Prediction of Forest Cover Type using Ensemble Techniques with Decision Trees

Submitted by

CHANDANA T.L	***11IT15***
NAVAMI K	***11IT48***
NISHA K.K	***11IT51***
PRUTHVI H.R	***11IT64***

Under the Guidance of

Mr. Biju R Mohan

Dept. of Information Technology

NITK Surathkal, Mangalore

In partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Information Technology

Department Of Information Technology

National Institute of Technology Karnataka, Surathkal

Srinivasnagar – 575025, Mangalore, India

2014-2015

Department of Information Technology
NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA
SURATHKAL, MANGALORE

DECLARATION

We hereby declare that the Project Work Report entitled **Prediction of Forest Cover Type using Ensemble Techniques with Decision Trees** which is being submitted to the **National Institute of Technology Karnataka, Surathkal** for the award of the Degree of Bachelor of Technology in **Information Technology** is a bonafide report of the work carried out by us. The material contained in this Project Work Report has not been submitted to any University or Institution for the award of any degree.

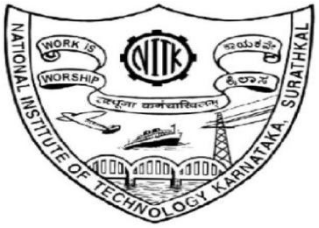
Name of the Student	Register No.	Signature with Date
1. Chandana T.L	11IT15	
2. Navami K	11IT48	
3. Nisha K.K	11IT51	
4. Pruthvi H.R	11IT64	

Department of Information Technology, NITK

Place: NITK, Surathkal

Date:

Department of Information Technology
NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA
SURATHKAL, MANGALORE



CERTIFICATE

This is to certify that the B.Tech Project Report entitled **Prediction of Forest Cover Type using Ensemble Techniques with Decision Trees** submitted by:

1. Chandana T.L 11IT15
2. Navami K 11IT48
3. Nisha K.K 11IT51
4. Pruthvi H.R 11IT64

as the record of the work carried out by them, is accepted as the B.Tech Project work Report submission in partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Information Technology.

Mr. Biju R Mohan	Prof. Ananthanarayana V.S
Project Guide, Dept. of IT	Chairman – DUGC, Dept. of IT
NITK Surathkal, Mangalore	NITK Surathkal, Mangalore

ACKNOWLEDGEMENTS

Throughout the course of this project, several people have helped directly and indirectly the project, contributing to the successful finish of our final year in the Bachelor of Technology course. Our guide, Mr. Biju R Mohan, was not only the guide, but also a motivator who ensured that we stayed on track, provided help when necessary and played an important role in making us overcome the deadlocks at various stages of our project. Our project would be incomplete without the mention of our Head of Department, Prof. Ananthanarayana V.S, who facilitated access to all the resources in the department. We would also like to thank Dr. Nagamma Patil, who provided timely suggestions and motivated us to think better with her valuable feedback during evaluations. Our faculty advisor, Mrs. Sowmya Kamath was always there to guide us to the right path by providing right suggestions through the major project evaluations. We also thank all the teachers and friends in the Masters courses for their feedback during the evaluation sessions, for providing insight when needed and guiding us out of deadlocks when we faced many. It is also our duty to thank our classmates and other friends who directly or indirectly played a part in the success of the project. Ultimately, thanks go to the institution of which we are proud of the National Institute of Technology Karnataka, Surathkal, and all the people associated, for being our haven and facilitating and inspiring us during the course of the major project.

Chandana T.L, Navami K, Nisha K.K, Pruthvi H.R

Dept of IT, NITK Surathkal, Mangalore

Abstract

The project aims to determine the forest cover type of the dataset containing strictly cartographic variables. The study evaluated four wilderness areas in the Roosevelt National Forest, located in northern Colorado. The source of cover type data was US Forest service inventory while the cartographic variables like elevation, slope, soil type and other data were derived from Geographic Information System(GIS). Dataset was analysed and feature engineering techniques were applied, which helped in getting more relevant features. Comparative study of various decision tree algorithms such as C4.5, C5.0, CART was conducted on the dataset. Random Forest and C5.0 gave better results compared to other decision trees. As a further improvement, we intend to build better ensemble techniques for decision trees.An ensemble technique which has simple brown boost implemented along with C5.0 is used to build the model. This ensemble learning approach constructs a composite hypothesis by rating individual hypothesis. The memory required to store these hypothesis is high and can be reduced by selecting a set of hypothesis which gives nearly the same performance. The package built includes Kappa Pruning in order to select the boosted classifiers.

Keywords:Decision Tree, C5.0, Ensemble Techniqiues, Pruning

Contents

List of Figures

List of Tables

1 Introduction

Classification is an important research topic in the field of data mining and knowl- edge discovery. It finds the common properties among a set of objects in a dataset and classifies them into different pre-identified classes. There have been many data classification methods including decision trees methods, statistical methods, neural networks, rough sets etc. One of the instances where classification techniques can be applied is for prediction of forest cover types. The prediction of the forest cover type (the predominant kind of tree cover) can be determined from strictly cartographic variables. The actual forest cover type for a given 30 x 30 meter cell was determined from US Forest Service (USFS) Region 2 Resource Information System data. Independent variables were then derived from data obtained from the US Geological Survey and USFS. The data is in raw form (not scaled) and contains binary columns of data for qualitative independent variables such as wilderness areas and soil type[1].
The study area for the project consisted of the Rawah (29 628 hectares or 73 213 acres), Comanche Peak (27 389 hectares or 67 680 acres), Neota (3904 hectares or 9647 acres), and Cache la Poudre (3817 hectares or 9433 acres) wilderness areas of the Roosevelt National Forest in northern Colorado. As shown in Figure 1, these areas are located 70 miles northwest of Denver, Colorado. These wilderness areas were selected because they contained forested lands that have experienced relatively little direct human management disturbances. As a consequence, the current composition of forest cover types within these areas are primarily a result of natural ecological processes rather than the product of active forest management.

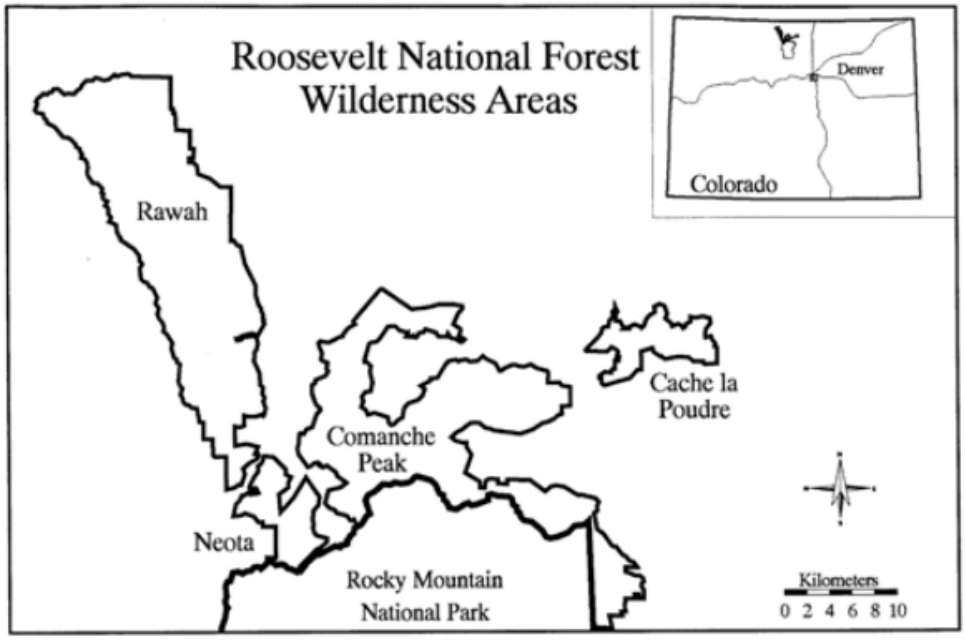


Figure 1.1: Roosevelt National Forest Wilderness Area

Some background information for the four wilderness areas: The Rawah (area 1) and Co-

manche Peak (area 3) areas would tend to be more typical of the overall dataset than either the Neota (area 2) or Cache la Poudre (area 4), due to their assortment of tree species and range of predictive variable values (elevation, etc.). Cache la Poudre would probably be more unique than the others, due to its relatively low elevation range and species composition.

This is a classification problem, where the forest cover type are predicted. The forest cover type is a predominant tree species which can take seven possible nominal values. The problem of identifying the category to which the new observation belongs to based on the a model built using a training set containing observations whose category membership is already known is called as Classification.

1.1 *Motivation for Work*

Accurate natural resource inventory information is vital to any private, state, or federal land management agency. Forest cover type is one of the most basic characteristics recorded in such inventories. Generally, cover type data is either directly recorded by field personnel or estimated from remotely sensed data. Both of these techniques may be prohibitively time consuming and/or costly in some situations. Furthermore, an agency may find it useful to have inventory information for adjoining lands that are not directly under its control, where it is often economically or legally impossible to collect inventory data. Predictive models provide an alternative method for obtaining such data.

2 Literature Survey

2.1 *Background*

Blackard et. al.[1] have compared two alternative techniques for predicting forest cover types. The results of the comparison indicated that a feed-forward artificial neural network model(70.58%) more accurately predicted forest cover type than a traditional statistical model based on Gaussian discriminant analysis (58.38%). B. Chandra et. Al [2] used the same dataset to evaluate the performance of the decision trees. The decision tree algorithm achieved a maximum classification accuracy of 84% as compared to that of 70.58%. Ragini Jain et. Al. suggested a hybridized rough set model that provides mechanism to trade-of between different performance parameters like - accuracy, complexity, number of rules and number of attributes in the resulting classifier for a large benchmarking dataset.

A number of supervised learning methods have been introduced in the last decade. Caruana et.al [3] presents a large-scale empirical comparison of ten Supervised Learning algorithms using eight performance criteria. They evaluate the performance of SVMs, neural network, logistic regression, naive bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees, and boosted stumps on eleven binary classification problems using a variety of performance metrics: accuracy, F-score, Lift, ROC Area, etc. They have concluded that learning methods such as boosting, random forests, bagging, decision tree and SVMs achieve excellent performance over others.

Entezari- Malecki et al. [4] have compared different classification methods based on type of attributes and sample size against performance criterion Area Under the Curve (AUC) of ROC. Their analysis shows that decision tree and C4.5, as an implementation of that show an effective performance in all datasets. Decision tree, C4.5 and SVM show excellent accuracies when the number of continuous attributes is higher that are discrete.

Quinlan[5] experimented the effect of boosting on 27 datasets with C4.5 as base learner. Boosting reduced the classification error by 15%, but improved the performance on 21 datasets. And also explains that one of the reasons for higher error rate in classification is over fitting of the tree. Works of Schapire[6] notes that the boosting technique fails if the weak learner achieves less than 50% accuracy on its own. Zhu et.al [7] worked on extending the adaboost technique to multiclass classification and has observed that the technique gives better results. Freund et. al[8] propose that boosting maybe helpful on learning algorithms having either of the two properties. The first property, which holds for many real-world problems, is that the observed examples tend to have varying degrees of hardness. For such problems, the boosting algorithm tends to generate distributions that concentrate on the harder examples, thus challenging the weak learning algorithm to perform well on these harder parts of the sample space. The second property is that the learning algorithm be sensitive to changes in the training examples so that significantly different hypotheses are generated for different training sets.

Margineantu et al [9] addresses the problem that deploying ensemble methods require a

large amount of memory to store all of the classifiers and the question of whether all of the decision trees constructed by AdaBoost are essential for its performance. He introduced the concept of pruning the ensemble which deals with discarding some of the trees constructed but still obtaining the same high level of performance. The paper introduces five different pruning algorithms and compares their performance on a collection of ten domains. The five different pruning algorithms used in this paper are: Early Stopping, KL-divergence Pruning, Kappa Pruning, Kappa-Error Convex Hull Pruning, and Reduce-Error Pruning with Backfitting. These five pruning techniques were tested on ten datasets using C4.5 as the weak learner. The experiments performed concluded that the ensemble produced by AdaBoost can be radically pruned in some domains and the best pruning methods were Kappa Pruning and Reduce-Error Pruning.

We propose to predict the forest cover type from strictly cartographic values. We intend to do so by comparing the different decision tree - C4.5, C5.0, CART and Random Forest. We have later on performed comparative study on different boosting techniques on decision trees and finally try to model the best ensemble learning for decision trees.

Table 2.1: Brief Description of the literature Survey Papers

PaperTitle	Authors	Description	Inference
Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables(2000)	Jock A.Blackard Denis J.Dean	*Feedforward artificial neural network model *Gaussian discriminant analysis *UCI Forest Cover Dataset	ANN model (70.58%) predicted forest cover type more accurately than Gaussian discriminant analysis (58.38%)
Prediction of Forest cover using Decision Trees(2007)	B.Chandra, Pallath Paul V.	*Improved SLIQ *UCI Forest Cover Dataset	Accuracy improvement to 84%
Drawing conclusion from Forest Cover Type data - Hybridised rough set model(2008)	Ragini Jain, Sonajhania Minz	*Hybridised Rough Set model(RS+DT) *UCI Forest Cover Dataset	Results comparable with published results for the dataset
An Empirical Comparison of Supervised Learning Algorithms(2006)	Rich Caruana, Alexandru Niculescu-Mizil	*SVM, ANN, logistic regression, NB, random forests, DT, bagged trees, boosted trees, and boosted stumps. *Multiple datasets *Multiple performance metrics	Boosting, Random Forests, Bagging, Decision Tree and SVMs achieve excellent performance over others.
Comparison of Classification Methods Based on the type of Attributes and Sample Size(2009)	Reza Entezari-Maleki, Arash Rezaei, Behrouz Minaei- Bidgoli	*DT, KNN, Logistic Regression, NB, C4.5, SVM *Performance metric: Area Under Curve (AUC) *Random generated dataset	DT(C4.5) provided higher AUC in the most cases. (Continuous attributes ¿ Discrete attributes)

Continued on next page

Table 2.1 – *Continued from previous page*

Bagging, Boosting, and C4.5(2006)	J. R Quinlan	*Boosting and Bagging on C4.5 *27 datasets	*Boosting reduced classification error by 15% for 21 datasets *Bagging reduced classification error by 10% for 24 datasets
A Brief Introduction to Boosting(1999)	Robert E.Schapire	*Boosting and C4.5 *UCI benchmark datasets	Boosting C4.5 improved performance significantly
Multi-class AdaBoost(2009)	Ji Zhu, Hui Zou, Saharon Rosset, Trevor Hastie	*Multi Class Adaboost using SAMME *Seven benchmark datasets	Adaboost can be extended for Multi Class classification problems.
Experiments with a New Boosting Algorithm(1996)	Yoav Freund, Robert E. Schapire	*Comparison of Bagging and Boosting on C4.5	Boosting and bagging improves performance of C4.5
Pruning Adaptive Boosting(1997)	Dragos D. Margineantu, Thomas G. Dietterich	*Adaptive Boosting algorithm AdaBoost *Pruning the ensemble *Five different pruning algorithms	Ensemble produced by AdaBoost radically pruned in few datasets and the best pruning methods were Kappa Pruning and Reduce-Error Pruning

2.2 *Outcome Of Literature Survey*

The extensive literature survey was done at every stage of the project which led to the following outcomes. Table 2.1 summarizes the literature survey and the inferences drawn from them.

