
Hillshade Noon	0 to 255 index	Hillshade index at noon, summer soltice
Hillshade 3pm	0 to 255 index	Hillshade index at 3pm, summer solstice
Horizontal Distance To Fire Points	meters	Horz Dist to nearest wildfire ignition points
Wilderness Area (4 binary columns)	0 (absence) or 1 (presence)	Wilderness area designation
Soil Type (40 binary columns)	0 (absence) or 1 (presence)	Soil Type designation
Cover Type	Classes 1 to 7	Forest Cover Type designation - *Response Variable*

The Kaggle training data was not a simple random sample of the entire dataset, but a stratified sample of the seven forest classes. The training data was equally distributed among the seven classes.

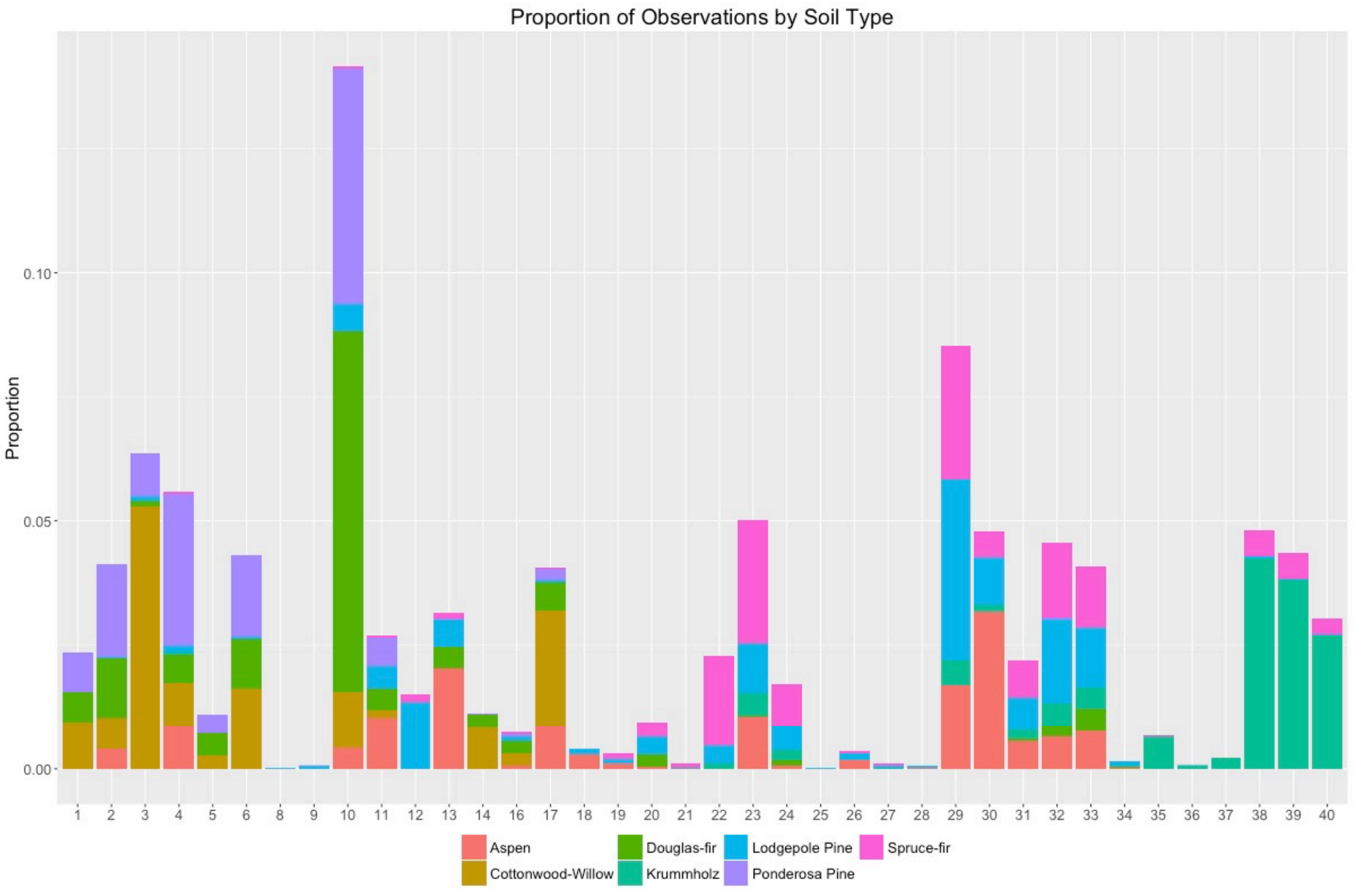
Some class seperation is clearly visible in the following plots of elevation.



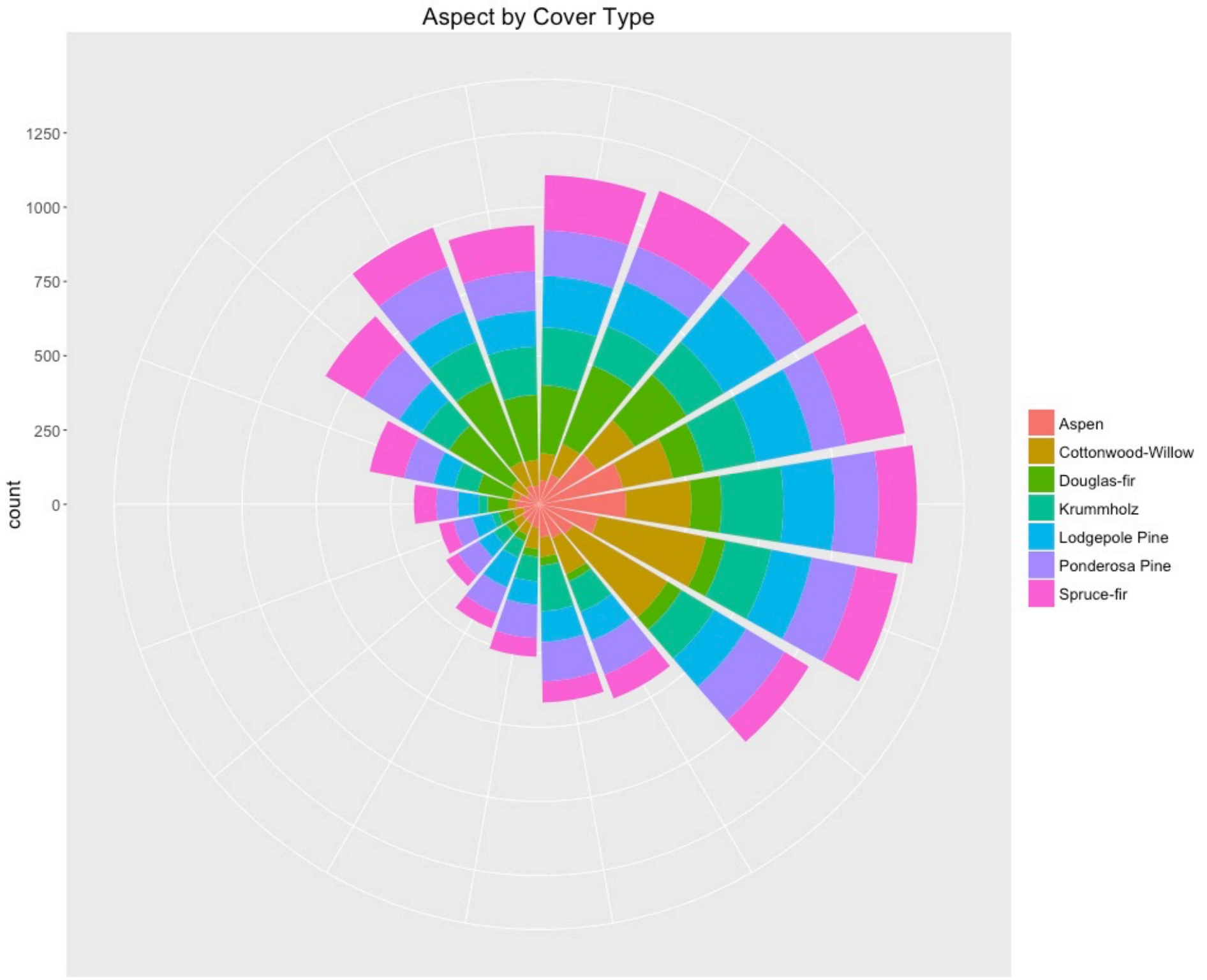
Across Wilderness areas, there seems to be some class seperation. Cottonwood Willow covers are found only in Cache la Poudre areas and Neota areas comprises of only Spruce-fir, Krummholz, and Lodgepole Pine covers.



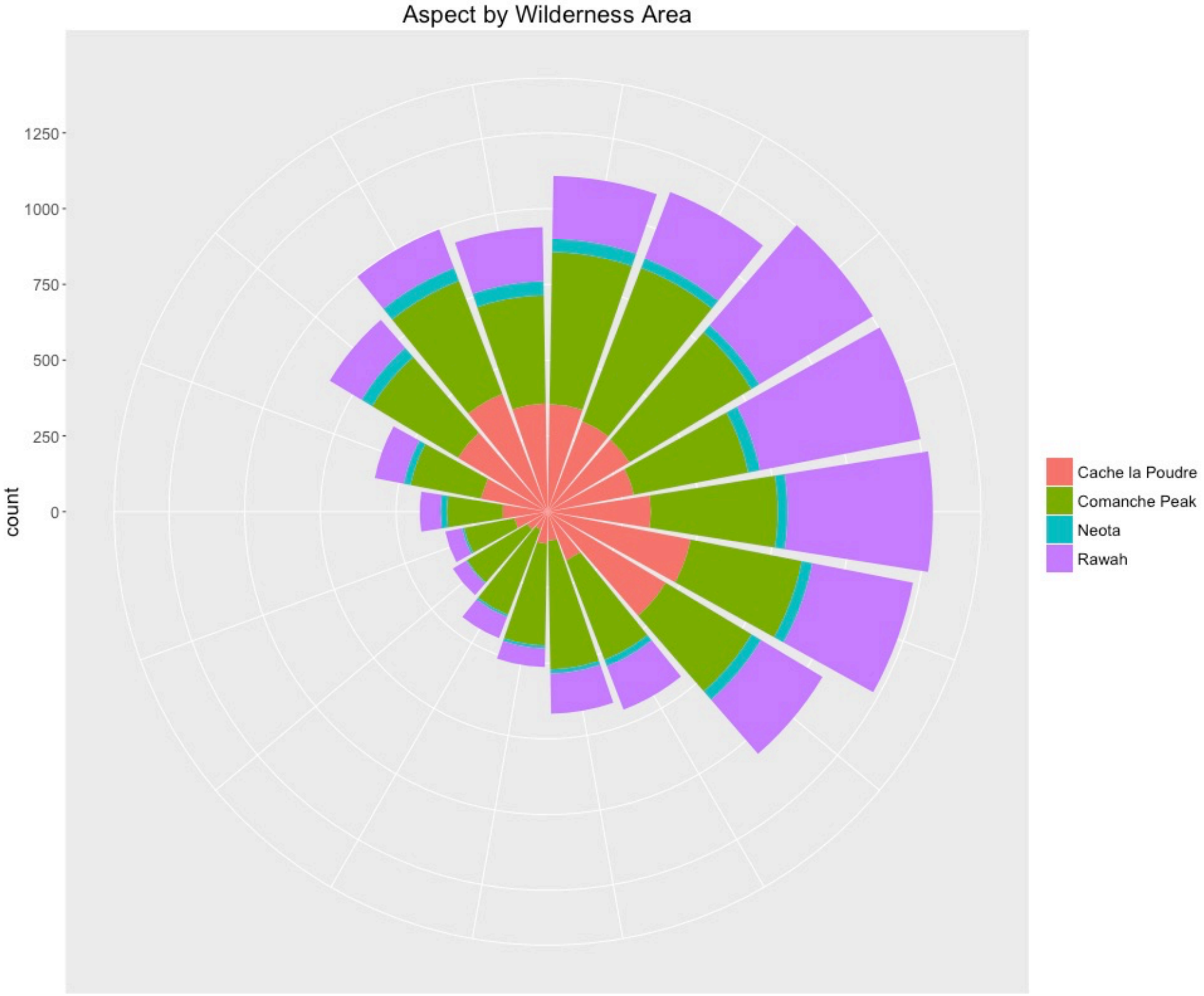
Within the training data, the presence of certain soil types reduces the probability of observing certain cover types. Soil type 7 and 15 had no observations in the training data, even though they were present in the test set. For the purposes of modeling, we had to ignore these features.



