Capstone Project Report - Forest Cover Type Prediction

Mohan Prasath Chinnasamy

January 6, 2017

1 Definition

1.1 Project Overview

Our Earth is an unique planet in the solar system holding the key to sustain Life. Even now many parts of the oceans, and forests remain unexplored. Several species are discovered periodically in remote forests across the globe. Under further genetic studies it is revealed that we humans do posses some characteristics of the DNA of the mammals!. Exploration of such remote regions can provide answers for many unanswered questions regarding origin of life, evolution, and life in unpolluted environments[1].

However such explorations are costly and even dangerous for humans. Aerial surveillance have improved much during these last decades. These technological improvements can assist in human explorations by capturing images of unexplored regions reducing cost and human effort. But most forests are dense with vegetation and such aerial surveillance can provide only the top view. We can then use the surveillance data(tabular data) to predict and restrict the search space for human explorations. From these explorations an extrapolation[3] can be done about missing species in similar areas, thus reducing the search space further.

1.2 Problem statement

This project aims to use surveillance data with cartographic information to train models that can predict and identify the forest cover types. The project also aims to create a classification model with more than 90% accuracy which can classify the forest cover type.

The performance of the model can be evaluated by calculating the prediction accuracy. For testing, 20% of the total data set will be used which is not used to train the model All classification models will be trained with 80% of the data. This huge amount of data used to train the models can increase the prediction accuracy over the unseen data. For example, a k-Nearest Neighbor model that can classify the features into k target variable can be applied here.

1.3 Evaluation Plan

Performance of various classifiers can be ranked and finally we can chose highest ranked classifier using all of the following properties,

- Precision
- Recall
- Accuracy
- F1-score

All above scores can be calculated by measuring the occurrence of true positive, true negative, false positive and false negatives. Also the true positive rate(Sensitivity) and true negative rates(Specifivity) are used in the above calculations.

2 Analysis

2.1 Data Exploration

The cover type data set was acquired from UCI Machine Learning Repository[2]. The data was collected in four major areas of wilderness in Roosevelt National Forest of northern Colorado. This region is affected minimally by human intervention. So this data also holds information about ecological changes occurring over time.

The data set has 12 major features with one multivariate target variable with information about forest cover type. Two features Ẃilderness_Area; and Śoil_Typeáre further divided into 4 and forty binary valued sub features respectively. Thus adding up all features to a count of 54. There are 581012 instances of the data with no missing values.

The features and the target variable of the data set are listed below,

- Elevation-quantitative data measured in meters
- Aspect-quantitative data measured in degrees
- Slope-quantitative data measured in degrees
- Horizontal_Distance_To_Hydrology-quantitative data measured in meters
- Vertical_Distance_To_Hydrology-quantitative data measured in meters
- Horizontal_Distance_To_Roadways-quantitative data measured in meters
- Hillshade_9am - quantitative data from 0 to 255 index
- Hillshade_Noon - quantitative data from 0 to 255 index
- Hillshade_3pm - quantitative data from 0 to 255 index

- Horizontal_Distance_To_Fire_Points - quantitative data measured in meters
- Wilderness_Area (4 binary columns) - binary data with 0 (absence) or 1 (presence)
- Soil_Type (40 binary columns) - binary data with 0 (absence) or 1 (presence)
- Cover_Type (7 types) - integer data from 1 to 7

Above features are measurements of a forest and the landmass around the forests. This information is provided to us as a tabular numerical data. All the above data contains their respective forest type(our target variable) preclassified.

2.2 Exploratory Visualization

In this section, the data is studied visually in-order to understand the hidden relationships of various attributes. However, the visual explorations can only be done effectively on non-binary data. So we avoid columns related to wilderness_area and soil_type. The following picture1 is a scatter plot of all the variables against all attributes themselves. From 1 we can observer various relationship among the data attributes. Some of them are listed below,

- (aspect, hillshade_3pm) and (aspect, hillshade_9am) are two pairs of attributes exhibiting a sigmoid relationships2.
- (hillshade_9am, hillshade_3pm) and (hillshade_noon, hillshade_3pm) are two pairs of attributes exhibiting an ellipsoid relationships3.
- (vertical_distance_to_hydrology), (horizontal_distance_to_hydrology) exhibit a linear pattern relationship. 4