- The previous works done on this dataset used decision trees, Artificial Neural Networks, Gaussian discriminant analysis and many other methods. However, the predictive models developed so far on Forest Cover Type dataset gave highest accuracy of 84%. There is a need to build a more accurate model.

- Many supervised learning methods have been proposed for prediction. A comparative study of such methods showed that Decision Trees, Bagging and Boosting techniques work well in general.
- Though the decision trees were known to give good results, literature survey indicated that Boosting and Bagging on C4.5 would possibly yield better results compared to the decision tree alone.
- Decision Trees(C4.5) showed the higher AUC value than others when the number of continuous attributes is higher than discrete ones. And since our dataset contains higher number of continuous attributes, Decision Tree algorithms can be employed for building the predictive model.
- Kappa Pruning and Reduce-Error Pruning showed the best results on AdaBoost with C4.5. Both the techniques reduced the number of classifiers chosen for prediction hence reducing the memory used to store the classifiers.

2.3 *Problem Statement*

We propose to predict the forest cover type from strictly cartographic values. We intend to do so by combining data analysis with decision tree models, further enhanced by boosting and bagging.

2.4 *Objective*

In order to achieve our main goal of building a more accurate model, we put forth the following objectives for the project.

- Build predictive models based on different decision trees like CART, C4.5, C5.0, Random Forest and draw conclusions on which decision tree works best for our dataset.
- Feature Engineering to generate new features which help in better learning.
- Feature selection to get the best set of features to build the model.
- Test boosting techniques with the selected decision tree as base learner.
- Improve the performance of the boosting technique for the selected decision tree.

3 Methodology

3.1 *Data Description*

Forest cover type dataset contains only cartographic variables (no remotely sensed data). The actual forest cover type for a given observation (30 X 30 meter cell) was determined from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. Independent variables were derived from data originally obtained from US Geological Survey (USGS) and USFS data. Data is in raw form (not scaled) and contains binary (0 or 1) columns of data for qualitative independent variables (wilderness areas and soil types).
The dataset for forest cover type prediction includes 54 features. The study area includes four wilderness areas located in the Roosevelt National Forest of northern Colorado. These areas represent forests with minimal human-caused disturbances, so that existing forest cover types are more a result of ecological processes rather than forest management practices. We intend to predict an integer classification for the forest cover type. The seven types are:

- Spruce/Fir
- Lodgepole Pine
- Ponderosa Pine
- Willow
- Aspen
- Douglas/Fir
- Krummholz

The training set (15120 observations) contains both features and the cover type. The test set contains only the features. We must predict the cover type for the test set (565892 observations). The dataset contains continuous, binary and nominal data. Table 3.1 gives the description of all the attributes in our dataset.

Table 3.1: Description of attributes for forest cover dataset

Attribute Name	Data Type	Measurement	Description
Elevation	Quantitative	Meters	Elevation in meters
Aspect	Quantitative	Azimuth	Aspect in degrees azimuth
Slope	Quantitative	Degrees	Slope in degrees
Horizontal_Distance _To_Hydrology	Quantitative	Meters	Horizontal distance to nearest surface water features
Vertical_Distance _To_Hydrology	Quantitative	Meters	Vertical distance to nearest surface water features
Horizontal_Distance _To_Roadways	Quantitative	Meters	Horizontal distance to nearest roadways
Hillshade_9am	Quantitative	0 to 255 index	Hillshade index at 9am,summer solstice
Hillshade_Noon	Quantitative	0 to 255 index	Hillshade index at noon,summer solstice
Hillshade_3pm	Quantitative	0 to 255 index	Hillshade index at 3pm, summer solstice
Horizontal_Distance _To_Fire_Points	Quantitative	Meters	Horizontal distance to nearest wildfire ignition points
Wilderness_Area(4 binary columns)	Qualitative	0 (absence) or 1 (presence)	Wilderness area designation
Soil_Type(40 binary columns)	Qualitative	0 (absence) or 1 (presence)	Soil type designation
Cover_Type(7 types)	Integer	1 to 7	Forest Cover Type designation

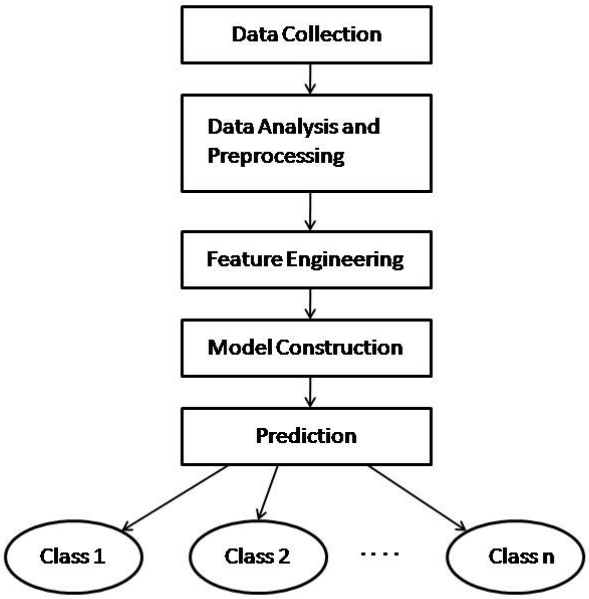


Figure 3.1: Flow Diagram

Figure 3.1 shows the flow diagram of the steps followed in classification in data mining. The first step of involves collecting data for classification. The dataset for this project is available in UCI machine learning repository. As mentioned earlier, the data is derived from USFS Region 2 RIS data.

Data analysis and feature engineering involves studying the dataset. Feature engineering involves understanding the properties of the task the user is trying to solve and how they might interact with the strengths and limitations of the model the user is going to use. Feature engineering consists of a cycle:

- Design a set of features.
- Run an experiment and analyze the results on a validation dataset.
- Change the feature set.

Feature engineering is followed by feature selection. Feature selection deals with identifying features that are redundant or irrelevant. Basic approaches of feature selection are:

- Filter: use the most promising features according to ranking resulting from a proxy measure. E.g. from mutual information or correlation coefficient
- Wrapper: search through the space of subsets, train a model for current subset, evaluate it on a held-out data, and then iterate. Simple greedy heuristics involve forward selection and backward selection.

- Embedded: feature selection is a part of model construction and incorporates the properties of both filter and wrapper methods.

The classification algorithm chosen for prediction is applied to the dataset. The records of the dataset are classified to one of the class labels.

3.2 *Feature Engineering*

3.2.1 *Feature Extraction*

Feature extraction derives more features from the initial set, which are intended to be more relevant, non-redundant, improving the subsequent learning, in some scenarios leading to better human interpretations.

When the data input to a model contains redundant features (e.g. length in feet and meters, or age and date of birth both), then it can be transformed into a reduced set of features. Such techniques used for transformation are feature extraction techniques. The extracted features contain more relevant information, hence they can be used for model building instead of the initial set of features.

The best results are obtained when domain knowledge is used to construct features. But, if such knowledge is not available, general dimensionality reduction methods might still help.

3.2.2 *Feature Selection*

Feature selection is a process whereby a subset of relevant features are chosen for model building. The need for feature selection arises only when the dataset contains many redundant or irrelevant features. The features which do not any more information than the currently subset of features are called redundant, whereas features which do not provide any useful information at all are called irrelevant.

Feature selection should not be confused with feature extraction. Feature extraction techniques generate new features from functions of the original features, whereas feature selection only selects a subset of the features. Feature selection techniques are highly useful for high-dimensional data.

3.2.3 *Subset selection*

Subset selection checks for the suitability of feature subset. Three approaches to subset selection are Wrappers, Filters and Embedded techniques. Wrappers search through all possible subsets and evaluate each subset by running a model on the subset. Wrapper algorithms are usually computationally expensive and might as well overfit to the selected model. Filters are

different from Wrappers as they evaluate against a simpler filter instead of a model. Embedded techniques are combination of both.

Subset evaluation uses a scoring metric to grade subset of features[10].
Various search approaches are:

- greedy hill climbing
- Exhaustive
- Best first
- Genetic algorithm
- Greedy forward selection
- Greedy backward elimination

Two popular filter metrics for classification problems are correlation and mutual information.

3.2.4 *Optimality criteria*

The choice of optimality criteria is difficult since feature selection task has to meet many objectives. Many common criteria incorporate accuracy related measures, the number of features selected (e.g. the Bayesian information criterion) etc...

3.3 *Decision Trees*

A decision tree is a simple model for classifying data objects. Decision tree learning is a popular technique used for classification. A decision tree is a tree like structure in which each internal node is represents a test on an attribute. The branches leading from the node represent the outcome of this condition.

The leaves of the tree are labeled with class names. A tree is constructed by partitioning the initial data set into subsets based on attribute test at the current node. This process is repeated on every derived subset recursively and is called recursive partitioning. The recursion get terminated when the subset at a node has pure parttions, or when partitioning no longer adds value to the classifications.

A number of decision trees have been proposed in the literature and they mainly differ by the measure used for attribute selection. In this section, we discuss these trees in the order of their evolution.

3.3.1 *ID3*

Quinlan *et al.* [11] suggested information gain of an attribute as a measure for its selection. The expected information (i.e Entropy) required to classify a sample in dataset D is given by

$$Info(D) = -\sum_{i=1}^{m} p_i log_2(p_i) \tag{1}$$

where p_i is the probability of a sample to belong to class C_i among m classes. After this partitioning, the information needed to reach exact classification is

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j) \tag{2}$$

where attribute A takes v distinct values, $a_i, a_{i+1}...a_v$. Information Gain is the difference between initial information requirement and requirement after the split. That is,

$$Gain(A) = Info(D) - Info_A(D) \tag{3}$$

The attribute with highest information gain value is chosen is chosen for splitting. Although ID3 algorithm gave quite good results, it's attribute selection criterion has a serious problem, it has strong bias in favour of tests with many outcomes. Also, ID3 does not provide a way to handle continuous and missing valued attributes.

3.3.2 *CART*

The CART methodology developed by Breiman et al.[12] is discussed in this section.

Tree growing procedure Let (X, Y) be an observation in the learning set L, where X is the set of attributes in the dataset, both continuous and categorical and Y is the response variable or the class label which takes values in the set $C(1, 2....J)$ with 1 to J being the class labels.

The starting point of the tree growing procedure is the root node which contains the entire learning set L. The tree is grown by finding the best attribute to split on. The attribute which is chosen for splitting is the one which has the largest value of gini diversity index over all the other variables.

Let $A1, A2....A_k$ be the set consisting of observations belonging to the class $C(1, 2....k)$. For node T, the gini diversity index is calculated as below:

$$i(T) = \sum_{k \neq k'}^{n} p(k/T)p(k'/T) = 1 - \sum_{k} p(k/T)^2 \tag{4}$$

Where $p(k|T)$ is an estimate of $P(X\varepsilon\Pi_k|\Gamma)$ the conditional probability that the observation X is in A_k given that it belongs to node T.

While growing a tree, splitting of attributes starts at the root node. Using the gini diversity index, the algorithm chooses the attribute which has the highest value as the splitting attribute. Next is to progress to each of the daughter nodes and find the best split in the same way. The computation of the gini diversity index is done for each daughter node on the observations that are specific to the node. This procedure which is carried out in sequence to build a tree layer by layer is called recursive partitioning.

The misclassification rate and pruning procedure In the initial stages of the tree growing procedure, the predictive accuracy typically increases as more and more nodes are created and the partitions get finer. But as the complexity of the tree increases, due to overfitting, the misclassification rate for future cases will increase. In order to compare the predictive accuracy of the different tree models, a measure called resubstitution estimate is used. Resubstitution estimate of the misclassification rate is obtained by using the tree to classify the members of the learning sample and observing the proportion that are misclassified. The resubstitution estimate of the misclassification rate $R(\tau)$ of an observation at node τ is calculated as follows:

$$R(\tau) = 1 - max_k p(k/\tau) \tag{5}$$

Pruning procedure: In the pruning of tree, a specific node for which the corresponding pruned nodes provide the smallest per node decrease in the resubstitution misclassification rate is selected. If two or more choices in the pruning process produces the same decrease in the resubstitution misclassification rate per node, pruning of the part which has larger number of nodes is preferred.

3.3.3 *C4.5*

3.3.4 *C4.5*

Gain Ratio Criterion C4.5 proposed by Quinlan [13] uses an extension of information gain called gain ratio which overcomes the bias towards multi valued attributes. It normalizes the information gain by using a "split information" value defined as

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2 \frac{|D_j|}{|D|} \tag{6}$$

This value tells the potential information generated by partitioning the training dataset, D, into V partitions, corresponding to the V outcomes of attribute test on A whereas information gain measures the information with respect to classification that is acquired based on the same

partitioning. The gain ratio is given by

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2 \frac{|D_j|}{|D|} \tag{7}$$

It gives the part of the information generated by the split which is useful for classification. The attribute with highest gain ratio is chosen for splitting. However, as the split information approaches 0, the GainRatio(A) becomes unstable. To avoid this, a constraint can be added such that the information gain for the attribute selected must be at least equal to the average gain of all the attributes examined.

Handling continuous and missing values In case of continuous valued attributes, the outcome of the attribute test can be based on a threshold(called split-point) on A. A possible split-point could be the average of adjacent values, when arranged in sorted order. Therefore, given v values of A, then V-1 possible splits are evaluated. If the values of A are sorted in advance, then determining the best split for A requires only one pass through the values. For each possible split-point for A, $Info_A(D)$ should be calculated with the number of partitions being two, i.e. V=2(or j=1,2). The point with minimum $Info_A(D)$ is selected as the split-point for A. To handle missing values, only the values defined can be considered for evaluating the gain ratio during tree construction. In using a decision tree for prediction, sample with missing attribute values can be estimated as the probability of the various possible results.

Pruning Decision Trees Since the decision tree built using the training set, it gives accurate results for most of tuples in the training set than unseen dataset. In order to get these accurate results, decision tree built tends to be complex with long and uneven path. Also, the branches that reflect anomalies in the training data may result in overfitting the data. Pruning of the decision tree, a solution to overfitting, is achieved by replacing a subtree by a leaf. The replacement is done when expected error rate in the subtree is greater than in the single leaf.

3.4 *Ensemble Learning*

Ensemble learning is a machine learning paradigm where multiple learners are trained to solve the same problem. In contrast to ordinary machine learning approaches which try to learn one hypothesis from training data, ensemble methods try to construct a set of hypotheses and combine them to use. In this section, the ensemble technique is discussed with decision trees as the weak learners.

