Forest Cover Type

ISDS 574

--

Group 5

Ami Thakkar
Jayantee Bhalerao
Harikrishnan Girikumar
Manasi Sakhadeo
Purva Bangad
Sagar Pathak
Vatan Shah
Vivek Gudapati

Under the guidance of Dr. Yinfei Kong

Table of Contents

Executive Summary

The project aims to analyze and classify forest cover types based on various cartographic variables. The training data we chose for our project include four wilderness areas located in the Roosevelt National Forest of northern Colorado. The actual sample data includes different forest cover types for an area of 30*30 meter cell. This data was determined from the US Forest Service (USFS) Region 2 Resource Information System. Independent variables were then derived from data obtained from the US Geological Survey and USFS. The area was chosen for training data because of the minimal disturbance by humans and to predict the accuracy in determining the cover type based on ecological evidence or practices rather than forest management.

The goal of this project is to create a model that predicts forest cover type using historical data already available. The data fields used by our model include various aspects of the forests like elevation, aspect, slope, soil type, horizontal and vertical distance to hydrology, etc. We have used three different methods to classify the given data. The methods used are- CART, Random Forest, and KNN. Based on the literature review we understand that using a supervised classification method can yield high accuracy. Our aim is to select a method that can yield maximum accuracy with the data available.

On implementing these methods to our training data set our analysis revealed various results. Using KNN on the training data set we get different accuracies for different values of k. We selected three different values of k, i.e. 4, 7, and 16 for testing our model. The results show that we get the best accuracy of 76.62 % when we select k=4. Using CART, we divided the dataset into a training dataset and validation dataset. The results gave an accuracy of 60.82% when we used the CART method for classification. Random forest is a technique that can perform both regression and classification. Using the Random forest method we get an accuracy of 82.14%.

Based on our results, we recommend the use of Random forest methodology for the classification of data. The fundamental idea behind the Random Forest method is to combine various decision trees in determining the final output instead of relying on individual decision trees. Also, from the implementation point of view, we can see that Random Forests are simpler to train as a model and are a crucial model to solve real-world problems. Random forest method not only yielded us a higher accuracy of 82.14% but also is a simpler model to train.

Introduction & Problem Description

Forest cover is one category of terrestrial land cover. Land cover is the observed physical features, both natural and manmade, that occupy the earth’s immediate surface. Forest cover is defined as 25% or greater canopy closure at the Landsat pixel scale.

Conducting a site survey where the prospectors physically examined the site was one of the first ways to classify the forest cover type. The limitation of this method was that the locations are remote and there are no roads available to reach these locations. The other method was aerial imaging using aircraft and drones. The drones have limited battery life and it is sometimes difficult to cover 50 to 150 acres per flight. The latest technology used is satellite imaging to get images of forest cover using satellites.

Satellite imaging is the most effective and advanced method to classify the forest cover type. But it is cost-prohibitive. So, we explore an alternative in Data analytics. Using historical data, we can predict forest cover with high accuracy. This method is considerably more cost-effective than all the other alternatives.

This dataset used in the project contains tree observations from four areas of the Roosevelt National Forest in Colorado. All observations are cartographic variables (no remote sensing) from 30-meter x 30-meter sections of forest. This dataset includes information on tree type, shadow coverage, distance to nearby landmarks (roads etcetera), soil type, and local topography. The goal of this project is to predict the forest cover type in four different wilderness areas of the Roosevelt National Forest of Northern Colorado with the best accuracy.

Seven categories numbered from 1 to 7 in the Cover_Type column, to be classified:

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

The natural successor to these methods is data analysis. Creating models to predict forest cover using historical data already available. There were mainly 3 methods used to classify the forest cover type.

(i) CART
(ii) Random Forest
(iii) KNN

Literature Review

The first article “Cover Classification by Optimal Segmentation of High-Resolution Satellite Imagery” investigates whether high-resolution satellite imagery is suitable for preparing a detailed digital forest cover map that discriminates forest cover at the tree species level. The images were classified using a traditional pixel-based, maximum-minimum classification approach. A reclassification was done using the segment-based approach with a majority rule.

The second article “Classifying and mapping forest cover types using IKONOS imagery in the NorthEastern United States” tries to increase the accuracy of mapping forest cover types. They try to increase the accuracy by incorporating image processing techniques such as hybrid forms of the supervised and unsupervised classification methods.

In the third article “Extracting and analyzing forest and woodland cover change in Eritrea based on Landsat data using supervised classification” remote sensing is used to capture high-resolution images of the forest. The methodology used to classify the data are supervised classification techniques. The results indicated that the forest and woodland cover extracted were with high overall accuracy of 96%.
Based on the research, we test our models using supervised classification methods like CART, KNN, and Random Forest.

Background

In 1630, the estimated area of U.S. forest land was 1,023 million acres which are about 46 percent of the total land area. By 1910, the area of forest land had declined to an estimated 754 million acres or 34 percent of the total land area. In 2012, forest land comprised 766 million acres or 33 percent of the total land area of the United States. The Forest area has been relatively stable since 1910.

In the early days, the classification of forest cover type was done by conducting a site survey where the prospectors physically visited the site. The prospectors did the classification by visually examining the site. If necessary, they would take biometric measurements considering the Biometric Protocol to support their choice of forest cover classification.

As time passed and technology advanced, a physical site survey was replaced by computer-aided classification of small scale aerial photography, which proved to be a promising technique for generating forest cover type maps efficiently and inexpensively. Cameras were capable of taking clear images of the ground from aircraft moving at incredibly high speeds. This coupled with computer-aided manipulation to map different sections of the photographs accurately made this method viable for many years.

There are thousands of satellites in orbit around the Earth. These perform a number of different functions but a good number of them can be used for satellite imaging. Thus a promising alternative to the direct mapping of forest cover types is using high-resolution satellite images, which have the advantage of covering relatively large landmasses on periodically repeating cycles. A number of earlier studies have used coarse spatial resolution satellite imagery for forest cover mapping, which allowed the classification of forest cover into broad categories, such as coniferous, deciduous, and mixed forests.

Data Fields

Elevation - Elevation in meters

Aspect - Aspect in degrees azimuth
Slope - Slope in degrees

Horizontal_Distance_To_Hydrology - Horz Dist to nearest surface water features
Vertical_Distance_To_Hydrology - Vert Dist to nearest surface water features
Horizontal_Distance_To_Roadways - Horz Dist to the nearest roadway
Hillshade_9am (0 to 255 index) - Hillshade index at 9am, summer solstice
Hillshade_Noon (0 to 255 index) - Hillshade index at noon, summer solstice
Hillshade_3pm (0 to 255 index) - Hillshade index at 3pm, summer solstice
Horizontal_Distance_To_Fire_Points - Horz Dist to nearest wildfire ignition points
Wilderness_Area (4 binary columns, 0 = absence or 1 = presence) - Wilderness area designation
Soil_Type (40 binary columns, 0 = absence or 1 = presence) - Soil Type designation
Cover_Type (7 types, integers 1 to 7) - Forest Cover Type designation

The wilderness areas are:
1 - Rawah Wilderness Area
2 - Neota Wilderness Area
3 - Comanche Peak Wilderness Area
4 - Cache la Poudre Wilderness Area

The soil types are:

1 Cathedral family - Rock outcrop complex, extremely stony.
2 Vanet - Ratake families complex, very stony.
3 Haploborolis - Rock outcrop complex, rubbly.
4 Ratake family - Rock outcrop complex, rubbly.
5 Vanet family - Rock outcrop complex complex, rubbly.
6 Vanet - Wetmore families - Rock outcrop complex, stony.
7 Gothic family.
8 Supervisor - Limber families complex.
9 Troutville family, very stony.
10 Bullwark - Catamount families - Rock outcrop complex, rubbly.
11 Bullwark - Catamount families - Rock land complex, rubbly.
12 Legault family - Rock land complex, stony.
13 Catamount family - Rock land - Bullwark family complex, rubbly.
14 Pachic Argiborolis - Aquolis complex.
15 unspecified in the USFS Soil and ELU Survey.
16 Cryaquolis - Cryoborolis complex.
17 Gateview family - Cryaquolis complex.
18 Rogert family, very stony.
19 Typic Cryaquolis - Borohemists complex.