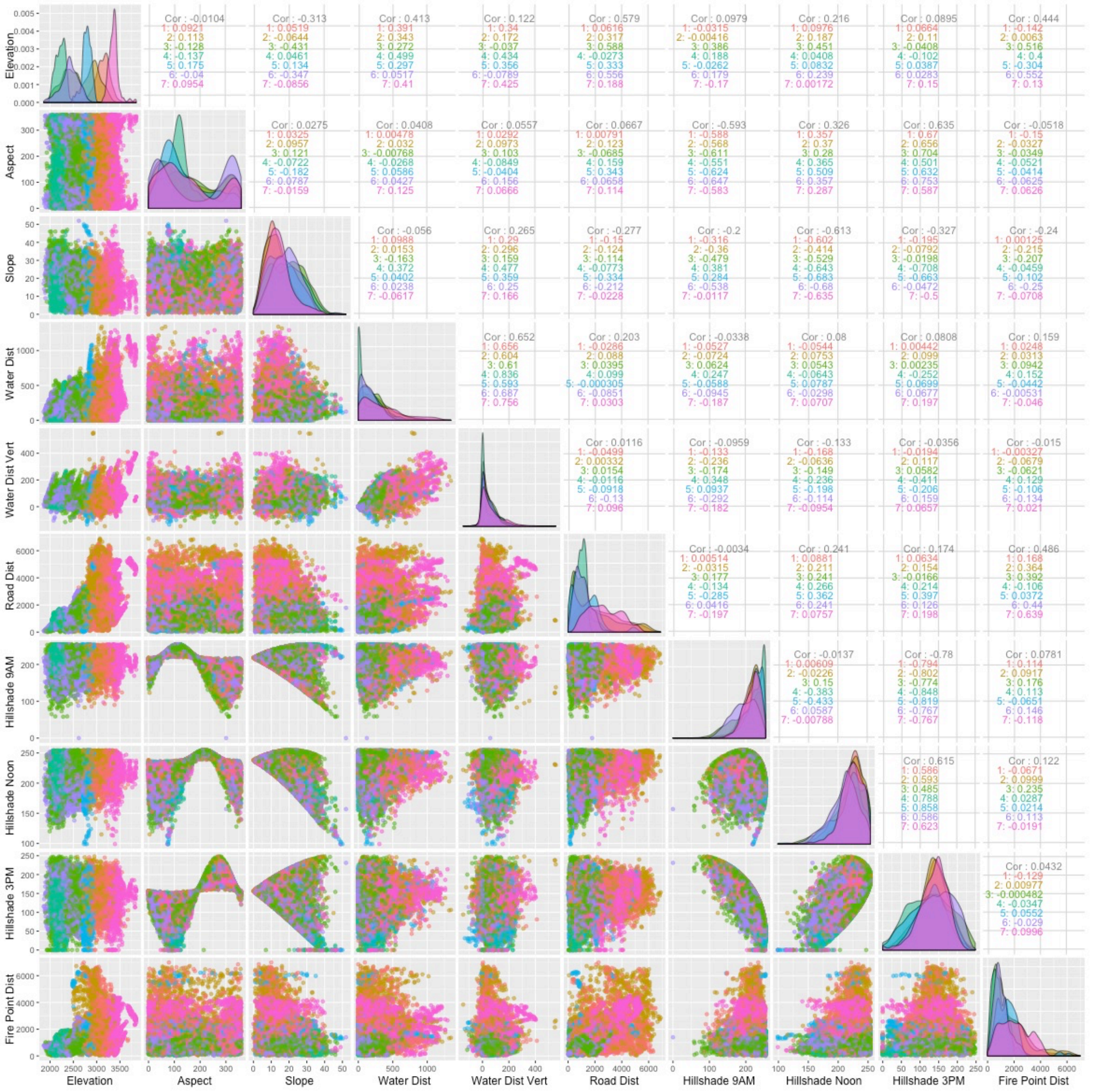
Another compelling variable is aspect, or the cardinal direction that the slope has the steepest gradient downwards. For example, in the rose diagram below, there is a greater prevalence of cottonwood/willow trees on patches where the slope faces an easterly direction.



When we look at the aspect by wilderness area, we find aspect to be equally distributed among the four wilderness locations.



Let us look at the correlation plots of the ten quantitative variables.



One of the clearer relationships in the dataset was the distribution of Hillshade luminance. Hillshade is measured on a scale from 0 to 255 (dark to bright).

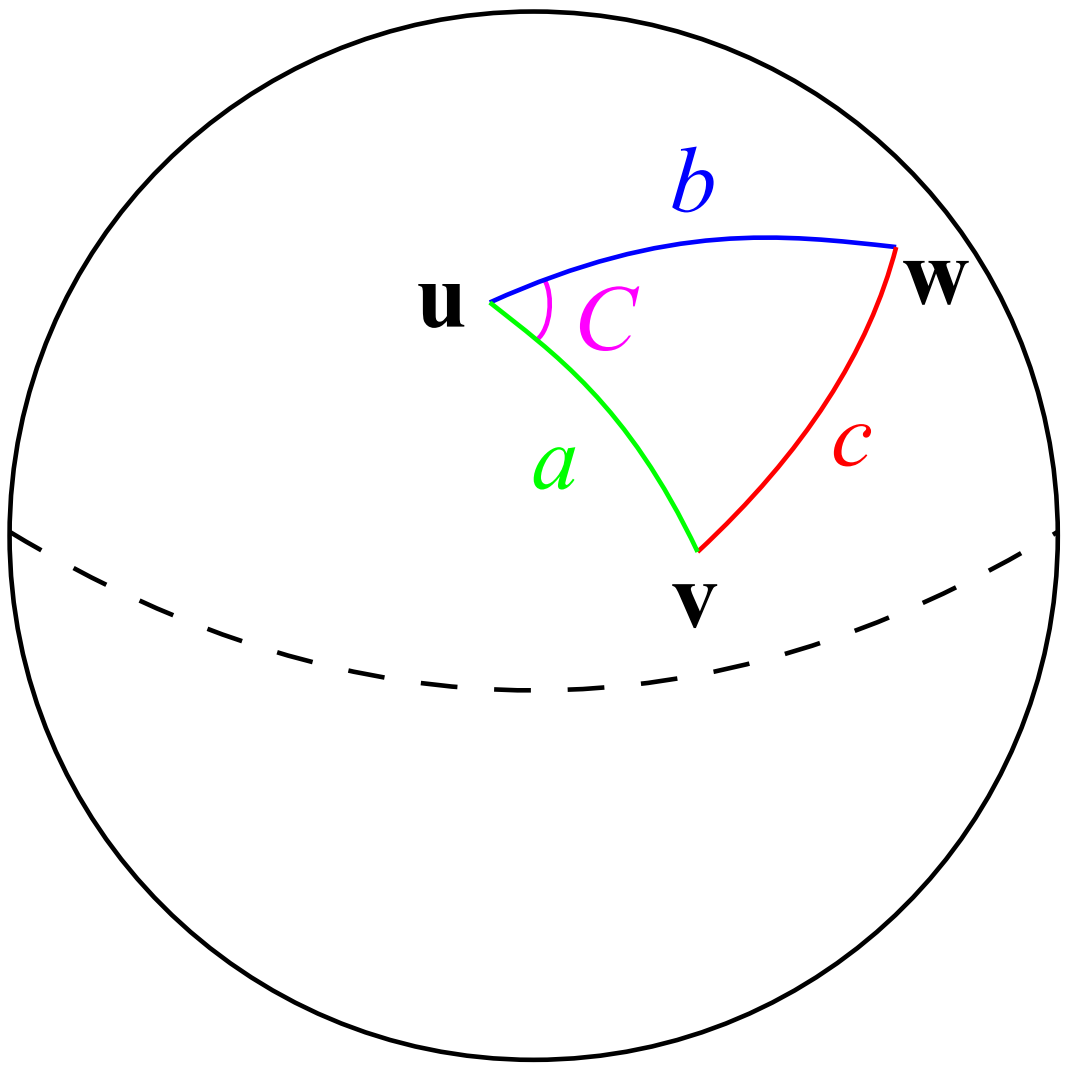
Hillshade at time *t* varies as a factor of:

$$\cos(slope)\cos(90 - Altitude) + \sin(slope)\sin(90 - Altitude)\cos(Azimuth - Aspect)$$

$$\cos(slope)\cos(90 - Altitude) + \sin(slope)\sin(90 - Altitude)\cos(Azimuth - Aspect)$$

where Altitude is the angle of the Sun relative to the horizon and Azimuth relates to the direction the Sun is facing: North, South, East, or West. Azimuth of 90 degrees corresponds to East.

This equation actually arises from a theorem in Spherical geometry known as “Spherical law of Cosines” relating the sides and angles of triangles constructed on spherical surfaces.



In a unit sphere, lengths *a*, *b*, *c* correspond to the angle subtended by those sides from the center of the sphere. If we know the two sides *a*, *b* and the angle between them *C*, then the cosine of *c*, is given by:

$$\cos(c) = \cos(a)\cos(b) + \sin(a)\sin(b)\cos(C).$$

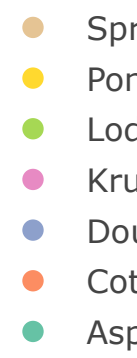
In short, the Illumination of the patch is related to alitude of the sun, slope of the terrain and the relative direction of the sun and the slope.

$Hillshade(t_{9AM}, t_{Noon}, t_{3PM})$$Hillshade(t_{9AM}, t_{Noon}, t_{3PM})$ has has been plotted below in 3 dimensions.

Click/drag on the plot to explore:

Spr
Por
Loc
Kru
Dou
Cot
Asp

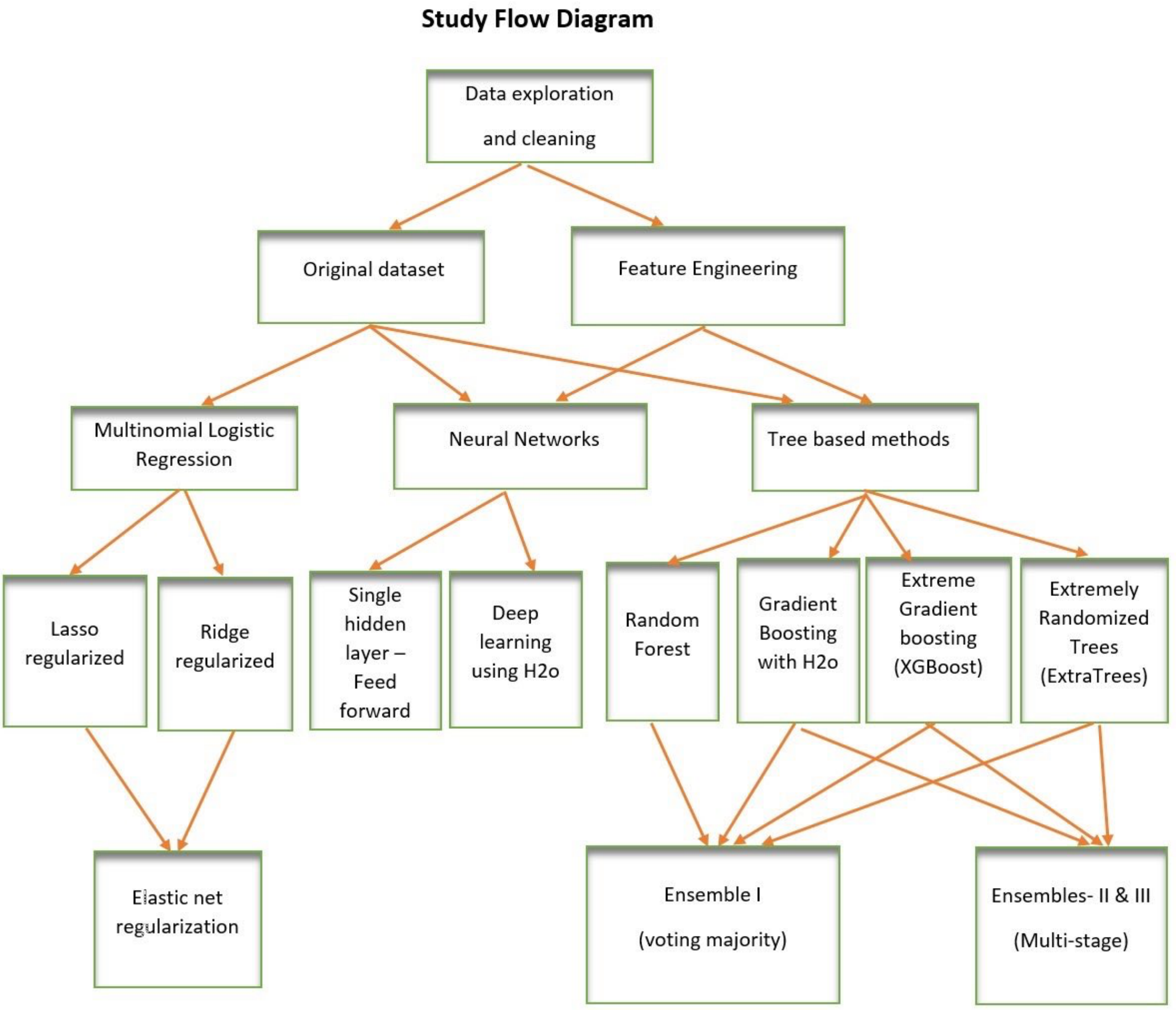
More importantly, in the context of class seperation, The following plot of elevation, slope, and hillshade clearly separates the forest cover types, demonstrating that elevation could be the most significant factor in determining cover type.