3.4.1 *C5.0*

C4.5 algorithm ensured better way of building a decision tree with the use of gain ratio has the spitting factor but it lacked many non-functional requirements to popularize among the appli-

cations. Hence, C5.0 [14] was developed as an improvement over the existing C4.5 algorithm. Many key aspects found in C5.0 makes it better than C4.5 Algorithm. Below are the list of its extended features:

- Rules Formation - Building the decision trees and using then for every test set prediction leads to high wastage of time. In order to overcome this, C5.0 has incorporated a ruleset formation instead of trees to save significant amount of time and space.
- Winnowing technique - used in C5.0 helps in reduction of memory by using lesser sample set, which was not available in C4.5.
- Smaller decision trees - C5.0 gets similar results to C4.5 with considerably smaller decision trees.
- Boosting - Support for boosting allows the creation of multiple decision trees . These trees formation is improvement over time till no more misclassification. Then based on the decision from all the classifiers a better prediction is made leading to better accuracy.
- Weighting - C5.0 allows you to vary the importance of different cases by giving different weights. Minimizing the weighted predictive error rate is the way to handle weighting.
- Misclassification Costs - In C4.5, all errors are treated as equal, but in practical applications some classification errors are more serious than others. Hence, C5.0 allows a separate cost to be defined for each predicted versus actual class pair ratio; if this option is used, C5.0 then constructs classifiers to minimize expected misclassification costs rather than error rates.
- Data Types - C5.0 has several new data types in addition to those available in C4.5, including dates, times, timestamps, ordered discrete attributes, and case labels. In addition to missing values, C5.0 allows values to be noted as not applicable. Further, C5.0 provides facilities for defining new attributes as functions of other attributes.

Working of C5.0 Algorithm are explained below[15]:

- Input: Training set T with n attributes i.e, $A_1, A_2, A_3.....A_n$ and m tuples i.e, $t_1, t_2, t_3, ...t_m$
- Output: a single decision tree M
- Model Building Algorithm with training set:
  * Check for the base case and exit if Base case is true.
  * If Base case = true :

* Constructing a Decision Tree using training data:-
  (a) Find the attribute with the highest gain ratio (A_2 is the best, so $A_2 == A_best$)
  (b) A_Best is assigned with Entropy minimization.
  (c) Partition the T into $T_1, T_2, T_3, ...$ according to the value of A_Best
* Repeat the steps for T_1, T_2, T_3

Prediction: For each t_i tuple belonging to D Test Set, apply the ti to the M model to predict the required value.

Base cases are the following:

- All the examples from the training set belong to the same class (a tree leaf labeled with that class is returned).
- The training set is empty (returns a tree leaf called failure).
- The attribute list is empty (returns a leaf labeled with the most frequent class of all the classes).

Boosting in C5.0 can be explained as follows:

- Input: Training set T with tuples $t_1, t_2, ...t_n$, Trials R
- Output: Boosted Decision trees M with $DT_1, DT_2, ...DT_r$
- Algorithm:
  * Every trial $r <= R$ where r is initialized to 1
  * Check if there is no improvement in the next DT formed or if the accuracy is lesser than random prediction then exit or else continue.
  * Every tuple t_i is initialized to weight wi such that weight is indicative of the sample's importance. The higher the weight is, the more the sample influence on the decision tree.
  * A new decision tree DT_r is constructed.
  * The weight of each sample is adjusted, such that the samples which are misclassified by the DT_r will be given higher weight in the next iteration.
  * Repeat the process.

3.4.2 *Random Forest*

Random forests are an ensemble technique for classification, regression and other tasks. It operates by constructing many decision trees at training time and then finding the mode of the classes (classification) or mean prediction (regression) to output the final class. It improves the stability and accuracy of the machine learning algorithms used in classification and regression. It also reduces variance and helps to avoid overfitting.

The training algorithm for random forests applies the general technique of bagging to tree learners. Given a training set $X = x_1, x_2, ..., x_n$ with responses $Y = y_1$ through y_n, bagging repeatedly selects a bootstrap sample of the training set and fits trees to these samples:

For b = 1 through B:

1. Sample, with replacement, n training examples from X, Y; call these X_b, Y_b.

2. Train a decision or regression tree f_b on X_b, Y_b.

After training, predictions for unseen samples can be made by averaging the predictions from all the individual regression trees on:

$$\hat{f} = 1/_B \sum_{b=1}^{B} \hat{f}_b(x') \tag{8}$$

or by taking the majority vote in the case of decision trees.

This bootstrapping procedure leads to a better model performance because it decreases the variance of the model. This means that the average predictions of many trees is not sensitive to noise even though the prediction of a single tree is in its training set. B is a free parameter which corresponds to the number of trees. Typically depending on the size and nature of the training set, a hundred thousand trees are used. Cross-validation can be used to find the optimal number of trees, B. After a few trees have been fit, the training and test error comes down.

The above procedure describes the original bagging algorithm for trees. Random forests are slightly different from this original procedure in terms of the feature bagging property which selects a random subset of features at each candidate split in the learning process.This is done because if one or a few features are a very strong predictors for the target output, these features will be selected in the B trees, causing them to become correlated. Typically, for a dataset with p features, $\sqrt{p}$ features are used in each split.

3.5 *Pruning Boosted Classifiers*

Adaptive boosting algorithms in combination with C4.5 and C5.0 have shown to be learning procedures which have high accuracy. The ensemble techniques work by generating set of classifiers and then having a voting measure to classify the data in the test set. However, one of the disadvantage of ensemble techniques is that it requires a large amount of memory to store all

the classifiers that are generated. Hence, there is a need to reduce the memory consumption. This can be achieved by selecting a subset of classifiers which would produce a performance comparable to the performance got when all the boosted classifiers are considered. Several algorithms like Early Stopping, KL-Divergence Pruning, Kappa Pruning, Reduced error pruning with backfitting can be used to find the subset of classifiers. In the package that we have developed, Kappa Pruning is used as an algorithm to find the subset of classifiers.

Kappa Pruning: The accuracy of the pruning method depends on constructing a diverse, yet accuratecollection of classifiers. One of the ways of selecting diverse classifiers is to measure how much the classification decisions differ. A measure of agreement between the classifiers called Kappa statistics can be used and is defined as follows.

For a dataset containing m examples, consider two classifiers namely ha and hb, A contingency table called C is constructed in which each cell C_{ij} contains the number of examples for which $ha(x) = i$ and $hb(x) = j$ where i and j are class labels. If ha and hb are similar, non-zero values will appear along the diagonal of the table. If ha and hb are different, the non-zero values would appear elsewhere other than the diagonal. The probability that two classifiers agree with each other can be defined as,

$$\theta_1 = \frac{\sum_{i=1}^{L} C_{ii}}{m} \tag{9}$$

In cases where one class is more common than the others, all the potential classifiers will tend to agree with each other, just by chance and those set of classifiers will obtain high value of θ_1. The measure of agreement should be high only for those classifiers which tend to agree with each other more than just random agreements. To correct this, we define,

$$\theta_2 = \sum_{i=1}^{L} (\sum_{j=1}^{L} \frac{C_{ij}}{m} . \sum_{j=1}^{L} \frac{C_{ij}}{m}) \tag{10}$$

Where θ_2 is the probability that the two classifiers agree by chance. K statistic is defined as

$$K = \frac{\theta_1 - \theta_2}{1 - \theta_1} \tag{11}$$

K = 0 when the agreement of the two classifiers equals that expected by chance, and K = 1,when the two classifiers agree on every example. Negative values occur when agreement is weaker than expected by chance, but this rarely happens.

4 Work Done

4.1 *Data Preprocessing*

1. *Wilderness Areas* According to [16], the Rawah and Comanche Peak areas would tend to be more typical of the overall dataset than either the Neota or Cache la Poudre, due to their assortment of tree species and range of predictive variable values (elevation, etc.) Cache la Poudre would probably be more unique than the others, due to its relatively low elevation range and species composition.

2. *Elevation with Wilderness Area* Neota (area 2) probably has the highest mean elevational value of the 4 wilderness areas. Rawah (area 1) and Comanche Peak (area 3) would have a lower mean elevational value, while Cache la Poudre (area 4) would have the lowest mean elevational value.

3. *Cover_type with wilderness area*

   As for primary major tree species in these areas, Neota would have spruce/fir (type 1), while Rawah and Comanche Peak would probably have lodgepole pine (type 2) as their primary species, followed by spruce/fir and aspen (type 5). Cache la Poudre would tend to have Ponderosa pine (type 3), Douglas-fir (type 6), and cottonwood/willow (type 4).

4. *Preprocessing based on Soil Type*

   Soil Types are numbered from 1 to 40 and are categorised based on the USFS Ecological Landtype Units (ELUs) for this study area as follows:

Table 4.1: USFS Ecological Landtype Units of soil types

Study Code	USFS ELU Code	Description
1	2702	Cathedral family Rock outcrop complex, extremely stony.
2	2703	Vanet - Ratake families complex, very stony.
3	2704	Haploborolis - Rock outcrop complex, rubbly.
4	2705	Ratake family - Rock outcrop complex, rubbly.
5	2706	Vanet family - Rock outcrop complex complex, rubbly.
6	2717	Vanet - Wetmore families - Rock outcrop complex, stony.
7	3501	Gothic family.
8	3502	Supervisor - Limber families complex.
9	4201	Troutville family, very stony.
10	4703	Bullwark - Catamount families - Rock outcrop complex, rubbly.
11	4704	Bullwark - Catamount families - Rock land complex, rubbly.
12	4744	Legault family - Rock land complex, stony.
13	4758	Catamount family - Rock land - Bullwark family complex, rubbly.
14	5101	Pachic Argiborolis - Aquolis complex.
15	5151	unspecified in the USFS Soil and ELU Survey.
16	6101	Cryaquolis - Cryoborolis complex.
17	6102	Gateview family - Cryaquolis complex.
18	6731	Rogert family, very stony.
19	7101	Typic Cryaquolis - Borohemists complex.
20	7102	Typic Cryaquepts - Typic Cryaquolls complex.
21	7103	Typic Cryaquolls - Leighcan family, till substratum complex.
22	7201	Leighcan family, till substratum, extremely bouldery.
23	7202	Leighcan family, till substratum - Typic Cryaquolls complex.
24	7700	Leighcan family, extremely stony.
25	7701	Leighcan family, warm, extremely stony.
26	7702	Granile - Catamount families complex, very stony.
27	7709	Leighcan family, warm - Rock outcrop complex, extremely stony.
28	7710	Leighcan family - Rock outcrop complex, extremely stony.
29	7745	Como - Legault families complex, extremely stony.
30	7746	Como family - Rock land - Legault family complex, extremely stony.
31	7755	Leighcan - Catamount families complex, extremely stony.
32	7756	Catamount family - Rock outcrop - Leighcan family complex, extremely stony.
33	7757	Leighcan - Catamount families - Rock outcrop complex, extremely stony.

Table 4.2: USFS Ecological Landtype Units of soil types

Study Code	USFS ELU Code	Description
34	7790	Cryorthents - Rock land complex, extremely stony.
35	8703	Cryumbrepts - Rock outcrop - Cryaquepts complex.
36	8707	Bross family - Rock land - Cryumbrepts complex, extremely stony.
37	8708	Rock outcrop - Cryumbrepts - Cryorthents complex, extremely stony.
38	8771	Leighcan - Moran families - Cryaquolls complex, extremely stony.
39	8772	Moran family - Cryorthents - Leighcan family complex, extremely stony.
40	8776	Moran family - Cryorthents - Rock land complex, extremely stony.

The first digit of the ELU code refers to climatic zone:

1. lower montane dry
2. lower montane
3. montane dry
4. montane
5. montane dry and montane
6. montane and subalpine
7. subalpine
8. alpine

The second digit of the ELU code refers to geologic zone:

1. alluvium
2. glacial
3. shale
4. sandstone
5. mixed sedimentary
6. unspecified in the USFS ELU Survey
7. igneous and metamorphic
8. volcanic

The third and fourth ELU digits are unique to the mapping unit and have no special meaning to the climatic or geologic zones.

The above categorical attribute Soil Type takes large number of distinct values, with no ordering among the values. We tried to group these values based on the ELU code for all values of cover types. This generated a concept hierarchy for categorical attribute soil type. Making use of the ELU values, we could generate 11 soil types.

The categorical attributes called climatic and geologic were created based on the first and second digits of the ELU code.

4.2 *Feature Engineering*

After a thorough data analysis and understanding the relationship between the features, new features were engineered from the existing ones. This section gives a detailed description of the feature engineering done on the data set[17]

1. C50 with categorical columns for soil type and wilderness area

   The dataset contained 44 binary columns for 40 soil types and 4 wilderness areas. This representation was changed to a Soil_Type attribute which included all distinct 40 soil type values and Wilderness_Area attribute which included all 4 distinct cover types.

2. Conversion of Aspect from Azimuth to 180 degrees. The aspect is in degree azimuthal (360 degrees) and can be shifted to 180 degrees. A new feature called Aspect2 is generated. The python code to carry out this step is as below.

   def r(x):
   if x+180>360:
   return x-180
   else:
   return x+180

   train['Aspect2'] = train.Aspect.map(r)
   test['Aspect2'] = test.Aspect.map(r)

3. Relationship between Vertical Distance to Hydrology and Elevation

   Elevation and Vertical_Distance_To_Hydrology are correlated to each other. A graph of Vertical_Distance_To_Hydrology versus Elevation was plotted using the code below.

   import numpy as np from IPython.display import Image

```
def plotc(c1,c2):
fig = plt.figure(figsize=(16,8))
sel = np.array(list(train.Cover_Type.values))
plt.scatter(c1, c2, c=sel, s=100)
plt.xlabel(c1.name)
plt.ylabel(c2.name)
plotc(train.Elevation, train.Vertical_Distance_To_Hydrology)
```

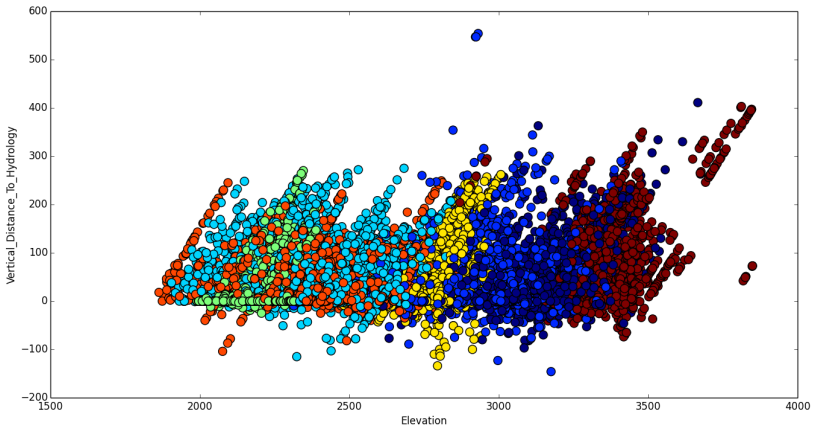


Figure 4.1: Graph of Vertical_Distance_To_Hydrology versus Elevation

In the above graph, coloring each cover type in different colors, seem to reveal a pattern of the plotted points. Hence, we create a new feature called VDTH which gives a simpler relation. Here, VdTH is given by subtracting Vertical_Distance_To_Hydrology from Elevation.

