Final Report Forest Cover Type Prediction

Krishna Vamsi Chandu
kchand23@uic.edu

Ativ Aggarwal
aaggar9@uic.edu

Neil Champakara
nchamp3@uic.edu

Syed Shariq Rehman
srahma35@uic.edu

Abstract

This document is a final report of the course project "Forest Cover Type Prediction" of CS-412- Introduction to Machine learning Fall 2018 under Professor Xinhua Zhang.

1 Summary

The goal of this project was to predict the forest cover type of a given 30 x 30 meter cell of forest land and given geographical features. The actual forest cover type was determined from US Forest Services (USFS). Area includes four wilderness areas located in the Roosevelt National Forest of northern Colorado. Independent variables were then derived from data obtained from the US Geological Survey and USFS [2].

1.1 Classifications

The seven types of predictions 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. The test set (565892 observations) contains only the features.

1.2 Data Features

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

Hill shade_9am (0 to 255 index) - Hill shade index at 9am, summer solstice

Hill shade Noon (0 to 255 index) - Hill shade index at noon, summer solstice

Hillshade_3pm (0 to 255 index) - Hill shade 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

2 Approach

Since our dataset had a lot of entries, the first step was to evaluate the dataset and to filter the important features, and the important entries. A basic evaluation has revealed that the data set did not have any missing data, which eliminated the necessity to perform imputation on the dataset. We then proceeded with evaluating each individual feature and deciding if it would make a difference in the final machine learning models we develop. We discovered a few features which were not very frequent and could be removed. Although removing these entries would have resulted in a better learning model, we have decided to keep them since our dataset was huge and the result wouldn't have altered because of a few entries.

After performing feature selection, our next step was to plan our machine learning models. Since our task was a multi class classification problem, we decided on few models which we believed would perform better for a huge dataset. The Algorithms we have implemented includes Naïve Bayes, K-Nearest Neighbors, Logistic Regression, SGD Classifier, SGD Classifier with RBF Sampling, XGBoost Classifier, and Neural Networks. We used the implementations available on the scikit-learn library for Python, for most of the models.

The dataset consisted of features which were on very different scales and units, and training machine learning models on such data would result in bad accuracy scores. We have implemented data standardization on all the features which modified all the features to be on a similar and comparable scale. After implementing the models, the final step was to evaluate. There were multiple measures we have considered when evaluating the model. The accuracy, cross validation scores, speed, and the scope of the model working better in the future if tweaked and updated.

3 Evaluation

The following list shows the results we obtained from each model:

3.1 Naïve Bayes

The measured accuracy we obtained from using this model is 8.7%. This model also gave us a 3-fold cross-validation score of 13%.

By using Naïve Bayes, we obtained the following confusion matrix:

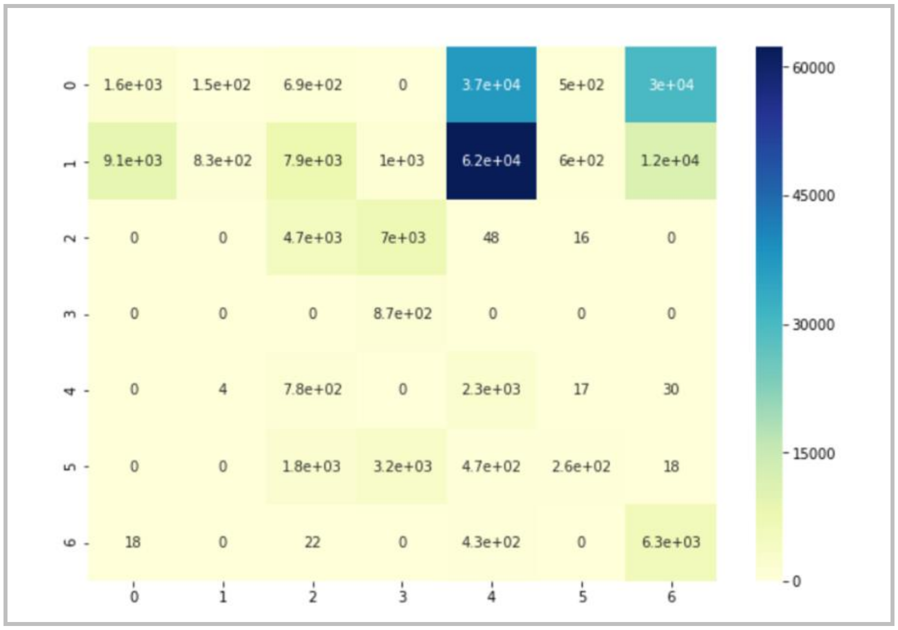

Figure 1: Confusion Matrix (Naïve Bayes)

3.2 SGD Classifier

Stochastic **G**radient **D**escent algorithm, inspired from the Machine Learning cheat sheet provided by the scikit-learn library [1], to try to improve the accuracy.

The training accuracy we obtained from using this model is 73.34%. We were able to get a testing accuracy of 70.19%. This model also gave us a 3-fold cross-validation score of 61%.

