MTH552A COURSE PROJECT:
FOREST COVER TYPE CLASSIFICATION

Classification of Forest Cover type using various Data Mining Techniques

Submitted by:

Abheet Aggarwal, 12012

Md. Enayat Ullah, 12407

Jitender Singh Shekhawat, 12324

Rounak Poddar, 12591

Tathagat Gupta, 12760

Zahid Sharief, 12836

Submitted to:

Dr. Amit Mitra,

Dept. of Mathematics and Statistics

Statement of the Problem

The task to predict the forest cover type (the predominant kind of tree cover) from strictly cartographic variables (as opposed to remotely sensed data). 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. This 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.

Dataset

The study area includes four wilderness areas located in the Roosevelt National Forest of northern Colorado. Each observation is a 30m x 30m patch. We would be predicting an integer classification for the forest cover type. The seven types are:

1 - Spruce/Fir

2 - Lodgepole Pine

3 - Ponderosa Pine

4 - Cottonwood/Willow

5 - Aspen

6 - Douglas-fir

7 - Krummholz

The training set (15120 observations) contains both features and the Cover_Type.

Data Description:

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 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

CLASSIFICATION TECHNIQUES

To complete the classification task we have used 4 different classification techniques and compared the accuracy of the techniques.

Multinomial Logistic Regression:

Multinomial logistic regression is a classification method that generalizes logistic regression to multiclass problems, i.e. with more than two possible discrete outcomes. That is, it is a model that is used to predict the probabilities of the different possible outcomes of a categorically distributed dependent variable, given a set of independent variables.

In this problem we are required to classify the Forest cover type into 7 categories.
The log odds of the outcomes i.e. the forest type are modeled as a linear combination of the predictor variables.

The final regression model:
Cover_Type~
Elevation+Aspect+Slope+Horizontal_Distance_To_Hydrology+Vertical_Distance_To_Hydrology+Horizontal_Distance_To_Roadways+Hillshade_Noon+Horizontal_Distance_To_Fire_Points+Soil_Type1

Hillshade_9am, Hillshade_Noon and Hillshade_3pm were highly correlated so, Hillshade_9am and Hillshade_3pm were eliminated from the model.

The estimated coefficients of the fitted model are given in the Appendix.

We calculated the corresponding probability of each test vector lying in each of the Forest Type by :

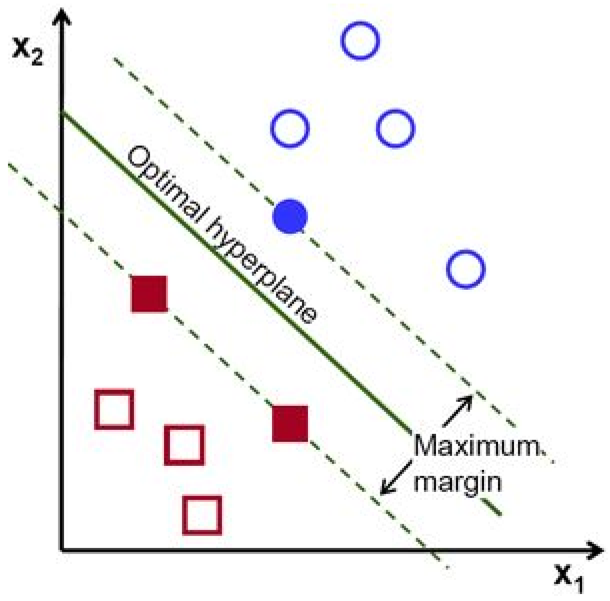
$$\Pr(Y_i = 1) = \frac{e^{\boldsymbol{\beta}_1 \cdot \mathbf{X}_i}}{1 + \sum_{k=1}^{K-1} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}}$$
$$\Pr(Y_i = 2) = \frac{e^{\boldsymbol{\beta}_2 \cdot \mathbf{X}_i}}{1 + \sum_{k=1}^{K-1} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}}$$
$$\cdots\cdots$$
$$\Pr(Y_i = K-1) = \frac{e^{\boldsymbol{\beta}_{K-1} \cdot \mathbf{X}_i}}{1 + \sum_{k=1}^{K-1} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}}$$
$$\Pr(Y_i = K) = \frac{1}{1 + \sum_{k=1}^{K-1} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}}$$

K=7 in our case.