The primary reason for the collection of cartographic data pertains to terrain mapping. It is ultimately useful for applying topographic correction to satellite images in remote sensing or as background information in scientific studies.

Topographic correction is necessary if, for example, we wish to identify materials on the Earth's surface by deriving empirical spectral signatures, or to compare images taken at different times with different Sun and satellite positions and angles. By applying the corrections, it is possible to transform the satellite-derived reflectance into their true reflectivity or radiance in horizontal conditions.

Modeling



Feature Engineering

Some of the features ultimately incorporated into our analysis are shown below.

New Feature	Description
Hillshade_mean	Mean hillshade of the hillshade at 9AM, Noon, and 3PM (0-255)
Euclidean_Distance_To_Hydrology	Square root of the sum of the squared horizontal & vertical distances to water
aspect_group	Grouping aspect into
log_elevation	Logarithm of elevation
Hillshade_9am_sq	9AM hillshade squared
Hillshade_noon_sq	Noon hillshade squared
Hillshade_3pm_sq	3PM hillshade squared
cosine_slope	The cosine of the slope, used to partially model the relationships between hillshade

aspect_group_class	Whether the aspect is large or small
soil_family	What soil family a soil belongs to
soil_rock_type	Whether soil is stony, rubbly, or neither
interaction_9amnoon	Product of hillshades at 9AM and Noon
interaction_noon3pm	Product of hillshades at Noon and 3PM
interaction_9am3pm	Product of hillshades at 9AM and 3PM

We also looked at linear combinations of many of the higher correlated variables. This proved to be useful for improving the accuracy of one of our models.

Logistic Regression

We first investigated multinomial logistic regression. Since classical multiple regression in sensitive to high variance of coefficient estimates in cases of correlated predictor variables, we decided to use the *glmnet* package, which executes logistic regressions with regularization. Using the dataset *x*, the probability of the response class *k* of 1 through 7 is the following:

$$\Pr(G = k|X = x) = \frac{e^{\beta_{0k}+\beta_k^T x}}{\sum_{\ell=1}^{7} e^{\beta_{0\ell}+\beta_\ell^T x}}$$

$$\Pr(G = k\,|\,X = x) = \frac{e^{\beta_{0k}+\beta_k^T x}}{\sum_{\ell=1}^{7} e^{\beta_{0\ell}+\beta_\ell^T x}}$$

We then transfer this into the elastic-net penalized negative log-likelihood function (the first term of the following equation):

$$\ell(\{\beta_{0k}, \beta_k\}_1^7) = -\left[\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{k=1}^{7} y_{il}(\beta_{0k} + x_i^T\beta_k) - \log\left(\sum_{k=1}^{7} e^{\beta_{0k}+x_i^T\beta_k}\right)\right)\right] + \lambda\left[(1-\alpha)||\beta||_F^2/2 + \alpha\sum_{j=1}^{p}||\beta_j||\right]$$

$$\ell(\{\beta_{0k}, \beta_k\}_1^7) = -\left[\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{k=1}^{7} y_{il}(\beta_{0k} + x_i^T\beta_k) - \log\left(\sum_{k=1}^{7} e^{\beta_{0k}+x_i^T\beta_k}\right)\right)\right] + \lambda\left[(1-\alpha)|\,|\beta|\,|_F^2/2 + \alpha\sum_{j=1}^{p}|\,|\beta_j|\,|\right]$$

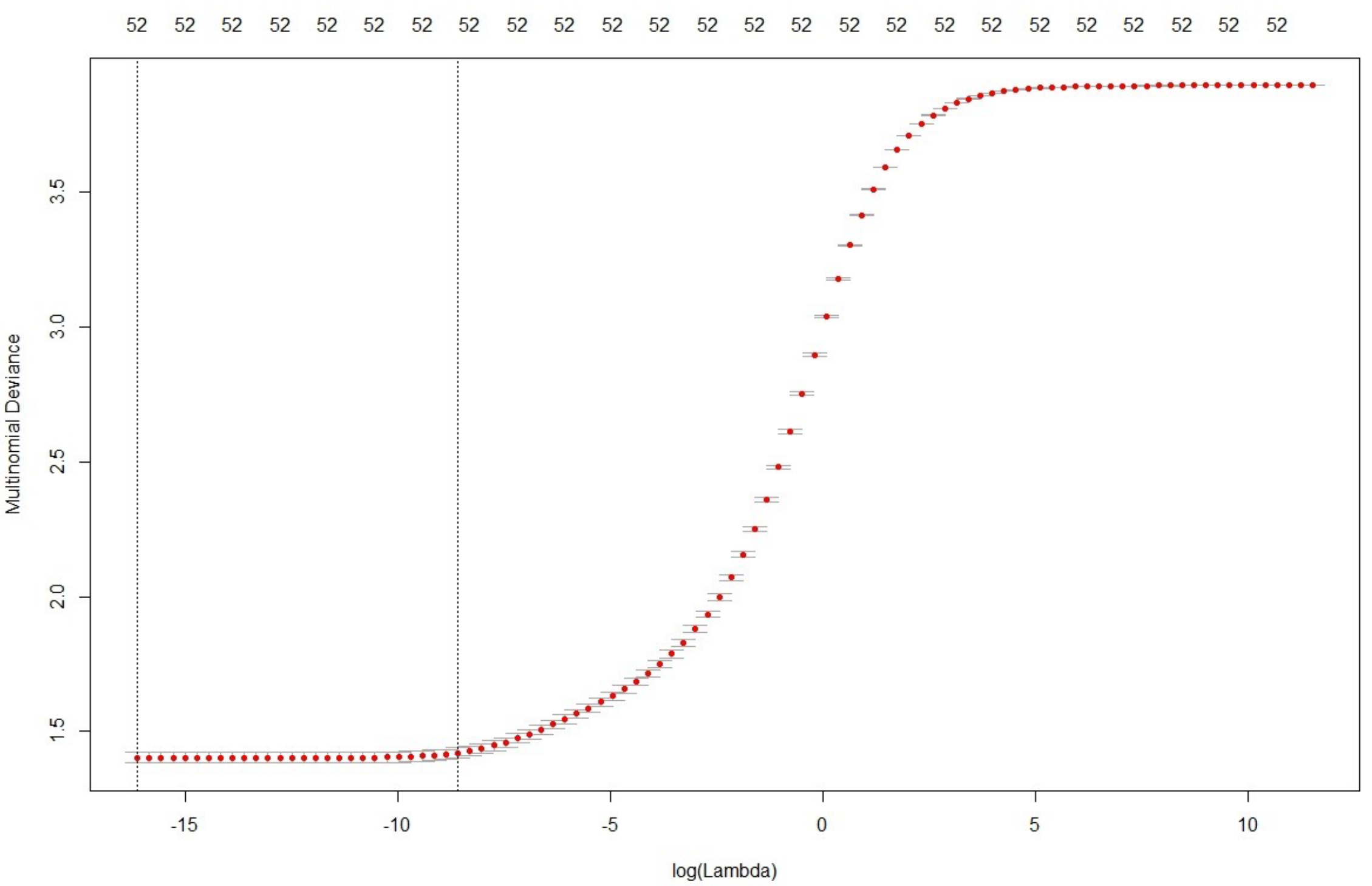
Where *k* is the response class (cover type from 1 to 7) and *N* is the number of observations. The second term in the equation represents the regularization term.

The elastic net penalty $\alpha\alpha$ varies from 0 to 1. When $\alpha = 0\alpha = 0$, the function reduces to Ridge regularization. When $\alpha = 1\alpha = 1$, Lasso regularization is used. The Ridge penalty shrinks the coefficients of closely correlated predictors wheras the Lasso tries to pick one and discard others.

Ridge Regularization

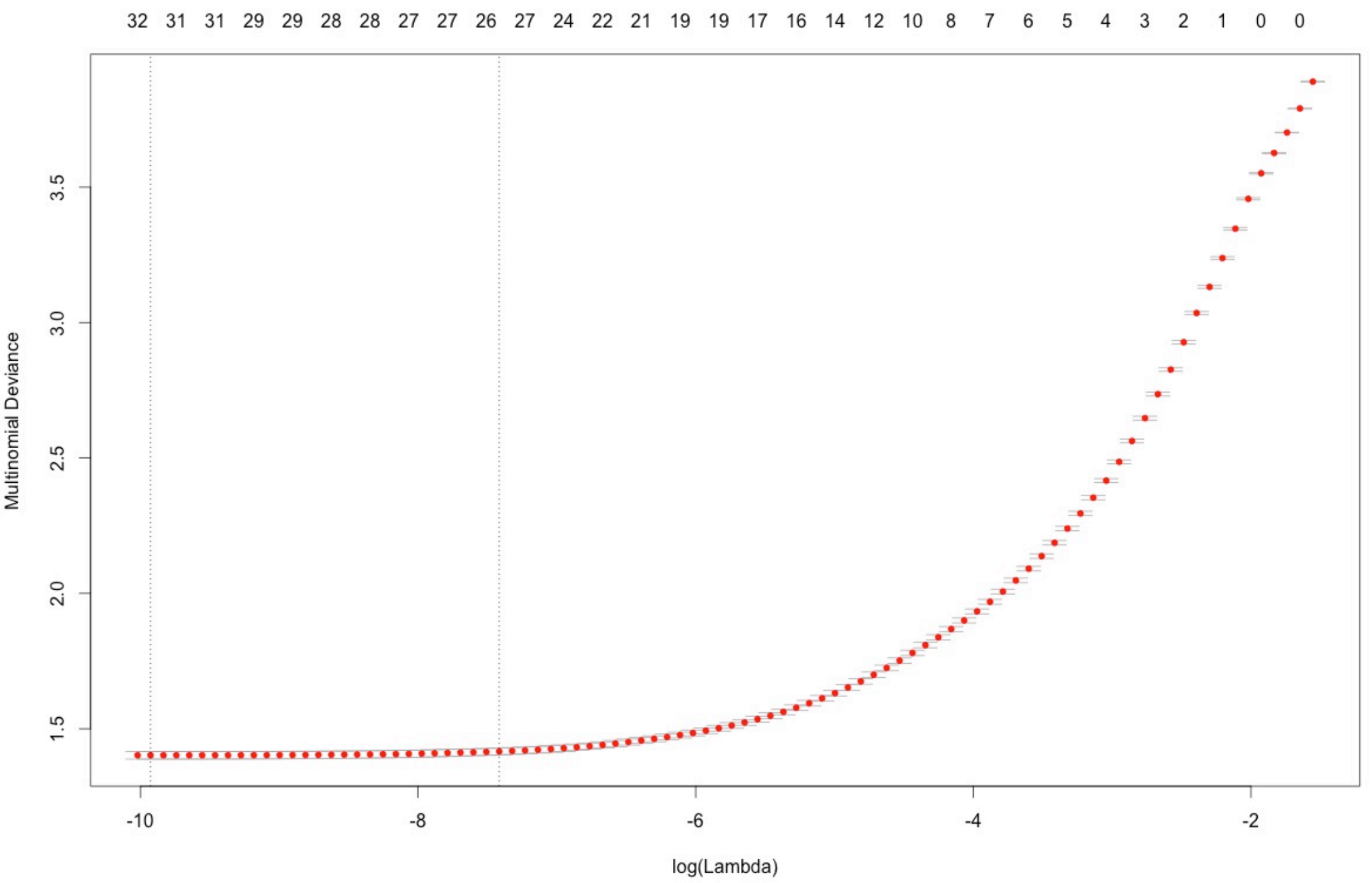
We performed 10-fold cross validation on a ridge regularization and on a grid of 100 values of $\lambda\lambda$, choosing the minimum $\lambda\lambda$ for each model. When we ran the regression with a 70-30 cross validation split on the training set, we got an accuracy on the test set on Kaggle of **55.70%**.

