Explore Forest Cover Type Prediction with Multiple Alogorithms

Xinyue,ZHANG
HKUST
Big Data Technology
20750194
xzhangfa@connect.ust.hk

Shuyu,QIAO
HKUST
Big Data Technology
20747563
sqiaoac@connect.ust.hk

Zhuoxuan,LI
HKUST
Big Data Technology
20740917
zlify@connect.ust.hk

***Abstract*—In this project, we predict the forest cover type (the predominant kind of tree cover) from strictly cartographic variables (opposed to remotely sensed data). To finish the classification task, we first analyze the dataset to equip the future data pre-processing and application. Then we select multiple machine learning algorithms including Logistic Regresion, Random Forests, LightGBM, Decision Tree on Spark and FNN from various machine learning packages such as Sklearn, Keras and Spark MLlib , to compare their performance. In this report, the process of data analysis, data cleaning, data normalization and hyperparameter tuning will be described to show how they affect the final classification accuracy.**

***Index Terms*—Feature Engineering; Classification; Spark; Machine Learning, Neural Network.**

I. Introduction

The Forest Cover type dataset [1] in the UCI Machine Learning Repository is a famous dataset in machine learning. It takes forestry data from four wilderness areas in Roosevelt National Forest in northern Colorado. The observations are taken from 30m by 30m patches of forest that are classified as one of seven cover types: Spruce/Fir, Lodgepole Pine, Ponderosa Pine, Cottonwood/Willow, Aspen, Douglas-fir, Krummholz. Each sample contains several characteristics describing each piece of land, including altitude, slope, distance to water sources, shading conditions, and soil types, and the corresponding type of forest vegetation. The data is in raw form (not scaled) and contains both continuous and binary columns. Based on these attributes, we will be able to classify its forest cover type to the given seven classes.

In order to explore the application of machine learning classification algorithms, we applied Logistic Regression, Random Forest, LightGBM algorithm from sklearn module, and neural network classifier from Keras. After comparing the results of these algorithms, we decided to combine Spark's MLlib [3] into our classification task to see if there is any benefit as Spark excels at iterative computation. MLlib contains high-quality algorithms that leverage iteration, and can yield better results than the one-pass approximations sometimes used on MapReduce [4]. And also, MLlib contains many algorithms and utilities. Based on the spark platform, decision tree algorithm algorithm is adopted and the parameters are optimized to obtain higher precision prediction results.

The remaining parts of the paper are organized as follows: Preprocessing Works like data cleaning and data preparation as well as StandardScaler have been presented in Section II. Subsequently, several regression models from sklearn, a neural network and the Decision Tree algorithm based on spark would be completed, their performance is presented in Section III. Then, section IV contains comparison of all experiments. Finally, conclude this work with future directions in Section V. Note that member contribution is set in last section VI.

II. Preliminary Works

Given the covertype dataset, we first conduct the exploratory data analysis and data visualization to have a first glance on the dataset. It is also important to check the dataset's integrity, correctness, and characteristics for future data cleaning. Then we clean and normalize the data for further classification.

A. Data set

The data has 581011 rows of data and 56 columns in the dataset. Besides the final prediction result and id column, there are 54 columns for prediction. We can divide them into 3 categories: geometric information (elevation, aspect, slope, distance to hydrology and roadways), light and shade information (hill shade at different times), wilderness and soil information (wilderness area tag and 40 kinds of soil type). Among them, wilderness and soil is binary, while others are actual numerical value.

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*

Fig. 1. Data Set Details.

From the statistical overview of the dataset (appendix 1), we can find the following hints:

1) For each column, the count numbers are the same. This suggests each row is complete.
2) Soil type 7 and 15 are all zero, so they do not provide any information and can be removed.
3) Scales of each attribute are varied. Standardisation may be necessary under some cases. At this stage, we will (a) drop soil type 7 and 15, (b) drop ID columns in further analysis.

B. Data Visualization

Next, Fig.2. shows we conduct correlation analysis for the continuous columns. The detail number will be shown in appendix 2(correlation_analysis_data.csv) We can find the largest absolute value of coefficient is less than 0.8. For a better visualization, we set the deepest color as 0.8.

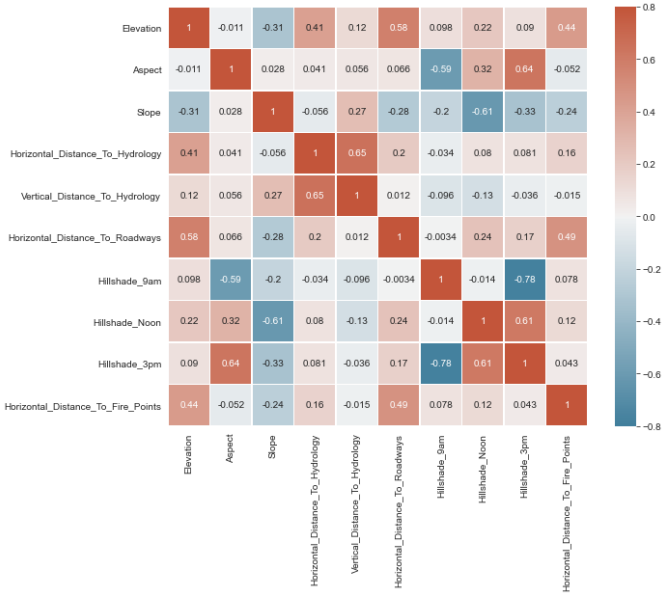


Fig. 2. Correlation analysis.