The python code to do this is as below.

```
train['EVDtH'] = train.Elevation-train.Vertical_Distance_To_Hydrology
test['EVDtH'] = test.Elevation-test.Vertical_Distance_To_Hydrology
```

4. Relationship between Elevation and Horizontal_Distace_to_Hydrology

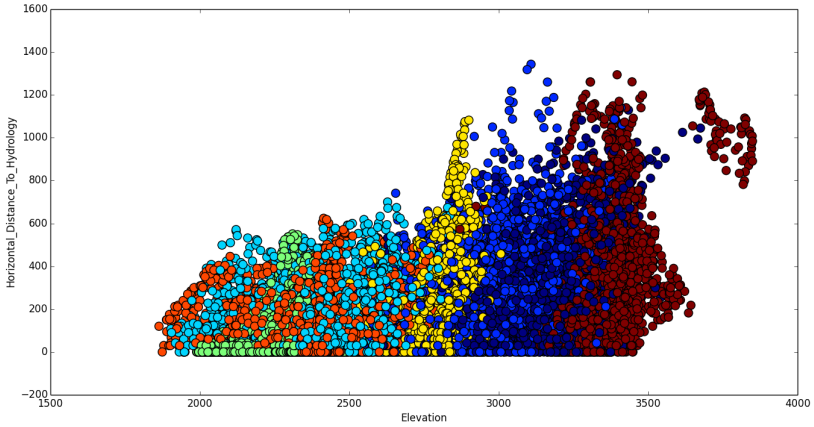


Figure 4.2: Graph of Horizontal_Distace_to_Hydrology versus Elevation

From the above graph, it has been observed that Elevation and Horizontal_Distance_to_Hydrology have a relationship between them, defined as a new attribute called EHDtH.

Where, EHDtH = Elevation - Horizontal_Distace_to_Hydrology*0.2

The python code to do this is as below.
train['EHDtH'] = train.Elevation-train.Horizontal_Distance_To_Hydrology*0.2
test['EHDtH'] = test.Elevation-test.Horizontal_Distance_To_Hydrology*0.2

5. Negative values of Vertical_Distance_To_Hydrology

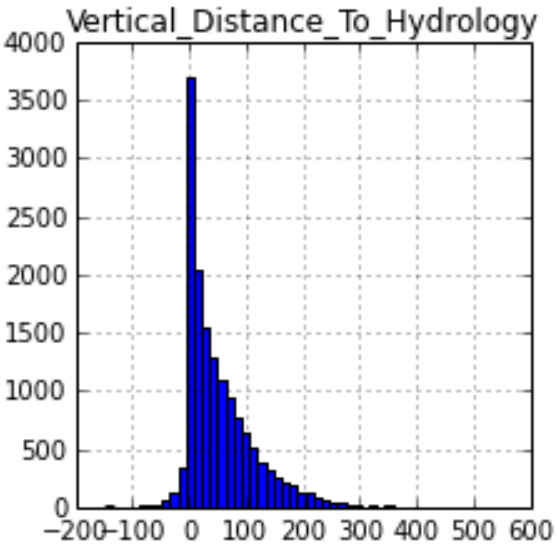


Figure 4.3: Plot of Vertical_Distance_To_Hydrology

Looking at the distribution of Vertical_Distance_To_Hydrology, we found that it has some negative values. We created another variable called 'Highwater', which indicates whether

this attribute has a positive or a negative value.

The python code to create the new feature Highwater is as below:
train['Highwater'] = train.Vertical_Distance_To_Hydrology ¡ 0
test['Highwater'] = test.Vertical_Distance_To_Hydrology ¡ 0

6. Missing values of Hillshade_3pm

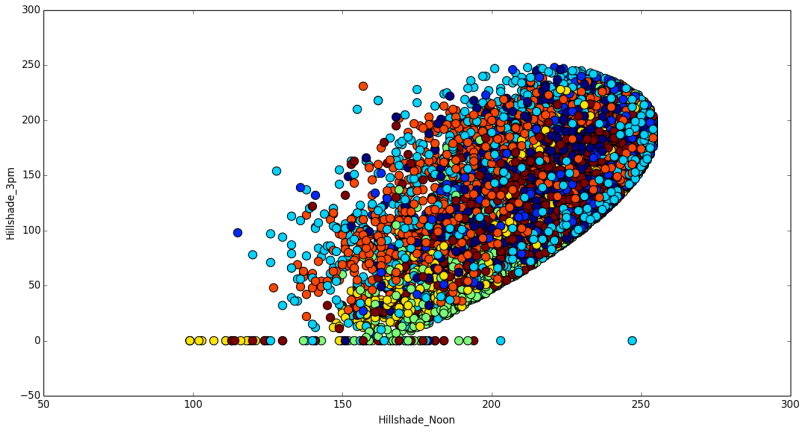


Figure 4.4: Graph of Hillshade_3pm versus Hillshade_Noon

When Hillshade_3pm was plotted against Hillshade_Noon, it was found to have some missing values as shown in the above graph. The missing values were filled with the mean of the values of Hillshade_3pm.

7. Other new features

A few features that were found to be important was given as input in building the predictive model by increasing the feature space of these attributes. The new features were created as follows.

train['Distanse_to_Hydrolody'] = (train['Horizontal_Distance_To_Hydrology'] ** 2 +
train['Vertical_Distance_To_Hydrology'] ** 2) ** 0.5
test['Distanse_to_Hydrolody'] = (test['Horizontal_Distance_To_Hydrology'] ** 2 +
test['Vertical_Distance_To_Hydrology'] ** 2) ** 0.5

train['Hydro_Fire_1'] = train['Horizontal_Distance_To_Hydrology'] +
train['Horizontal_Distance_To_Fire_Points']
test['Hydro_Fire_1'] = test['Horizontal_Distance_To_Hydrology'] +
test['Horizontal_Distance_To_Fire_Points']

train['Hydro_Fire_2'] = abs(train['Horizontal_Distance_To_Hydrology'] -
train['Horizontal_Distance_To_Fire_Points'])
test['Hydro_Fire_2'] = abs(test['Horizontal_Distance_To_Hydrology'] -
test['Horizontal_Distance_To_Fire_Points'])

train['Hydro_Road_1'] = abs(train['Horizontal_Distance_To_Hydrology'] +
train['Horizontal_Distance_To_Roadways'])
test['Hydro_Road_1'] = abs(test['Horizontal_Distance_To_Hydrology'] +
test['Horizontal_Distance_To_Roadways'])

train['Hydro_Road_2'] = abs(train['Horizontal_Distance_To_Hydrology'] -
train['Horizontal_Distance_To_Roadways'])
test['Hydro_Road_2'] = abs(test['Horizontal_Distance_To_Hydrology'] -
test['Horizontal_Distance_To_Roadways'])

train['Fire_Road_1'] = abs(train['Horizontal_Distance_To_Fire_Points'] +
train['Horizontal_Distance_To_Roadways'])
test['Fire_Road_1'] = abs(test['Horizontal_Distance_To_Fire_Points'] +
test['Horizontal_Distance_To_Roadways'])

train['Fire_Road_2'] = abs(train['Horizontal_Distance_To_Fire_Points'] -
train['Horizontal_Distance_To_Roadways'])
test['Fire_Road_2'] = abs(test['Horizontal_Distance_To_Fire_Points'] -
test['Horizontal_Distance_To_Roadways'])

8. Missing Soil_Type

   Soil_Type_7 and Soil_Type_15 have their value to be 0 in all the tuples of the training set. Hence, we removed the tuples which had Soil_Type_7 and Soil_Type_15 value to be 1 in the test set.

4.3 *Feature selection*

Feature selection aims to choose a small subset of the relevant features from the original ones according to certain relevance evaluation criterion, which usually leads to better learning performance (e.g., higher learning accuracy for classification), lower computational cost, and better model interpretability[18]

Gain Ratio is a measure of information that is provided by an attribute. Decision tree uses Gain Ratio as attribute selection measure for tree construction. Hence, the most relevant attributes are taken for tree construction. This eliminates the need for using any other feature selection technique.
Also, upon plotting graphs we observed that the attributes are independent of each other.

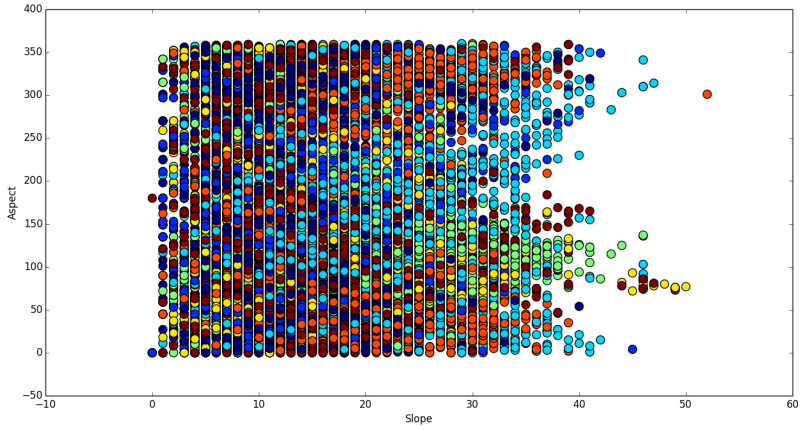


Figure 4.5: Graph of Aspect versus Slope

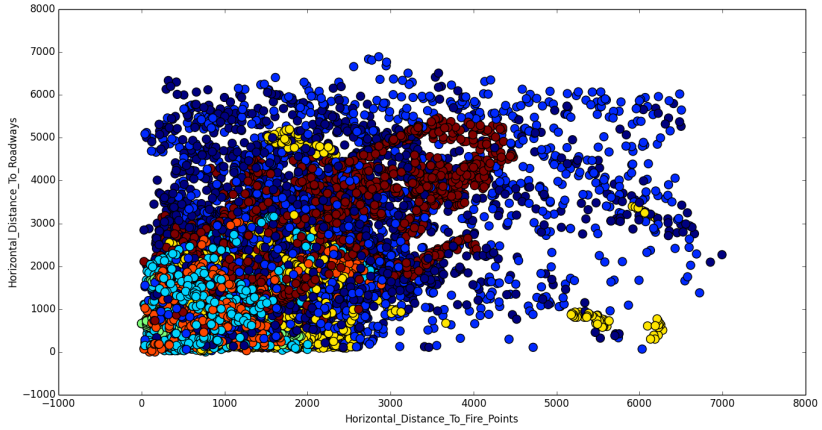


Figure 4.6: Horizontal_Distance_To_Roadways vs Horizontal_Distance_To_Fire_Points

From Figure 4.5 and 4.6, the pair of attributes Aspect, Slope and Horizontal_Distance_To_Hydrology, Horizontal_Distance_To_Fire_Points are found to be independent of each other and hence are included in the dataset.

4.4 *Pruning Boosted Classifier(MC5.0)*

The ‘C50’ package in R builds C5.0 decision trees and rule-based models for classification. The model can take the form of a full decision tree or a collection of rules or boosted versions

of either. The framework of 'C50' package with highlighted boosting section is as shown in Figure4.7:

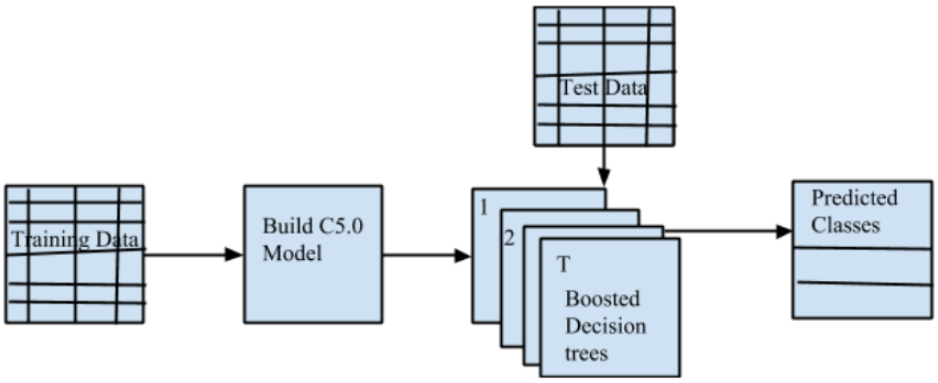


Figure 4.7: C5.0 Package Framework

In the model building stage, C50 takes a parameter trials(T) to specify the number of boosting iterations. A value of one indicates that a single model is used. Hence, T number of boosted decision trees make up the ensemble used for predicting classes.

The kappa pruning algorithm has been implemented as follows. Kappa K on the training set, for every pair of classifiers produced by adaboost. Once all kappas are caluculated, choose pairs of classifiers starting with the pair that has the lowest K and considering them in increasing order of K until M classifiers are obtained.

The source code of C50 package in R was modified to prune the boosted classifiers. Kappa pruning algorithm was applied on T boosted classifiers to obtain M pruned boosted classifiers $(M <= T)$. The modified framework of the C50 model is shown as in Figure4.8:

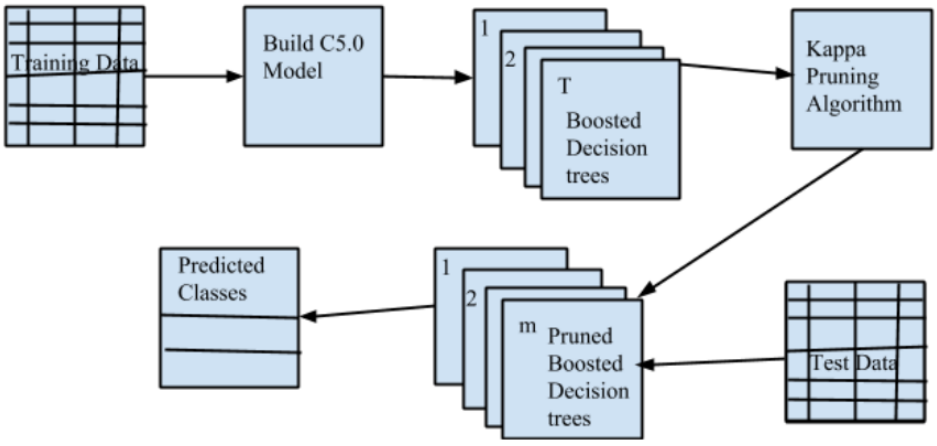


Figure 4.8: MC5.0 Package Framework

5 Results and Analysis

This section shall give a detailed description and analysis of the results obtained from the work done from the various experiments conducted as stated in the work done.