When we reran the ridge regression with an 85-15 split on the training data, we got a Kaggle accuracy of **59.56%**. Since the test set is so much larger than our training set, it was beneficial to incorporate a larger proportion of the data for learning within the training data before runnint the model on the larger testing data.



Lasso Regularization

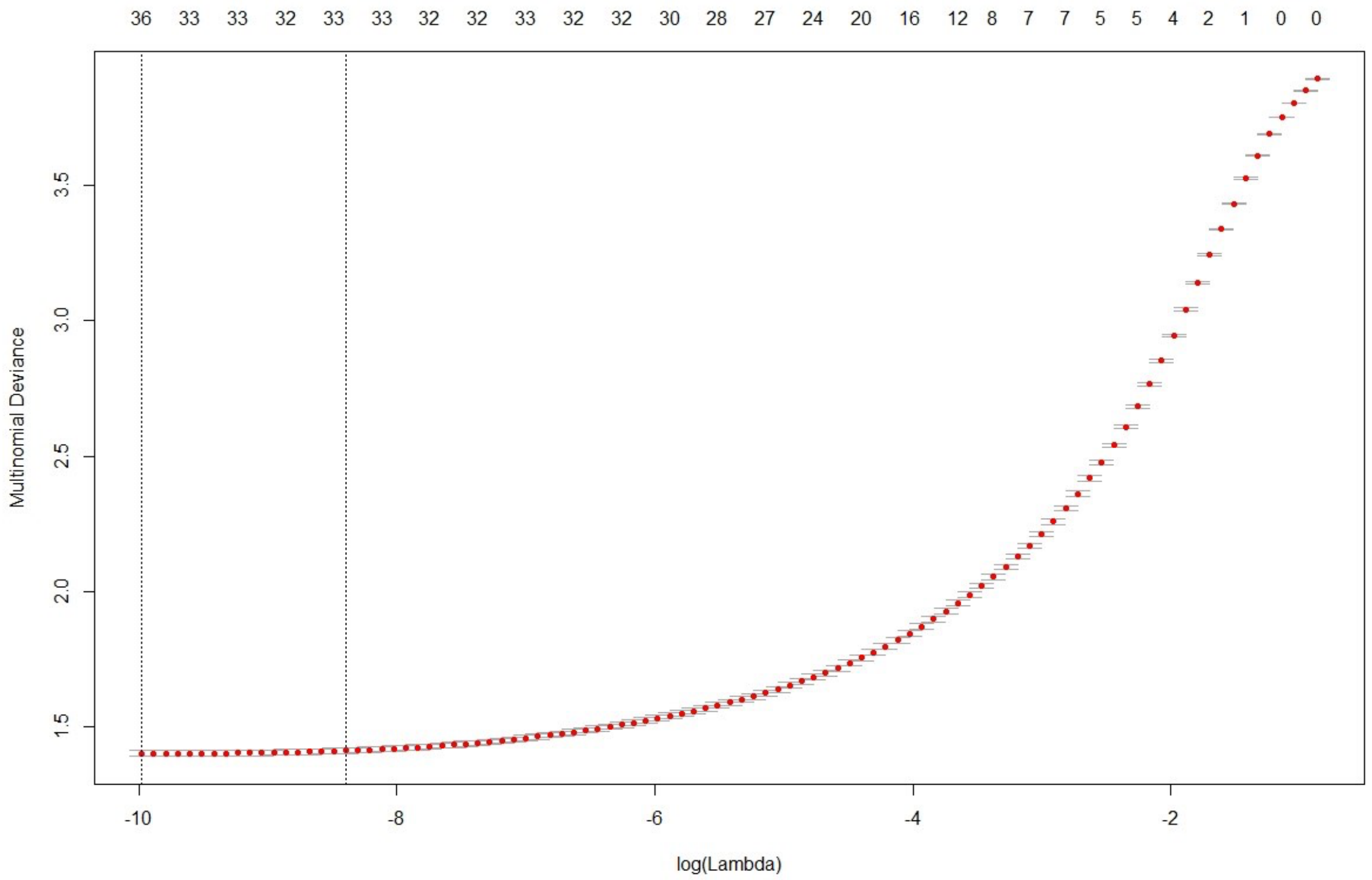
A 10-fold cross validated lasso-regularized logistic regression kaggle score was **59.59%** for 70-30 split kaggle score of **59.53%**, for 85-15 split kaggle score of **59.44%** if we took the lowest 50 percentile of lambdas and calculated the mode of the predictions.



Elastic-net Regularization

Using both the ridge and lasso regularization terms concurrently produces an elastic-net regularization. We ran this hybrid method with both a 70-30 split and 85-15 split in the training set. When running these models on the testing set, we got accuracies on Kaggle of **59.59%** and **59.52%**, respectively.

We also tried an elastic-net regression using all 15,120 observations in the training set with an $\alpha\alpha$ parameter of 0.5. It earned an accuracy score on the testing set of **59.50%**.



In summary, we were able to attain a best accuracy of **59.59%** using lasso/elastic net and regularization. Whatever value $\alpha\alpha$ takes does not have huge impact on model accuracy on this dataset.

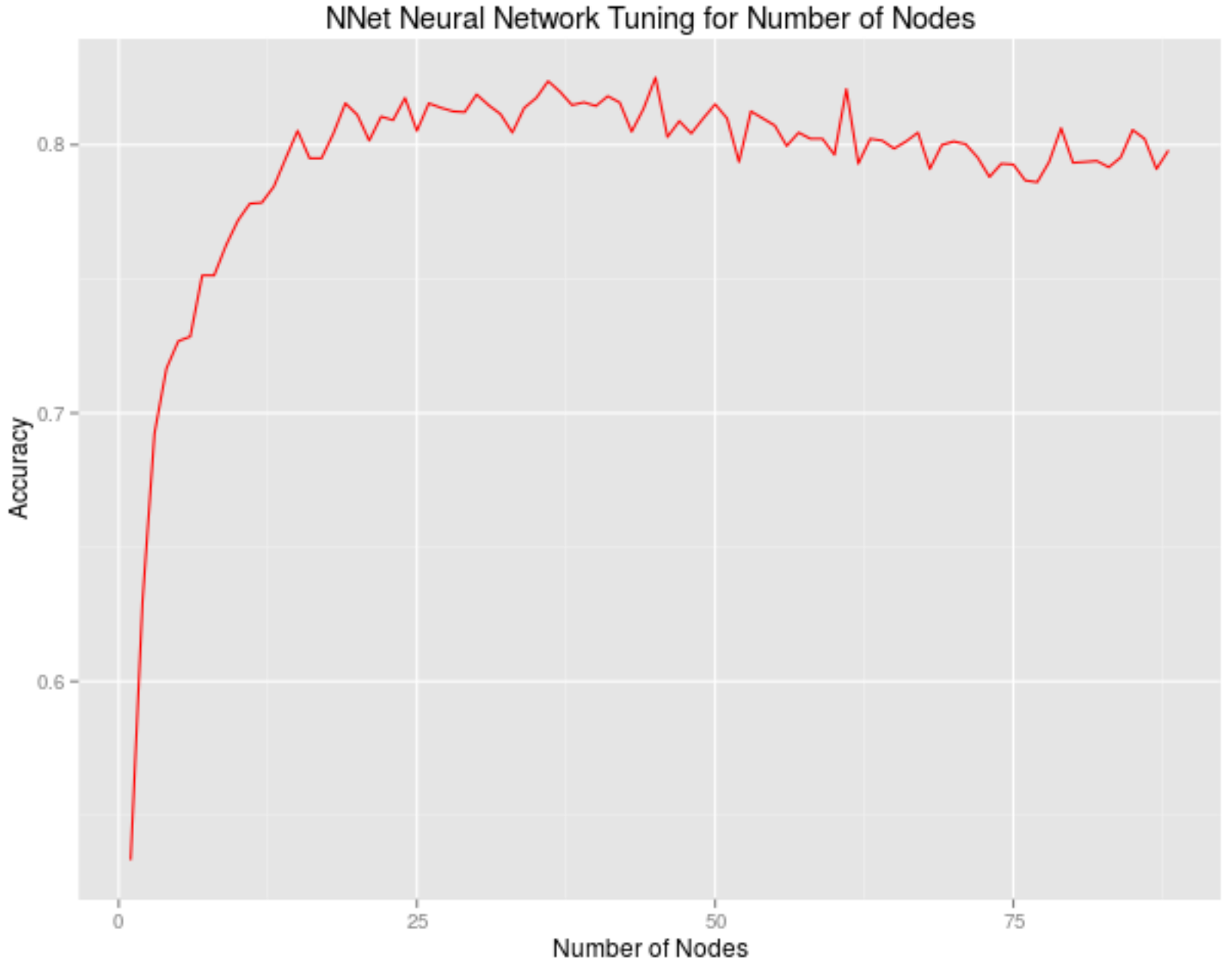
Artificial Neural Networks

We first modeled the data in a feed-forward neural network with a single layer of nodes. Later, we created deep learning models where we added more layers of hidden nodes.

Single Hidden Layer Using the nnet Package

The nnet R library allows the construction of feed-forward single-layer nerual networks. After scaling our variables and running a basic model, we decided to tune the nnet function based on the following parameters - Number of nodes - Number of iterations - Maximum number of weights (since this depends on the number of nodes)

We experimented to see the optimal number of nodes for our hidden layer by checking our accuracy levels over various number of nodes.



We finally chose a 54-100-7 node neural nets on our training data achieving **52.11%** accuracy for 300 maximum iterations and **52.43%** accuracy for 500 maximum iterations.

Deep Learning using H2o

For adding multiple hidden layers, we used the deeplearning function using the H2o package. H2o is an R package that runs parallel-distributed machine learning algorithms to improve computational speed. Later, we also used H2o's Gradient Boosting Machine (GBM) algorithm to acheive our second best accuracy rate.

Some features of H2o include:

- In-memory compression techniques to handle large datasets
- Automatic standardization
- Randomized Grid Search
- Fast algorithms deployed in Java
- Anomaly detection to identify outliers

When we first ran a basic model, our validation results were poor and we decided to experiment with varying the default parameters using a grid search. We tried variations of one to three hidden layers with between 30 to 200 nodes each. We also varied the input dropout ratio, the fraction of training features to omit for improving generalizability. The next parameter was the rate, or how much the network adjusts over time. Finally, we tuned the rate annealing parameter that reduces the liklihood that the learning rate chooses local minima while optimizing. H2o allows for "hyper-parameter" tuning in the following format:

```
hyper_params <- list(
  hidden=list(c(30),c(50),c(70),c(100),c(130),c(160),c(200),c(250),c(30,60),c(60,90),c(90,120),c(120,150),c(
30,60,90),c(60,90,120),c(90,120,160)),
  input_dropout_ratio=c(0,0.05),
  rate=c(0.01,0.02),
  rate_annealing=c(1e-8,1e-7,1e-6))
```

From our first tuning run, the neural network had an accuracy of **60.19%** with a 54-120-150-7 network.