Let's set the threshold as +/- 0.5 for further filtering. If the correlation coefficient has an absolute value larger than 0.5, that means it has some correlation (either positive or negative).In summary, there are several kinds of conclusions: Hill-shade at 9am has a negative correlation with hill-shade at 3pm. (-0.78)

1) Horizontal distance to hydrology has a positive correlation with the corresponding vertical distance. (0.65)
2) Hill-shade at 3pm has a positive correlation with aspect. (0.64)
3) Hill-shade at noon has a positive correlation with hill-shade at 3pm.(0.61)
4) Hill-shade at noon has a negative correlation with slope. (-0.61)
5) Hill-shade at 9am has a negative correlation with aspect. (-0.59)
6) Horizontal distance to the roadway has a positive correlation with elevation. (0.58)

Based on the above conclusions, let's draw a scatter plot of them. These plots can give us very intuitive conclusions of classification.

C. StandardScaler

After the features are digitized, due to different values, the distance of the sample points in the feature space will be dominated by individual feature values, while the influence of other features is relatively small; for example: feature 1 = [1, 3, 2, 6, 5 , 7, 9], feature 2 = [1000, 3000, 5000, 2000, 4000, 8000, 3000], when calculating the distance between two samples in the feature space, it is mainly determined by feature 2; therefore, all data is mapped to the same scale. In the data preparation part, only the training set is directly normalized. In the evaluation part, cannot directly normalize the test data set according to the formula, but use the mean and variance of the training data set to normalize the test data set.

III. Model Training and Evaluation

A. Logistic Regression

Multinomial logistic regression [5] is chosen for this task. In this multiclass case, the cross-entropy loss is considered as the set of 'multinomial'. Liblinear library, lbfgs and netwon-cg are the three common solvers for regularizing logistic regression, but here we chose 'lbfgs' as the solver applied in this project to handle the multinomial loss, with its support of L2 regularization. Set 500 as the maximum number of iterations taken for 'lbfgs' solver to converge, 17 as the pseudo random generators while shuffling the data and 4 CPU cores for parallelizing over classes. After the training, the accuracy for testing data we get is 70.7%.

B. Random Forest

In the supervised learning algorithm of machine learning [6] , our goal is to learn a stable model that performs well in all aspects, but the actual situation is often not so ideal. Sometimes we can only get multiple models with preferences (The weakly supervised model performs better in some aspects). Ensemble learning is to combine multiple weakly-supervised models here in order to get a better and more comprehensive strong-supervised model.

Random forest is a kind of ensemble learning.In this model, we set 100 trees in the forest, with 17 as the pseudo random generators while shuffling the data and 2 CPU cores for paralleling over classes.

We fit the training data without normalized, the accuracy for testing is improved to 86.0% compared with the logistic regression. To check whether normalization affects the accuracy, we also fit the data with normalization to get a slight(0.1%) increase in accuracy to 86.1%, which could be ignored. We inspect the importance of features to observe which variables have the most effect in this model, shown below(Fig.3.) as descending order. Feature 'Evaluation' takes up the most important effect as about 22%, and the 'Wilderness_Area4' is the least important with coefficient only 3.8%.

	Importance
Elevation	0.221297
Horizontal_Distance_To_Roadways	0.093678
Horizontal_Distance_To_Fire_Points	0.073004
Horizontal_Distance_To_Hydrology	0.062592
Hillshade_9am	0.052744
Vertical_Distance_To_Hydrology	0.052035
Aspect	0.050237
Hillshade_3pm	0.047294
Hillshade_Noon	0.045997
Wilderness_Area4	0.038577

Fig. 3. Feature importance ranking.

C. *LightGBM*

Light Gradient Boosted Machine(LightGBM) [7] is one algorithm that implements the gradient boosting algorithm effectively. The accuracy is only 23.1% as we fit the simple LightGBM model with 17 pseudo random generators while shuffling the data. Tuning the parameters in this model to improve the accuracy is the first stage. We choose Grid Search with Cross-Validation in LightGBM which helps to find the best hyper-parameters for this specific data set. Setting the value of cross-validation to 5, the grid of parameters with number of leaves and the maximum depth, then the best accuracy score receives a large improvement to 84.7%. In the second stage, we try the convergence test with a learning rate as 0.2. Letting the model to do 200 iterations, then the accuracy reaches to 87.0%, a slight raise. We also inspect the importance of features to observe which variables have the most effect in this stage, shown as the descending order, 'Elevation' effects the most and 'Slope' is the least.