The above relationship have to be verified, so the correlation score among those attributes are calculated and higher correlation(absolute values above 0.5) are listed below:

- hillshade_9am and hillshade_3pm = -0.78
- aspect and hillshade_3pm = 0.65
- horizontal_distance_to_hydrology and vertical_distance_to_hydrology = 0.61
- hillshade_noon and hillshade_3pm = 0.59
- aspect and hillshade_9am = -0.58
- slope and hillshade_noon = -0.53

Further examinations can be done by viewing the attached images with this project report or by viewing the plotted results in the Ipython notebook named **Capstone Project Forest Cover Type Prediction.ipynb**

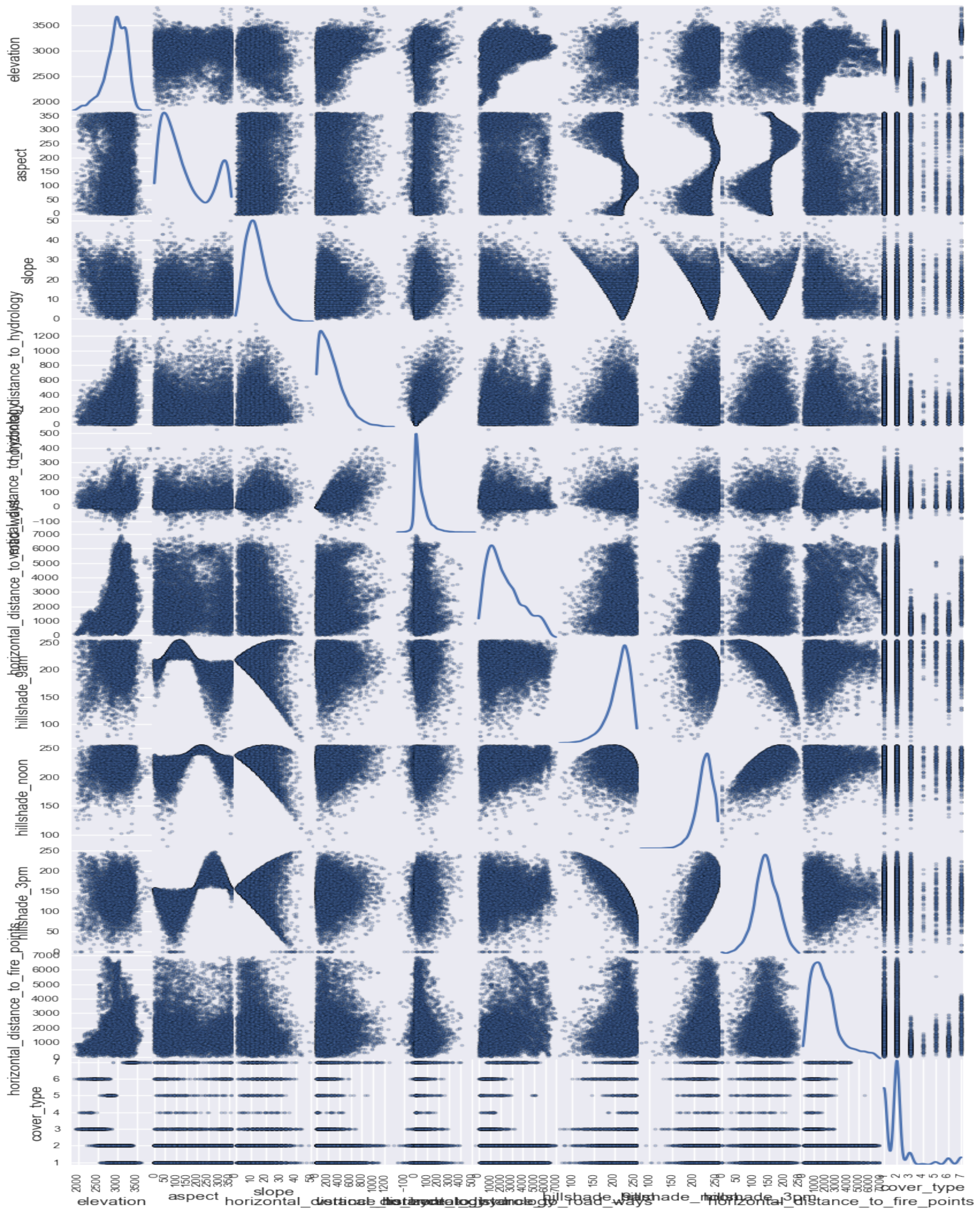
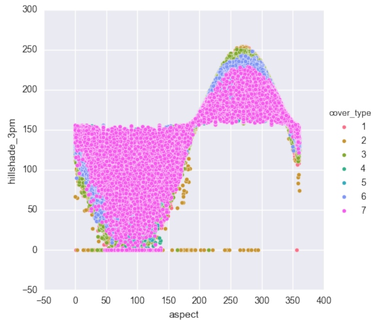
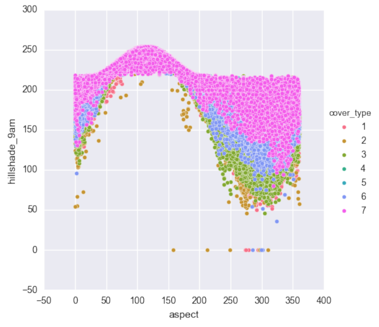