20 Typic Cryaquepts - Typic Cryaquolls complex.
21 Typic Cryaquolls - Leighcan family, till substratum complex.
22 Leighcan family, till substratum, extremely bouldery.
23 Leighcan family, till substratum - Typic Cryaquolls complex.
24 Leighcan family, extremely stony.
25 Leighcan family, warm, extremely stony.
26 Granile - Catamount families complex, very stony.
27 Leighcan family, warm - Rock outcrop complex, extremely stony.
28 Leighcan family - Rock outcrop complex, extremely stony.
29 Como - Legault families complex, extremely stony.
30 Como family - Rock land - Legault family complex, extremely stony.
31 Leighcan - Catamount families complex, extremely stony.
32 Catamount family - Rock outcrop - Leighcan family complex, extremely stony.
33 Leighcan - Catamount families - Rock outcrop complex, extremely stony.
34 Cryorthents - Rock land complex, extremely stony.
35 Cryumbrepts - Rock outcrop - Cryaquepts complex.
36 Bross family - Rock land - Cryumbrepts complex, extremely stony.
37 Rock outcrop - Cryumbrepts - Cryorthents complex, extremely stony.
38 Leighcan - Moran families - Cryaquolls complex, extremely stony.
39 Moran family - Cryorthents - Leighcan family complex, extremely stony.
40 Moran family - Cryorthents - Rock land complex, extremely stony.

Data Preprocessing

Data can be noisy, incomplete, consists of outliers or missing values. Data preprocessing helps to clean data for further usage. The chosen dataset consists of 15120 observations of 56 variables, which are reduced to 14888 observations and 53 variables after the preprocessing is completed. The raw dataset was huge, and hence data preprocessing was a critical task. The dataset was clean to a certain extent, but it needed a closer look for better predictions.

Name	Measurement	Description

Elevation	meters	Elevation in meters
Aspect	azimuth	Aspect in degrees azimuth
Slope	degrees	Slope in degrees
Horizontal Distance To Hydrology	meters	Horz Dist to nearest surface water features
Vertical Distance To Hydrology	meters	Vert Dist to nearest surface water features
Horizontal Distance To Roadways	meters	Horz Dist to nearest roadway
Hillshade 9am	0 to 255 index	Hillshade index at 9am, summer solstice
Hillshade Noon	0 to 255 index	Hillshade index at noon, summer soltice
Hillshade 3pm	0 to 255 index	Hillshade index at 3pm, summer solstice
Horizontal Distance To Fire Points	meters	Horz Dist to nearest wildfire ignition points
Wilderness Area (4 binary columns)	0 (absence) or 1 (presence)	Wilderness area designation

Soil Type (40 binary columns)	0 (absence) or 1 (presence)	Soil Type designation
Cover Type	Classes 1 to 7	Forest Cover Type designation - *Response Variable*

Categorical variable

The seven types for a cover type are which are to be classified in Cover_Type column:

1 - Spruce/Fir

2 - Lodgepole Pine

3 - Ponderosa Pine

4 - Cottonwood/Willow

5 - Aspen

6 - Douglas-fir

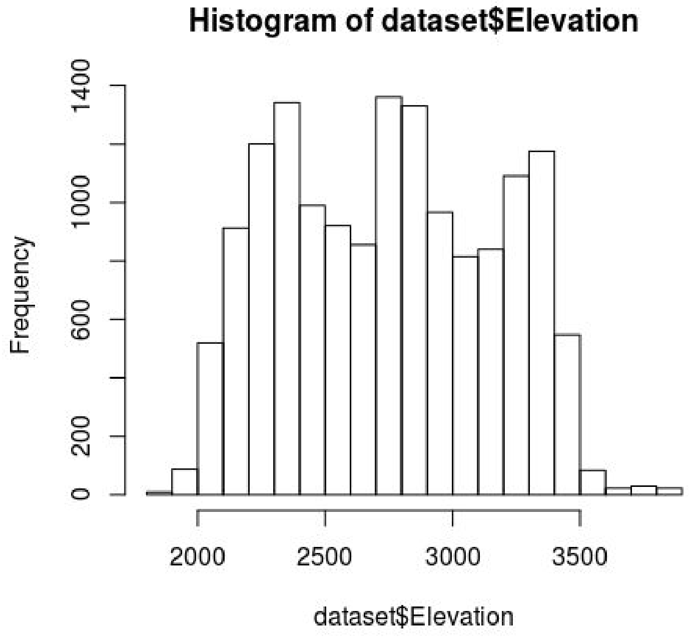
7 - Krummholz

Columns with index numbers 1, 22, 30, which were ID, Soil_Type7, and Soil_Type15 are removed as the values were varied and 1.

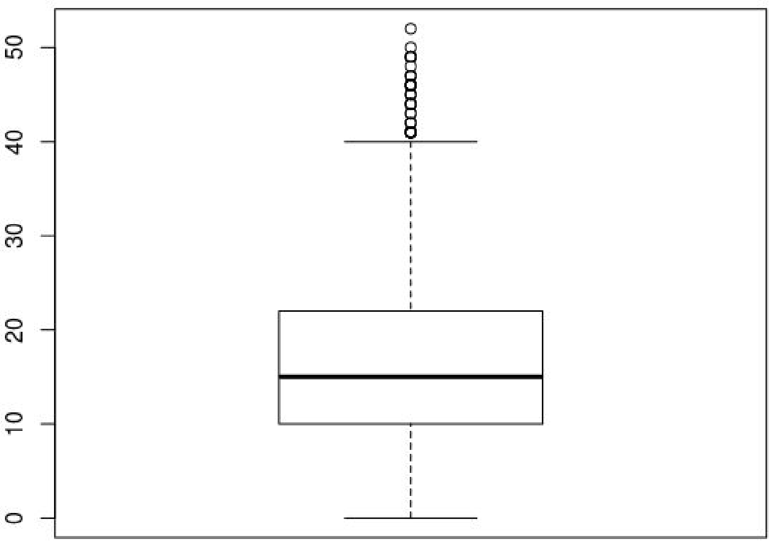
Data exploration methods

A histogram is a graph that describes the frequency of the occurrences in the desired range of values, which are arranged at specific intervals as per the requirement of the data.

Class separation is clearly visible in the following plots of elevation derived by plotting a histogram as below:

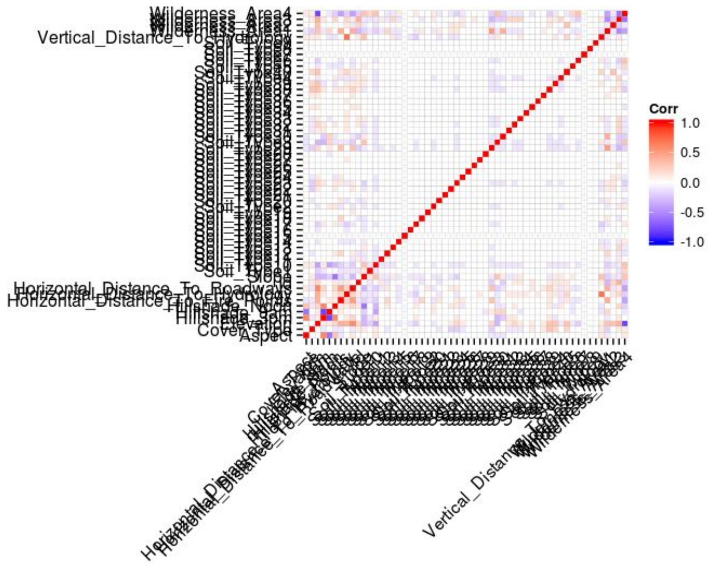


Boxplot is a graphical representation of how the values in the data are spread. Boxplots provides the variability and dispersion of the data. Boxplots are helpful to determine the outliers and provide their values too.



Data reduction

The correlation matrix was created and determined that the red where correlation is more i.e., one and blue where correlation is less i.e., -1.