D. *Neural Network*

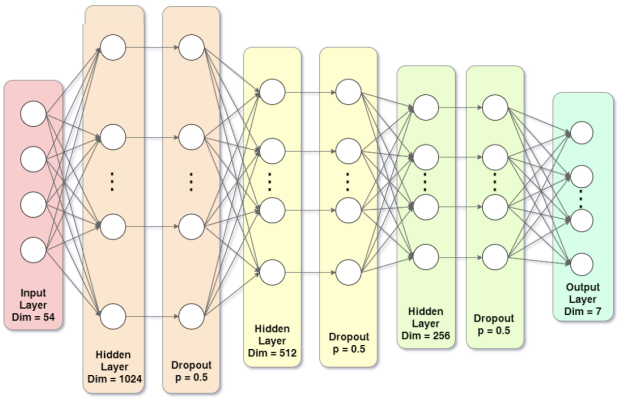


Fig. 4. Structure of metwork we used.

As we learn FNN [6] in this semester, we decided to apply it to this classification task. When adopting neural networks, tuning is always the most critical part. We tried random search and grid search while tuning epochs, batch size, learning rate, activation function, and optimizer. Although we do a lot of computations and find some clues of improving the correctness, our accuracy can only achieve around 50% (random guess in around 14.29%). This suggests our model still does not fit the fact well. As auto-tuning requires a lot of computation, especially tuning multiple hyper-parameters together (each layers' dimension, and its activation function), we make a reference of others parameters and this improves our accuracy a lot. We adopt the model with the optimizer Adam which has 3 hidden layers and with 1024, 512, 256 neurons each. Besides the final layer, all the other layers use ReLu as the activation function. Between each layer, we add a dropout layer to prevent over-fitting. We run the model with a batch size of 2048 and epochs of 100. Finally achieve an accuracy around 93.5%. In each epoch, the training model will be tested by the test set, while the test set didn't participate in the training. During training, as loss decreases, the accuracy is increasing. Similar behaviour shows in testing.

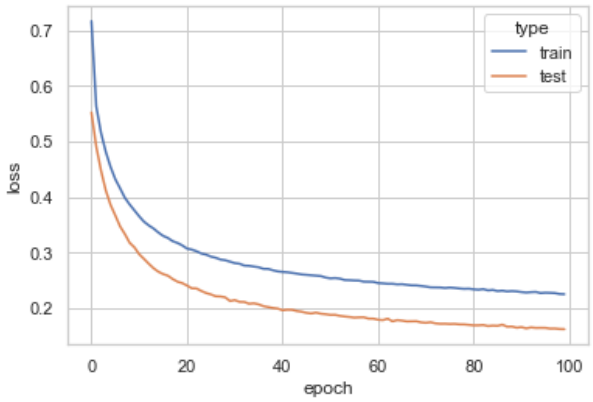


Fig. 5. Loss of neural network with epoch.

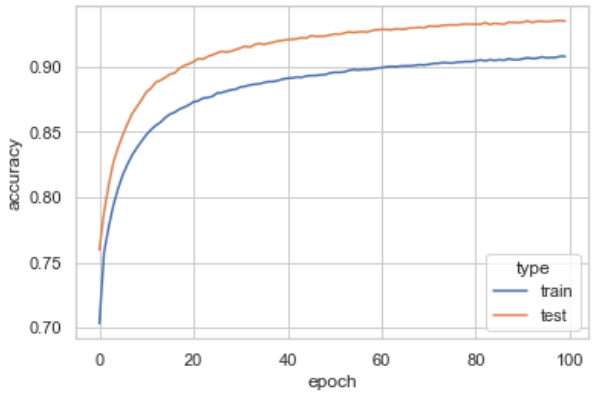


Fig. 6. Accuracy of neural network with epoch.

E. Decision Tree on Spark

1) Decision Tree: The decision tree algorithm family can naturally handle categorical and numerical features. Decision tree algorithms are easy to parallelize. They are robust to outliers in the data, which means some extreme or possibly wrong data points will not affect the prediction at all. The algorithm can accept data of different types and dimensions, and does not require preprocessing or normalization for data types and scales that are different. Another advantage of algorithm based on decision tree is that it is relatively intuitive to understand and reason.

In order to explain the decision tree in more detail, some features (altitude, slopeaspect) of vegetation are selected for reference.

Id	Elevation	Slope	Cover_Type
1	2596	3	5
2	2590	2	5
3	2804	9	2
4	2785	18	2
5	2595	2	5

Fig. 7. Example.