Final Outcome: A 5-fold cross validation on the 15120 training samples is performed, and the accuracy obtained using a logistic regression classifier is about 0.74.

Support Vector Machines:

Support Vector Machine is a supervised learning model which builds a linear classifier to segregate the data into two classes. Given p-dimensional feature vector data points, SVM constructs a p-1 dimensional hyperplane to separate the data points. Since there are many hyperplanes that might classify the data, the reasonable choice is to select the hyperplane represents the largest separation, or margin, between the two classes. The data points lying closest to the margin are called support vectors.



Source: http://docs.opencv.org/

For a linearly non-separable data, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. However, instead of explicitly computing the coordinates in the high-dimensional feature space, the inner products between the images of all pairs of data in the feature space is computated, which is computationally cheaper. The inner product is called a kernel, which essentially is similarity function over pairs of data points in the projected space.

Some popular choice of kernels used in SVM are:

1. Linear Kernel:

$$k(x, y) = x^T y + c$$

2. Polynomial Kernel:

$$k(x, y) = (\alpha x^T y + c)^d$$

3. Sigmoid Kernel:

$$k(x, y) = \tanh(\alpha x^T y + c)$$

4. Gaussian Kernel:

$$k(x, y) = \exp\left(-\frac{||x - y||^2}{2\sigma^2}\right)$$

Multi-class SVM:

An SVM is a binary classifier, separating the data into two labels(0,1). However, many real-world problems involve classification into more than two classes. In our problem definition, we need to classify the data into 7 classes. The most common approach employed for a multi-class classification is One-Versus the Rest method. For the one-versus-the-rest approach, classification is done by a max-wins voting strategy, in which every classifier assigns the instance to one of the two classes, then the vote for the assigned class is increased by one vote, and finally the class with the most votes determines the instance classification.

Result:

A 5-fold cross validation on the 15120 training samples is performed, and following are the results obtained using different kernels:

Kernel Type	Linear	Polynomial	Sigmoid	Radial
Accuracy	0.686	0.701 (d=2) 0.752 (d=3) 0.725 (d=4)	0.407	0.756

Naive-Bayes Classifier

Naive Bayes Classifier is a simple probabilistic Bayesian classifier with strong (naive) independence assumptions between the features.

Given an n dimensional feature vector *x*, to be classified into *j* classes. The class conditional independence is given as follows:

$$p(x|cj) = p(x1 \, |cj\,) * p(x2 \, |cj\,) * \ldots * p(xn \, |cj\,)$$

This assumption reduces the bayesian classifier equation to the following:

$$p(C_k|x_1,\ldots,x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^{n} p(x_i|C_k)$$

where *Z=P(x)* is the normalization constant.

The decision rule to pick the hypothesis that is most probable(also known as maximum a posteriori or MAP) is a Bayes classifier, and the function that assigns a class label $\hat{y} = C_k$ for some k is as follows:

$$\hat{y} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} \ p(C_k) \prod_{i=1}^{n} p(x_i|C_k).$$

Result:

A 5-fold cross validation on the 15120 training samples is performed, and the accuracy obtained using a naive-Bayes classifier is 0.602.