Outliers

length(boxplot(dataset$Slope)$out) - 57: to check the number of outliers and verification is more comfortable with the function written.
The group wrote a function to detect upper and lower outliers by calculating the interquartile range. The following outliers for each column were calculated as follows:

```
"0  is the number of outliers in  Elevation"
"0  is the number of outliers in  Aspect"
"0  is the number of outliers in  Slope"
"53  is the number of outliers in  Horizontal_Distance_To_Hydrology"
"49  is the number of outliers in  Vertical_Distance_To_Hydrology"
"3  is the number of outliers in  Horizontal_Distance_To_Roadways"
"7  is the number of outliers in  Hillshade_9am"
"20  is the number of outliers in  Hillshade_Noon"
```

```
"0  is the number of outliers in  Hillshade_3pm"
"132  is the number of outliers in  Horizontal_Distance_To_Fire_Points"
"3597  is the number of outliers in  Wilderness_Area1"
"499  is the number of outliers in  Wilderness_Area2"
"0  is the number of outliers in  Wilderness_Area3"
"0  is the number of outliers in  Wilderness_Area4"
"355  is the number of outliers in  Soil_Type1"
"623  is the number of outliers in  Soil_Type2"
"962  is the number of outliers in  Soil_Type3"
"843  is the number of outliers in  Soil_Type4"
"165  is the number of outliers in  Soil_Type5"
"650  is the number of outliers in  Soil_Type6"
"1  is the number of outliers in  Soil_Type8"
"10  is the number of outliers in  Soil_Type9"
"2142  is the number of outliers in  Soil_Type10"
"406  is the number of outliers in  Soil_Type11"
"227  is the number of outliers in  Soil_Type12"
"476  is the number of outliers in  Soil_Type13"
"169  is the number of outliers in  Soil_Type14"
"114  is the number of outliers in  Soil_Type16"
"612  is the number of outliers in  Soil_Type17"
"60  is the number of outliers in  Soil_Type18"
"46  is the number of outliers in  Soil_Type19"
"139  is the number of outliers in  Soil_Type20"
"16  is the number of outliers in  Soil_Type21"
"345  is the number of outliers in  Soil_Type22"
"757  is the number of outliers in  Soil_Type23"
"257  is the number of outliers in  Soil_Type24"
"1  is the number of outliers in  Soil_Type25"
"54  is the number of outliers in  Soil_Type26"
"15  is the number of outliers in  Soil_Type27"
"9  is the number of outliers in  Soil_Type28"
"1291  is the number of outliers in  Soil_Type29"
"725  is the number of outliers in  Soil_Type30"
"332  is the number of outliers in  Soil_Type31"
"690  is the number of outliers in  Soil_Type32"
"616  is the number of outliers in  Soil_Type33"
"22  is the number of outliers in  Soil_Type34"
"102  is the number of outliers in  Soil_Type35"
```

```
"10  is the number of outliers in  Soil_Type36"
"34  is the number of outliers in  Soil_Type37"
"728  is the number of outliers in  Soil_Type38"
"657  is the number of outliers in  Soil_Type39"
"459  is the number of outliers in  Soil_Type40"
"0  is the number of outliers in  Cover_Type"
```

Removed the following outliers:

- Horizontal_Distance_To_Hydrology
- Horizontal_Distance_To_Roadways
- Horizontal_Distance_To_Fire_Points
- Vertical_Distance_To_Hydrology

Wilderness and soil_type are binary so they have not been removed.

Data cleaning

The data cleaning process with a systematic approach helped the group to attain cleaner data. The code checked for missing values, and the group derived there were none. Unwanted observations and variables were removed, making the data more structured. With the help of boxplots, outliers were detected and removed accordingly. Scaling of columns 1 - 10 was done in order to normalize the data.

Data Splitting

The whole dataset is split into training and test data with 60% and 40% of the whole respectively. First, the seed is set to ‘123’ so that the data is not split randomly for each run. Then the subset is created for both test and train.

Data Mining tasks

CART

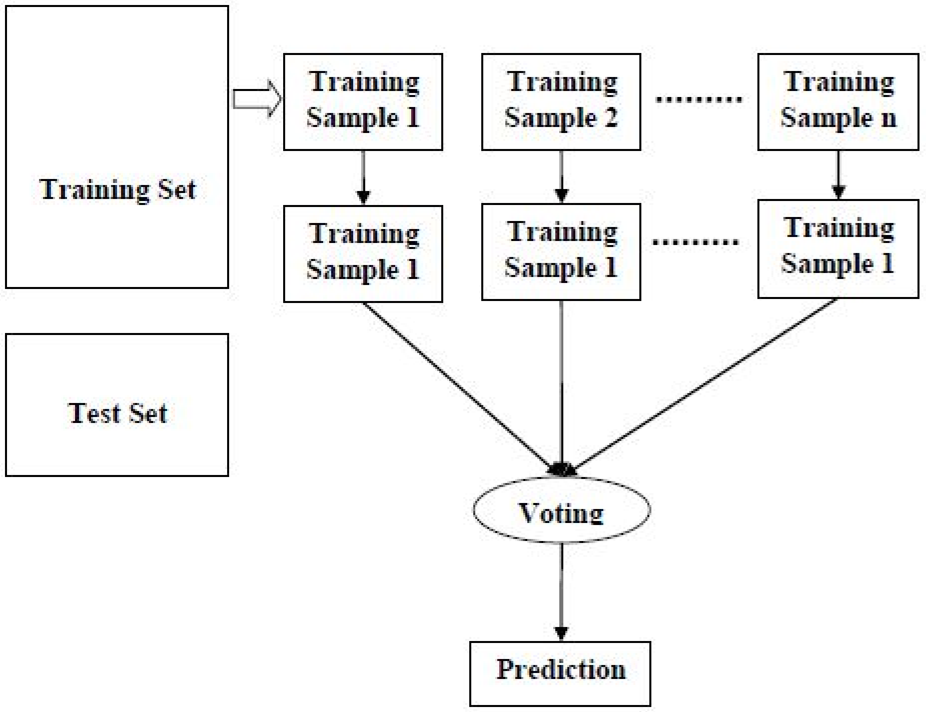
Classification and Regression Trees or CART model is the representation of a binary tree. A node represents a single input variable (X) and a split point on that variable, assuming the variable is numeric. The leaf nodes (also called terminal nodes) of the tree contain an output variable (y), which is used to make a classification. Once created, a tree can be navigated with a new row of data following each branch with the splits until a final classification is made. Creating a binary decision tree is a process of dividing the input space. Splitting continues until nodes contain a minimum number of training examples, or the maximum tree depth is reached.
Here, we have used 53 variables to create a tree by dividing the data into training and validation datasets. The training data and validation data consisted of 9072 and 6048 observations, respectively. We then selected Best Pruned Tree using the validation data. This method helped in demonstrating a simpler model, which is easier to interpret and understand.

Random Forest

A Random Forest is an ensemble technique that can perform both regression and classification tasks with the use of multiple decision trees and a method called Bootstrap Aggregation, also known as bagging. The fundamental idea behind this is to combine various decision trees in determining the final output instead of relying on individual decision trees.
Working of Random Forest Algorithm

- Step 1 − Start with a selection of random samples from a given dataset.
- Step 2 − The random Forest Algorithm will construct a decision tree for every sample. Then it will get the classification result from every decision tree.
- Step 3 − Here, voting will be performed for every classified result.

- Step 4 − At last, the most voted classification outcome will be selected as the final cover type.
  .



KNN

K Nearest Neighbor is a Supervised Machine Learning algorithm that classifies a new data point into the target class, depending on the features of its neighboring data points.

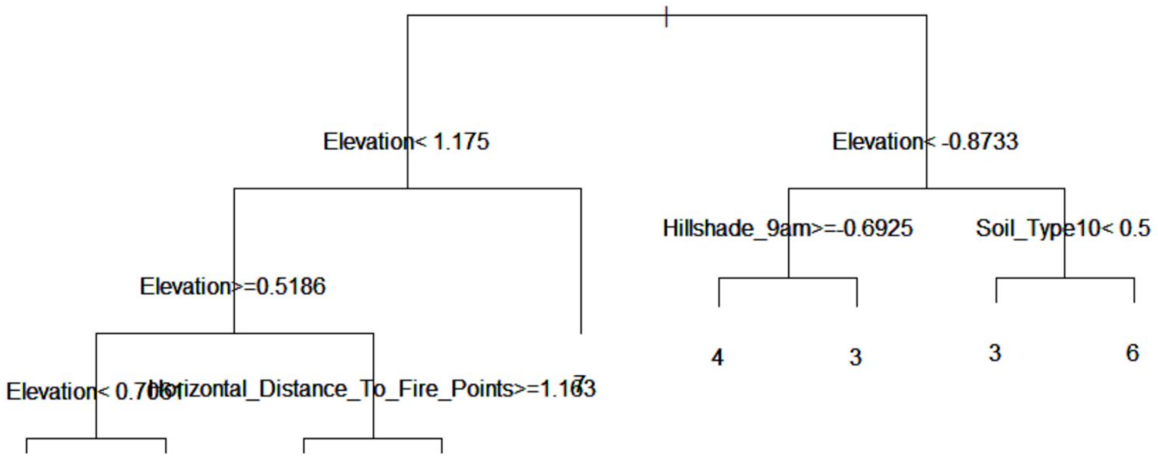
- **Step 1**− After loading the dataset and splitting it into training and test data, we need to choose the value of K that is, the nearest data points. Here, K can be any integer.
- **Step 2** − For each point in the test data, the following is done −
  a − The distance between test data and each row of the training data is calculated with the help of any of the methods, namely: Euclidean, Manhattan, or Hamming distance. The most commonly used method to determine distance is the Euclidean method.
  b − Based on the distance value, sort them in ascending order.

c − Next, it will choose the top K rows from the sorted array.
d − Now, it will assign a class to the test point based on the most frequent class of these rows.