Figure 1: A scatter plot of all non binary attributes in the data

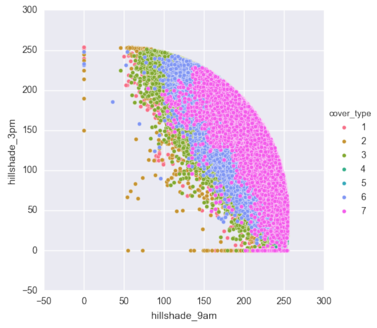


(a) Aspect vs Hillshade3PM

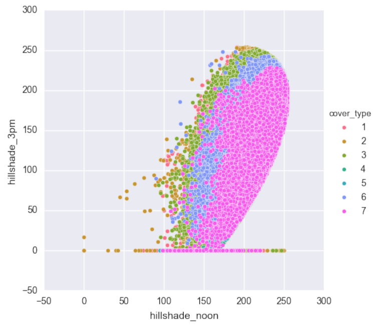


(b) Aspect vs Hillshade9AM

Figure 2: Aspect VS Hillshade 3PM and 9AM

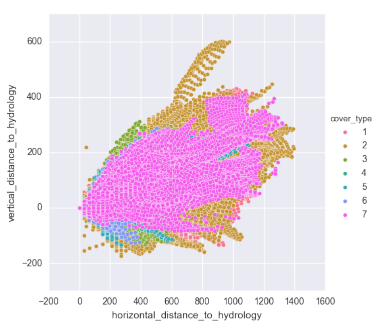


(a) Hillshade9AM vs Hillshade3PM

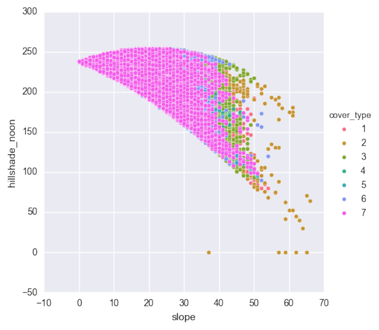


(b) HillshadeNoon vs Hillshade3PM

Figure 3: Hillshade VS Hillshade



(a) Horizontal vs Vertical distance to hydrology



(b) Slope vs HillshadeNoon

Figure 4: Distance to Hydrology and Slope VS Hillshade

3 Algorithms and Techniques

3.1 Classification

Let's consider, we are given a data set with x, y pairs of data where value y is a function of x. The values of y are categorical and discrete and are called as classes. We have to identify groups of x data point that belongs to a particular class of y. We can train a model to do the classification process and later test the model for its accuracy of classification on an unseen data. This type of classification are called as Supervised learning. Also depending upon the number of classes in y, we can also classify this model as Binary classification(y has two classes) or Multi-class classification.