After returning to the model, we decided to experiment with a wider range of the tuning parameters. Our best model was a 54-500-800-7 network with a learning rate of 0.02, rate annealing of 1e-05, input dropout ratio of 0, and an accuracy of **67.4%**.

Notable studies from the past:

The earliest use of the dataset was in a doctoral dissertation. Jock Blackard and Denis Dean at Colorado State University used Linear Discriminant Analysis to obtain a 58% accuracy and Artificial Neural network to achieve 70% accuracy. The paper concludes by saying that ANN does have its drawbacks, it is still more viable than traditional methods, by which we think he refers to logistic regression, LDA and Support Vector Classifiers.

In 2007, Jain & Minz (Journal of Indian Agricultural Statistics) applied Rough Set Theory as an alternative to Machine learning approaches to achieve equally good but using lesser attributes (only 29) to achieve 78% accuracy for the forest cover dataset. This is especially relevant in real world classification applications in remote sensing involving large datasets.

In 2004, in Systems, Man and Cybernetics an IEEE International Conference publication, Xiaomei Liu, Kevin W. Bowyer of the University of Notre Dame use the forest cover dataset to conjecture something interesting. In their own words:

- *"Based on some experiments with the Forest Cover Type Data Set, we conjectured a class of problems that are extremely difficult for FFBP NNs but relatively simple for DTs. The features of such problems are that some minority classes are surrounded and overlapped by some majority classes. We generated synthetic data sets with the above description. In our experiments on the synthetic data sets, we varied the size and the number of the islands for the minority class. We also varied the ratio of the minority class and the majority class for each island. We tried a plain feed forward back propagation NN, a Rprop NN, and a cascade correlation NN. With all of these experiments, the DT performed better than each kind of FFBP NN"*[1] [1]

Based on these findings, we decided to move on from neural networks in favor of tree-based methods.

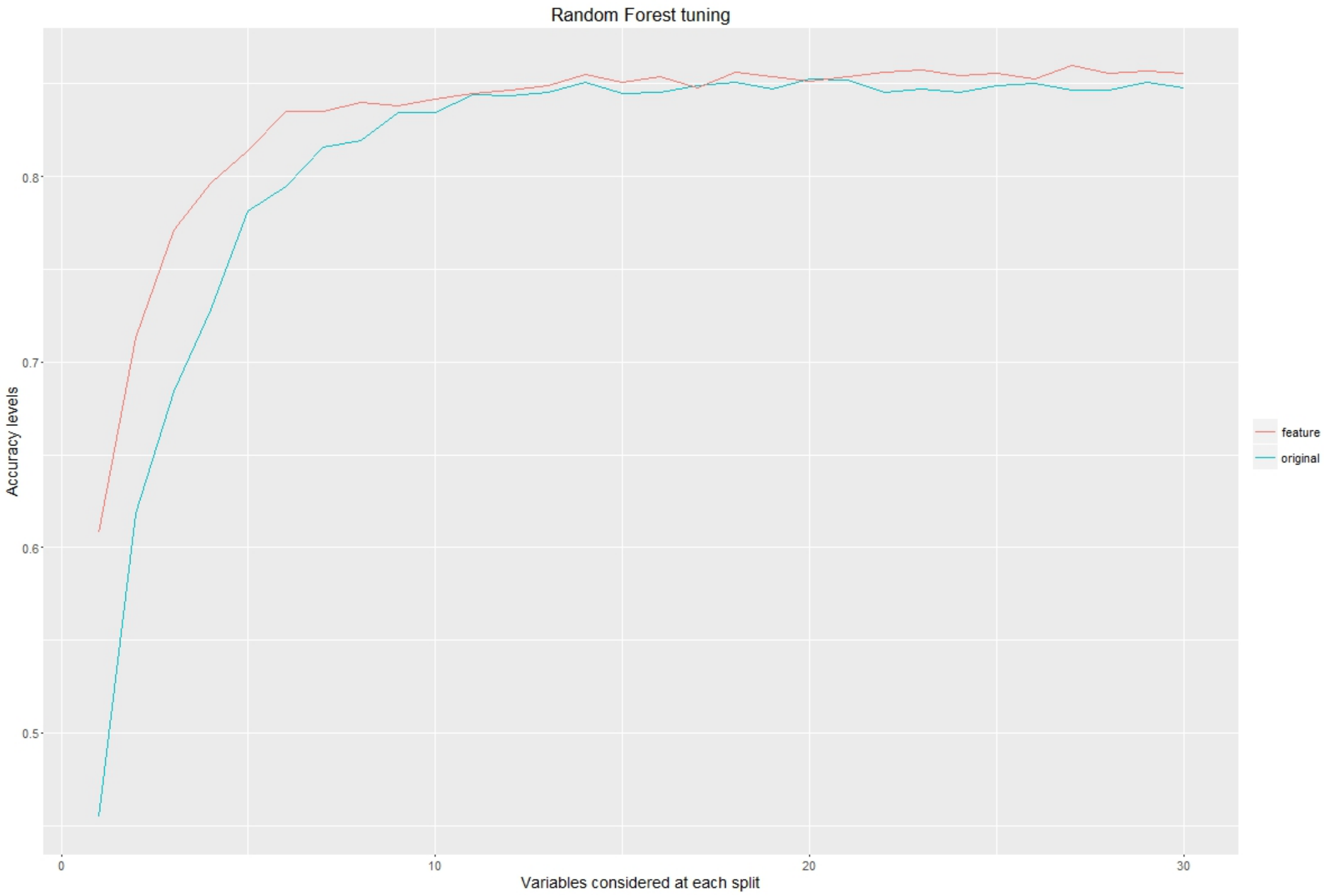
Tree Based Methods

No.	Package	Essential features of the tree based methods in R
1	Random Forest	• Bootstrapped bagging • Randomly samples rows /columns to build a tree • Averages or takes the voting ensembles of the samples together • Lowers variance without much of an increase in bias
2.	GBM (Vanilla)	• Fits consecutive trees, dependent upon one another (unlike Random forests), a new decision tree is created out of the net error of previous trees.
3.	XGBoost	• Gradient Boosting implemented in C++ • Uses OpenCP which parallels on a multi-threaded CPU • Also includes regularized linear models • Stores data on a data structure called DMatrix for faster iteration
4.	GBM (H2o)	• Easily scalable with small clusters to handle billions of rows of data • Individual tree fitting is performed in parallel • Fast and memory-efficient Java implementations of the underlying algorithms • grid search for hyper parameter optimization and model selection • Randomized hyper parameter search and Check pointing capabilities
5.	Extremely Randomized Trees (ExtraTrees)	• Two main differences with other tree based ensemble methods are - It splits nodes by choosing cut-points fully at random - It uses the whole learning sample (rather than a bootstrap replica) to grow the trees.