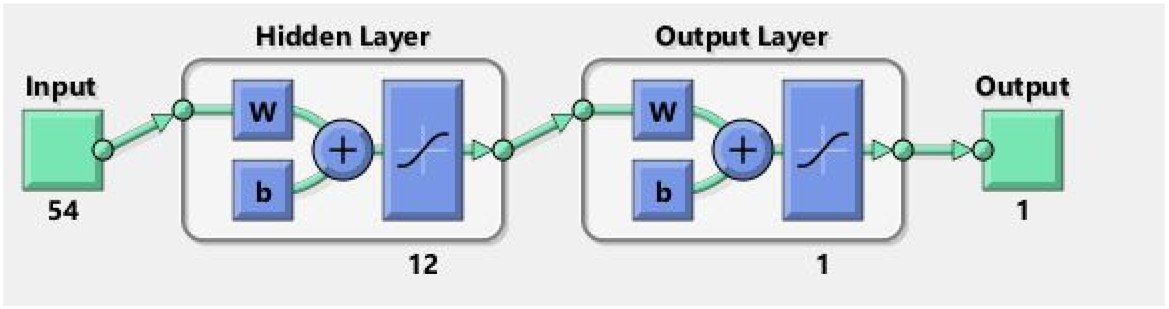
Classification by Neural Networks

Artificial neural networks are generally presented as systems of interconnected "neurons" which can compute values from inputs, and are capable of machine learning as well as pattern recognition thanks to their adaptive nature.
In neural networks, we have three types of layers:

1) Input layer
2) Hidden layers
3) Output layer

Data is sent via neurons from input layer to hidden layer and then from hidden layer to output layer. This structure is fully connected. The parameters attached to these neurons are known as weights. Through training, the network learns these weights and then it uses these updated weights to estimate the output of test data.

Architecture:



Procedure and Result:

We applied a neural network structure consisted of a input layer , one hidden layer and an output layer. We tried for three different number of nodes in the hidden layer.

No of layers	No of nodes	Error	epoch
1	8	18.83 %	1000
1	12	15.20 %	1000
1	16	16.07 %	1000

We found that for these three, the error is minimum for nodes 12 in the hidden layer. As you can see that for nodes 16 the error is increasing , that might be due to some overfitting.

Transfer function: We used tansig function as transfer function.

<u>K-Nearest Neighbour approach</u>

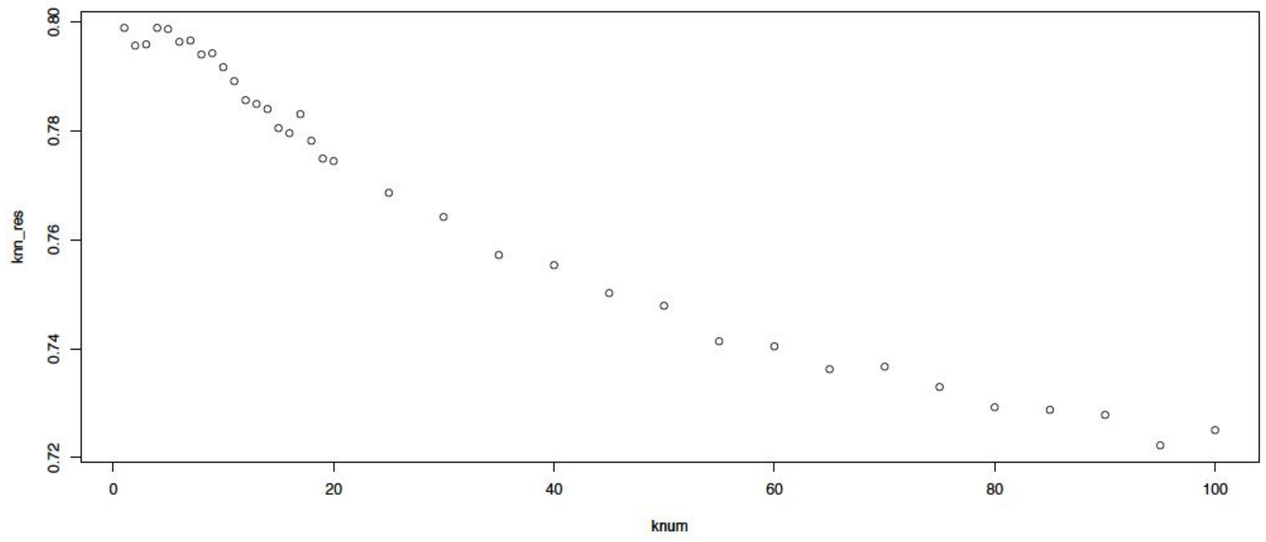
We have used the k-NN algorithm which uses the euclidean distance to get the k nearest neighbour.

Further, the test vector is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbours. We have plotted the accuracy vs the value of k which concludes that the accuracy falls by increasing the value of k.

A shortcoming of the k-NN algorithm is that it is sensitive to the local structure of the data.

