Machine Learning Engineer Nanodegree

Capstone Project Report

Bin Luo
January 11, 2016

Definition

Domain Background

The United States Forest Service (USFS) is an agency of the U.S. Department of Agriculture that administers the nation's 154 national forests and 20 national grasslands, which encompass 193 million acres (780,000 km^2). Major divisions of the agency include the National Forest System, State and Private Forestry, Business Operations, and the Research and Development branch. Managing approximately 25% of federal lands, it is the only major national land agency that is outside the U.S. Department of the Interior.

In our project, we used the data/features was derived from the US Geological Survey and USFS, the forest types, our the targeted predication results, was determined by USFS.

Problem Statement

The problem this project is trying to solve is 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 study area where the data derived from includes four wilderness areas located in the Roosevelt National Forest of northern Colorado. Each observation is a 30m x 30m patch. Our goal is to predict 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

This is a problem of supervised learning – and more specifically multi-categories classification problem, there are many methods could be explored, will be discussed in the below section. Being able to explore different algorithms is the major motivation I selected this competition as this will help me to establish a systematic view of the supervised learning techniques, and get some hands-on experience on the algorithms I was not able to try out during the courses.

Benchmark Model

To benchmark our model, we will use Kaggle's benchmark model to check our predication accuracy. In Public Leaderboard, the **top 50th**'s performance is **0.81626**, we will compare our final solution with the leaderboard to see where we stand. We will aim to have our final solution have an accuracy higher than 50th.

Evaluation Metrics

Since that each instance would only have one label, and the submissions are evaluated by Kaggle based on a simple multi-class classification accuracy, which is defined as the number of correctly classified instances divided by the total number of instances (the# of predictions is 565,892), I think the accuracy of prediction is good as the evaluation metrics.

Analysis

Datasets and Inputs description

In this study, we are using the publicly available dataset downloaded from Cagle.

The training set (15120 observations) contains both features and the Cover_Type. The test set contains only the features. We must predict the Cover_Type for every row in the test set (565892 observations).

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. Below are the details about the metadata.

Feature	**Description**	**Type**
Elevation	Elevation in meters	numerical

Aspect	Azimuth from true north	numerical
Slope	Slope in degrees	numerical
Horizon- tal_Distance_To_Hydrology	Horz Dist to nearest surface water features	numerical
Vertical_Distance_To_Hydrology	Vert Dist to nearest surface water features	numerical
Horizon- tal_Distance_To_Roadways	Horz Dist to nearest roadway	numerical
Horizon- tal_Distance_To_Fire_Points	Horz Dist to nearest wildfire ignition points	numerical
Hillshade_9am	Hillshade index at 9am, summer solstice 0 to 255 index	numerical
Hillshade_Noon	Hillshade index at noon, summer solstice 0 to 255 index	numerical
Hillshade_3pm	Hillshade index at 3pm, summer solstice 0 to 255 index	numerical
Wilderness_Area	Wilderness area designation 4 binary columns, 0 = absence or 1 = presence	nominal
Soil_Type	Soil Type designation 40 binary columns, 0 = absence or 1 = presence	nominal
Cover_Type	Forest Cover Type designation 7 types, integers 1 to 7	nominal

Data exploration

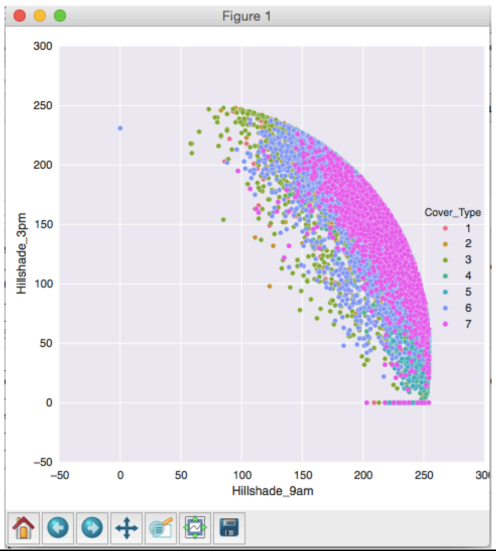
1. **Data quality check**
   The data contains no null values or missing value, all attributes have a count of 15120; also, there is no imbalance among the class distribution, 7 classes have equal instances; however, there are two attributes ('Soil_Type7', 'Soil_Type15') have constant values, we will exclude them during the training.
2. **Feature correlation**
   Explore the correlation between all the features with continuous values by calculating the co-efficient for all the feature pairs, below are 7 feature pairs have co-efficient more than 0.5, this shows strong correlation, and represents an opportunity to reduce the feature set through transformations such as PCA or feature selection.