It is easy to think of a simple decision tree to fit the training data set. This decision tree only makes decisions based on ”altitude”. This series of ”yes/no” judgments is the prediction process embodied by the decision tree algorithm. Of course, we can continue to add decision rules until all samples can be predicted correctly. But the decision tree obtained in this way may be unreasonable. In other words, the decision tree may be: ”If the elevation of the vegetation is less than 2600 and the slope is less than 4, it may be type 5.”

Although it can perfectly fit a given sample, this decision tree cannot predict a land with an altitude of 2500 and a slope of 5 as type 5. In order to avoid this phenomenon called over-fitting, we still need to continue to improve.

Unlike calling the sklearn module, Spark MLlib abstracts the feature vector as LabeledPoint, which consists of a Spark MLlib Vector containing multiple feature values and a target value called a label. The target is Double type, and Vector is essentially an abstraction of multiple Double type values. This shows that LabeledPoint is only suitable for numerical features. But as long as it is properly coded, LabeledPoint can also be used for categorical features.

Although all the columns of the Cover-type data set are numerical values, in essence, the data set is not entirely composed of numerical features. The covtype.info [2] file shows that 4 columns are generated by one-hot encoding of the same categorical feature Wilderness_Type, and it is assigned 4 possible values. Similarly, there are 40 columns with the same categorical feature Soil_Type. The target itself is also a categorical value, coded from 1 to 7. The rest are numerical features measured by different units, such as meters, degrees, or a qualitative index value.

First, in order to facilitate the use of untrained data to evaluate the model, the original data set is divided into three complete parts: training set80%, cross-validation set (CV10%) and test set10%. Before training, we didn't know how to use 54 features to describe a land parcel in Colorado that had never been seen before, nor how to predict the type of forest vegetation on the parcel.

In this case, let's first try to construct a DecisionTreeModel on the training set, with default values for the parameters, and use the CV set to calculate the index of the result model. It should be noted that trainClassifier indicates that the target in each LabeledPoint should be treated as a different class label, rather than a numerical feature value. The reason is that the original data set is one-hot encoded, so the feature values represent different categories.

```
(0.6879105188005712,0.6724202102912441)
(0.7224695369618197,0.7905422222222223)
(0.6314069838676741,0.8600834492350486)
(0.3257328990228013,0.398406374501992)
(0.9411764705882353,0.016194331983805668)
(0.0,0.0)
(0.6995073891625616,0.4190850959173635)
```

Fig. 8. The accuracy of each category relative to other categories.

Although the 70% accuracy sounds good compared to the blind guessing results, it is not accurate enough compared with the previous algorithm results in the sklearn module and neural network.

2) Parameters tuning: It should be noted that the above accuracy is obtained with the default parameters in DecisionTree.trainClassifier(). If try other values of hyperparameters during the construction of the decision tree, the accuracy can be improved.

Here we use trainClassifier, first look at what parameters trainClassifier needs:

- input: LabeledPoint of the original data
- numClasses: number of categories
- categoricalFeaturesInfo: specifies which features are categorical and how many categorical values each of those features can take. E.g., Map(0 - 2, 4 - 10) specifies that feature 0 is binary (taking values 0 or 1) and that feature 4 has 10 categories (values 0, 1, . . . , 9).

Note that we do not have to specify categoricalFeaturesInfo. The algorithm will still run and may get reasonable results. However, performance should be better if categorical features are properly designated.

- Impurity: gini or entropy, impurity is used to measure the quality of a rule, a good rule can divide the data into two parts of equal value, and a bad rule is the opposite.
  Gini formula:

$$I_G(p) = 1 - \sum_{i=1}^{N} p_i^2. \quad (1)$$

  Entropy formula:

$$I_E(p) = \sum_{i=1}^{N} p_i \log\left(\frac{1}{p}\right) = -\sum_{i=1}^{N} p_i \log\left(p_i\right). \quad (2)$$

- maxDepth: only limits the number of layers of the decision tree. It is the maximum number of series of judgments made by the classifier in order to classify the sample. Limiting the number of judgments helps avoid over-fitting the training data.
- maxBins: The number of decision rules used in each layer. The more rules, the more accurate and the more time it takes. The smallest number of buckets should not be less than the largest number of choices in the category feature.

The traditional method of performing hyper-parameters optimization is grid search or parameter sweep. Grid search algorithms must be guided by certain performance metrics, usually measured by cross-validation on the training set or evaluation of the retained validation set. Here, when we train the model, by selecting different parameter combinations, feedback the model accuracy results of each combination training, then we can choose the best parameter combination to build the model.

```
((entropy,20,300),0.9380098861985638)
((gini,20,300),0.9319721451536285)
((entropy,20,10),0.9273681094366382)
((gini,20,10),0.9195954644654499)
((gini,1,10),0.633916339077334)
((gini,1,300),0.6335772755123819)
((entropy,1,300),0.48759922342395684)
((entropy,1,10),0.48759922342395684)
```

Fig. 9. Parameter combination result.