Results

CART

Here we consider Cover Type as the outcome variable and build classification tree using rpart() with method=class. We use plot() function followed by text() function to build a naïve classification tree.

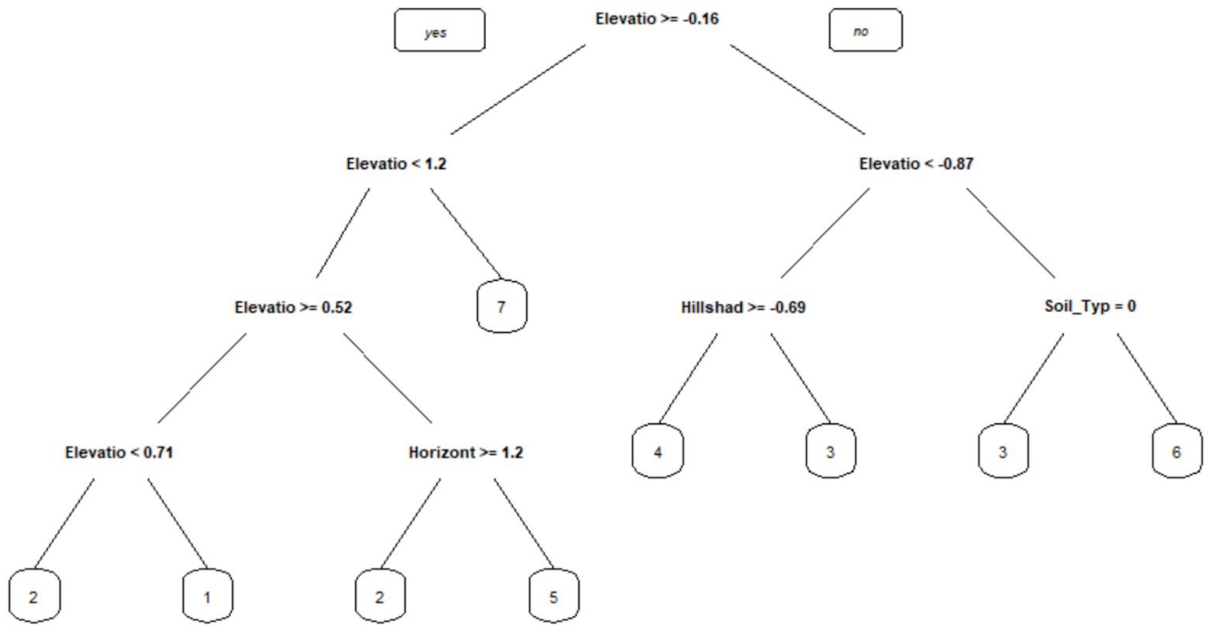


Based on the values of predictors, a plot with a set of rules is created to classify train data records into different Cover Type.
Rules of plotted classification tree are as follows:

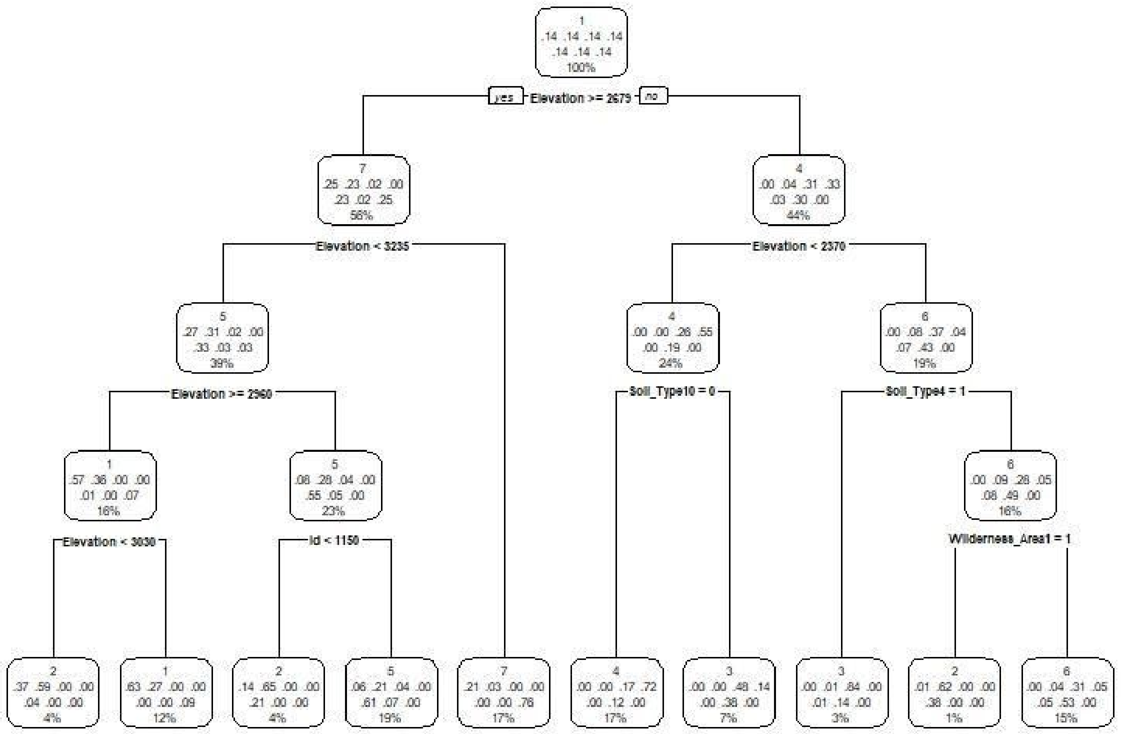
```
   Cover_Type
   1    2    3    4    5    6    7
1 [.64 .27 .00 .00 .00 .00 .09] when Elevation is  0.71 to  1.17
2 [.35 .62 .00 .00 .03 .00 .00] when Elevation is  0.52 to  0.71
2 [.25 .66 .00 .00 .08 .02 .00] when Elevation is -0.16 to  0.52&
Horizontal_Distance_To_Fire_Points        >= 1.2
3 [.00 .00 .45 .16 .00 .40 .00] when Elevation <  -0.87 & Hillshade_9am <  -0.69
3 [.00 .10 .45 .05 .10 .30 .00] when Elevation is -0.87 to -0.16 & Soil_Type10 is 0
4 [.00 .00 .16 .72 .00 .11 .00] when Elevation <  -0.87 & Hillshade_9am >= -0.69
5 [.06 .22 .04 .00 .61 .07 .00] when Elevation is -0.16 to  0.52 &
Horizontal_Distance_To_Fire_Points <  1.2
```

6 [.00 .03 .21 .01 .00 .74 .00] when Elevation is -0.87 to -0.16 & Soil_Type10 is 1
7 [.21 .04 .00 .00 .00 .00 .74] when Elevation >=1.17

plot() produces a couple of black clouds of overlaid text. We will use prp() for better results and produce a better tree structure.



Now, we use rpart.plot() which combines and extends plot.rpart() and text.rpart()



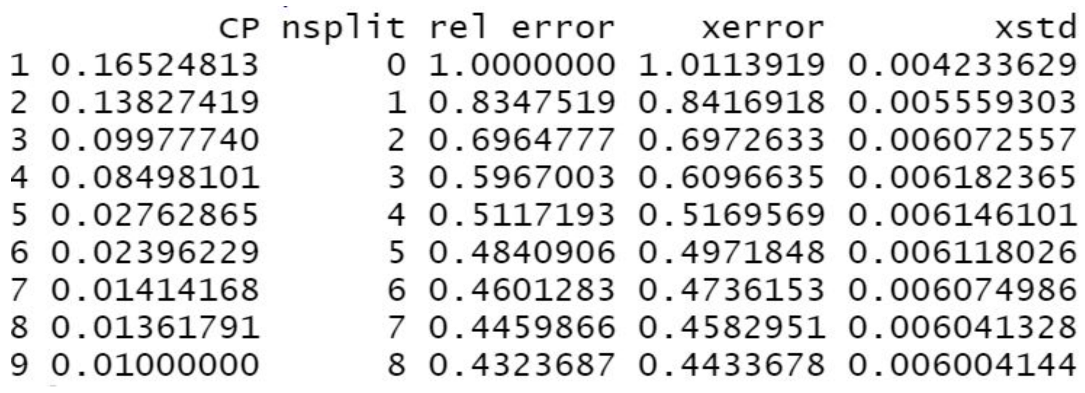
Here, each node describes classified forest cover type and percentage of data in each node.