A small value of k means that noise can have a higher influence on the result.

Result: A 5-fold cross validation on the 15120 training samples is performed, and the accuracy obtained using a K-Nearest Neighbour classification Approach is 0.795.



Random Forest and CART

Random forests (ensemble learning method) is used for classification which operates by constructing a multitude of decision trees at training time and outputting the class that is the majority of the classes (classification) of the individual trees.
Bagging is used for creating different learning sets which in turn generate different classification trees.
Each learning set is created by randomly picking up vectors from the original learning set with replacement.
On an average, each learning set of the tree consist of about 63% of vectors from the original learning set.

It reduces variance and helps to avoid overfitting.

Mtry parameter is kept four, which randomly picks up m number of attributes and splitting at each node is done exclusively by those attributes.

Random forests correct for decision trees' habit of overfitting to their training set.

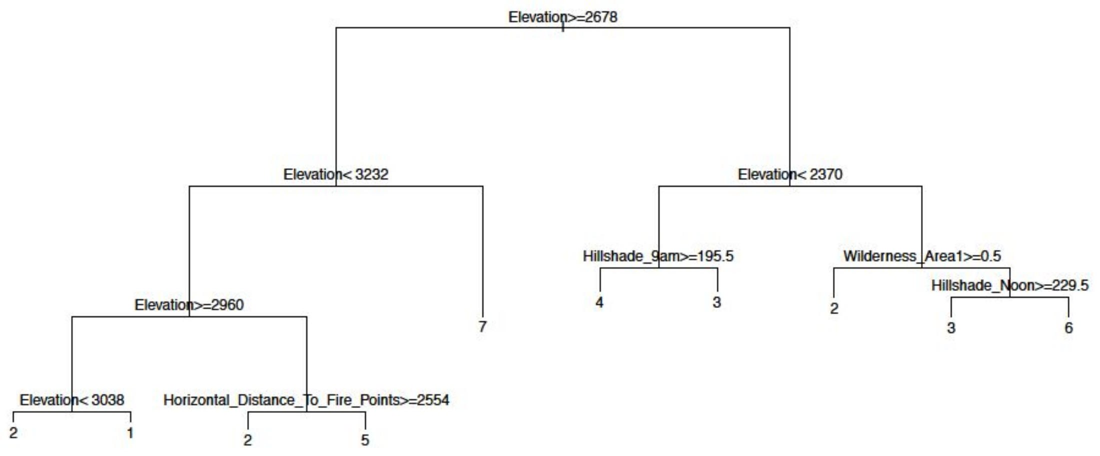
Result:

Random Forest:

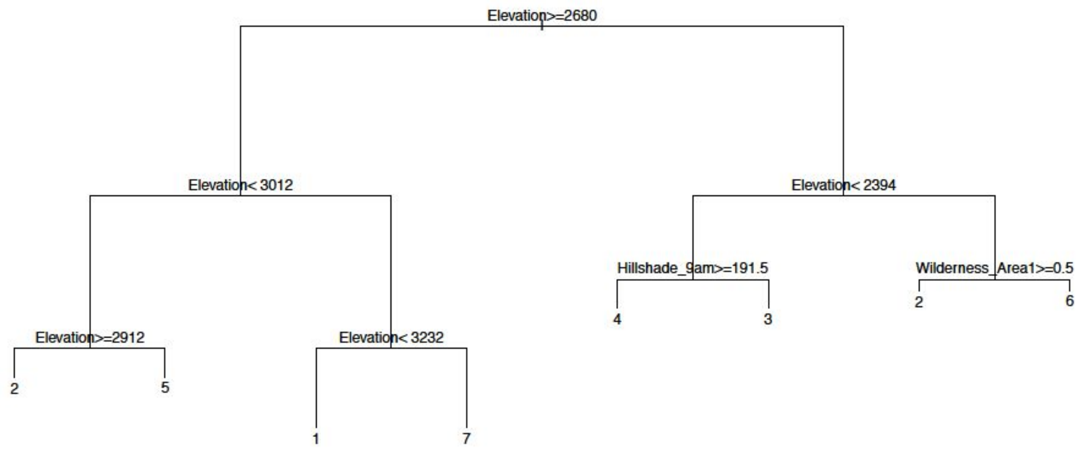
A plot of accuracy vs number of trees is plotted which shows that as the number of trees are increased the accuracy increases monotonically and finally becomes stable at about 87%.The out of bag error (a good approximation to the generalisation error) of the forest comes out to be 13.24% for m=4 and number of trees as 100.