Forest cover type data set's target variable contains seven unique values. They are 1, 2, 3, 4, 5, 6, and 7. Also we are already given the target variable, so this project is a Supervised classification project. Since there are more than two classes in the target variable we consider this classification model as Multi-class classification. Some random algorithms are chosen to predict the target variable and from their basic predictions a mean accuracy score is calculated. Those score are listed below.

- LinearSVC
- SVC with sigmoid kernel
- SVC with rbf kernel
- SGDClassifier
- KNeighborsClassifier
- RandomForestClassifier
- XGBClassifier

From the above list of classifiers, we can notice that three classifiers, namely KNeighborsClassifier, RandomForestClassifier, and XGBClassifier shows highest classification accuracy(more than 0.7). We focus on these algorithms and fine tune their attributes to increase the accuracy further.

3.2 Benchmark Model

The forest cover type data set is split into test and train set at 80% and 20% respectively. This would create a training set with 464809 samples and test set with 116203 samples. We focus on training a classification model, with more than 90% accuracy because the implication of a wrong classification can be huge. For example, spending is huge in human explorations including medical, transport, insurance, machinery, navigation, etc., Explorations done based upon results of classifier is supposed to yield good results. Also we use the existing target variable for measuring the test set. This gives us scores for precision, recall, accuracy, and f1-score. We plan to select a best classifier that can score consecutively higher in all the scores and at low cost(running time).

3.3 KNeighborsClassifier

KNeighborsClassifier is developed based upon the concept of Nearest Neighbor Classification. Initially this classifier mode is trained with the training data. When unseen data is given as input to the model, it selects **k** nearest neighbors to a data point and identifies the common class for those n neighbors. The common class is then assigned as the appropriate class for a decision tree being investigated.

3.3.1 Attributes to fine tune

The attributes for KNeighborsClassifier are listed below:

- n_neighbors was changed to 10, default was 5.
- weights was changed to distance, default was uniform. This gives more importance to closest neighbors rather than the count of most common class among the neighbors.
- algorithm remains unchanged, default is auto. This allows the model to pick the best approach to select nearest neighbors from the available options.
- leaf_size is set to 10000, default value was 30. This value is increased to a higher value, but the optimal value cannot be known.
- p uses the default value.
- metric uses the default 'minkowski' distance metric.
- metric_params was set to default value None.
- n_jobs was set to -1, default was 1. This makes the model to utilize all the cores in the hardware for calculations.

3.3.2 Results

Before fine tuning the algorithm generated a mean accuracy score of **0.968701324406**. After fine tuning the mean accuracy score remained unchanged with a value of **0.961154187069**. Using the scaled data, the algorithm had a mean accuracy score of **0.916361883944**. This is a reduced score. However this algorithm has achieved our initial goal to create a classifier model with more than 90% accuracy. Finally, it seems that the algorithm cannot be fine tuned anymore. So we will look into the next algorithm.

3.4 RandomForestClassifier

RandomForestClassifier is developed based upon the concept of random decision trees. Random Forest is an ensemble learning method which builds decision trees during training time and later during testing time, outputs the modes of the most occurring class for a data point. Both Random Forest classifiers and k-NN can be viewed as *weighted neighborhoods schemes*. The following paragraph(Quoted from scikitlearn webpage) best explains the Random Forest further.

"In random forests (see RandomForestClassifier and RandomForestRegressor classes), each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. In addition, when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. As a result of this randomness, the bias of the forest usually slightly increases (with respect to the bias of a single non-random tree) but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model."

3.4.1 Attributes to fine tune

The attributes for RandomForestClassifier are listed below:

- n_estimators was set to 500, default was 10.
- criterion was set to default 'gini'.
- max_depth was set to default value None, this allows the tree to expand unbounded.
- min_samples_split was set to default value 2, at least two samples are required to split a node in the tree to two other sub nodes.
- min_samples_leaf was set to default value 1, at least one sample is required to form a leaf node.
- min_weight_fraction_leaf was set to default value 0.0
- max_features was set to default value 'auto', that is square root of number of features, $\tilde{7}$.35.
- max_leaf_nodes was set to default value None, allowing the tree to grow to the maximum size.
- min_impurity_split was set to default value 1e-07, thus stopping the tree from further growth.
- bootstrap was set to default value True.