5.1 *Decision Trees*

The decision trees was selected has the classifier for our forest cover type prediction from the literature survey. Experimental study for the selection of the right decision tree was conducted using the existing packages in R.

Table 5.1 summarizes the comparison of decision trees under the selected metrics. The classification metrics taken in to consideration include:

- Accuracy: Gives a measure of the number of the samples are correctly predicted.
- Tree size: Indicates the number of nodes that appear in the constructed decision tree.
- AUC: Area under Curve for Receiver Operating Characteristic

Table 5.1: Performance Evaluation

Performance Metrics/Decision trees	C4.5	C5.0	CART	Random Forest
Accuracy in %	80.78	91.11	79.45	83.58
Size of the Tree(nodes)	2111	772	951	NA*
AUC	0.92	0.88	0.94	0.97

NA* refers to the condition that Random Forest follows multiple iteration with k random attributes each time to create decision tree. So, the nodes at every iteration is less when compared to other trees but has a overhead of multiple decision tree creation.

Among the decision trees, C5.0 was found to give higher accuracy. The reason for improved accuracy with C5.0 is because of boosting the predictive model. Boosting algorithm sets weight for each sample, which presents its importance. This varies iteration, based on the error rate of previous iteration, due to which the prediction ability is improved.

The size of the tree is found to be very high in C4.5 and the least in case of C5.0. As already mentioned the important additions to C5.0 were the in-built pruning that helps to reduce

the tree size and memory usage. This becomes very productive for the simple applications to use the C5.0 classifier.

Based on all the measures mentioned above, important point is to note that techniques like pruning, boosting and bagging improve tree performance and reduce the memory requirements for building the model.

5.2 *Feature engineering*

As stated in the previous sections, our model needed pre-processing and feature extraction. Table 5.2 gives the modifications made to the features and the improvements to the model when compared to the base case.

Base case: Accuracy of the Model when tested on Test set with no changes made to the Data set. This case gave an accuracy of 68.44%

Table 5.2: Data Pre-Processing and Feature Extraction

Expt No.	Feature Description	Feature Modifications	Reason	Results
1	Missing Soil Types7 and Soil Types15 in the training set	Removal of s7 and s15 from test set (There were 105 samples with soil type s7).	The relevance of these features seemed to be very low according to the attribute usage obtained from C50 tree built from test set.	Improved to 69.13%
2	Soil Type (Qualitative) with 40 binary columns	Generalization of Soil Type to 11 Columns	Generalization is based on the ELU codes of soil types	Decreased to 67.32%
3	Soil Type (Qualitative) with 40 binary columns	Generalization based on geologic and climatic zones	Generalization is based on the ELU codes of soil types	Decreased further to 66.95%
4	Wilderness Area (Qualitative) with 4 binary columns	Generalization to one categorical feature	To lower this features used due to its lower relevance.	Improvement to 68.89%
5	Missing values in Hillshade_3pm	Replaced those values with the mean of all Hillshade_3pm	To remove the outliers	Improvement to 69.17%
6	Soil type (Qualitative) with 40 binary columns	Generalized to one categorical feature	To lower the features used due to its lower relevance	Improved to 69.13%

The experiments performed above showed only very slight improvements over the primary data-set. Based on the accuracy changes, the features were selected for the new dataset.

The detailed data analysis with decision trees has showed the need for enhancing the feature space with more relevant features. So, the further experimentations were performed in building new features from the given feature set. Table 5.3 shows the lists of new features obtained from the feature engineering:

Table 5.3: Description of the new additional features for Forest Cover Dataset

Attribute Name	Data type	Description
Wilderness Area	Qualitative	Wilderness area designation
Soil type	Qualitative	Soil type designation
Aspect2	Qualitative	Aspect in degrees azimuth
Highwater	Qualitative	Indicative for positive or negative values to Vertical_Distance_To_Hydrology
EVDtH	Qualitative	Elevation - Vertical_Distance_To_Hydrology
EHDtH	Qualitative	Elevation - Horizontal_Distance_To_Hydrology*0.2
Distance_To_Hydrology	Qualitative	(Horizontal_Distance_To_Hydrology^2 + Vertical_Distance_To_Hydrology^2)$^{1/2}$
Hydro_Fire_1	Qualitative	Horizontal_Distance_To_Hydrology + Horizontal_Distance_To_Fire_Points
Hydro_Fire_2	Qualitative	Horizontal_Distance_To_Hydrology - Horizontal_Distance_To_Fire_Points
Hydro_Road_1	Qualitative	Horizontal_Distance_To_Hydrology + Horizontal_Distance_To_Roadways
Hydro_Road_2	Qualitative	Horizontal_Distance_To_Hydrology - Horizontal_Distance_To_Roadways
Fire_Road_1	Qualitative	Horizontal_Distance_To_Fire_Points + Horizontal_Distance_To_Roadways
Fire_Road_2	Qualitative	Horizontal_Distance_To_Fire_Points- Horizontal_Distance_To_Roadways

The new feature set when tested on the C5.0 decision tree with no boosting gave an improvement in accuracy to 69.22%.

5.3 *Feature selection*

Feature Selection is a technique to select the best subset of features from the given set in order to maximize the accuracy. The decision trees are known to have the build in feature selection based on the splitting parameter. These are in fact known as embedded feature selectors.

In order to prove the non-requirement of the feature selection, a correlation based attribute selection was performed. Correlation based attribute selection evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. Subsets of features that are highly correlated with the class while having low inter-correlation are preferred.

Table 5.4: Comparison of Decision trees with feature Selection

Classification Technique	No. of. Features	Accuracy
Random Forest	54	81.5%
Random Forest	27	82.552%
Random Forest	13	81.65%

The results of the test conducted with the original data-set with cross-validation can be seen in Table 5.4. The results clearly stated that there can only be memory reduction with the feature selection and no more accuracy improvement when used along with a decision tree classifier.

5.4 *Pruning*

Pruning is a way of reducing the size of the decision tree. This will reduce the accuracy on the training data, but (in general) increase the accuracy on unseen data. It is used to mitigate over fitting, where you would achieve perfect accuracy on training data, but the model (i.e. the decision tree) you learn is so specific that it doesn't apply to anything but that training data.
In our case, varying the confidence parameter from 0.25 to 0.15 for the C5.0 Decision tree is found to improve the accuracy to 69.25%

5.5 *Ensemble Learning*

The new features obtained from the previous steps were found to contain the most relevant attributes. But the improvement was only by 1%. Ensemble learning is the technique to enhance the learning of weak base learners. In this direction, the Random forest and C5.0 was used for further testing. Random forest gave an improvement in accuracy to 77.24% (81.54648% in Scikit) which was found to be better than simple decision tree. While C5.0 with boosting parameter set to 10 gave an improved accuracy to 76.02%.

The above boosting and bagging based ensemble techniques showed to increase the prediction rate. The detailed study of these techniques in R may help in improving the over-all model

for better prediction.

5.6 *MC5.0*

The Modified C5.0 (MC5.0) was tested on ten data sets. All these data sets were taken from the UCI Repository. The AdaBoost of MC5.0 was run on each data set to generate 100 classifiers. Then the Kappa pruning technique was evaluated by generating a set of 20, 40, 60, 80 and 100 classifiers. This corresponds to 80%, 60%, 40%, 20% and 0% pruning. The MC5.0 was also run on each data set in the figures and the resulting performance was plotted which was evaluated by 10-fold cross-validation. The plot corresponds to 100% pruning.
In the plotted overall performance figures, the Gain is defined as the difference in percentage points between the performance of full boosted MC5.0 and the performance of MC5.0 alone. The Gain was always positive for all of our ten domains. The relative performance of the method is defined as the difference between its performance and MC5.0 divided by the Gain. Hence, a relative performance of 1.0 indicates that the alternative method obtains the same gain as AdaBoost. A relative performance of 0.0 indicates that the alternative method obtains the same performance as MC5.0 alone.

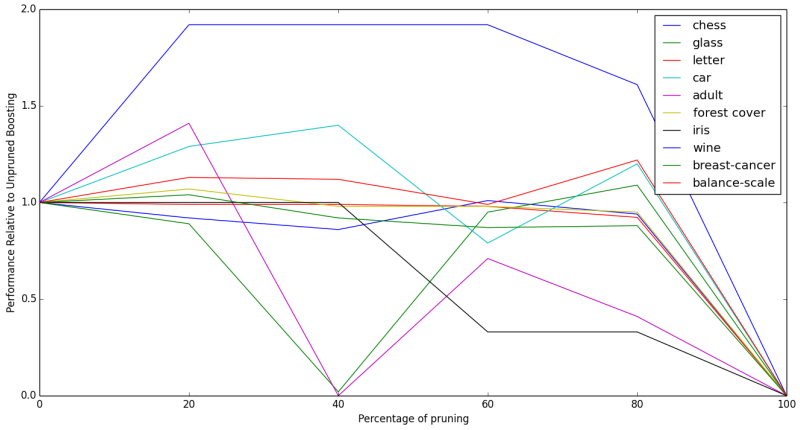


Figure 5.1: Relative performance of MC5.0 with various amounts of pruning

FigureB.2 shows the performance of Kappa Pruning on ten data sets. Pruning improves performance over AdaBoost for Chess, Adult, Car, Letter, Wine and Forest-Cover. The only data sets that show very bad behaviour are Adult and Breast-Cancer , which appears to be very unstable. Hence, in many cases, significant pruning does not hurt performance very much.

6 Conclusions and Future Work

The purpose of this project is to use the decision tree classification algorithms for predicting the forest cover type. The forest cover data of the Roosevelt National Forest of northern Colorado was used to evaluate the performance of various Decision Tree algorithms. Among the decision trees, C5.0 was found to give higher accuracy. Various feature engineering techniques performed on the dataset showed improvement over the primary data-set. The new feature set when tested with C5.0 decision tree with no boosting gave an improvement in accuracy to 69.22%. Random forest and C5.0 gave an improvement in accuracy to 77.24% and 76.02% respectively. These show that ensemble techniques can enhance the performance of decision trees considerably.
The ensemble learning approach constructs a composite hypothesis by rating individual hypothesis. The memory required to store these hypothesis is high and can be reduced by selecting a set of hypothesis which gives nearly the same performance.The selection of the hypothesis was implemented in the package using kappa pruning.

There are many more pruning techniques available for boosted classifiers, which seem to work under different conditions. So, an experimentation with other pruning techniques like back-fitting along with kappa pruning can be performed.

7 Project Plan

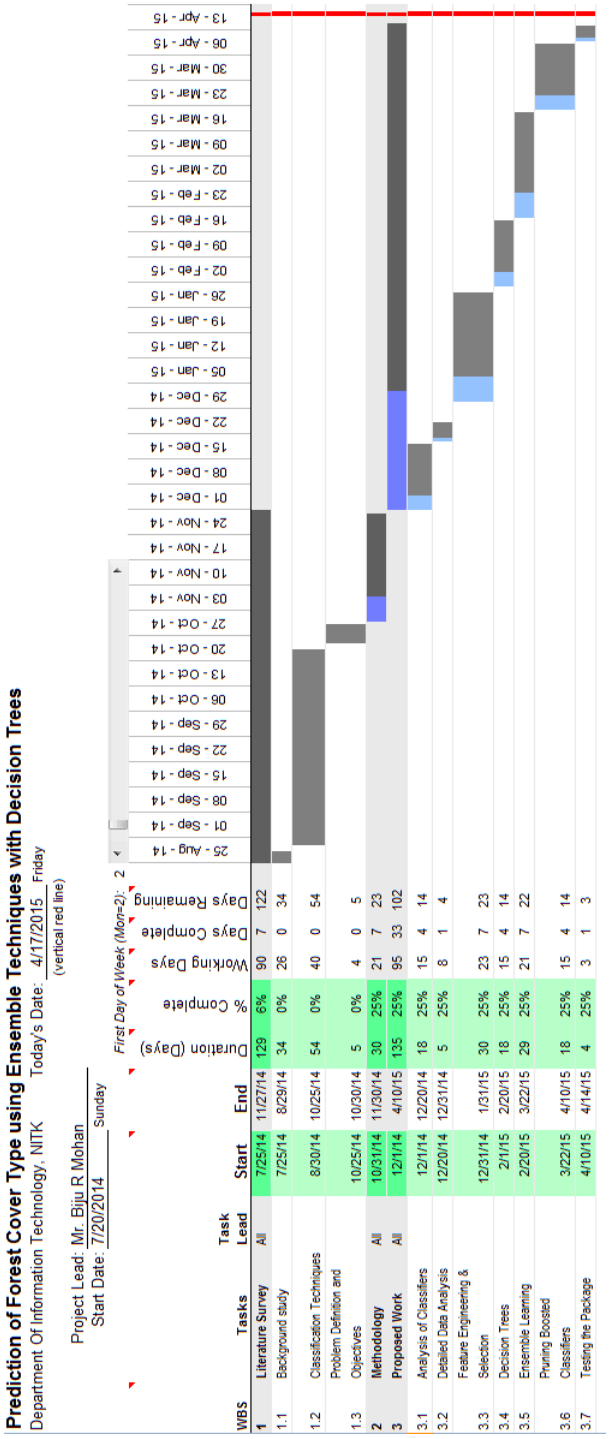


Figure 7.1: Gantt Chart

References

[1] J. A. Blackard and D. J. Dean, "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables," *Computers and electronics in agriculture*, vol. 24, no. 3, pp. 131–151, 1999.

[2] B. Chandra and V. Pallath Paul, "Prediction of forest cover using decision trees," *J. Ind. Soc. Agril. Statist*, vol. 61, no. 2, pp. 192–198, 2007.

[3] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proceedings of the 23rd international conference on Machine learning*, pp. 161–168, ACM, 2006.

[4] R. Entezari-Maleki, A. Rezaei, and B. Minaei-Bidgoli, "Comparison of classification methods based on the type of attributes and sample size," *Journal of Convergence Information Technology*, vol. 4, no. 3, 2009.

[5] J. R. Quinlan, "Bagging, boosting, and c4. 5," in *AAAI/IAAI, Vol. 1*, pp. 725–730, 1996.

[6] R. E. Schapire, "A brief introduction to boosting," in *Ijcai*, vol. 99, pp. 1401–1406, 1999.

[7] J. Zhu, H. Zou, S. Rosset, and T. Hastie, "Multi-class adaboost," *Statistics and Its*, 2009.

[8] Y. Freund, R. E. Schapire, *et al.*, "Experiments with a new boosting algorithm," in *ICML*, vol. 96, pp. 148–156, 1996.

[9] D. D. Margineantu and T. G. Dietterich, "Pruning adaptive boosting," in *ICML*, vol. 97, pp. 211–218, Citeseer, 1997.