CART:

1. Ginni Index: Accuracy: 64.656%



2. Information Index: 63.452%



CONCLUSION

We used seven classification methods.

The most accurate classification of the validation set was achieved by the **Random Forest** Classifier. The highest classification accuracy of about **87%** was obtained.

Then the next best result was shown by **Neural Network** with an accuracy of **84.8** % when the we took 12 nodes in the hidden layer.

K-NN gives an accuracy of **79.5** %.

SVM with radial function showed some good results and the accuracy came out to be **75.6** % .

Multinomial Logistic Regression also found out be useful with the accuracy of **74** %.

The Least accurate classification is done by the **Bayes Classifier**- **60.4**%.

Appendix

Logistic Regression Call:

mlogit (formula = Cover_Type ~ 1 | Elevation + Aspect + Slope + Horizontal_Distance_To_Hydrology + Vertical_Distance_To_Hydrology + Horizontal_Distance_To_Roadways + Hillshade_Noon + Horizontal_Distance_To_Fire_Points + Soil_Type1, data = long0, method = "nr", print.level = 0)

Coefficients:

```
          2:(intercept)            3:(intercept)            4:(intercept)
            2.1258e+01               8.5736e+01               1.0646e+02
          5:(intercept)            6:(intercept)            7:(intercept)
            3.6515e+01               9.0506e+01              -4.9594e+01
           2:Elevation              3:Elevation              4:Elevation
           -9.7437e-03              -3.2976e-02              -4.3677e-02
           5:Elevation              6:Elevation              7:Elevation
           -1.4993e-02              -3.2788e-02               1.3574e-02
             2:Aspect                 3:Aspect                 4:Aspect
           -1.2327e-03               1.6354e-03              -5.0481e-03
             5:Aspect                 6:Aspect                 7:Aspect
           -1.2093e-03               3.3240e-03              -1.9117e-03
              2:Slope                  3:Slope                  4:Slope
            3.0888e-02               9.8767e-02               5.5050e-02
              5:Slope                  6:Slope                  7:Slope
            6.2469e-02               4.3356e-02               1.1273e-02
2:Horizontal.._Hydrology 3:Horizontal.._Hydrology 4:Horizontal.._Hydrology
            2.2695e-03               2.5539e-03              -3.5964e-03
5:Horizontal.._Hydrology 6:Horizontal.._Hydrology 7:Horizontal_..Hydrology
            6.3864e-04               1.5651e-03              -1.4474e-03
```

```
  2:Vertical_..Hydrology     3:Vertical_..Hydrology     4:Vertical_..Hydrology
              1.2707e-03                 1.5619e-02                 2.6361e-02
  5:Vertical_..Hydrology     6:Vertical_..Hydrology     7:Vertical_..Hydrology
              6.4928e-03                 1.2785e-02                -8.9933e-04
 2:Horizontal_..Roadways    3:Horizontal_..Roadways    4:Horizontal_..Roadways
              1.0766e-04                 6.8475e-04                 1.6727e-03
 5:Horizontal_..Roadways    6:Horizontal_..Roadways    7:Horizontal_..Roadways
             -2.1534e-04                 9.0959e-04                -3.6493e-05
        2:Hillshade_Noon           3:Hillshade_Noon            4:Hillshade_Noon
              2.8914e-02                 2.2626e-02                 4.3485e-02
        5:Hillshade_Noon           6:Hillshade_Noon            7:Hillshade_Noon
              3.2529e-02                -3.0498e-03                 3.6797e-03
2:Horizontal_..Fire_Points  3:Horizontal_._Fire_Points  4:Horizontal_..Fire_Points
             -4.0421e-05                -1.1584e-03                -3.0768e-04
5:Horizontal_..Fire_Points  6:Horizontal_..Fire_Points  7:Horizontal_..Fire_Points
             -4.1242e-04                -8.2965e-04                 1.7001e-04
             2:Soil_Type1               3:Soil_Type1               4:Soil_Type1
              2.4593e-02                -1.4975e-01                -1.3967e-01
             5:Soil_Type1               6:Soil_Type1               7:Soil_Type1

              1.5049e-02                -4.6378e-02                 1.5033e-01
```