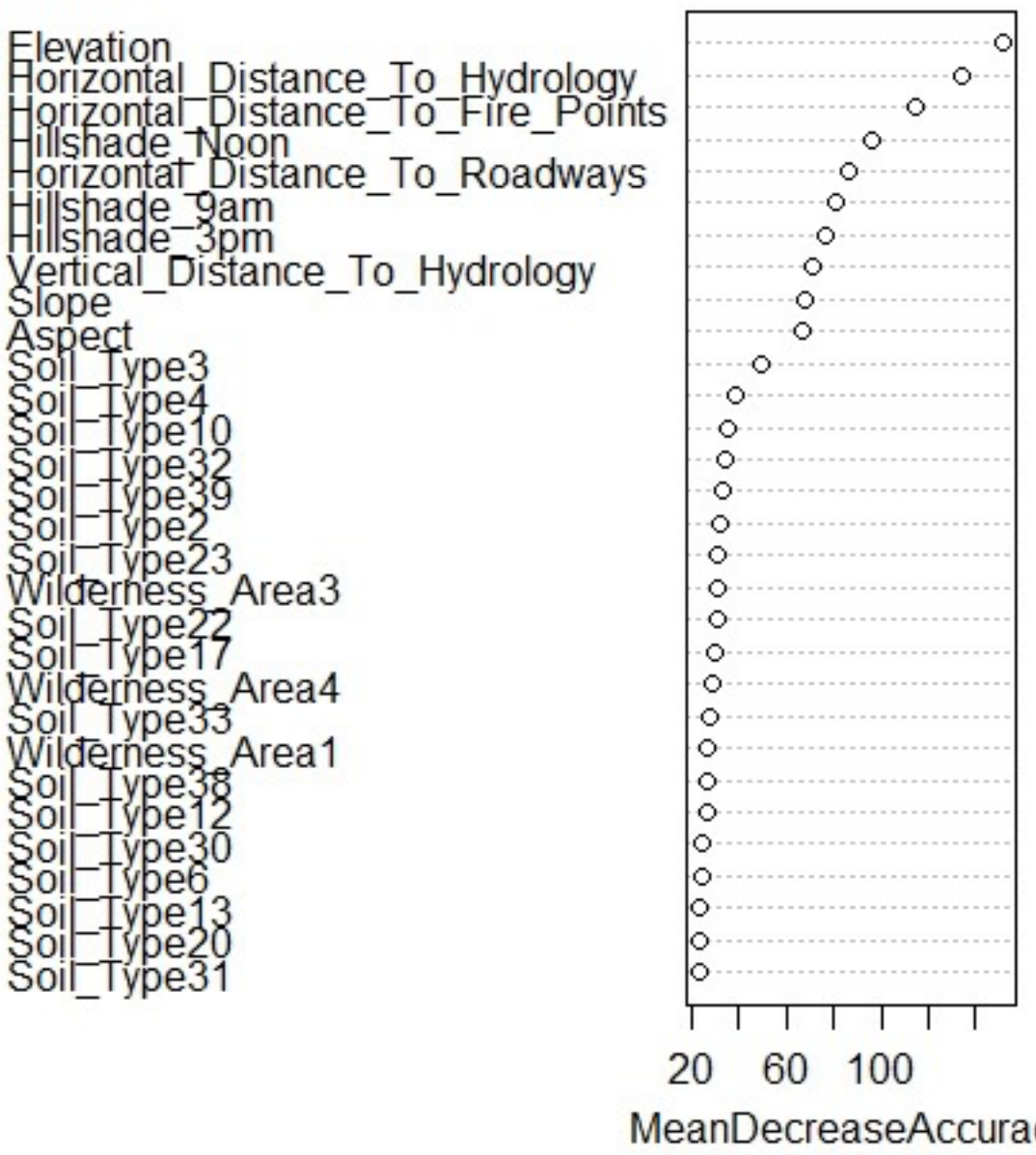
Random Forest

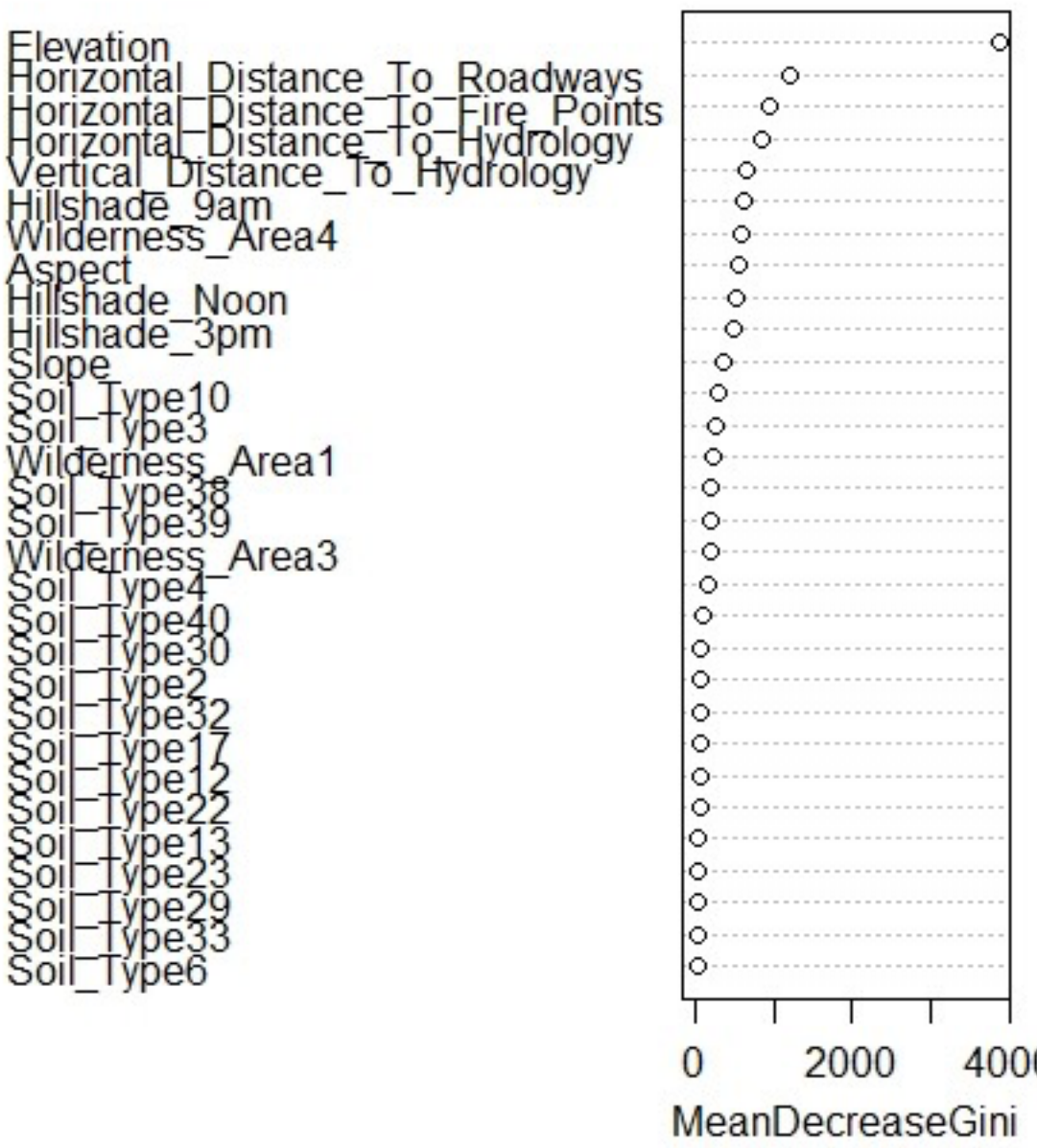
Our initial submission of the Random forest algorithm without tuning any parameters improved our score to 68%.

The tuning was performed by varying the number of variables randomly sampled as candidates at each split.



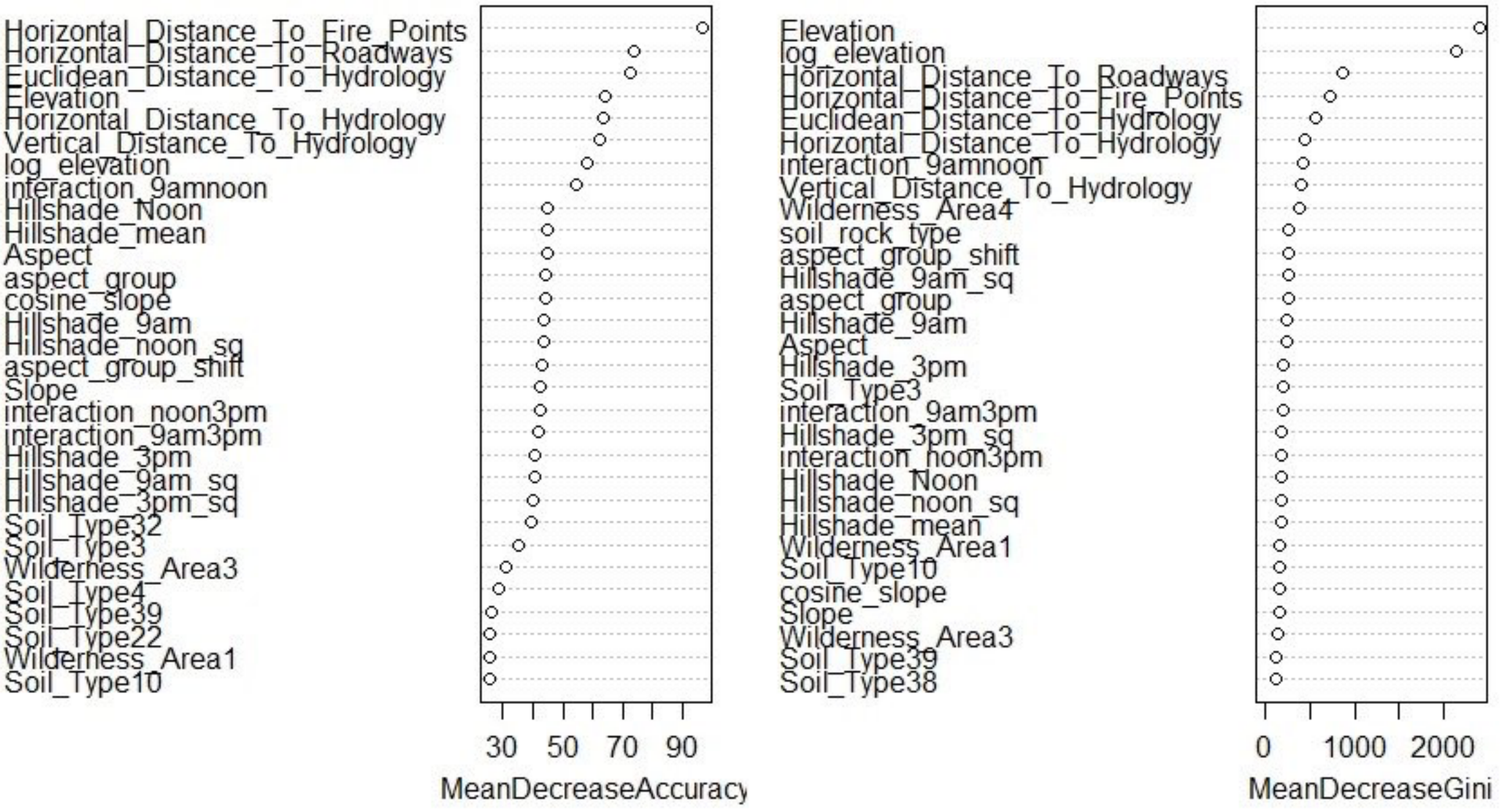
rf.tuned17





Elevation clearly shows up as the most significant variable both in terms of mean decrease in Accuracy as well as contributing to reduction in node impurity.

After tuning the number of trees, we arrived an accuracy of **75.17%**, our first benchmark.



After tuning the feature-engineered dataset we were only able to acheive around **71.93%** accuracy.

Boosting Using XGBoost

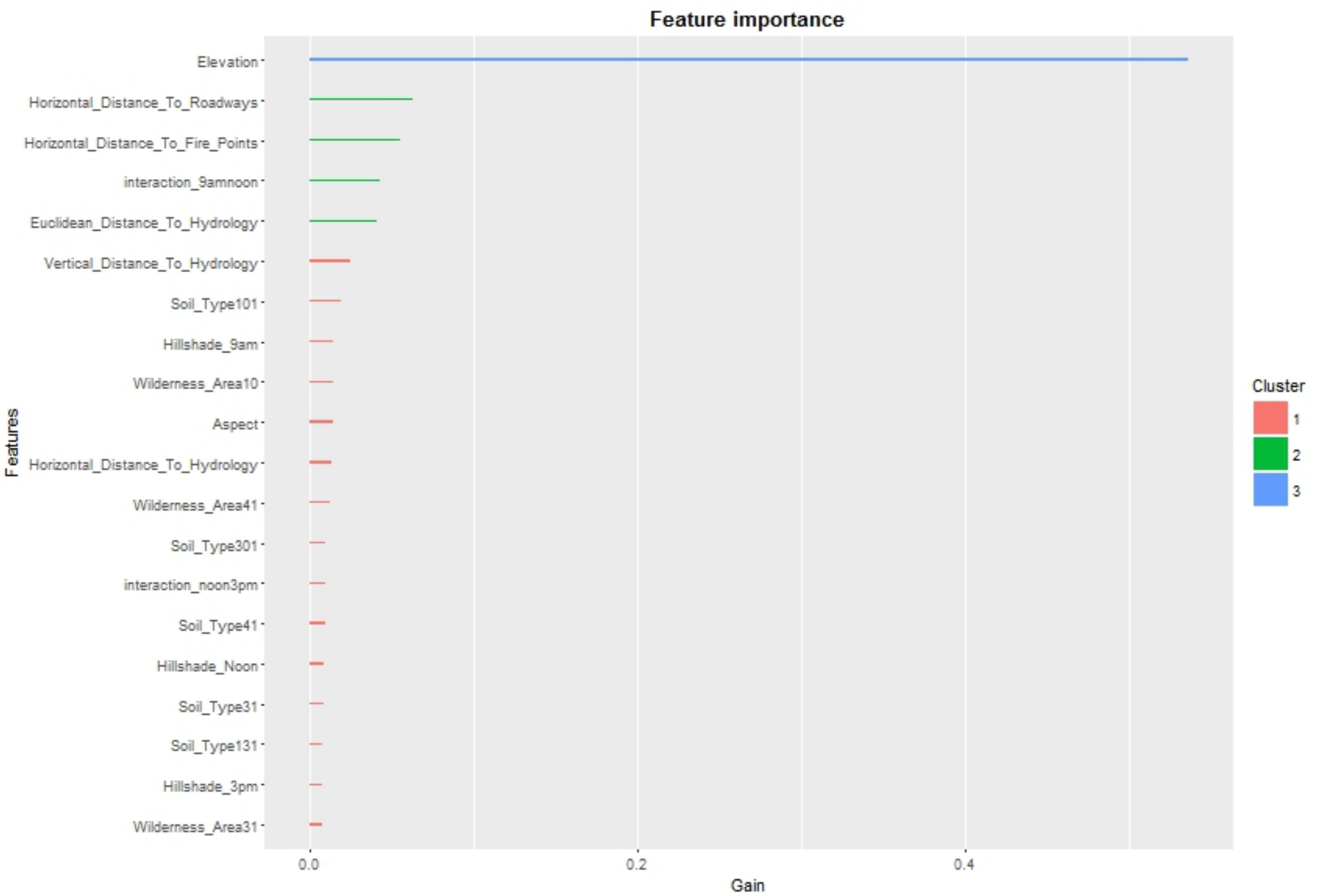
The following steps were followed while implementing the XGBoost algorithm - An untuned algorithm performed slightly better than the untuned Random Forest with **70%** accuracy - The tuning metric we used was the multinomial logloss

$$multinomial\ logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j}\log(p_{i,j})$$