Obviously, the maxdepth 1 is too small, and the results obtained are relatively poor. The number of bins is a bit helpful, but not very helpful. At a reasonable maxdepth, the results of the two impurity measures are similar. Generally speaking, the more bins the better, but it will slow down the model construction process and increase memory usage. Increasing the maxdepth can improve the accuracy, but there is an inflection point here, after which it is useless to increase the depth. It can be seen that the best parameter combination is the first one, and the accuracy is about 94%. We can check the model trained with the best parameters on the test data set. The accuracy of the adjusted model applied to the test set is 91.6%. It is not very different from the 93.8% obtained from the training set, which is acceptable.

IV. Comparison

In Table I, we compare the performance of all our models from two dimension, accuracy and efficiency.The three algorithms, namely logistic regression, random forest and lightGBM in the sklearn library, have a satisfied accuracy over 70% (random classification has the expected accuracy of 14.29%). However, the accuracy range of these three models before tuning are from 70.7% to 84.7%, which shows the selection of the best classifier during the test progress is needed to improve the effect. It can also be clearly seen that the model has a more obvious improvement after adjusting the parameters. The most typical one is FNN, which has an increase of accuracy from 50% to 93.5%. This shows tuning has a huge effect on FNN's accuracy.

However, it is worth noting that due to the characteristics of the neural network structure, a large number of parameters need to be optimized, and the increase in accuracy also brings an increase in time cost. At this time, the advantages of distributed decision trees are reflected: on the premise of achieving accuracy close(91.6%) to that of neural networks, decision trees are built in a breadth-first manner, and the Split and Bin of features are calculated in advance. In the case of a large amount of data, the optimal split can be approximated by Bin, instead of traversing all possible split points of the training data, using the known Bin and the statistics required for each Bin to construct a one-dimensional array. Typical parallelization ideas(partition-local-merge) are applied to achieve a significant increase in training speed.

TABLE I
Five models' performance

Accuracy / Model	Default	After Tuning
Logistic Regression	70.7%	None
Random Forest	86.0%	None
LightGBM	84.7%	87.0%
FNN	50.0%	93.5%
Desicion Tree on Spark	70%	91.6%

^{None}only one result.

V. Summary

In this task, we analyze and finish the classification task on the classic forest cover type data, and then compare them from accuracy and efficiency. We make use of multiple methods as logistic regression, random forest, lightGBM, FNN and distributed decision tree based on spark, then compare them from the accuracy and speed cost.

In this project, the accuracy are given before and after tuning. All tested model have an increase in their accuracy after tuning. Among them, FNN has the highest increase after tuning. The accuracy of FNN has increased nearly 40%, while the lightGBM has the lowest increase. For the final accuracy

(i.e., after tuning) FNN gets the highest accuracy among all the tested model.

Time complexity of models are highly related to the parameters of the classifier, thus is hard to compare. BUT In general, logistic regression, random forest, and lightGBM has the same time cost level. As neural network requires thousands of times back and forth training with the whole training set, it has a longer time consuming when applying to the same task, Also, when training, finishing a search for its various kinds of parameters are painfully slow, although better computation devices and a better searching candidates might be helpful to reduce the cost. But its flexibility and possibility are still attractive.

The parallel level of computing is also a factor which related to speed. In this project, we use decision tree based on Spark's MLlib library. It significantly decreases the time consuming while increases the accuracy. This will be another possible solutions when finishing machine learning tasks. The original data set has actually been one-hot encoded. Several categorical features have been converted into multiple binary values using one-hot methods, such as 40 numerical features instead of a categorical feature with 40 values Will increase memory usage and slow down decision-making speed, so if one-hot encoding is removed, the effect may be improved.

VI. Contribution

Xinyue ZHANG(20750194): Literature Review, Data Preprocessing, Decision tree model Setup/Coding, Report writing, Video Presentation.

Zhuoxuan LI(20740917): Literature Review, Neural network model Setup/Coding, Data Analyzing / Visualization, Report writing, Video Presentation.

Shuyu QIAO(20747563): Literature Review, Sklearn models Setup/Coding, Report writing, Video Presentation.

References

[1] "Covertype Data Set",http://archive.ics.uci.edu/ml/datasets/Covertype/, 1998.

[2] J.A.Blackard, "Description of The Forest CoverType Dataset", http://ftp.ics.uci.edu/pub/machinelearningdatabases/covtype/covtype.info, 2001.

[3] S. A. Eschrich, "Learning From Less: A Distributed Method for Machine Learning ", Dissertation, U. of South Florida, 2003.

[4] A. Lazarevic and Z. Obradovic, "The distributed boosting algorithm", Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 311-316, 2001.