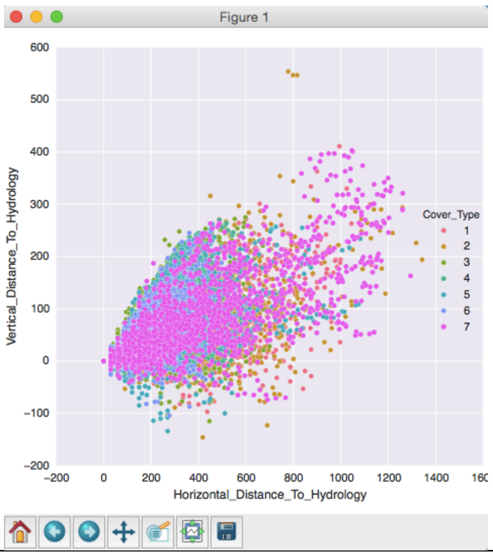
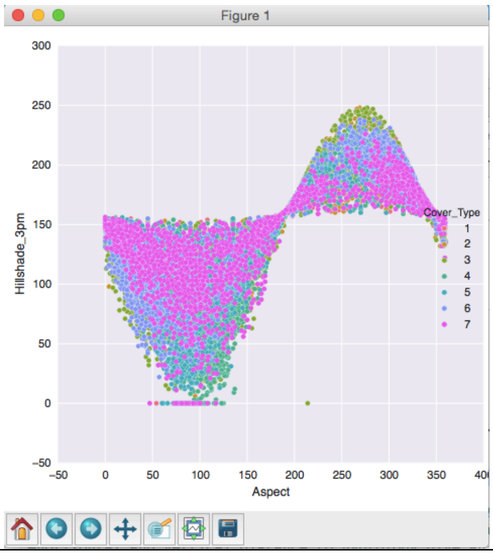
Feature1	**Feature 2**	**Co-efficient**

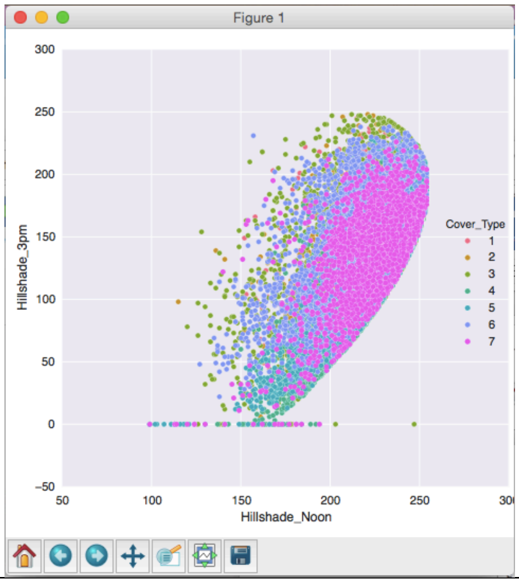
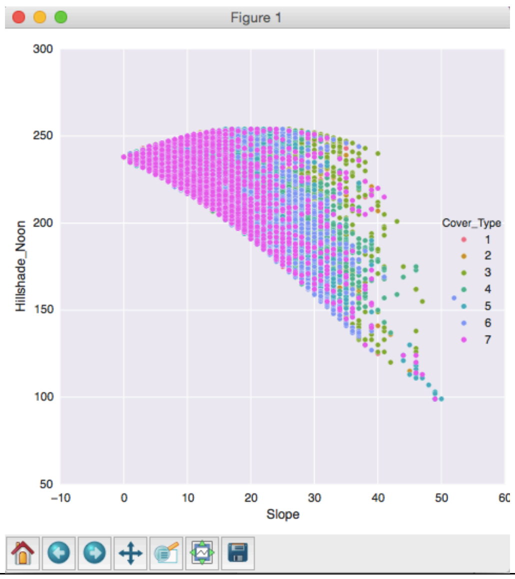
Hillshade_9am	Hillshade_3pm	-0.78
Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrology	0.65
Aspect	Hillshade_3pm	0.64
Hillshade_Noon	Hillshade_3pm	0.61
Slope	Hillshade_Noon	-0.61
Aspect	Hillshade_9am	-0.59
Elevation	Horizontal_Distance_To_Roadways	0.58