- oob_score was set to default value False.
- n_jobs was set to -1, default was 1. This makes the model to utilize all the cores in the hardware for calculations.
- random_state was set to 42, default value was None.
- verbose was set to default value 0
- warm_start was set to default value False
- class_weight was set to default value None. This is set to 'None' because the target variable is not balanced. It has more data points for class 1 and 2. So building a tree for the target variable in balanced manner is not possible.

3.4.2 Results

Before fine tuning the algorithm generated a mean accuracy score of **0.943142603891**. After fine tuning the mean accuracy score was slightly increased with a value of **0.955982203558**. Using the scaled data, the algorithm had an increase in mean accuracy score with a value of **0.956042442966**. The algorithm responds to fine tuning and attribute scaling.

This is the fastest algorithm for classifying the forest cover type dataset. So RandomForestClassifier can be recommended to be used on similar data sets in the future for accurate predictions. This algorithm has achieved our initial goal to create a classifier model with more than 90% accuracy.

3.5 XGBClassifier

XGBClassifier is based upon the famous machine learning concept called Gradient boosting. It was originally created on the idea of improving a weak learner into a better learner. Usually a weak learner is defined as the one whose performance is slightly better than random choice. Gradient boosting has three components: optimizing a loss function, a weak learner making predictions, and an additive model to add weak learners to minimize the loss function. Based upon these functions a booted classifiers is built and it's source code can be found in GitHub.

3.5.1 Attributes to fine tune

The attributes for KNeighborsClassifier are listed below:

- max_depth was set to 50, default value as 3.
- n_estimators was set to 500, default value as 100.
- silent uses default value True.

- objective uses ”multi softmax”, default was ”binary logistic”. Since we use multiclass classification, binary classification wont work here. More related discussion.
- nthread was set to -1, default was 1. This makes the model to utilize all the cores in the hardware for calculations.
- gamma uses default value 0.
- min_child_weight uses default value 1.
- max_delta_step uses default value 0.
- subsample uses default value 1.
- colsample_bytree was changed to 0.3, from default value of 1.
- colsample_bylevel uses default value 1.
- reg_alpha uses default value 0.
- reg_lambda uses default value 1.
- scale_pos_weight uses default value 1.
- seed uses default value 0.
- base_score uses default value 0.5.
- missing uses default value None.

3.5.2 Results

Before fine tuning the algorithm generated a mean accuracy score of **0.744025541509**. After fine tuning the mean accuracy score was increased highly with a value of **0.95894254021**. Using the scaled data, the algorithm had a increase in mean accuracy score with a value of **0.95895114584**. The algorithm responds to fine tuning only. **This is the slowest algorithm for the forest cover type data set.** This algorithm has also achieved our initial goal to create a classifier model with more than 90% accuracy.

4 conclusion

From the above three classifiers, we can notice that after fine tuning, XGBClassifier performance is increased by a huge margin. Also an accuracy increase from $\tilde{0}.74$ to $\tilde{0}.95$ is observed. Also our initial plan to achieve an accuracy of above 90% is also achieved here. However, **RandomForestClassifier** performs consistently better when considered to all other classifiers. It produces an accuracy score of $\tilde{0}.96$ which is the highest observed in this project. Also it is the fastest classifier considered in this project. **So from the list of classifiers used in**

this project I would recommend RandomForestClassifier to be used for further analysis in similar type of forest cover identification. Since it has consistent highest accuracy score.

References

[1] Christopher W. Dick and W. John Kress. Dissecting tropical plant diversity with forest plots and a molecular toolkit. *BioScience*, 59(9):745–755, 2009.

[2] M. Lichman. UCI machine learning repository, 2013.

[3] Rosanne Tackaberry, Nicholas Brokaw, Martin Kellman, and Elizabeth Mallory. Estimating species richness in tropical forest: the missing species extrapolation technique. *Journal of Tropical Ecology*, 13(3):449–458, 005 1997.