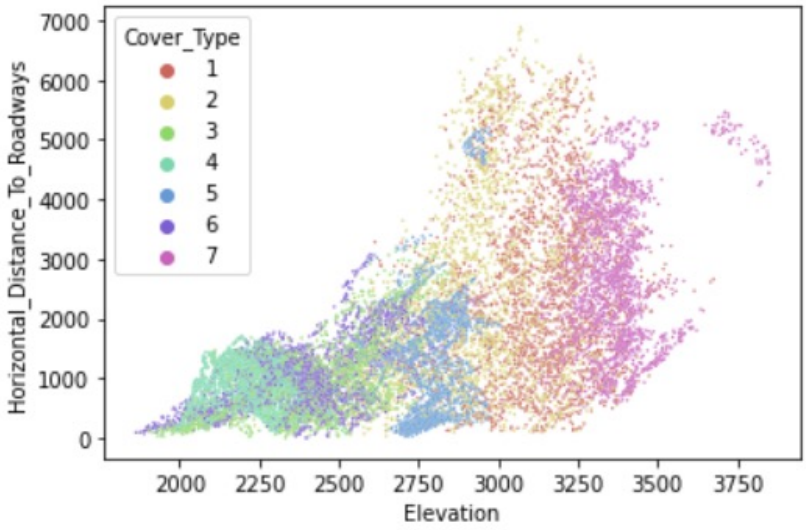
[5] J. R. Quinlan, "C4.5: Programsfor Machine Learning", Morgan Kaufmann Publishers, Inc., pp. 35-43, 1993.

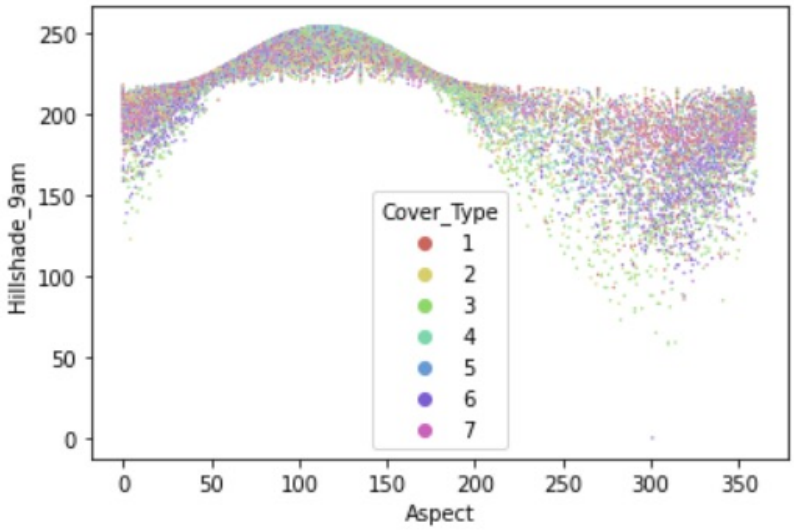
[6] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm", Proceedings of IEEE Int'l Conference on Neural Networks, pp. 586-591, 1993.

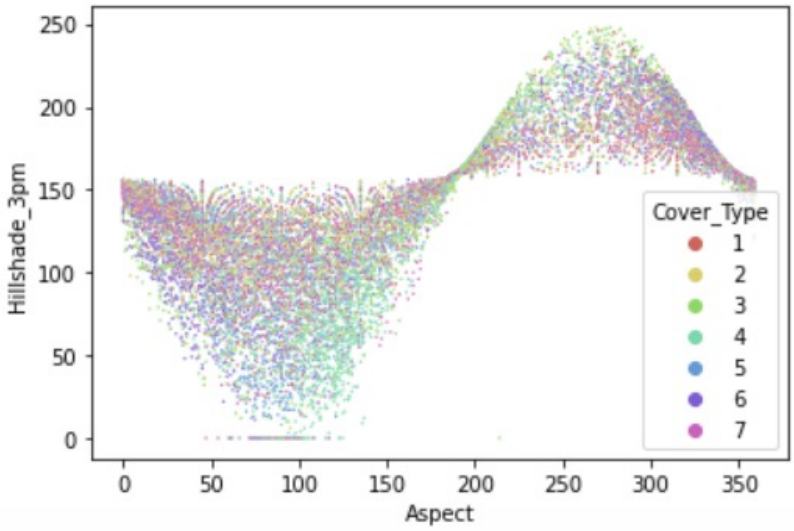
[7] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. Annals of statistics, pages 1189–1232, 2001.

[8] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". Advances in Neural Information Processing Systems 30 (NIPS 2017), pp. 3149-3157.