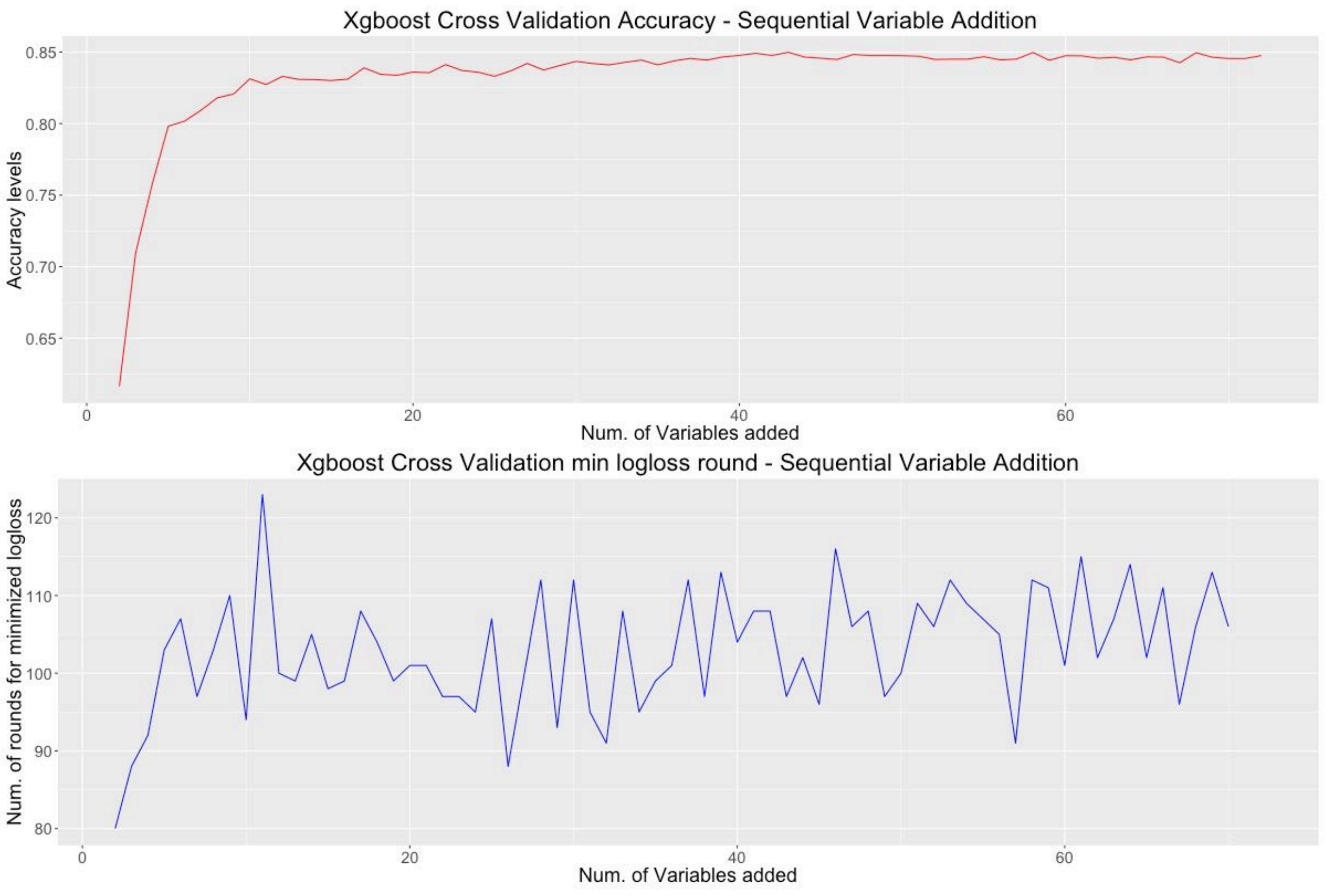
$$multinomial\ logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j}\log(p_{i,j})$$

where *N* is the number of observations, *M* is the number of class labels, loglog is the natural logarithm, $y_{i,y}y_{i,y}$ is 1 if observation *i* is in class *j* and 0 otherwise, and $p_{i,j}p_{i,j}$ is the predicted probability that observation *i* is in class *j*.

- After running a 3-fold crossvalidation with 800 rounds, we estimated the logloss minimization occured on around 104
- Our new model used the above parameters and improved the accuracy to **73.44%**



Based on this information we tried another model for which we sequentially added the most important variables using all our variables



Our accuracy was **72.66%**, lower than the one acheived using the original dataset.

Gradient Boosting

- After some initial tuning of H2o's Gradient Boosting Algorithm, an accuracy of **76.64%** was reached
- The Grid search tuning looked as follows:

```
hyper_params_gbm = list(
ntrees = c(40, 100, 200),
max_depth = 20,
min_rows = c(5, 10, 15),
learn_rate = c(.01, .1, .3))
```

max_depth is the maximum depth (length of the longest path from node to leaf of a Tree) *min_rows* is the minimum number of rows to assign to terminal nodes

- Hyperparameter tuning gives an impvroment to **78.05%**.
- Using a second set of feature engineered variables leads to a further improvement of **79.01%**.

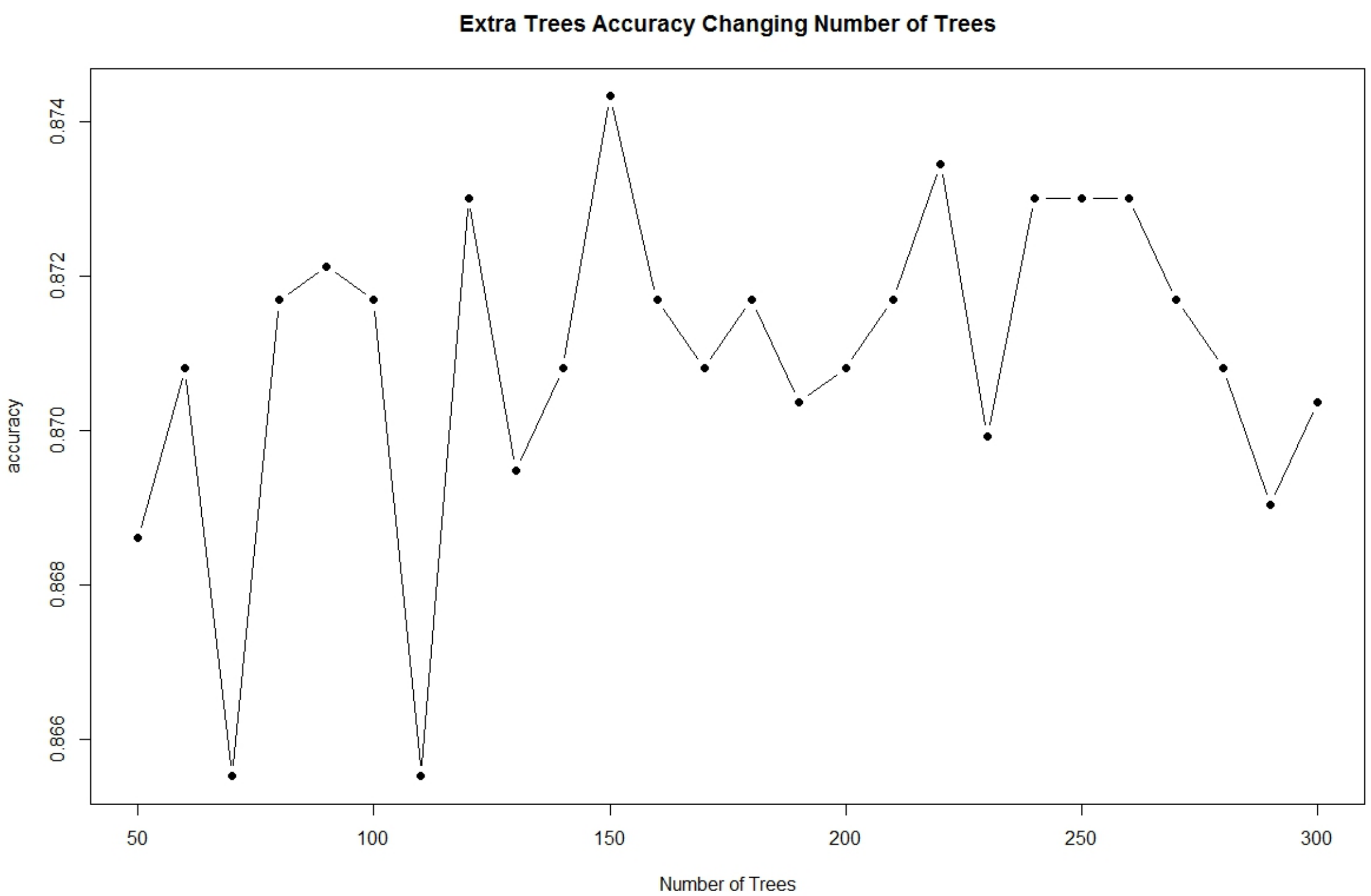
Extremely Randomized Trees

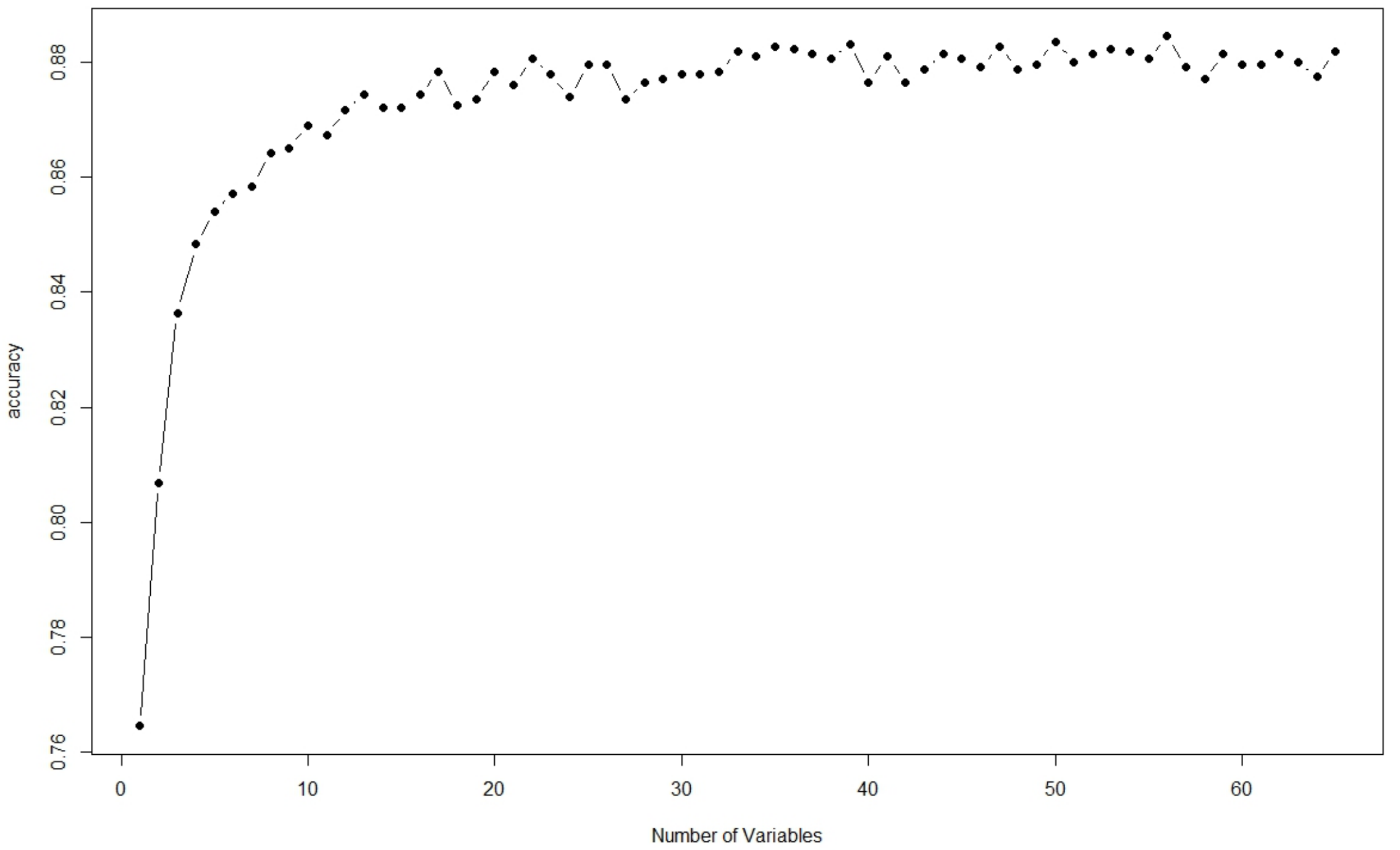
As mentioned earlier, the Extratrees algorithm includes another level of randomness to the mix by choosing cutpoints for each split in random while using the whole of the in-sample data to build the tree. The feature also allows us to choose the number of cutpoints.