[10] J. Tang, S. Alelyani, and H. Liu, "Feature selection for classification: A review," *Data Classification: Algorithms and Applications. Editor: Charu Aggarwal, CRC Press In Chapman & Hall/CRC Data Mining and Knowledge Discovery Series*, 2014.

[11] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[12] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[13] J. R. Quinlan, *C4. 5: programs for machine learning*, vol. 1. Morgan kaufmann, 1993.

[14] S.-l. Pang and J.-z. Gong, "C5.0 Classification Algorithm and Application on Individual Credit Evaluation of Banks," *Systems Engineering - Theory & Practice*, vol. 29, pp. 94–104, Dec. 2009.

[15] A. S. Galathiya, A. P. Ganatra, and C. K. Bhensdadia, “Improved Decision Tree Induction Algorithm with Feature Selection , Cross Validation , Model Complexity and Reduced Error Pruning,” vol. 3, no. 2, pp. 3427–3431, 2012.

[16] UCI, “Uci data,” 15-01-2015.

[17] Kaggle.com, “Description - forest cover type prediction forums — kaggle,” 13-02-2015.

[18] I. Guyon and A. Elisseeff, “An introduction to variable and feature selection,” *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

A Appendix I

1. Feature Engineering:

```
 def plotc(c1,c2):

    fig = plt.figure(figsize=(16,8))
    sel = np.array(list(train.Cover_Type.values))

    plt.scatter(c1, c2, c=sel, s=100)
    plt.xlabel(c1.name)
    plt.ylabel(c2.name)

    plotc(train.Elevation,0.2*train.
       Horizontal_Distance_To_Hydrology)
    plt.show()

def r(x):
    if x+180>360:
        return x-180
    else:
        return x+180

train['Aspect2'] = train.Aspect.map(r)
test['Aspect2'] = test.Aspect.map(r)

train['Highwater'] = train.Vertical_Distance_To_Hydrology
   < 0
test['Highwater'] = test.Vertical_Distance_To_Hydrology <
   0

train['EVDtH'] = train.Elevation-train.
   Vertical_Distance_To_Hydrology
test['EVDtH'] = test.Elevation-test.
   Vertical_Distance_To_Hydrology

train['EHDtH'] = train.Elevation-train.
   Horizontal_Distance_To_Hydrology*0.2
```

```
test['EHDtH'] = test.Elevation-test.
    Horizontal_Distance_To_Hydrology*0.2

train['Distanse_to_Hydrolody']=(train['
    Horizontal_Distance_To_Hydrology']**2+train['
    Vertical_Distance_To_Hydrology']**2)**0.5
test['Distanse_to_Hydrolody']=(test['
    Horizontal_Distance_To_Hydrology']**2+test['
    Vertical_Distance_To_Hydrology']**2)**0.5

train['Hydro_Fire_1']=train['
    Horizontal_Distance_To_Hydrology']+train['
    Horizontal_Distance_To_Fire_Points']
test['Hydro_Fire_1']=test['
    Horizontal_Distance_To_Hydrology']+test['
    Horizontal_Distance_To_Fire_Points']

train['Hydro_Fire_2']=abs(train['
    Horizontal_Distance_To_Hydrology']-train['
    Horizontal_Distance_To_Fire_Points'])
test['Hydro_Fire_2']=abs(test['
    Horizontal_Distance_To_Hydrology']-test['
    Horizontal_Distance_To_Fire_Points'])

train['Hydro_Road_1']=abs(train['
    Horizontal_Distance_To_Hydrology']+train['
    Horizontal_Distance_To_Roadways'])
test['Hydro_Road_1']=abs(test['
    Horizontal_Distance_To_Hydrology']+test['
    Horizontal_Distance_To_Roadways'])

train['Hydro_Road_2']=abs(train['
    Horizontal_Distance_To_Hydrology']-train['
    Horizontal_Distance_To_Roadways'])
test['Hydro_Road_2']=abs(test['
    Horizontal_Distance_To_Hydrology']-test['
    Horizontal_Distance_To_Roadways'])
```

```
train['Fire_Road_1']=abs(train['
    Horizontal_Distance_To_Fire_Points']+train['
    Horizontal_Distance_To_Roadways'])
test['Fire_Road_1']=abs(test['
    Horizontal_Distance_To_Fire_Points']+test['
    Horizontal_Distance_To_Roadways'])

train['Fire_Road_2']=abs(train['
    Horizontal_Distance_To_Fire_Points']-train['
    Horizontal_Distance_To_Roadways'])
test['Fire_Road_2']=abs(test['
    Horizontal_Distance_To_Fire_Points']-test['
    Horizontal_Distance_To_Roadways'])

feature_cols = [col for col in train.columns if col not in
    ['Cover_Type','Id']]

X_train = train[feature_cols]
X_test = test[feature_cols]
y = train['Cover_Type']
test_ids = test['Id']
```

2. Python code for Random Forest :

```
  forest = ensemble.ExtraTreesClassifier(n_estimators=400,
      criterion='gini', max_depth=None,
    min_samples_split=2, min_samples_leaf=1, max_features=
      'auto',
    bootstrap=False, oob_score=False, n_jobs=-1,
      random_state=None, verbose=0,
    min_density=None)

forest.fit(X_train, y)

with open('features_engineering_benchmark.csv', "wb") as
    outfile:
    outfile.write("Id,Cover_Type\n")
    for e, val in enumerate(list(forest.predict(X_test))):
        outfile.write("%s,%s\n"%(test_ids[e],val))
```

```
print pd.DataFrame(forest.feature_importances_,index=
    X_train.columns).sort([0], ascending=False) [:10]
```

3. R code for C5.0 and Random Forest :

```
 library(C50)
library(caret)
train = read.csv("train_sw_categorical.csv")
test = read.csv("test_sw_categorical.csv")

train$Wilderness_Area1 = as.factor(train$Wilderness_Area1)
    #Wilderness area - nominal
train$Soil_Type1 = as.factor(train$Soil_Type1) #Soil type
   - nominal
train$Highwater= as.factor(train$Highwater)

test$Wilderness_Area1 = as.factor(test$Wilderness_Area1)
test$Soil_Type1 = as.factor(test$Soil_Type1)
test$Highwater= as.factor(test$Highwater)

c5tree = C5.0(train[,2:23],factor(train$Cover_Type),trials
    = 1, control = C5.0Control(CF = 0.15,winnow = TRUE)) #
   sw categorical
predicted = predict(c5tree,test[,2:23])

confusionMatrix(factor(test$Cover_Type),predicted)
```

B Appendix II

MC5.0 Package

1. Requirements:
   There are two main prerequisites for building or modifying R packages:

   - GNU software development tools including a C/C++ compiler; and
   - LaTeX for building R manuals.

2. Adding Package:
   To enable RStudio's package development tools for an existing package, create a new RStudio Project associated with the package's (existing) directory.

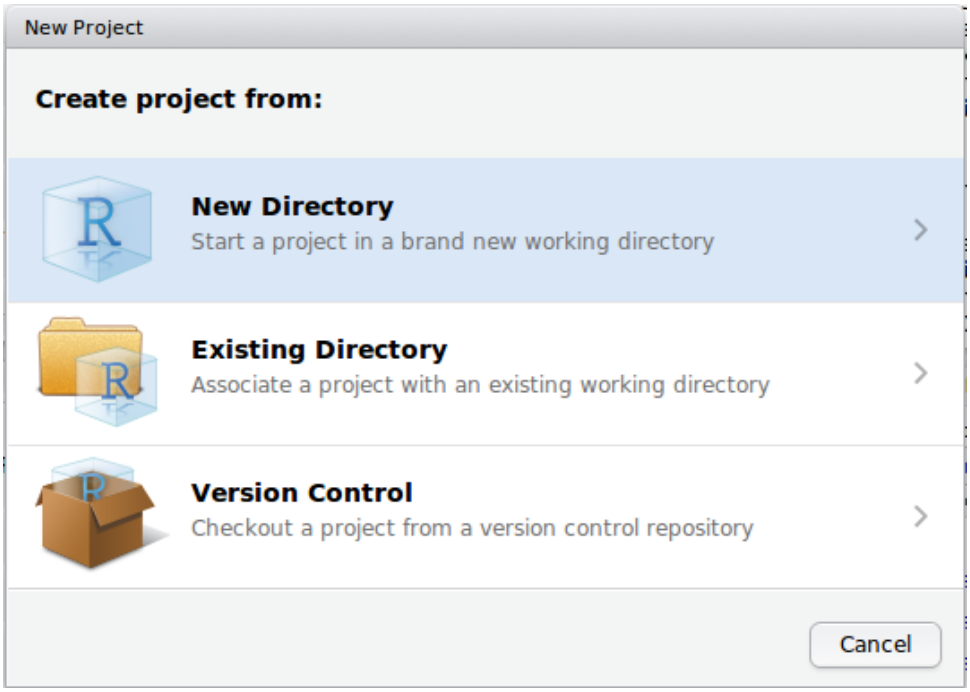


Figure B.1: Creating Package

3. Building Package:
   The Build pane includes a number of tools for building and testing packages. When modifying or developing a package in RStudio, Build and Reload command can be used to re-build the package and reload.

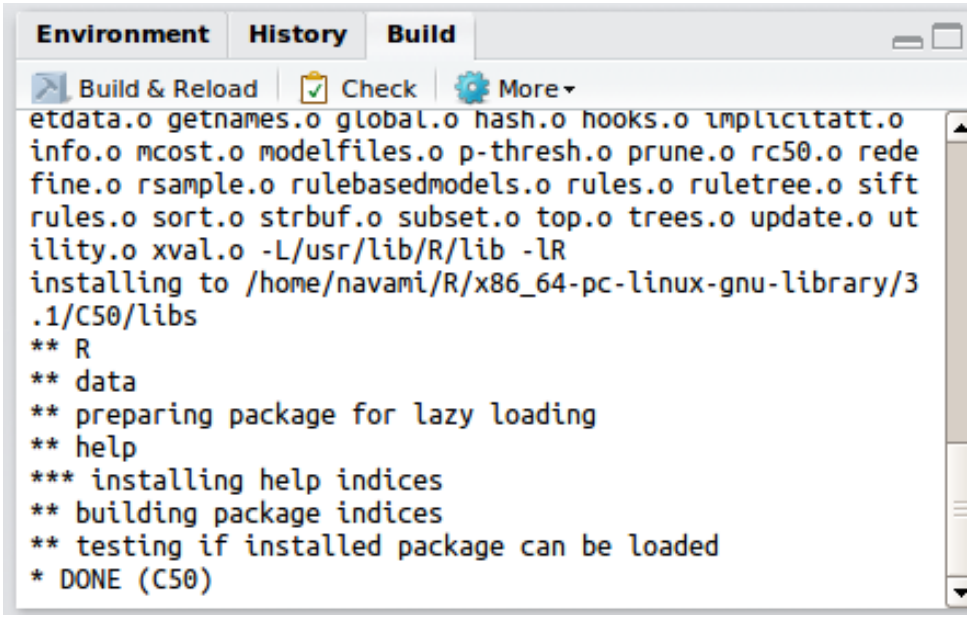


Figure B.2: Buiding and Reloading Package

The Build and Reload command performs several steps in sequence to ensure a clean and correct result:

- Unloads any existing version of the package (including shared libraries if necessary).
- Builds and installs the package using R CMD INSTALL.
- Restarts the underlying R session to ensure a clean environment for re-loading the package.
- Reloads the package in the new R session by executing the library function.

4. Kappa Functions added to construct.c in C5.0:

```
 int kappanotexists(int kappadata,int m)
{
        int i;
        ForEach(i,0,m-1)
        {
                if(pruned_classifier_no[i]==kappadata){
                        return 0;
                }
        }
        return 1;
}
float FindKappa(int ha,int hb)
{
        int tupleNo,ClassNo,i,j;
        int C[MaxClass+1][MaxClass+1];
        float k,SigmaOne,SigmaTwo,ThetaTwo=0.0,ThetaOne
           =0.0;
        ForEach(i,1,MaxClass)
                ForEach(j,1,MaxClass)
                        C[i][j] = 0;

        ForEach(tupleNo,0,MaxCase)
        {
                 i = ( RULES ? RuleClassify(Case[tupleNo],
                    RuleSet[ha]) :
                              TreeClassify(Case[tupleNo],
                                 Pruned[ha]) );
                 j = ( RULES ? RuleClassify(Case[tupleNo],
                    RuleSet[hb]) :
```

```
                                TreeClassify(Case[tupleNo],
                                    Pruned[hb]) );
                C[i][j]+=1;
        }

        ForEach(ClassNo,0,MaxClass-1)
        {
                ThetaOne+=C[ClassNo][ClassNo];
        }

        ThetaOne/=MaxCase;

        ForEach(i,0,MaxClass-1)
        {
                SigmaOne = 0.0;
                SigmaTwo = 0.0;

                ForEach(j,0,MaxClass-1)
                {
                        SigmaOne+=((float)C[i][j]/(MaxCase
                            +1));
                        SigmaTwo+=((float)C[j][i]/(MaxCase
                            +1));
                }
                ThetaTwo+=(SigmaOne*SigmaTwo);
        }

        k = (ThetaOne - ThetaTwo)/(1-ThetaTwo);
        return k;
}


float Prune_BoostClassifiers()
{
        struct KappaData{
        float kappa;
        int ha;
        int hb;
        };
```

```
struct KappaData* Kdata;
int Maxkappa = TRIALS*(TRIALS-1)/2;
pruned_classifier_no = (int*)malloc(Maxkappa*
   sizeof(int));
Kdata = (struct KappaData*)malloc(Maxkappa*sizeof(
   struct KappaData));
float temp;int count = 0;
int temp1,trial1,trial2,k=0;
ForEach(trial1,0,TRIALS-1)
{
   ForEach(trial2,trial1+1,TRIALS-1)
          {     count++;
                Kdata[k].kappa = FindKappa(trial1,
                   trial2);
                Kdata[k].ha = trial1;
                Kdata[k].hb = trial2;
                k++;
         }
}

for(int i=0;i<Maxkappa;i++)
{
         for(int j=i+1;j<Maxkappa;j++)
         {
                if(Kdata[i].kappa>Kdata[j].kappa)
                {
                        temp=Kdata[i].kappa;
                        Kdata[i].kappa=Kdata[j].
                           kappa;
                        Kdata[j].kappa=temp;
                        temp1=Kdata[i].ha;
                        Kdata[i].ha=Kdata[j].ha;
                        Kdata[j].ha=temp1;
                        temp1=Kdata[i].hb;
                        Kdata[i].hb=Kdata[j].hb;
                        Kdata[j].hb=temp1;
                }
         }
}
```

```
        for(int maxkappa=0,m=0;m<M;maxkappa++)
        {
                if(kappanotexists(Kdata[maxkappa].ha,m))
                        pruned_classifier_no[m++] = Kdata[
                            maxkappa].ha;

                if(kappanotexists(Kdata[maxkappa].hb,m))
                        pruned_classifier_no[m++] = Kdata[
                            maxkappa].hb;
        }
}
```

5. Adding a new parameter to the package:
   A parameter called 'prunem' which is specified by the user is used in the function call. 'prunem' should always be less than or equal to 'TRAILS'. The prediction is done using the pruned boosted classifiers, which are 'prunem' in number.