APPENDIX-I

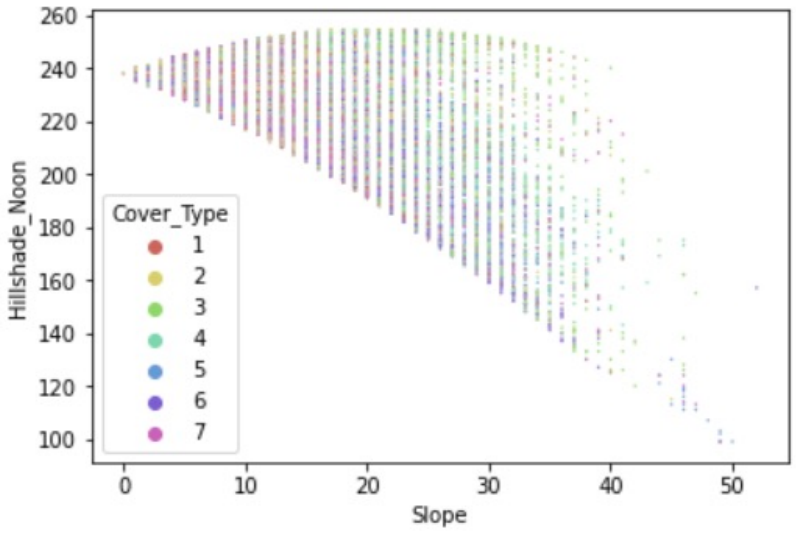

Cover_Type
1
2
3
4
5
6
7
Horizontal_Distance_To_Roadways
7000
6000
5000
4000
3000
2000
1000
0
2000
2250
2500
2750
3000
3250
3500
3750
Elevation

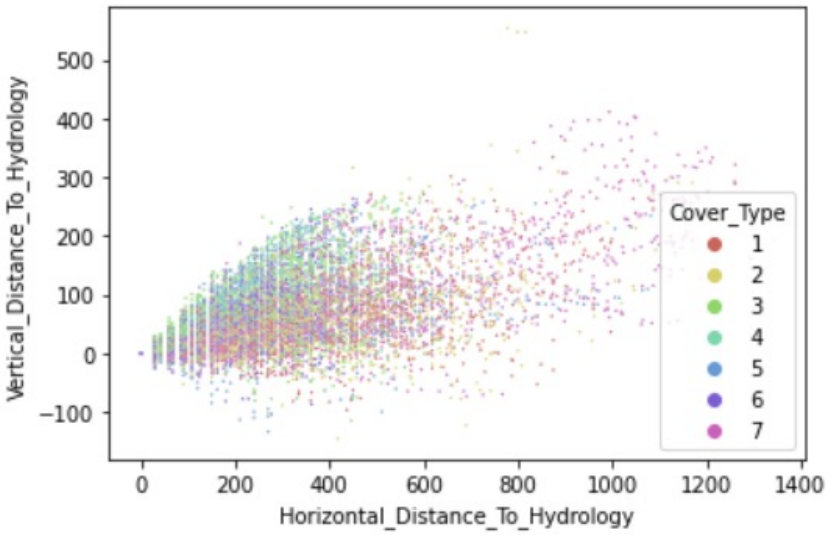

Cover_Type
1
2
3
4
5
6
7
Hillshade_9am
250
200
150
100
50
0
0
50
100
150
200
250
300
350
Aspect

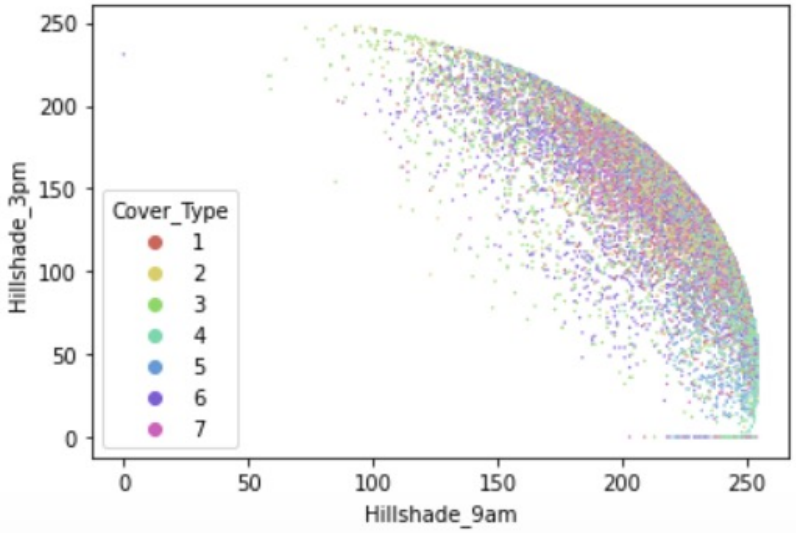

Cover_Type
1
2
3
4
5
6
7
Hillshade_3pm
250
200
150
100
50
0
0
50
100
150
200
250
300
350
Aspect

]

APPENDIX-II


Hillshade_Noon
Slope
Cover_Type
1
2
3
4
5
6
7


Vertical_Distance_To_Hydrology
Horizontal_Distance_To_Hydrology
Cover_Type
1
2
3
4
5
6
7


Hillshade_3pm
Hillshade_9am
Cover_Type
1
2
3
4
5
6
7

]