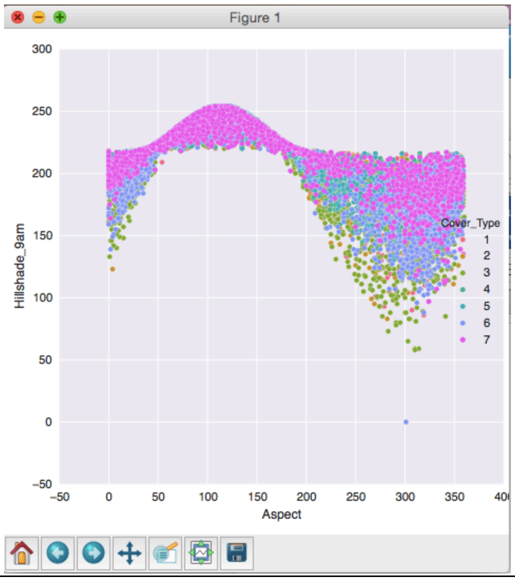
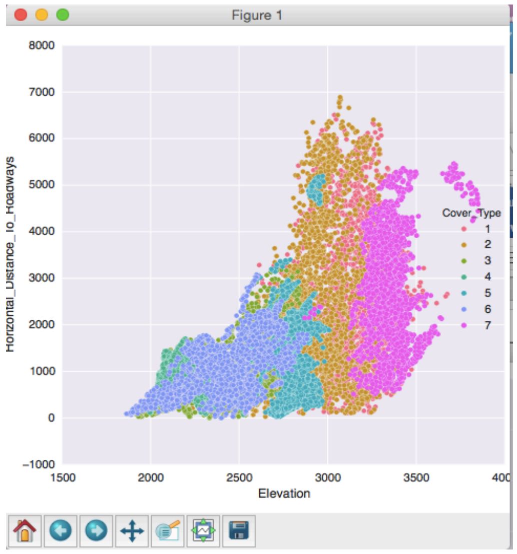
Exploratory Visualization

Apply a scatter plot to all the high-correlation feature pairs selected by above steps, below plot shows which class does a point belong to. The class distribution overlaps in the plots.

Scatter plot	**Notes**
	Hillshade shows a nice ellipsoid patterns with each other

	Vertical distance and historical distance almost shows linear pattern
	Aspect and Hillshades attributes form a sigmoid pattern, maybe some sigmoid kenel with prediction might fit.

	Hillshade shows a nice ellipsoid patterns with each other
	Hillshade and Slope shows a negative linear pattern (not very strong one though)

	Aspect and Hillshades attributes form a sigmoid pattern.
	Horizontal distance with Elevation shows a strong classification relationship – each class split based on Elevation value clearly.

Algorithms and Techniques

This is a classic supervise learning algorithm, and it is a classification problem, since the output of the prediction would be types – discrete values. The most important criterias for selecting algorithms is to be able to support multi-class classification, not regression.