Confusion matrix for full tree is as follow:

```
predict_fulltree
        1   2   3   4   5   6   7
  1 422 171   0   0   0   0  70
  2 130 228   0   0  24   3   0
  3   2  81 551 106  67 411   0
  4   0   0 182 758   0 138   0
  5  81 306  57   0 760  61   4
  6   3  12  74   0   3 251   0
  7 207  29   0   0   0   0 763
```

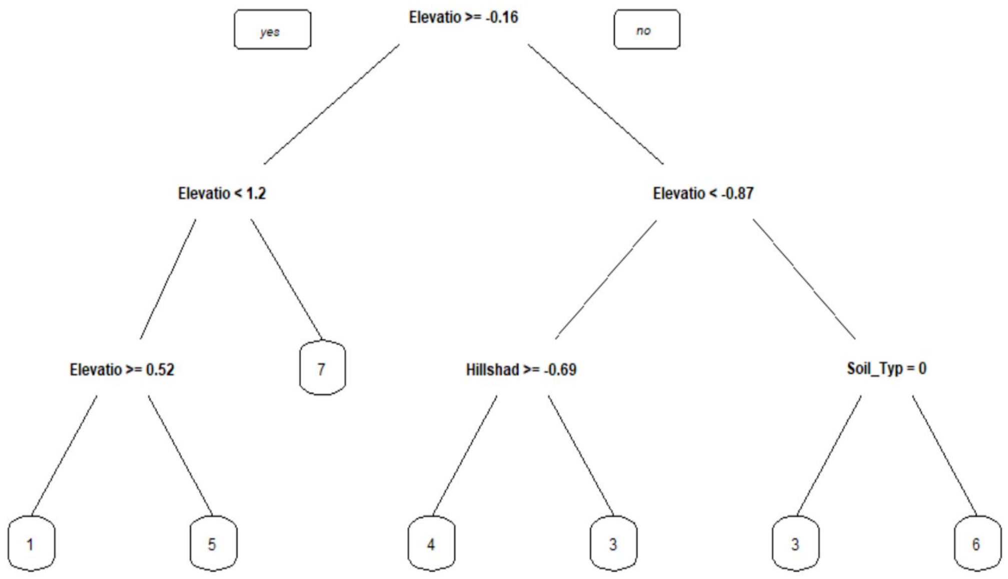
Accuracy for full tree is 62.68%

Following is output of CP, split and error rates.



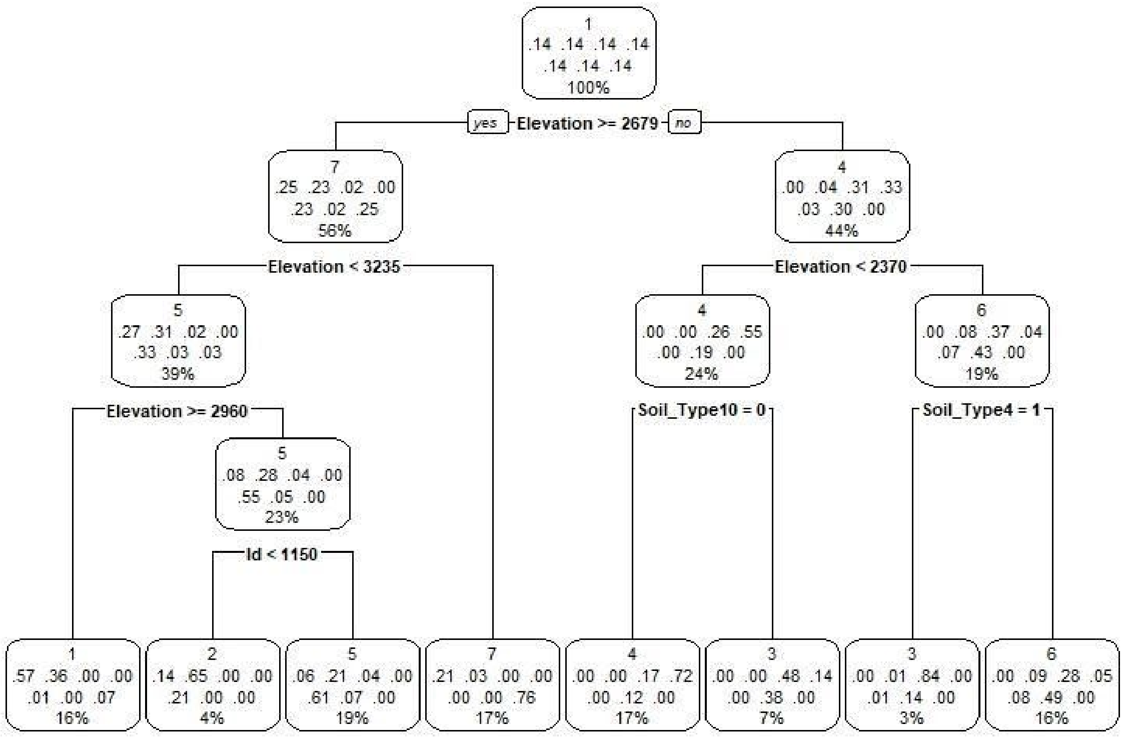
```
          CP nsplit rel error    xerror        xstd
1 0.16524813      0 1.0000000 1.0113919 0.004233629
2 0.13827419      1 0.8347519 0.8416918 0.005559303
3 0.09977740      2 0.6964777 0.6972633 0.006072557
4 0.08498101      3 0.5967003 0.6096635 0.006182365
5 0.02762865      4 0.5117193 0.5169569 0.006146101
6 0.02396229      5 0.4840906 0.4971848 0.006118026
7 0.01414168      6 0.4601283 0.4736153 0.006074986
8 0.01361791      7 0.4459866 0.4582951 0.006041328
9 0.01000000      8 0.4323687 0.4433678 0.006004144
```

Further, to avoid overfitting, we plot prun tree at cp=0.016524813. After pruning tree, rules of classification are as follows:



```
C          1   2   3  4  5   6   7
     1 [.56 .37 .00 .00 .01 .00 .07] when Elevation is  0.52 to  1.17
     3 [.00 .00 .45 .16 .00 .40 .00] when Elevation <  -0.87 & Hillshade_9am <  -0.69
     3 [.00 .10 .45 .05 .10 .30 .00] when Elevation is -0.87 to -0.16 & Soil_Type10 is 0
     4 [.00 .00 .16 .72 .00 .11 .00] when Elevation <  -0.87 & Hillshade_9am >= -0.69
     5 [.08 .26 .04 .00 .56 .06 .00] when Elevation is -0.16 to  0.52
     6 [.00 .03 .21 .01 .00 .74 .00] when Elevation is -0.87 to -0.16 & Soil_Type10 is 1
     7 [.21 .04 .00 .00 .00 .00 .74] when Elevation >= 1.17
```

Plotting rpart.plot for prune tree

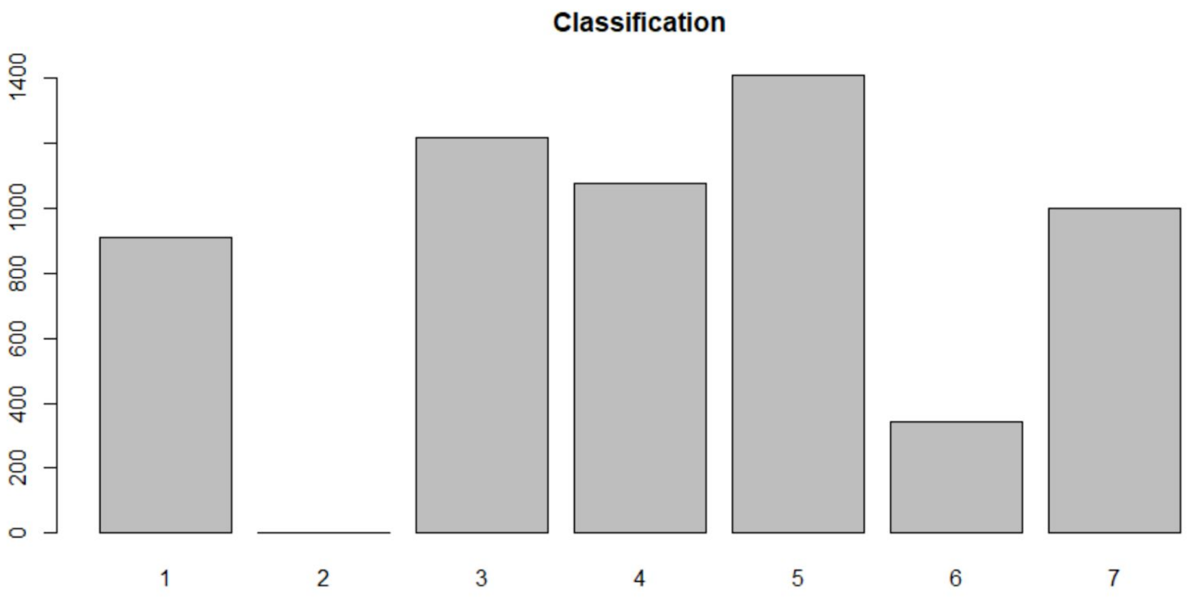


Here, we can note a lesser number of leaf nodes compare to the unpruned tree.