The following confusion matrix was obtained by using SGD Classifier:

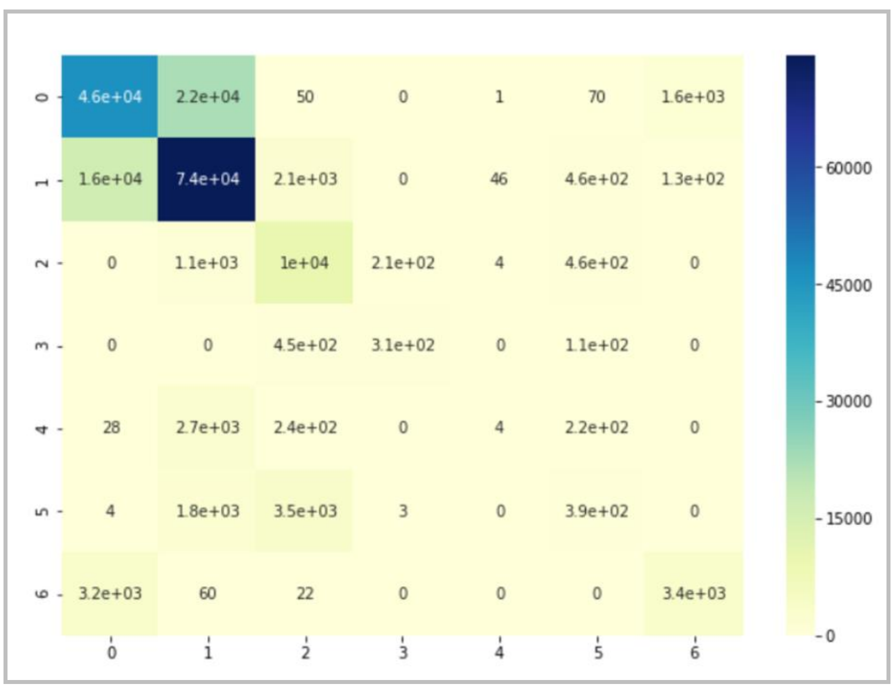

Figure 2: Confusion Matrix (SGD Classifier)

3.3 SGD Classifier with RBF Sampling (Kernel approximation)

This model uses an optimization technique which aims at increasing the accuracy of the SGD Classifier. It tries to map the 54 features to 100 features, and then uses the SGD Classifier.

This model resulted in a training accuracy of 51.34% and a testing accuracy 48.65%. This model also gave us a 3-fold cross-validation score of 59.88%.

3.4 K Nearest Neighbors

In this model, we implemented the K Nearest Neighbors classifier on the data set using the implementation provided by the scikit-learn library. But, Since the dataset was huge, and the algorithm is a lazy algorithm, we could not compute the confusion matrix, and cross validation scores.

This model gave us Train Accuracy of 94.34% and Test Accuracy of 92.33%.

3.5 Logistic Regression

In this model we performed Logistic Regression implementation available in the scikit-learn library. To prevent overfitting, we had to set the L2 regularization constant to 1.

This model gave us a Training accuracy of 31.99 %, Testing accuracy of 31.86 % and 3-fold cross validation score of 6.25.

3.6 XGBoost Classifier

Extreme Gradient Boosting, which uses a tree classifier, provided by xgboost library in python.

This model gave us a training accuracy of 74.44% and a testing accuracy of 74.55%. We also obtained a 3-fold cross-validation score of 61.8%.

3.7 2 layered Neural Network

We also decided to implement a simple 2 layered Neural Network. For this we used the keras library to create layers and computations. The Structure of our neural network consisted 54 inputs and a Fully Connected hidden layer with 8 neurons and which is further connected to an output layer of 7 neurons., which predicts using the softmax activation function.

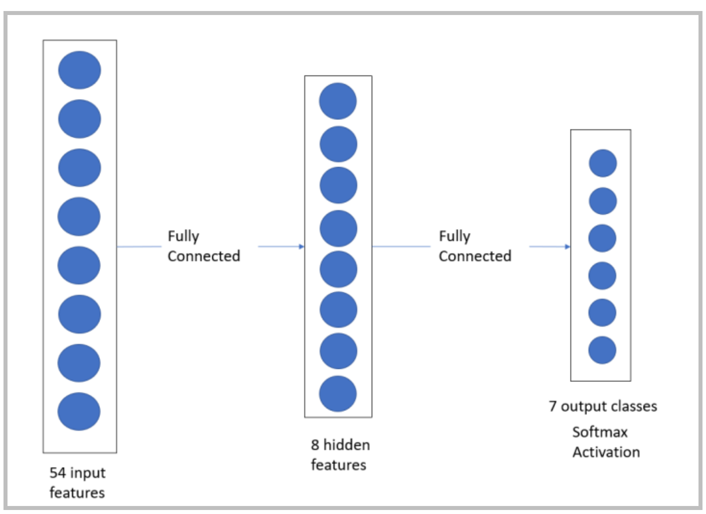

Figure 3: 2 Layered NN

4 Result

- K Nearest Neighbours provided the best accuracy scores.
- The neural network has the stable testing and training scores, which can be further improved with a different structure of neural network and different activation function.
- XG Boost is another promising algorithm with good training and testing accuracies.
- Naïve Bayes was not a good model, because the data is not expected to be conditionally independent which is the basic principle of the algorithm.

4.1 What we learned

- We were able to try out various machine learning models, and able to relate it to our database.
- Feature selection and data standardization was very important and helped the accuracies much better.
- Learned how to evaluate the performance of a model, and how to decide on what model is the better one for a given dataset.

4.2 What can be done in the future

- Future selection can be done in a much better way, by dropping few entries which are not very important.
- Different structures of neural networks can be tried because it was a promising model.
- Confusion matrix and cross validation should be performed on K Nearest Neighbours Classifier to evaluate it much better as a viable model for the data set.
- Try more approaches like Random Forests, and other decision trees, etc.

References

[1] Scikit-Learn Library - https://scikit-learn.org/stable/tutorial/machine_learning_map/

[2] DataSet - Bache, K. & Lichman, M. (2013). UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Scienc