Algorithms	Descriptions	Strength & weakness
LinearDiscriminantAnalysis	A classifier with a linear decision boundary, generated by fitting class conditional densities to the data and using Bayes' rule. Default parameter: None	+ inherently multiclass, and have no hyperparameters to tune
LogisticRegression	Generally used to estimate the probability of a binary response based on the feature. In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme Inital parameter: n_jobs = -1, randomstate = 42, C = 100 (less regularization since we have enough training set)	+ does not require the feature to be normally distributed, and it return the probability of each label; and it is easy to computed and understood. - normally required more data for achieve a stable prediction results
K-nearest neighbor	An instance is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. Initial parameter: n_neighbors=1 (in our case, one instance only predicted based on the closet neighbor)	+ Effective for large training dataset + Robust to noisy data - need to determine how to calculate distance - computation requirement is high
CART(DecisionTree)	A non-parametric supervised learning method used for classification. Split data based on information gain. 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. Initial parameter: Max_depth=13 (randomly pick any value between 10-15)	+ Simple to understand and to interpret. Trees can be visualised. + Able to handle both numerical and categorical data - easy to overfitting - Decision trees can be unstable because small variations in the data might result in a completely

		different tree being generated.
SVM	An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. Default parameter: C=10 (required some regularization to avoid overfitting)	Strengths: good for multi-dimensional input space, low error rates and easy to compute. Weakness: sensitive to parameters and kernel choice.
RandomForest	General application: an ensemble learning method by constructing many decision tree at training time for classification, regression and other tasks. Default parameter: n_estimators = 100	Strengths: control overfitting, easy to explain the the decision process, fast to train Weekness: sometime slow to create predictions once trained. More accurate ensembles require more trees.
ExtraTreesClassifier	Extra-trees differ from classic decision trees in the way they are built. When looking for the best split to separate the samples of a node into two groups, random splits are drawn for each of the max_features randomly selected features and the best split among those is chosen. Initial parameter: n_estimators=100	+ avoid overfitting + lesser computational cost vs RandomForest - hyperparameter tuning take long time
AdaBoostClassifier	An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases. **Initial parameter:** n_estimators=100	+ able to correct upon its mistakes - noisy data/outliers impacts a lot on the outcome
GradientBoostingClassifier	GradientBoosting produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. it generalizes them by allowing	+ new tree helps to correct errors made by previously trained tree, normally better performance

	optimization of an arbitrary differentiable loss function. Initial parameter: max_depth=13	- take longer to train - prone to overfitting - hard to tune