Confusion matrix for prune tree is as follow:

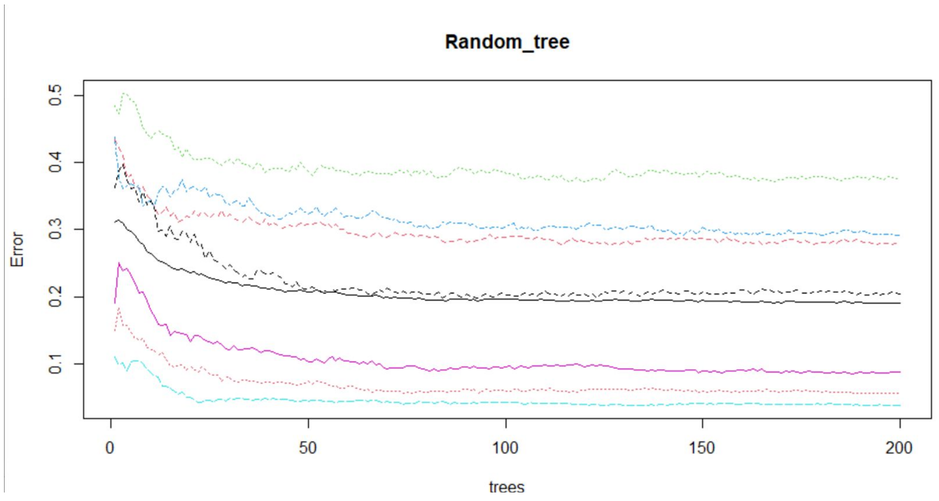
```
 predict_classification
     1   2   3   4   5   6   7
  1 528   0   2   0 105   3 207
  2 298   0  81   0 407  12  29
  3   0   0 551 182  57  74   0
  4   0   0 106 758   0   0   0
  5  13   0  67   0 771   3   0
  6   0   0 411 138  64 251   0
  7  70   0   0   0   4   0 763
```

Accuracy of prune tree is 60.82%
Here, we note that accuracy decreases by pruning tree.
For prune tree, classification done for cover_type for test data is as follows:



Random Forest

We build a random forest using training data and set 200 as the number of trees. Following graph displays error for 1:200 number of trees.



Now, we classify cover_type for test data using a trained random forest.

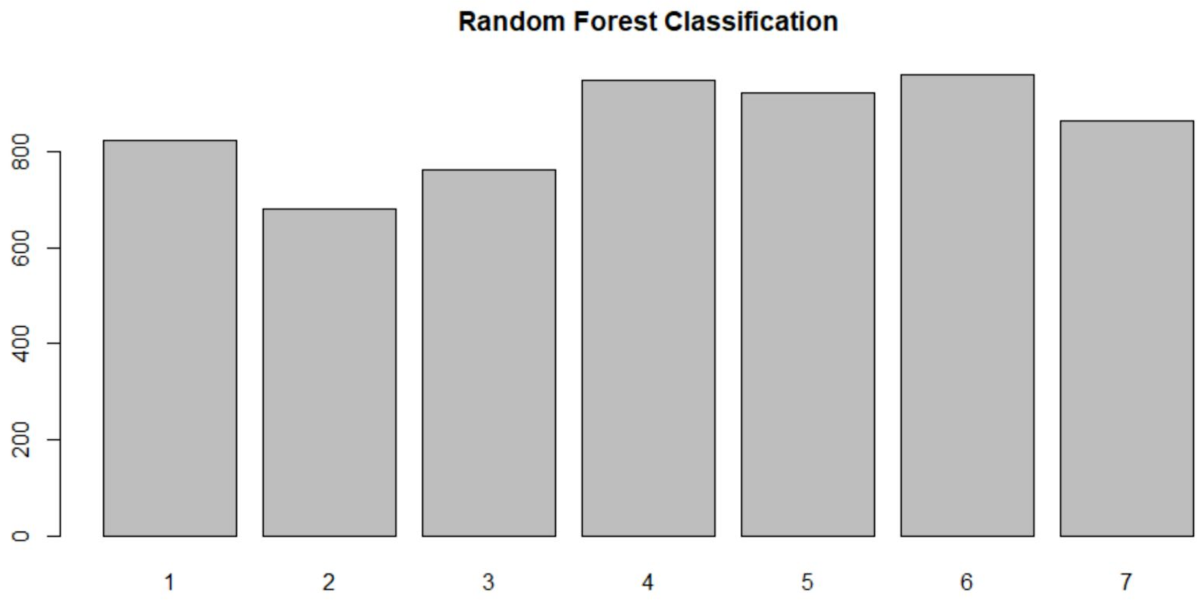
Confusion matrix for classification is as follows:

```
        predict_randomforest
             1   2   3   4   5   6   7
```

```
1 616 125   0   0  28   5  71
2 158 527  13   0  78  44   7
3   0   0 611  72   7 174   0
4   0   0  10 833   0  21   0
5   3  24   9   0 798  20   0
6   0   1 117  42  10 694   0
7  46   3   2   0   1   0 785
```

Accuracy of the random forest: 81.67%

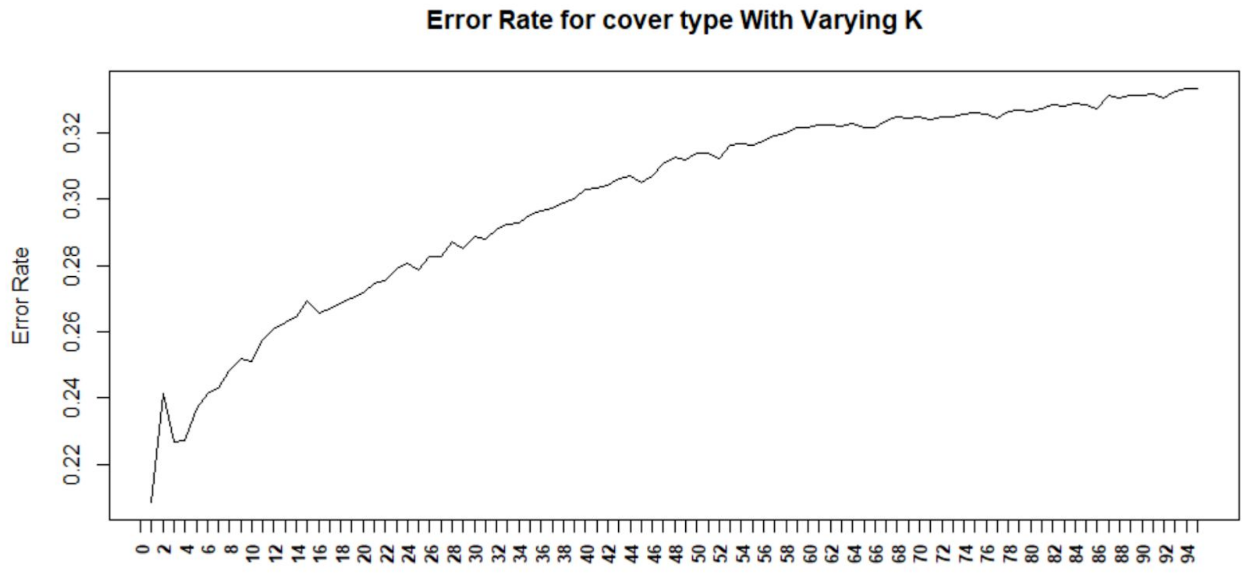
Random forest classification for test data is as follows:



KNN

Considering the value of K should not be very large enough to create overfitting and should not be very small to produce noise. We will use the square root of the total number of training data records to run KNN algorithm on. Here, 95 is sqrt(length of training data records). Hence, we will run the KNN algorithm from 1:95. We calculate the error for each k which classified cover type for test data.

Graph of error v/s K value is as follows:



Here we can see the deflection of error at k=4, k=7, and k=16. Now, we will find accuracy for classification using these three k values.