APPENDIX-III

	count	mean	std	min	25%	50%	75%	max
Id	15120.0	7560.500000	4364.912370	1.0	3780.75	7560.5	11340.25	15120.0
Elevation	15120.0	2749.322553	417.678187	1863.0	2376.00	2752.0	3104.00	3849.0
Aspect	15120.0	156.676653	110.085801	0.0	65.00	126.0	261.00	360.0
Slope	15120.0	16.501587	8.453927	0.0	10.00	15.0	22.00	52.0
Horizontal_Distance_To_Hydrology	15120.0	227.195701	210.075296	0.0	67.00	180.0	330.00	1343.0
Vertical_Distance_To_Hydrology	15120.0	51.076521	61.239406	-146.0	5.00	32.0	79.00	554.0
Horizontal_Distance_To_Roadways	15120.0	1714.023214	1325.066358	0.0	764.00	1316.0	2270.00	6890.0
Hillshade_9am	15120.0	212.704299	30.561287	0.0	196.00	220.0	235.00	254.0
Hillshade_Noon	15120.0	218.965608	22.801966	99.0	207.00	223.0	235.00	254.0
Hillshade_3pm	15120.0	135.091997	45.895189	0.0	106.00	138.0	167.00	248.0
Horizontal_Distance_To_Fire_Points	15120.0	1511.147288	1099.936493	0.0	730.00	1256.0	1988.25	6993.0
Wilderness_Area1	15120.0	0.237897	0.425810	0.0	0.00	0.0	0.00	1.0
Wilderness_Area2	15120.0	0.033003	0.178649	0.0	0.00	0.0	0.00	1.0
Wilderness_Area3	15120.0	0.419907	0.493560	0.0	0.00	0.0	1.00	1.0
Wilderness_Area4	15120.0	0.309193	0.462176	0.0	0.00	0.0	1.00	1.0
Soil_Type1	15120.0	0.023479	0.151424	0.0	0.00	0.0	0.00	1.0
Soil_Type2	15120.0	0.041204	0.198768	0.0	0.00	0.0	0.00	1.0
Soil_Type3	15120.0	0.063624	0.244091	0.0	0.00	0.0	0.00	1.0
Soil_Type4	15120.0	0.055754	0.229454	0.0	0.00	0.0	0.00	1.0
Soil_Type5	15120.0	0.010913	0.103896	0.0	0.00	0.0	0.00	1.0
Soil_Type6	15120.0	0.042989	0.202840	0.0	0.00	0.0	0.00	1.0
Soil_Type7	15120.0	0.000000	0.000000	0.0	0.00	0.0	0.00	0.0
Soil_Type8	15120.0	0.000066	0.008133	0.0	0.00	0.0	0.00	1.0
Soil_Type9	15120.0	0.000661	0.025710	0.0	0.00	0.0	0.00	1.0
Soil_Type10	15120.0	0.141667	0.348719	0.0	0.00	0.0	0.00	1.0
Soil_Type11	15120.0	0.026852	0.161656	0.0	0.00	0.0	0.00	1.0
Soil_Type12	15120.0	0.015013	0.121609	0.0	0.00	0.0	0.00	1.0
Soil_Type13	15120.0	0.031481	0.174621	0.0	0.00	0.0	0.00	1.0
Soil_Type14	15120.0	0.011177	0.105133	0.0	0.00	0.0	0.00	1.0
Soil_Type15	15120.0	0.000000	0.000000	0.0	0.00	0.0	0.00	0.0
Soil_Type16	15120.0	0.007540	0.086506	0.0	0.00	0.0	0.00	1.0
Soil_Type17	15120.0	0.040476	0.197080	0.0	0.00	0.0	0.00	1.0
Soil_Type18	15120.0	0.003968	0.062871	0.0	0.00	0.0	0.00	1.0
Soil_Type19	15120.0	0.003042	0.055075	0.0	0.00	0.0	0.00	1.0
Soil_Type20	15120.0	0.009193	0.095442	0.0	0.00	0.0	0.00	1.0
Soil_Type21	15120.0	0.001058	0.032514	0.0	0.00	0.0	0.00	1.0
Soil_Type22	15120.0	0.022817	0.149326	0.0	0.00	0.0	0.00	1.0
Soil_Type23	15120.0	0.050066	0.218089	0.0	0.00	0.0	0.00	1.0
Soil_Type24	15120.0	0.016997	0.129265	0.0	0.00	0.0	0.00	1.0
Soil_Type25	15120.0	0.000066	0.008133	0.0	0.00	0.0	0.00	1.0
Soil_Type26	15120.0	0.003571	0.059657	0.0	0.00	0.0	0.00	1.0
Soil_Type27	15120.0	0.000992	0.031482	0.0	0.00	0.0	0.00	1.0
Soil_Type28	15120.0	0.000595	0.024391	0.0	0.00	0.0	0.00	1.0
Soil_Type29	15120.0	0.085384	0.279461	0.0	0.00	0.0	0.00	1.0
Soil_Type30	15120.0	0.047950	0.213667	0.0	0.00	0.0	0.00	1.0

]