6. Distributing R Package:
   R packages can be distributed in a variety of ways:

   - Source packages (requires that users build the project from source on their workstation)
   - Binary packages (can be installed as-is but require building binaries for both Mac and Windows)
   - Contributing packages to CRAN
   - Publishing on R-Forge or GitHub

Turnitin Originality Report

MJP Report_One more copy by Navami K

From Forest Cover (B.Tech Major Project)

Processed on 01-May-2015 08:05 IST
ID: 536208273
Word Count: 12169

Similarity Index	Similarity by Source	
29%	Internet Sources:	23%
	Publications:	17%
	Student Papers:	16%

sources:

1 5% match (Internet from 27-Nov-2012)
http://archive.ics.uci.edu/ml/machine-learning-databases/covtype/covtype.info

2 2% match (publications)
Blackard, J.A.. "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables", Computers and Electronics in Agriculture, 199912

3 1% match (Internet from 15-May-2014)
http://www.docstoc.com/docs/10961467/Data-Mining

4 1% match (Internet from 19-Aug-2010)
http://homepages.cae.wisc.edu/~ece539/project/f01/leske.doc

5 1% match (student papers from 07-Oct-2014)
Submitted to Deakin University on 2014-10-07

6 1% match (Internet from 12-Dec-2014)
http://openml.org/d/180

7 1% match (student papers from 09-Mar-2015)
Submitted to Tampereen teknillinen yliopisto on 2015-03-09

8 1% match (Internet from 04-Jul-2009)
http://www.icml2006.org/icml_documents/camera-ready/021_An_Empirical_Compari.pdf

9 1% match ()
http://www2.boosting.org/boosting/papers/FreSch96.pdf

10 1% match (Internet from 22-Apr-2015)
http://en.wikipedia.org/wiki/Random_forest

11 1% match (publications)
Izenman. "Recursive Partitioning and Tree-Based Methods", Springer Texts in Statistics, 2008

12 1% match (Internet from 09-Feb-2015)
http://stackoverflow.com/questions/10865372/why-does-the-c4-5-algorithm-use-pruning-in-order-to-reduce-the-decision-tree-and

13 1% match (Internet from 06-Mar-2015)
http://en.wikipedia.org/wiki/Feature_selection

14 1% match (Internet from 20-Mar-2010)
http://web.engr.oregonstate.edu/~tgd/publications/mlj-randomized-c4.pdf

15 < 1% match (publications)
Srinivasa Prasad, Kalli and Ramakrishna, Seelam. "An Efficient Traffic Forecasting System Based on Spatial Data and Decision Trees", International Arab Journal of Information Technology (IAJIT), 2014.

16 < 1% match (Internet from 05-Nov-2014)
http://www.kaggle.com/c/forest-cover-type-prediction/data

17 < 1% match (Internet from 04-Oct-2009)
http://www.cs.waikato.ac.nz/~abifet/MOA/StreamMining.pdf

< 1% match (Internet from 04-Aug-2014)

Feature Engineering on Forest Cover Type Data with Ensemble of Decision Trees

Pruthvi H.R*, Nisha K.K†, Chandana T.L‡, Navami K§ and Biju R M¶
¶Faculty, Department Of Information Technology
National Institute of Technology, Karnataka, India,
* phr.11it64@nitk.edu.in,†nishakk94@gmail.com,‡tlchandana@gmail.com,§knavami@gmail.com,¶bijurmohan@gmail.com

***Abstract*—The paper aims to determine the forest cover type of the dataset containing strictly cartographic variables. The study evaluated four wilderness areas in the Roosevelt National Forest, located in northern Colorado. The source of cover type data is US Forest service inventory while the cartographic variables like elevation, slope, soil type and other data were derived from Geographic Information System(GIS). Dataset was analysed and feature engineering techniques were applied, which helped in getting more relevant features. Comparative study of various decision tree algorithms such as C4.5, C5.0, CART was conducted on the dataset.With the new dataset built from feature engineering, Random Forest and C5.0 improved the accuracy by 9% compared to the raw dataset.**

I. INTRODUCTION

The area under the study consists of the Rawah (73 213 acres), Comanche Peak (67 680 acres), Neota (9647 acres), and Cache la Poudre (9433 acres) wilderness areas of the Roosevelt National Forest in northern Colorado. As shown in Figure 1, these areas are located 70 miles northwest of Denver, Colorado. These wilderness areas are selected because they contain forested lands that have experienced relatively little direct human management disturbances. As a consequence, the current composition of forest cover types within these areas are primarily a result of natural ecological processes rather than the product of active forest management.

Blackard et al.[3] have compared two alternative techniques for predicting forest cover types.The results of the comparison indicated that a feedforward artificial neural network model(70.58%) more accurately predicted forest cover type than a traditional statistical model based on Gaussian discriminant analysis (58.38%). B. Chandra et al. [5] used the same dataset to evaluate the performance of the decision trees. The decision tree algorithm achieved a maximum classification accuracy of 88.143% as compared to that of 70.58%. Ragini Jain et al. suggested a hybridized rough set model that provides mechanism to trade-off between different performance parameters like - accuracy, complexity, number of rules and number of attributes in the resulting classifier for a large benchmarking dataset.

A number of supervised learning methods have been introduced in the last decade. Caruana et al.[4] present a large-scale empirical comparison of ten Supervised Learning algorithms using eight performance criteria. They evaluate the performance of SVMs, Neural Network, Logistic Regression, Naive Bayes, Memory-based learning, Random Forests, Decision Trees, Bagged Trees, Boosted Trees, and Boosted Stumps on eleven binary classification problems using a variety of performance metrics: accuracy, F-score,Lift, ROC Area, etc. They have concluded that learning methods such as Boosting, Random Forests, Bagging, Decision Tree and SVMs achieve excellent performance over others.

Entezari - Malecki et al.[6] have compared different classification methods based on type of attributes and sample size against performance criterion Area Under the Curve (AUC) of ROC. Their analysis shows that decision tree and C4.5, as an implementation of that show an effective performance in all datasets. Decision tree, C4.5 and SVM show excellent accuracies when the number of continuous attributes is higher that are discrete.

We propose to predict the forest cover type from strictly cartographic variables. We intend to do so by applying preprocessing and feature engineering techniques on the dataset followed by decision tree models. We also make a performance comparison of different decision trees - C4.5, C5.0, and CART, based on different metrics - accuracy, area under roc curve and number of nodes on the current dataset.

The rest of the paper has been organized as follows: The section II explains the proposed model in details, followed by the section III, which gives the results of the various experiments performed at every stage of the model on the forest cover type dataset using decision tree for classification. Section V concludes the paper.

II. PROPOSED WORK

This section shall give a detailed description of all the steps followed in our model. Figure 2 shows all the steps undertaken for the prediction of the forest cover type using decision tree has a classifier.

A. Data Collection

In the data collection phase, data is collected from various sources and integrated. The current dataset is available in the UCI Machine Learning Repository and is derived from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. Forest cover type dataset contains only cartographic variables (no remotely sensed data). Data is in raw form (not scaled) and contains binary (0 or 1) columns of data for qualitative independent variables (wilderness areas and soil types). The dataset for forest cover type prediction

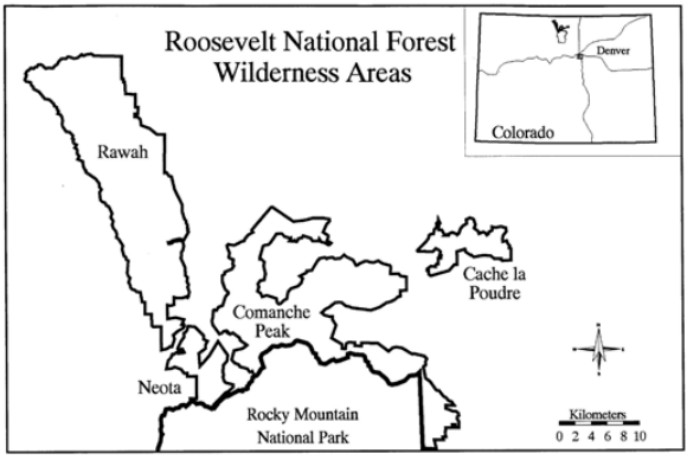


Fig. 1: Roosevelt National Forest Wilderness Area

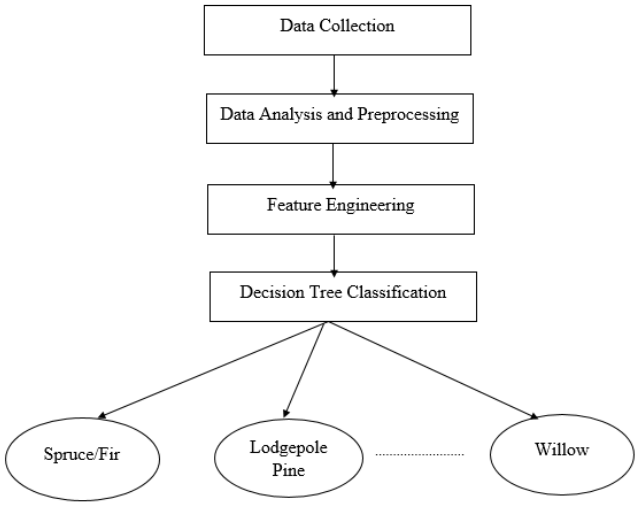


Fig. 2: Process followed in building our model.

includes 54 features. There are forty soil types and four wilderness areas which are binary attributes and other attributes are numeric in nature. We intend to predict an integer classification for the forest cover type. The seven types are Spruce/Fir, Lodgepole Pine, Ponderosa Pine, Willow, Aspen, Douglas/Fir and Krummholz and are represented as integers. The dataset contains continuous, binary and nominal data. Table I gives the description of all the attributes in our dataset.

B. Data Analysis and preprocessing

After collecting the data, it is important to analyze the data and understand the relationships between various attributes. Such an analysis will help in data preprocessing to remove the redundant and irrelevant attributes. The changes made to the dataset after data analysis and preprocessing is as stated below:

1) According to [2], the wilderness areas which are more typical over the dataset are Rawah and Comanche Peak, due to their assortment of tree species and range of predictive variable values (elevation, etc.) Cache la Poudre has relatively low elevation range and species composition and hence would probably be more unique than the others.
2) Elevation with Wilderness Area: Among the four wilderness areas, Neota (area 2) probably has the highest mean elevation value. The second highest mean elevation value would be in the area of Rawah (area 1) and Comanche Peak (area 3) followed by Cache la Poudre (area 4), where it is found to have the lowest mean elevation value.
3) Cover type with wilderness area: Neota would have its primary tree species to be spruce/fir (type 1), while Rawah and Comanche Peak would probably have Lodgepole pine (type 2) as their primary species, followed by spruce/fir and aspen (type 5). Cache la Poudre would tend to have Ponderosa pine (type 3), Douglas-fir (type 6), and cottonwood/willow (type 4).
4) Conversion of binary attributes into categorical attributes: The 40 soil types and 4 wilderness areas which were binary attributes were converted into corresponding categorical attributes. The categorical attribute corresponding to soil types took values from s1 to s40, while the attribute corresponding to wilderness areas took values from w1 to w4.
5) Missing values of Hillshade_3pm: When Hillshade_3pm was plotted, it was found to have certain missing values,

TABLE I: Description of attributes for forest cover dataset

Attribute Name	Data Type	Description
Elevation	Quantitative	Elevation in meters
Aspect	Quantitative	Aspect in degrees azimuth
Slope	Quantitative	Slope in degrees
Horizontal_Distance _To_Hydrology	Quantitative	Horizontal distance to nearest surface water features
Vertical_Distance _To_Hydrology	Quantitative	Vertical distance to nearest surface water features
Horizontal_Distance _To_Roadways	Quantitative	Horizontal distance to nearest roadways
Hillshade_9am	Quantitative	Hillshade index at 9am,summer solstice
Hillshade_Noon	Quantitative	Hillshade index at noon,summer solstice
Hillshade_3pm	Quantitative	Hillshade index at 3pm, summer solstice
Horizontal_Distance _To_Fire_Points	Quantitative	Horizontal distance to nearest wildfire ignition points
Wilderness_Area(4 binary columns)	Qualitative	Wilderness area designation
Soil_Type(40 binary columns)	Qualitative	Soil type designation

which were filled with the median of the values.

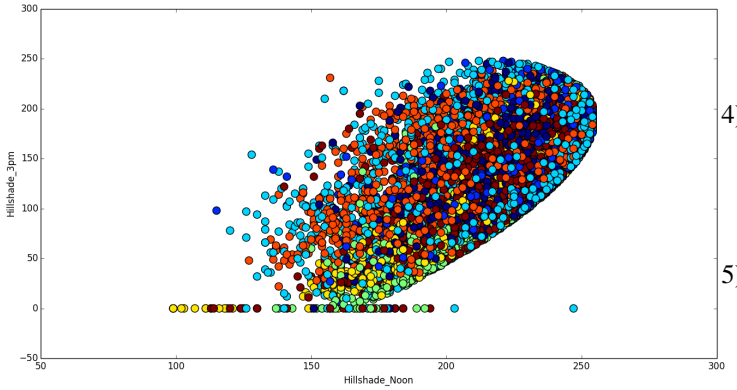


Fig. 3: Hillshade_3pm Vs Hillshade_Noon

C. Feature Engineering

1) Feature Extraction: Feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative, non-redundant, facilitating the subsequent learning and generalization steps, in some cases leading to better human interpretations.

After a thorough data analysis and understanding the relationship between the features, new features were engineered from the existing ones. This section gives a detailed description of the feature engineering done on the data set.