Confusion Matrix for k=4 is as follow:

```
    1   2   3   4   5   6   7
1 524 159   2   0  54  13  93
2 174 447  22   1 118  50  15
3   1   5 566  74  32 186   0
4   0   0  27 808   0  29   0
5  14  29   9   0 784  18   0
6   2   8 138  48  17 651   0
7  37  12   0   0   5   0 783
```

Confusion Matrix for k=7 is as follow:

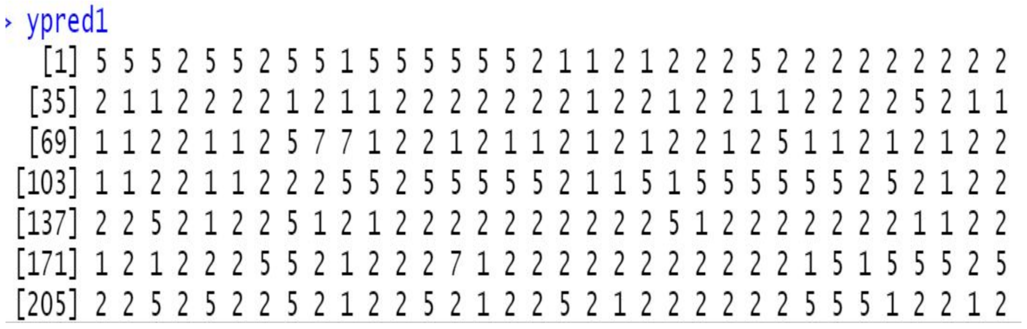
```
    1   2   3   4   5   6   7
1 512 148   3   0  62  19 101
2 166 433  24   2 131  53  18
3   0   4 530  89  29 212   0
4   0   0  16 816   0  32   0
5   7  25  11   0 784  27   0
6   0   2 127  52  20 663   0
7  48   7   0   0   8   0 774
```

Confusion Matrix for k=16 is as follow:

```
    1   2   3   4   5   6   7
1 495 143   4   0  72  23 108
2 166 407  22   1 147  57  27
3   0   5 484 115  35 225   0
4   0   0  22 819   0  23   0
5  14  31  15   0 758  35   1
6   0   2 135  69  19 639   0
7  47  12   0   0  12   0 766
```

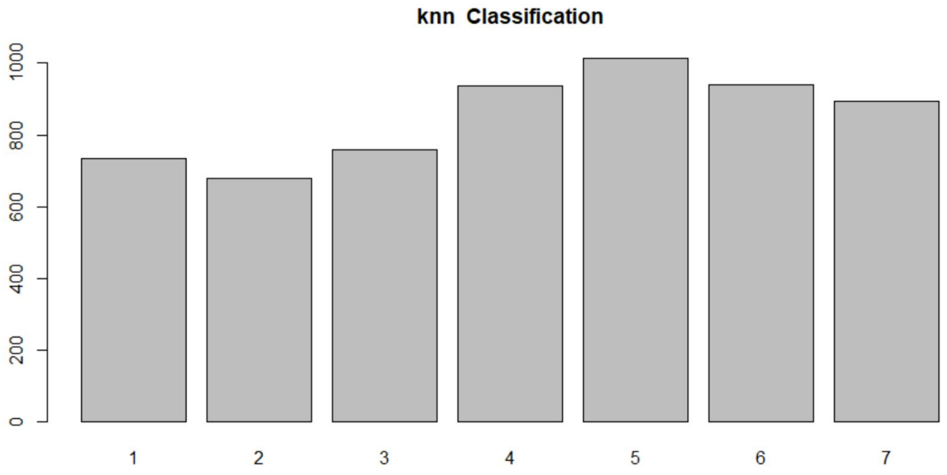
Accuracy for k=4 is **76.62%**
Accuracy for k=7 is **75.76%**
Accuracy for k=16 is **73.35%**

Hence choosing, K=4 will be efficient here. Output for KNN with K=4 is as follows:



```
> ypred1
  [1] 5 5 5 2 5 5 2 5 5 1 5 5 5 5 5 5 2 1 1 2 1 2 2 2 5 2 2 2 2 2 2 2 2 2
 [35] 2 1 1 2 2 2 2 1 2 1 1 2 2 2 2 2 2 2 1 2 2 1 2 2 1 1 2 2 2 2 5 2 1 1
 [69] 1 1 2 2 1 1 2 5 7 7 1 2 2 1 2 1 1 2 1 2 1 2 2 1 2 5 1 1 2 1 2 1 2 2
[103] 1 1 2 2 1 1 2 2 2 5 5 2 5 5 5 5 5 2 1 1 5 1 5 5 5 5 5 5 2 5 2 1 2 2
[137] 2 2 5 2 1 2 2 5 1 2 1 2 2 2 2 2 2 2 2 2 2 5 1 2 2 2 2 2 2 2 1 1 2 2
[171] 1 2 1 2 2 2 5 5 2 1 2 2 2 7 1 2 2 2 2 2 2 2 2 2 2 2 2 1 5 1 5 5 5 2 5
[205] 2 2 5 2 5 2 2 5 2 1 2 2 5 2 1 2 2 5 2 1 2 2 2 2 2 2 5 5 5 1 2 2 1 2
```

For K=4, Classified records for respective cover_type is as follows:



Conclusion

After considering various assumptions and implementing different methods, we as a team were able to infer that the Random Forest was a better model in comparison with k Nearest Neighbour(kNN) and Classification and Regression Tree (CART). We identified problematic data and processed it to achieve results giving a higher accuracy rate to determine the best output amongst the varied methods in supervised learning. The accuracy rate is 82.14% for Random forest,60.82% for CART, and approximately 76% for kNN. As we can see, Random Forests are simpler to train as a model and is a crucial model to solve real-world problems.

Through the process of research and implementation of the study, we learned that every step is of crucial importance in building a strong output with supporting facts and evidence. We believe that sharing the data and comparing the results of different models helps in making responsible recommendations. It gives way to assess and better the model implementation by resolving the incompleteness in a dataset.