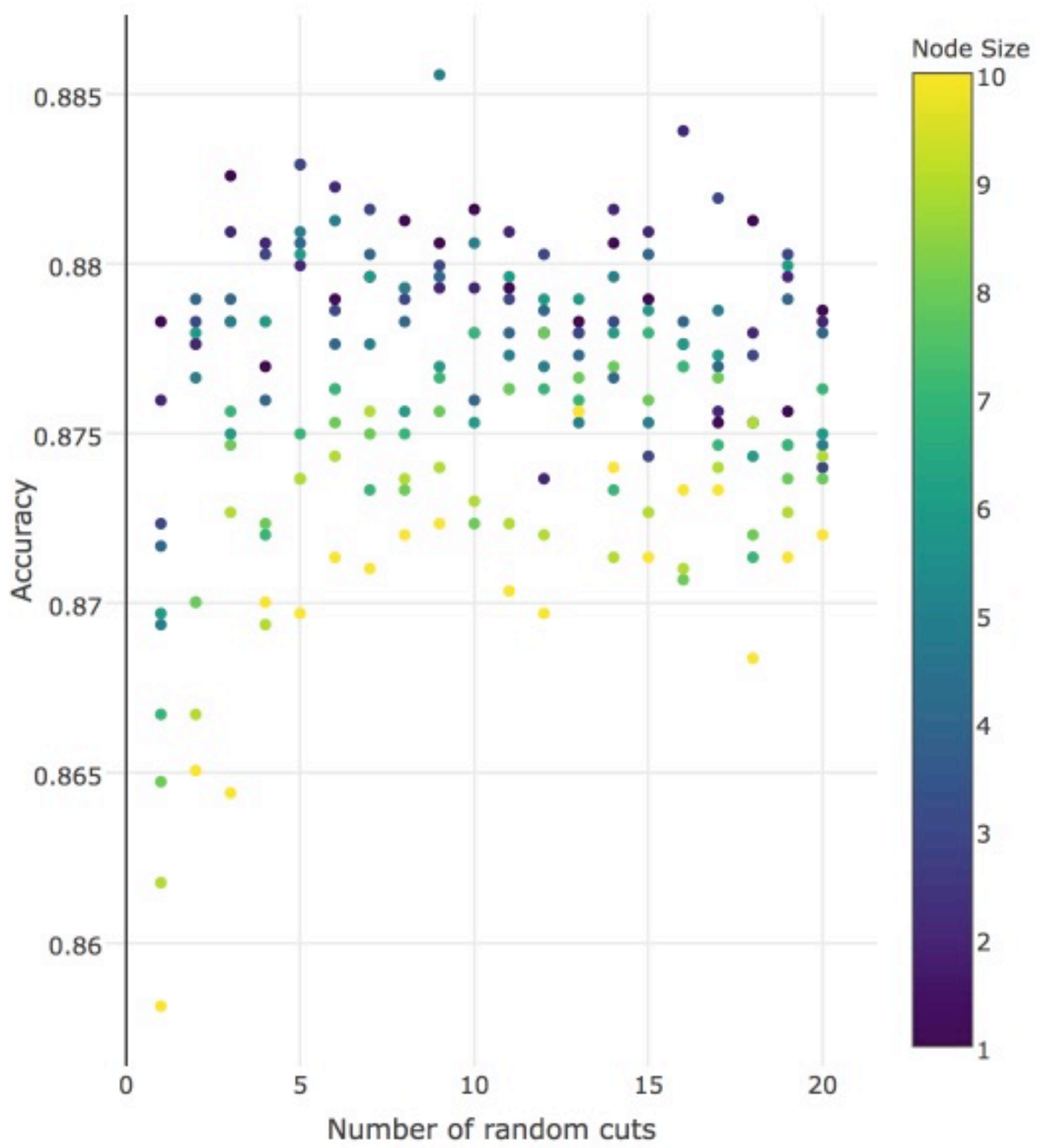
A simple implementation of ExtraTrees in R , with 13 features tried at each node, 2 random cuts led an accuracy of **79.22%**, our second best result.

The procedure for tuning the model was to first estimate an optimal number of trees , and then estimate the number of features considered for each split , and finally simultaneously estimate both number of cuts and node size to get the final model .

The model selected with the parameters mtry = 10 , ntrees = 200 , Node size = 2, Number of random cuts = 5 resulted in an improvement to our earlier model with an accuracy of **79.25%**







Ensembling

- For the first ensemble we used a majority of the best models from XGBOOST, GBM H2o, Extratrees, and Randomforest. Our final accuracy was around **76%**, less than the Extratrees standalone.
- We fed our model outputs of ExtraTrees into GBM and that led to a score of **79.182%**, the best we could do with ensembling.

Performance, Benchmarks and Accuracy

- Our final Kaggle rank was 224, achieving a top 13% ranking. This was satisfactory considering the time constraint on the project.
- Dong Ying of Technische UniversitÃ¤t Darmstadt & Shivanshu Agrawal et all at Indian Institute of Technology, Kanpur achieved similar scores for their models in 2015, but were able to improve their ExtraTrees accuracy to 82% through better feature selection and engineering.
- Our GBM model performed better than both the projects cited above largely due to H2o's improved algorithm.

Learning and Improvements

- The tuning of Neural networks can be more cumbersome and time consuming especially when it involves deep learning. This must be taken into account and techniques such as Adaptive learning rate and Randomized hyper-parameter tuning must be utilized.
- To save time when one is implementing techniques like multistage ensembles, it is important better to store model parameters, fitted values, and predicted values of the best models beforehand for each method including the probability values for each prediction.
- It is important to consider all aspects of the data and if/whether the classification problem can be solved effectively by solely machine learning approaches or alternative algorithms.
- One of the important learnings from the Project was effective utilization of Github for workflow management and coordinating as a team.
- Learning to use LateX for publishing was very useful.
- The key drivers to success in a competition like on Kaggle are creative feature engineering, effective model tuning, optimal ensembling, computational resources, and time.

Conclusion

The forest cover dataset is an extremely rich dataset for anyone who wishes to implement multi-class classification algorithms across a wide range of methods. It will continue to be a benchmark for classifying algorithms for some time.

Appendix

Models in bold denote highest test set accuracy for a specific model type

Classifier	Feature Engineering	Parameters	Accuracy	Rank
Logistic Regression - Ridge	No	70-30 split	0.55696	1,489
Logistic Regression - Ridge	No	85-15 split	0.59550	1,414
Logistic Regression - Lasso	**No**	**70-30 split best lambda**	**0.59594**	**1,411**
Logistic Regression - Lasso	No	85-15 split best lambda	0.59526	1,415
Logistic Regression - Lasso	No	85-15 split,lowest 50 percintile of lambdas and calculated the mode of the predictions	0.59443	1,418
Logistic Regression - Elastic-net	No	70-30 split	0.59588	1,412
Logistic Regression - Elastic-net	No	85-15 split	0.59520	1,415
Logistic Regression - Elastic-net	No	using whole training set	0.59496	1,417
Random Forest	No	80-20 split	0.68932	1,286
Random Forest	No	using whole training set	0.70748	1,210
Random Forest	No	ntree=500, mtry=13	0.75168	792
Random Forest	**No**	**ntree=500, mtry=17**	**0.754**	**672**
Random Forest	Yes	ntree=500, mtry=13	0.69115	1,282
Random Forest	Yes	ntree=500, mtry=20	0.71929	1,221
Random Forest	Yes	ntree=500, mtry=30	0.71831	1,133

Neural Network	No	nnet pack, whole train, maxit=300	0.50867	1,554
Neural Network	No	nnet package, whole train set, maxit=500	0.51487	1,549
Neural Network	No	nnet package, 80% training set, maxit=300	0.52113	1,545
Neural Network	No	nnet package, 80% training set, maxit=500	0.52432	1,541
Neural Network - Deep Learning	No	H2o package,	0.5946	1,418
Neural Network - Deep Learning	No	H2o package, 54-100-120-7	0.60186	1,403
Neural Network - Deep Learning	No	H2o package, 54-120-150-7	0.60186	1,403
Neural Network - Deep Learning	**No**	**H2o package, 54-500-800-7, rate=0.02, rate-annealing=1e-05, rate-decay=1**	**0.67428**	1,325
XGBoost	No	nround=50	0.70033	1,258
XGBoost	**No**	**nround=104**	**0.73438**	**991**
XGBoost	Yes	after 800 run	0.72949	1,040
XGBoost	Yes	cv run by sequentially choosing the 62 variables based on importance and chose the optimal set of parameters	0.72656	1,070
GBM - H2o	No	ntrees=250, max_depth=18, min_rows=10, learn_rate=.1	0.76640	467
GBM - H2o	No	ntrees=200, max_depth=20, min_rows=10, learn_rate=.1	0.76119	521
GBM - H2o	No	ntrees=200, max_depth=20, min_rows=10, learn_rate=.1, cv 80/20	0.74509	911
GBM - H2o	No	w/o stopping_rounds=2, stopping_tolerance=0.01, score_each_iteration=T, 10-fold	0.76586	479
GBM - H2o	No	w/o stopping_rounds=2, stopping_tolerance=0.01, score_each_iteration=T, 15-fold	0.78052	356
GBM - H2o	**No**	**after reducing row sampling rate=0.4, column sampling rate=0.3 (roughly sqrt(n)/n), 9-fold**	**0.72656**	1,070
ExtraTrees	No	mtry=13, numRandomCuts = 2, nodesize = 3, numThreads = 3, ntree=50	0.79220	224
ExtraTrees	**No**	**mtry=10, numRandomCuts = 5, nodesize = 2, numThreads = 3, ntree=200**	**0.79247**	**224**
ExtraTrees	Yes	mtry=13, numRandomCuts = 4, nodesize = 3, numThreads = 3, ntree=700	0.78739	308
ExtraTrees	Yes	tuned: mtry=56, numRandomCuts = 4, nodesize = 3, numThreads = 3, ntree=200	0.757000	613
Ensemble I	No	basic voting of RF, GBM, ExtraTrees and XGBoost	0.76787	448
Ensemble II	**No**	**Feeding ExtraTrees to H2o XGB**	**0.79182**	**230**
Ensemble III	**No**	**Feeding ExtraTrees to H2o GBM then to XGB**	**0.79182**	**230**

GitHub Repository

()https://github.com/nycdatasci/bootcamp004_project/tree/master/Project4-Machinelearning/Aravind_thomas (https://github.com/nycdatasci/bootcamp004_project/tree/master/Project4-Machinelearning/Aravind_thomas)

References:

- https://scientiaplusconscientia.wordpress.com/2014/12/16/hillshading-useful-fun-with-digital-elevation-models/ (https://scientiaplusconscientia.wordpress.com/2014/12/16/hillshading-useful-fun-with-digital-elevation-models/)
- https://scientiaplusconscientia.wordpress.com/2014/08/06/working-with-modis-l1b-from-scratch-4-topographic-and-illumination-correction/ (https://scientiaplusconscientia.wordpress.com/2014/08/06/working-with-modis-l1b-from-scratch-4-topographic-and-illumination-correction/)
- http://www.geography.hunter.cuny.edu/~jochen/GTECH361/lectures/lecture11/concepts/Hillshade.htm (http://www.geography.hunter.cuny.edu/~jochen/GTECH361/lectures/lecture11/concepts/Hillshade.htm)
- https://upload.wikimedia.org/wikipedia/commons/3/38/Law-of-haversines.svg

(https://upload.wikimedia.org/wikipedia/commons/3/38/Law-of-haversines.svg)

- https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html (https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html)
- http://H2o-release.s3.amazonaws.com/H2o/rel-turan/3/docs-website/H2o-docs/booklets/DeepLearning_Vignette.pdf (http://H2o-release.s3.amazonaws.com/H2o/rel-turan/3/docs-website/H2o-docs/booklets/DeepLearning_Vignette.pdf)

- http://www.isas.org.in/jsp/volume/vol62/Apr-08/Apr-08-9.pdf (http://www.isas.org.in/jsp/volume/vol62/Apr-08/Apr-08-9.pdf)
- https://www.researchgate.net/publication/222499246_Comparative_Accuracies_of_Artificial_Neural_Networks_and_Discriminant_Analys (https://www.researchgate.net/publication/222499246_Comparative_Accuracies_of_Artificial_Neural_Networks_and_Discriminant_Analy
- http://www.csee.usf.edu/~hall/papers/smc04nn.pdf (http://www.csee.usf.edu/~hall/papers/smc04nn.pdf)

- http://www.ke.tu-darmstadt.de/lehre/arbeiten/studien/2015/Dong_Ying.pdf (http://www.ke.tu-darmstadt.de/lehre/arbeiten/studien/2015/Dong_Ying.pdf)
- http://home.iitk.ac.in/~shefalig/g8p5.pdf (http://home.iitk.ac.in/~shefalig/g8p5.pdf)