Support Vector Machine Call:

```
sv=svm(x=cv.train[,-13],y=as.factor(cv.train[,13]),kernel="polynomial",degree=4)
```

Naive-Bayes Classifier:

Call:

```
nb=naiveBayes(x=cv.train[,-13],y=as.factor(cv.train[,13]))
```

A-priori probabilities:

1	2	3	4	5	6	7
0.1417824	0.1432705	0.1441799	0.1402116	0.1440146	0.1419478	0.1445933

Conditional Probabilities: Mean and SD of all the parameters

Elevation

	Mean	SD
1	3129.094	156.27260
2	2920.973	185.76293
3	2395.972	191.48719
4	2221.857	103.15956
5	2786.656	96.67337
6	2422.627	188.45529
7	3363.316	109.79736

Aspect

	Mean	SD
1	160.1155	117.77899
2	149.5534	106.84652
3	174.6938	107.98593
4	139.4292	89.54789
5	138.3209	91.76898
6	179.6255	132.97943
7	155.2653	111.36028

Slope

	Mean	SD
1	13.16268	6.792886
2	13.44662	7.006526
3	20.60608	8.844950
4	18.63149	9.295753
5	16.80195	8.492476
6	18.87944	7.736031
7	14.05260	7.181496

Horizontal_Distance_To_Hydrology

	Mean	SD
1	271.7271	222.4067
2	287.3087	214.0645
3	211.3859	141.5999

4	102.1415	136.3383
5	212.1791	182.0374
6	159.7175	125.6346
7	351.0343	296.7857

Vertical_Distance_To_Hydrology

	Mean	SD
1	41.73120	55.98027
2	47.54357	59.71279
3	65.04530	59.20445
4	39.63090	57.90050
5	50.98393	58.88641
6	44.15842	46.46740
7	69.51001	80.84529

Horizontal_Distance_To_Roadways

	Mean	SD
1	2569.6006	1498.4438
2	2422.4951	1640.4135
3	971.2838	614.5343
4	921.4393	358.2531
5	1330.5718	1038.2198
6	1077.2073	579.7967
7	2724.7444	1208.9704

Hillshade_9am

	Mean	SD
1	211.7743	25.14357
2	214.2256	24.98629
3	201.5281	40.33456
4	227.9133	24.31024
5	223.4277	22.51984
6	194.1846	33.78718
7	216.4631	23.30654

Hillshade_Noon

	Mean	SD
1	223.0612	18.07482
2	225.1477	18.31833
3	216.7437	27.26454
4	216.7913	20.97719

5	218.0918	26.04925
6	210.0513	23.85200
7	222.5798	19.05896

Hillshade_3pm

	Mean	SD
1	143.7353	36.37265
2	142.5118	35.13904
3	141.9151	51.89264
4	111.5896	49.18671
5	120.9369	50.39378
6	147.1194	45.36063
7	136.6747	37.09484

Horizontal_Distance_To_Fire_Points

	Mean	SD
1	2007.5073	1257.8867
2	2163.5366	1412.8824
3	915.3796	533.2196
4	856.4233	482.7984
5	1533.8628	960.5727
6	1060.6546	574.5753
7	2074.4334	1094.1840

Wilderness_Area1

	Mean	SD
1	1.917784	0.9507743
2	1.931333	0.9956106
3	3.610665	0.4877393
4	4.000000	0.0000000
5	2.207807	0.9784507
6	3.558532	0.4967068
7	2.393939	0.8542449

Soil_Type1

	Mean	SD
1	27.626822	6.104265
2	24.480092	8.354742
3	6.365252	3.784979
4	6.977594	5.615778
5	21.679104	9.563825

6	10.067560	6.491529
7	36.727844	4.643125