Forest Cover Type Prediction

Example from past class

1. Problem definition

In this project, the study area includes four wilderness areas located in National Forest, and each observation is a 30m x 30m patch. We are going to predict the predominant kind of tree cover of each patch from raw form data[2], which contains binary columns of qualitative independent variables such as wilderness areas and soil type. We will use the training set to set the model, and predict an integer classification for the forest cover type of each line in test set, and the seven types are:

1 - Spruce/Fir; 2 - Lodgepole Pine; 3 - Ponderosa Pine; 4 - Cottonwood/Willow;

5 – Aspen; 6 - Douglas-fir; 7 – Krummholz

2. Background

In machine learning and statistics, classification is the problem of identifying to which of a set of categories a new observation belongs, on the basis of a training set of data containing observations whose category membership is known. Examples of classification algorithms include: Naive Bayes classifier, k-nearest neighbor, Decision trees, Neural networks, etc.

But classifier performance depends greatly on the characteristics of the data to be classified and there is no single classifier that works best on all given problems. For example, knn needs expensive testing of each instance, which is problematic for datasets with a large number of attributes. And it's sensitive to noisy attributes and unbalanced datasets, so it is not suitable for this case. Our aim is to find the suitable classifier for this problem, and evaluate its performance and accuracy.

3. Dataset

Our problem is from Kaggle[1] and dataset is from UCI machine learning repository[2]. The training set (15120 observations) contains both features and the Cover_Type. The test set (565893 observations) contains only the features.

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 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

4. Method

a. Tools

scikit-learn[3]

Scikit-learn is an open source machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, logistic regression, naive Bayes, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

We use scikit-learn to implement decision trees and random forest in this project.

Pandas[4]

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

It's a good tool for reading and writing data between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format.

We use pandas to load data from CSV files in this project.

b. Algorithms

In this project, we used two algorithms to implement the prediction, decision trees[5] and random forest[6].

Decision Trees

Decision trees are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. We chose decision tree as one of the algorithms because this method is simple to understand and to interpret, requires little data preparation and able to handle both numerical and categorical data.

The cost of using the tree to predict data is logarithmic in the number of data points used to train the tree.

We use DecisionTreeClassifier from scikit-learn[3] package, a class that is capable of performing multi-class classification on a dataset to do the classification.

This classifier takes as input two arrays: an array X of size [n_samples, n_features] holding the training samples, and an array Y of integer values, size [n_samples], holding the class labels for the training samples. After being fitted, the model can then be used to predict new values.

But sometimes using decision-tree can create over-complex trees that do not generalize the data well, which is called over fitting, and it can be unstable because small variations in the data might result in a completely different tree being generated, so we tried random forest to improve the performance.

Random Forest

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

We use sklearn.ensemble.RandomForestClassifier from scikit-learn[3] package to do the classification.

ensemble.RandomForestClassifier(n_estimators = 500, n_jobs = -1)

n_estimators stands for the number of trees in the forest.

n_jobs stands for the number of jobs to run in parallel for both fit and predict. If -1, then the number of jobs is set to the number of cores.

We use fit(X, y, sample_weight=None) to build a forest of trees from the training set (X, y) and predict(X) to predict class for X.

5. Experiment: performance analysis and results

Control the size of the test dataset

In order to analyze the difference of the results easily. We keep the same size of the test data, which is 10000, and try different size of the training datasets to test the running time. The red line is the running time of the DT and the blue line is the running time of the RF. The chart is as follows:

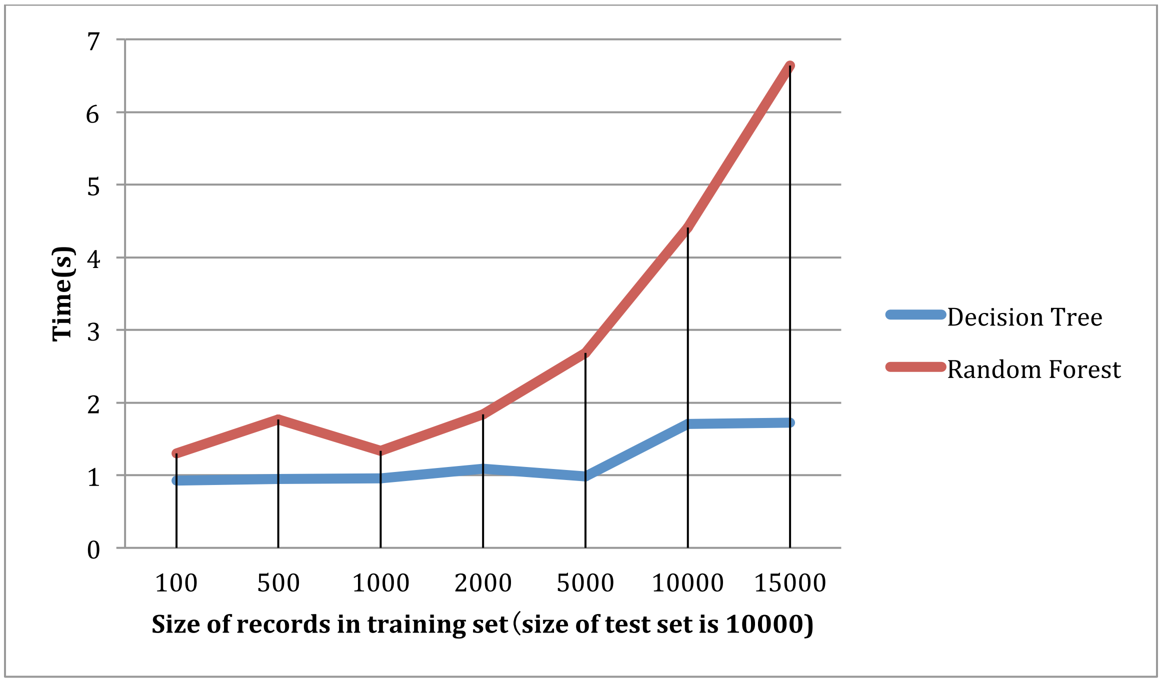

Chart1. The running time of two algorithms when changes the size of records in training set and control the size of test set

Through the chart we can see that as the number the records of training set grow larger, the running time of both grow larger, however, RT's line grow much faster and is always above the DT's line.

Control the size of the training dataset

In this case, we keep the same size of the training datasets, which is 10000, and try different size of the test datasets to test the running time. The chart is as follows:

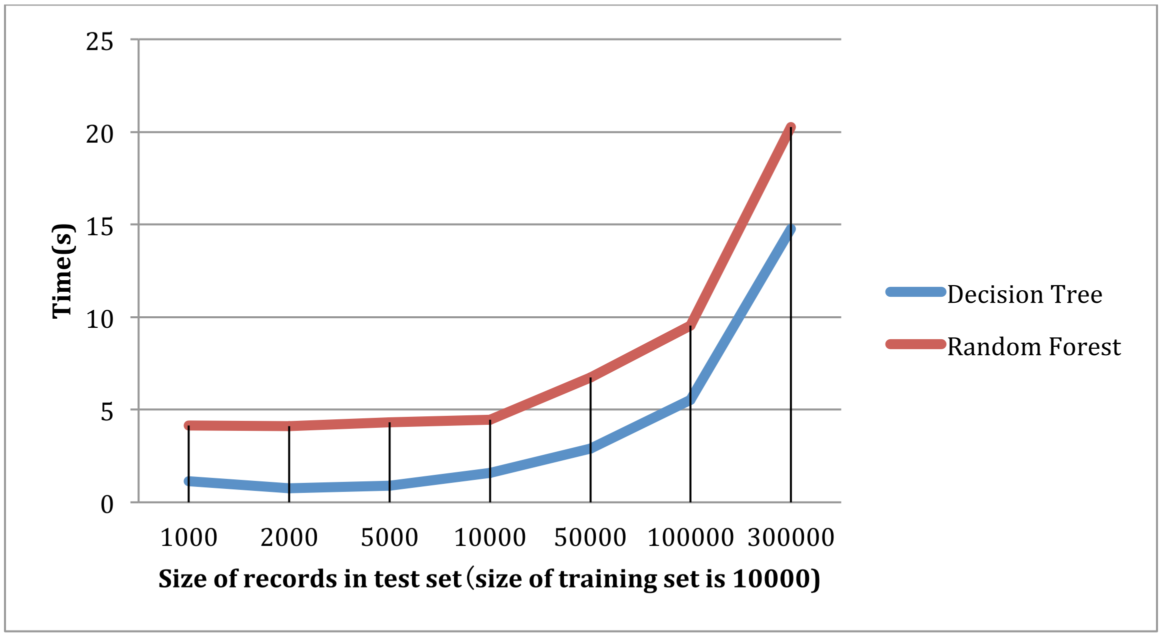

Chart2. The running time of two algorithms when changes the size of records in test set and control the size of training set

Through this chart, we get that both lines grow fast as the number of the records grow and the red line is always above the blue line, however the difference between them almost keeps the same with the same size of records of test datasets.

Accuracy Analysis of Decision Trees Method and Random Forest Method:

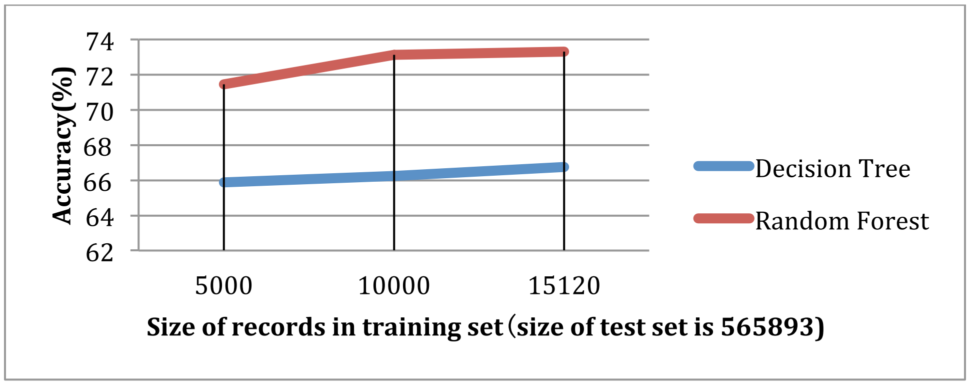

Chart3. The accuracy of two algorithm when changes the size of records in training set and control the size of test set

Through the chart, we get that the red line is always way above the blue line, which indicates that the accuracy of RF is much better than DTs. And as the size of records grows, the accuracy both grows.

	Time(s)	Accuracy(%)
Decision Trees	28.112	66.745
Random Forest	37.446	73.119

Table1.Time and accuracy of decision trees and random forest using the whole dataset

We run all the record this time, through this table, we can get that although the RT algorithm takes a little bit longer than DTs algorithm, they don't have big difference between them. Thus RF algorithm is better and more accurate than DTs algorithm.

6. Conclusion

Decision Trees are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. The deeper the tree, the more complex the decision rules and the fitter the model. What's more, decision trees tend to overfit on data with a large number of features. Getting the right ratio of samples to number of features is important, since a tree with few samples in high dimensional space is very likely to overfit.

In random forests, each tree in the ensemble is built from a sample drawn with replacement 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 but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model.

In the both decision trees method and random forest method, as the size of datasets grows bigger, the running time becomes larger. If we use the same size between these two methods, the average running time is larger in the random forest method because random forest method generates better model, thus take relatively long running time.

7. References

[1]Forest Cover Type Prediction. https://www.kaggle.com/c/forest-cover-type-prediction

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

[3]scikit-learn.

http://scikit-learn.org/stable/supervised_learning.html#supervised-learning

[4]Pandas. http://pandas.pydata.org

[5]Decision Trees. http://en.wikipedia.org/wiki/Decision_tree

[6]Random Forest. http://en.wikipedia.org/wiki/Random_forest

[7]Blackard, Jock A. and Denis J. Dean. 2000. "Comparative Accuracies of Artificial Neural Networks and Discriminant Analysis in Predicting Forest Cover Types from Cartographic Variables." Computers and Electronics in Agriculture 24(3):131-151.