Appendix

```
# Data Preprocessing
rm(list=ls())


# Importing the dataset
dataset =read.csv('C:/Users/Ami
Thakkar/Downloads/forest-cover-type-prediction/train.csv', head=T, stringsAsFactors=F,
na.strings='')
#removing id,Soil_Type_7 and Soil_Type_15
dataset=dataset[,c(-1,-22,-30)]

#view(dataset)
length(boxplot(dataset$Slope)$out) #understand more

#check and delete outliers
i = 1
dataset[,1:53] = lapply(dataset[,1:53],as.numeric)
vizu = round(cor(dataset),1) #why does it give a warnng message? --> In cor(dataset) :
the standard deviation is zero
library(ggcorrplot)
#checking for correlation
ggcorrplot(vizu)

#function for finding outliers
while(i <= ncol(dataset)){


  thirdQuantile = quantile(dataset[,i],0.75)
  upperOutlier = (IQR(dataset[,i]) * 3) + thirdQuantile
  #print(upperOutlier)
  x = length(which(dataset[,i]>upperOutlier))



  firstQuantile = quantile(dataset[,i],0.25)
  lowerOutlier =  firstQuantile -(IQR(dataset[,i])* 3)
  #print(lowerOutlier)
  y = length(which(dataset[,i] < lowerOutlier))

  print(paste(x+y," is the number of outliers in ", colnames(dataset)[i]))

  i = i+1

}
thirdQuantile = quantile(dataset$Horizontal_Distance_To_Hydrology,0.75)
upperOutlier = (IQR(dataset$Horizontal_Distance_To_Hydrology) * 3) + thirdQuantile
firstQuantile = quantile(dataset$Horizontal_Distance_To_Hydrology,0.25)
lowerOutlier =  firstQuantile -(IQR(dataset$Horizontal_Distance_To_Hydrology)* 3)

removeUpper = which(dataset$Horizontal_Distance_To_Hydrology > upperOutlier)
removeLower = which(dataset$Horizontal_Distance_To_Hydrology < lowerOutlier)
#removing upper outlier for Horizontal_Distance_To_Hydrology
if(length(removeUpper)>0){
  dataset = dataset[-removeUpper,]
}
#removing lower outlier for Horizontal_Distance_To_Hydrology
if(length(removeLower)>0){
  dataset = dataset[-removeLower,]

}

thirdQuantile = quantile(dataset$Horizontal_Distance_To_Roadways,0.75)
upperOutlier = (IQR(dataset$Horizontal_Distance_To_Roadways) * 3) + thirdQuantile
firstQuantile = quantile(dataset$Horizontal_Distance_To_Roadways,0.25)
lowerOutlier =  firstQuantile -(IQR(dataset$Horizontal_Distance_To_Roadways)* 3)
removeUpper = which(dataset$Horizontal_Distance_To_Roadways > upperOutlier)
removeLower = which(dataset$Horizontal_Distance_To_Roadways < lowerOutlier)
#removing upper outlier for Horizontal_Distance_To_Roadways
```

```
if(length(removeUpper)>0){
  dataset = dataset[-removeUpper,]
}
#removing lower outlier for Horizontal_Distance_To_Roadways
if(length(removeLower)>0){
  dataset = dataset[-removeLower,]

}

thirdQuantile = quantile(dataset$Horizontal_Distance_To_Fire_Points,0.75)
upperOutlier = (IQR(dataset$Horizontal_Distance_To_Fire_Points) * 3) + thirdQuantile
firstQuantile = quantile(dataset$Horizontal_Distance_To_Fire_Points,0.25)
lowerOutlier =  firstQuantile -(IQR(dataset$Horizontal_Distance_To_Fire_Points)* 3)
removeUpper = which(dataset$Horizontal_Distance_To_Fire_Points > upperOutlier)
removeLower = which(dataset$Horizontal_Distance_To_Fire_Points < lowerOutlier)
#removing upper outlier for Horizontal_Distance_To_Fire_Points
if(length(removeUpper)>0){
  dataset = dataset[-removeUpper,]
}
#removing lower outlier for Horizontal_Distance_To_Fire_Points
if(length(removeLower)>0){
  dataset = dataset[-removeLower,]

}

thirdQuantile = quantile(dataset$Vertical_Distance_To_Hydrology,0.75)
upperOutlier = (IQR(dataset$Vertical_Distance_To_Hydrology) * 3) + thirdQuantile
firstQuantile = quantile(dataset$Vertical_Distance_To_Hydrology,0.25)
lowerOutlier =  firstQuantile -(IQR(dataset$Vertical_Distance_To_Hydrology)* 3)
removeUpper = which(dataset$Vertical_Distance_To_Hydrology > upperOutlier)
removeLower = which(dataset$Vertical_Distance_To_Hydrology < lowerOutlier)
#removing upper outlier for Vertical_Distance_To_Hydrology
if(length(removeUpper)>0){
  dataset = dataset[-removeUpper,]
}
#removing lower outlier for Vertical_Distance_To_Hydrology
if(length(removeLower)>0){
  dataset = dataset[-removeLower,]
}
#checking dimensions for cleaned data
dim(dataset)

#scaling cleaned dataset
dataset[,c(1:10)]= scale(dataset[,c(1:10)])
rownames(dataset) = seq.int(nrow(dataset))
#importing libraries for CART and Random forest
library(caTools)
library(rpart)
library(rpart.plot)
library(caret)
library(randomForest)

#splitting dataset into test and train
set.seed(123)
splitting = sample.split(dataset$Cover_Type , SplitRatio = 0.6)
dataset_train = subset(dataset , splitting == TRUE)
dataset__test  = subset(dataset ,splitting == FALSE)
#applying CART on train data
classtree = rpart(Cover_Type ~., method = "class", data = dataset_train )
#predict full tree
predict_fulltree= predict(classtree , newdata = dataset__test, type = "class")
#confusion matrix for predicted full tree
confusion_matrix_fulltree = table(predict_fulltree, dataset__test$Cover_Type)
#finding accuracy for full tree
Accuracy_fulltree = sum(diag(confusion_matrix_fulltree))/nrow(dataset__test)

#plotting tree using prp()
prp(classtree)
#plotting tree using plot() and text()
plot(classtree)
text(classtree)
#plotting tree using rpart.plot()
rpart.plot(classtree, box.palette = 0)
names(classtree)
#finding CP for full tree
classtree$cptable
#prunning tree at lowest xerror
```

```
prune_classtree = prune(classtree , cp =0.01715333 )
#plotting prune tree using rpart.plot()
rpart.plot(prune_classtree,box.palette = 0)
#plotting prune tree using prp()
prp(prune_classtree)
#predict test data using prune tree
predict_classification = predict(prune_classtree, newdata = dataset__test, type =
"class")
#find accuracy for prune tree
confusion_matrix_class=table(dataset__test$Cover_Type, predict_classification)
confusion_matrix_class
Accuracy = sum(diag(confusion_matrix_class))/nrow(dataset__test)
Accuracy
#classification of test data with prune tree
plot(predict_classification, main ="Classification")

#plotting random forest with nt=200
Random_tree = randomForest(factor(Cover_Type) ~. , data =dataset_train,ntree=200 )
plot(Random_tree)
#factoring Cover_Type
dataset__test$Cover_Type = as.factor(dataset__test$Cover_Type)
#predict test data with random forest
predict_randomforest = predict(Random_tree , newdata = dataset__test)
#finding accuracy for RAndom forest
confusion_matrix_RForest=table(dataset__test$Cover_Type , predict_randomforest)
confusion_matrix_RForest
Accuracy_RF = sum(diag(confusion_matrix_RForest))/nrow(dataset__test)
Accuracy_RF
#summary for predicted random forest
summary(predict_randomforest)
#plotting predicted random forest
plot(predict_randomforest, main = "Random Forest Classification")
 ##
probs= treeres

#KNN
#divide test and train data based on outcome variable
library(class)
Xtrain = dataset_train[1:52]
```

```
Xtest = dataset__test[1:52]
ytrain = dataset_train[53]
ytest = dataset__test[53]
#try k values for sqrt(train)
klen=round(sqrt(nrow(Xtrain)),0)

tpredict<-numeric() #Holding variable
for(j in 1:klen){
  #Apply knn with k = i
  ypred<-knn(Xtrain,Xtest,ytrain$Cover_Type,k=j)
  #finding accurately predicted values
  tpredict<-c(tpredict,
            mean(ypred==ytest$Cover_Type))
}
#validation error
error<-1-tpredict
#plotting validation error with K values
plot(1-tpredict,type="l",ylab="Error Rate",xlab="K",xaxt="n",main="Error Rate for cover
type With Varying K")
axis(1, seq(0,klen, 1),las=2, font=2,cex.axis=0.8)

#predicting results and accuracy for k=4
ypred1 = knn(Xtrain, Xtest, ytrain$Cover_Type, k=4, prob=T)
cm1=as.matrix(table(ytest$Cover_Type, ypred1)) #confusion matrix
AccuracyK4=sum(diag(cm1))/length(ytest$Cover_Type) #accuracy

#predicting results and accuracy for k=12
ypred2 = knn(Xtrain, Xtest, ytrain$Cover_Type, k=12, prob=T)
cm2=as.matrix(table(ytest$Cover_Type, ypred2))
Accuracyk12=sum(diag(cm2))/length(ytest$Cover_Type)

#predicting results and accuracy for k=16
ypred3 = knn(Xtrain, Xtest, ytrain$Cover_Type, k=16, prob=T)
cm3=as.matrix(table(ytest$Cover_Type, ypred3))
Accuracyk16=sum(diag(cm3))/length(ytest$Cover_Type)

#test data classification with k=4
plot(ypred1, main = "knn  Classification")
```

References

k-Nearest Neighbor Classification." *A Course and Community Designed to Take You from Computer Vision Beginner to Guru.*, gurus.pyimagesearch.com/lesson-sample-k-nearest-neighbor-classification/.

Imran. "How to Choose the Value of K in Knn Algorithm." *Data Science, Analytics and Big Data Discussions*, 2 Aug. 2015, discuss.analyticsvidhya.com/t/how-to-choose-the-value-of-k-in-knn-algorithm/2606.

Rotter, Charles, et al. "Predicting Climate Change." *Watts Up With That?*, 16 Feb. 2019, wattsupwiththat.com/2019/02/16/predicting-climate-change/.

Kim, So-Ra, et al. "Forest Cover Classification by Optimal Segmentation of High Resolution Satellite Imagery." *Sensors (Basel, Switzerland)*, Molecular Diversity Preservation International (MDPI), 2011, www.ncbi.nlm.nih.gov/pmc/articles/PMC3274007/.

Ghebrezgabher, Mihretab G., et al. "Extracting and Analyzing Forest and Woodland Cover Change in Eritrea Based on Landsat Data Using Supervised Classification." *The Egyptian Journal of Remote Sensing and Space Science*, Elsevier, 28 Jan. 2016, www.sciencedirect.com/science/article/pii/S111098231500037X.

"Forest Cover Type Prediction." *Kaggle*, www.kaggle.com/c/forest-cover-type-prediction.