1) Preprocessing based on Soil Type: Soil Types are numbered from 1 to 40 and are categorized based on the USFS Ecological Landtype Units (ELUs). The ELU code of each soil type is a four digit number, where the first and second digits refer to the climatic and geologic zone respectively. The third and the fourth digit refer to a certain mapping unit and does not have any relationship with the climatic or geologic zone. The two changes made to the dataset using the ELU codes are as follows:
   - Soil types which had same climatic(first digit) and geologic zone(second digit) were found to have similar characteristics and were grouped as one. There were 11 such groups and hence, 40 soil types were converted into 11 soil types.
   - The first digit and the second digit of the ELU code is used to create two categorical attributes called 'Climatic' and 'Geologic', which take eight distinct values [2]
2) Negative values of Vertical_Distance_To_Hydrology [1]: Looking at the distribution of Vertical_Distance_To_Hydrology in Figure 4, we found that it has some negative values. We created another variable called 'Hg_wter', which indicates whether this attribute has a positive or a negative value.
3) Relationship between Vertical_Distance_To_Hydrology and Elevation: Elevation and Vertical_Distance_To_Hydrology are correlated to each other. In the Figure 5, coloring each cover type in different colors, seem to reveal a pattern of the plotted points. Hence, we create a new feature called EV_DTH which gives a simpler relation seen in Figure 6. Here, EV_DTH is given by subtracting Vertical_Distance_To_Hydrology from Elevation.
4) Relationship between Elevation and Horizontal_Distance_To_Hydrology: From a similar graph, it has been observed that Elevation and Horizontal_Distance_To_Hydrology are correlated. Hence, we define a new attribute called EH_DTH.
5) Other new features: A new set features HyF_1, HyF_2, HyR_1, HyR_2, FiR_1, FiR_2 were derived from combining all distance based attributes. The final feature set is shown in Table II.

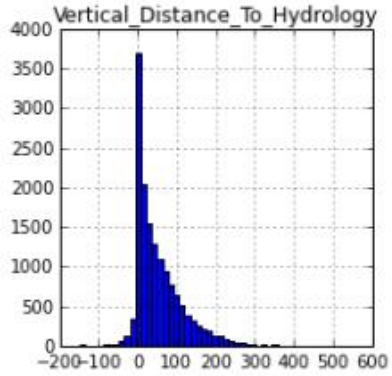


Fig. 4: Plot of Vertical_Distance_To_Hydrology

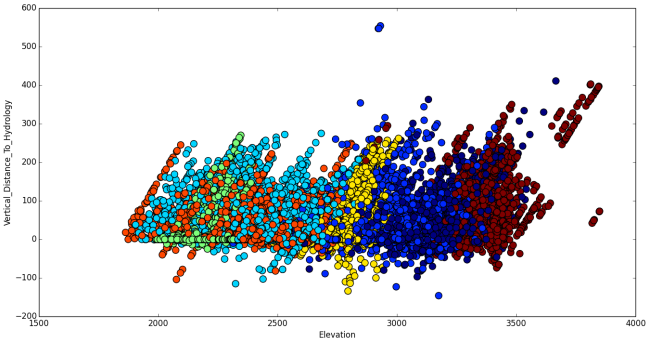


Fig. 5: Graph of Vertical_Distance_To_Hydrology Vs Elevation

2) Feature selection: Feature selection aims to select a subset of relevant features for use in model construction. The central assumption when using a feature selection technique is that the data contains many redundant or irrelevant features. Redundant features are those which provide no more information than the currently selected features, and irrelevant features provide no useful information in any context.

On plotting features against each other, we observed that the features are independent of each other as in Figure 7, hence

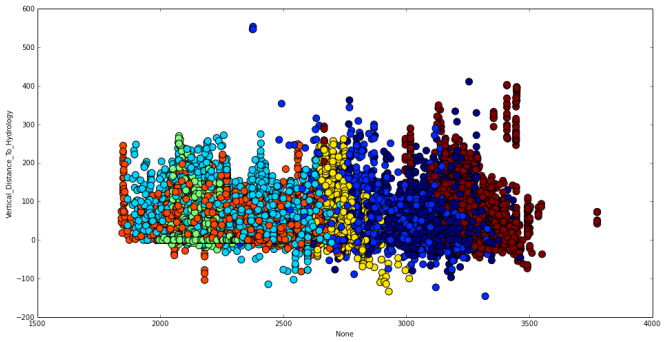


Fig. 6: Graph of EV_DTH Vs Elevation

TABLE II: Description of the new additional features for Forest Cover Dataset

Attribute Name	Data type	Description
Wilderness Area	Qualitative	Wilderness area designation
Soil type	Qualitative	Soil type designation
Aspect2	Qualitative	Aspect in degrees azimuth
Hg_wter	Qualitative	Indicative for positive or negative values to Vertical_Distance_To_Hydrology
EV_DTH	Qualitative	Elevation - Vertical_Distance_To_Hydrology
EH_DTH	Qualitative	Elevation - Horizontal_Distance_To_Hydrology*0.2
Dis_To_Hy	Qualitative	(Horizontal_Distance_To_Hydrology^2 + Vertical_Distance_ To_Hydrology^2)$^{1/2}$
HyF_1	Qualitative	Horizontal_Distance_To_Hydrology + Horizontal_Distance_To_Fire_Points
HyF_2	Qualitative	Horizontal_Distance_To_Hydrology - Horizontal_Distance_To_Fire_Points
HyR_1	Qualitative	Horizontal_Distance_To_Hydrology + Horizontal_Distance_To_Roadways
HyR_2	Qualitative	Horizontal_Distance_To_Hydrology - Horizontal_Distance_To_Roadways
FiR_1	Qualitative	Horizontal_Distance_To_Fire_Points + Horizontal_Distance_To_Roadways
FirR_2	Qualitative	Horizontal_Distance_To_Fire_Points- Horizontal_Distance_To_Roadways

not redundant.

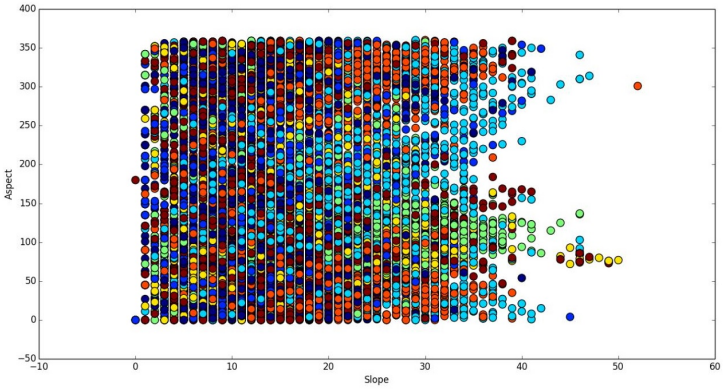


Fig. 7: Plot of Aspect vs Slope

Also, decision trees use attribute selection measures like Gain Ratio during tree construction. Thus, only the relevant attributes appear in the final model. This eliminates the need for using any other feature selection technique.

D. Decision Trees

A decision tree is a simple representation for classifying examples. Decision tree learning is one of the most successful techniques for supervised classification learning. A decision tree or a classification tree is a tree in which each internal (non-leaf) node is labeled with an input feature. The arcs coming from a node labeled with a feature are labeled with each of the possible values of the feature. Each leaf of the tree is labeled with a class or a probability distribution over the classes.A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node has all the same value of the target variable, or when splitting no longer adds value to the predictions.

A number of decision trees have been proposed in the literature and they mainly differ by the measure used for attribute selection. For example, C4.5 and C5.0 use Gain Ratio whereas CART uses Ginni as the attribute selection measure, in the tree construction phase. One such important Decision tree is C5.0.

1) C5.0: C4.5 algorithm ensured better way of building a decision tree with the use of gain ratio has the spitting factor but it lacked many non-functional requirements to popularize among the applications. Hence, C5.0 [7] was developed as an improvement over the existing C4.5 algorithm. Many key aspects found in C5.0 makes it better than C4.5 Algorithm. Below are the list of its extended features:

- Rules Formation - Building the decision trees and using then for every test set prediction leads to high wastage of time. In order to overcome this, C5.0 has incorporated a ruleset formation instead of trees to save significant amount of time and space.
- Winnowing technique - used in C5.0 helps in reduction of memory by using lesser sample set, which was not available in C4.5.
- Smaller decision trees - C5.0 gets similar results to C4.5 with considerably smaller decision trees.
- Boosting - Support for boosting allows the creation of multiple decision trees . These trees formation is improvement over time till no more misclassification. Then based on the decision from all the classifiers a better prediction is made leading to better accuracy.
- Weighting - C5.0 allows you to vary the importance of different cases by giving different weights. Minimizing the weighted predictive error rate is the way to handle weighting.
- Misclassification Costs - In C4.5, all errors are treated as equal, but in practical applications some classification

errors are more serious than others. Hence, C5.0 allows a separate cost to be defined for each predicted versus actual class pair ratio; if this option is used, C5.0 then constructs classifiers to minimize expected misclassification costs rather than error rates.

- Data Types - C5.0 has several new data types in addition to those available in C4.5, including dates, times, timestamps, ordered discrete attributes, and case labels. In addition to missing values, C5.0 allows values to be noted as not applicable. Further, C5.0 provides facilities for defining new attributes as functions of other attributes.

III. Result and Analysis

This section shall give a detailed description and analysis of the results obtained from the various experiments conducted as stated in the proposed work.

All the experimentations were carried out in the R programming using the CRAN built in packages and the python codes were used for the feature engineering and data analysis.

A. Decision Trees

The decision trees was selected has the classifier for our forest cover type prediction from the literature survey. Experimental study for the selection of the right decision tree was conducted using the existing packages in R.

The Table V summarizes the comparison of decision trees under the selected metrics. The classification metrics taken in to consideration include:

- Accuracy: Gives a measure of the number of the samples are correctly predicted.
- Tree size: Indicates the number of nodes that appear in the constructed decision tree.
- AUC: Area Under Curve for Receiver Operating Characteristic

TABLE III: Performance Evaluation

Performance Metrics/Decision trees	C4.5	C5.0(No Boosting)	CART
Accuracy in %	80.78	91.11	79.45
Size of the Tree(nodes)	2111	772	951
AUC	0.92	0.88	0.94

Based on all the measures mentioned above, important point is to note that C5.0 has higher accuracy and less tree size due to the various built-in techniques like pruning, boosting and winnowing to improve tree performance and possibly reduce the memory requirements for building the model.

B. Feature engineering

As stated in the Section II, our dataset needed preprocessing and feature extraction. Table IV gives the modifications made to the features and the improvements to the model when compared to the base case.

Base case: Accuracy of the model when tested on real test set with no changes made to the raw data. This case gave an accuracy of 68.44%.

TABLE IV: Data Pre-Processing and Feature Extraction

Expt No.	Feature Description	Feature Modifications	Reason	Results
1	Missing Soil Types7 and Soil Types15 in the training set	Removal of s7 and s15 from test set (There were 105 samples with soil type s7).	The relevance of these features seemed to be very low according to the attribute usage obtained from C50 tree built from test set.	Improved to 69.13%
2	Soil Type (Qualitative) with 40 binary columns	Generalization of Soil Type to 11 Columns	Generalization is based on the ELU codes of soil types	Decreased to 67.32%
3	Soil Type (Qualitative) with 40 binary columns	Generalization based on geologic and climatic zones	Generalization is based on the ELU codes of soil types	Decreased further to 66.95%
4	Wilderness Area (Qualitative) with 4 binary columns	Generalization to one categorical feature	To lower this features used due to its lower relevance.	Improved to 68.89%
5	Missing values in Hillshade_3pm	Replaced those values with the mean of all Hillshade_3pm	To remove the outliers	Improved to 69.17%
6	Soil type (Qualitative) with 40 binary columns	Generalized to one categorical feature	To lower the features used due to its lower relevance	Improved to 69.13%

The experiments performed above showed very slight improvements over the primary dataset. Based on the accuracy improvement, the features were selected for the new dataset.

The detailed data analysis with decision trees also showed the need for enhancing the feature space with more relevant features. So, the further experimentations were performed in building new features from the given feature set. Table II shows the lists of new features obtained from the feature engineering:

The new feature set when tested on the C5.0 decision tree with no boosting gave an improvement in accuracy to 69.22%.

C. Feature selection

Feature Selection is a technique to select the best subset of features from the given set in order to maximize the accuracy. The decision trees are known to have the build in feature selection based on the splitting parameter. These are in fact known as embedded feature selectors.
Hence, no additional feature selection was performed.

D. Pruning

Pruning is a way of reducing the size of the decision tree. This will reduce the accuracy on the training data, but (in general) increase the accuracy on unseen data. It is used to reduce over fitting, where you would achieve perfect accuracy on training data, but the model (i.e. the decision tree) that is

learned is very specific that it doesn't apply to anything but that training data.
In our case, varying a parameter called confidence factor(which controls the amount of pruning) from 0.25 to 0.15 for the C5.0 Decision tree is found to improve the accuracy to 69.25%

E. Ensemble Learning

The new features obtained from the previous steps were found to be more relevant. Yet the model showed only 1% improvement . Ensemble learning is a technique to enhance the learning of weak base learners like decision trees. Hence, ensemble techniques Random forest and C5.0 were used for further testing. Random forest gave an improvement in accuracy to 77.24% which was found to be better than that of single decision tree. Whereas C5.0 with boosting iterations set to 10 gave an improved accuracy to 76.02%.The Table V gives the detailed comparison of both the techniques.

TABLE V: Performance Evaluation for Ensemble Techniques

Performance Metrics/Ensemble Learning with 10 trials	C5.0	Random Forest
Accuracy in %	76.02	77.24
AUC	0.82	0.85

The above boosting and bagging techniques showed to increase the prediction performance. The detailed study of these ensemble techniques and their modification may help in improving the overall model for better prediction.

IV. Conclusions

The purpose of this project is to use the decision tree classification algorithms for predicting the forest cover type. The forest cover data of the Roosevelt National Forest of northern Colorado was used to evaluate the performance of various Decision Tree algorithms. Among the decision trees, C5.0 was found to give higher accuracy. Various feature engineering techniques performed on the dataset showed improvement over the primary data-set. The new feature set when tested with C5.0 decision tree with no boosting gave an improvement in accuracy to 69.22%. Random forest and C5.0 gave an improvement in accuracy to 77.24% and 76.02% respectively. These show that ensemble techniques can enhance the performance of decision trees considerably. These ensemble techniques can be improved further.

Acknowledgment

We would like to thank Mr. Biju R Mohan for guiding us and giving his valuable insights throughout the project.

References

[1] "Kaggle." [Online]. Available: https://www.kaggle.com/c/forest-cover-type-prediction/forums/t/10693/features-engineering-benchmark

[2] "Uci repository database." [Online]. Available: https://archive.ics.uci.edu/ml/machine-learning-databases/covtype/covtype.info

[3] J. A. Blackard and D. J. Dean, "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables," *Computers and electronics in agriculture*, vol. 24, no. 3, pp. 131–151, 1999.

[4] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 161–168.

[5] B. Chandra and V. Pallath Paul, "Prediction of forest cover using decision trees," *J. Ind. Soc. Agril. Statist*, vol. 61, no. 2, pp. 192–198, 2007.

[6] R. Entezari-Maleki, A. Rezaei, and B. Minaei-Bidgoli, "Comparison of classification methods based on the type of attributes and sample size," *Journal of Convergence Information Technology*, vol. 4, no. 3, 2009.

[7] S.-l. Pang and J.-z. Gong, "C5.0 Classification Algorithm and Application on Individual Credit Evaluation of Banks," *Systems Engineering - Theory & Practice*, vol. 29, no. 12, pp. 94–104, Dec. 2009. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1874865110600920