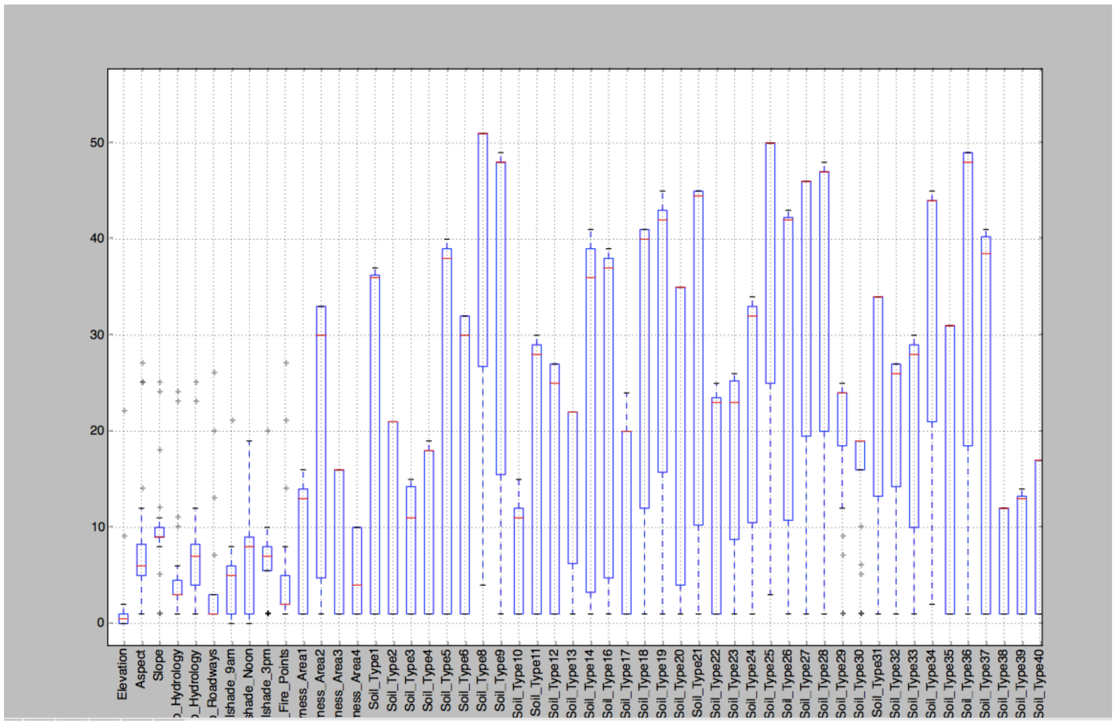
Methodology

Data Preprocessing

1. **Data cleaning**
   Drop the columns with constant data and also the 'ID' columns
2. **Data transformations**
   Since the value of the first 10 features are in different scale, apply 3 different scalers to normalized the values or the first 10 features and then concatenate with the rest of feature values.
3. **Feature selection**
   Since there are high correlations observed from prior data expletory, there are possibility to use a subset of features to make the train and predict the data. 3 feature selection techniques were leveraged to prioritize the feature and looking at different ratios – 50% , 75%, 100%
   - RandomForest.feature_importances_
   - ExtraTress.feature_importances
   - RecrusiveFeatureElimination(RFE).ranking_

   Below plot summarizes the rankings according to the standard feature selection techniques, top ranked attributes are ...

   - first 10 attributes
   - Wilderness_Area1,4
   - Soil_Type 3,4,10,38-40.

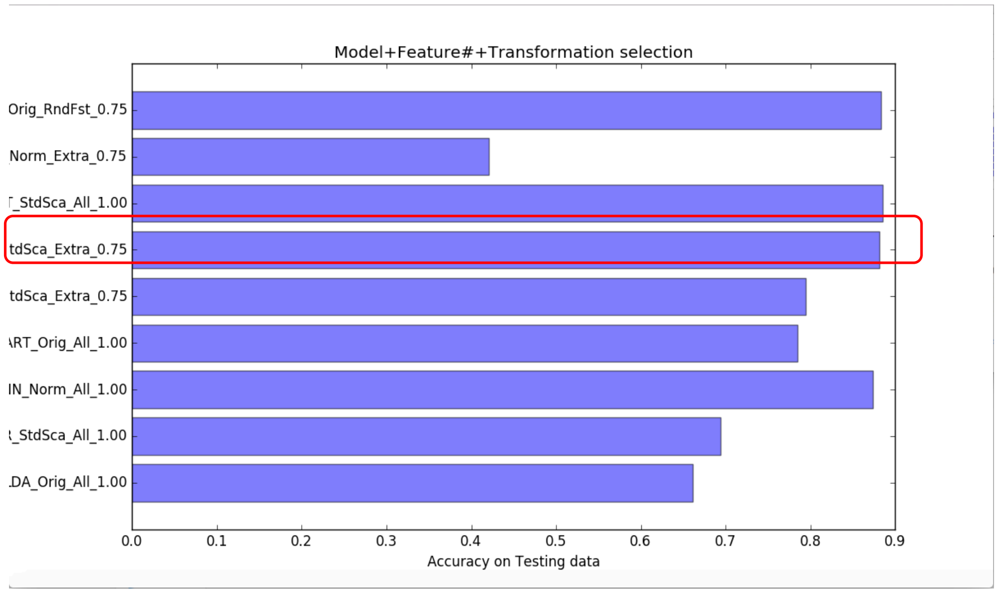


Implementation

1. **Model selection**
   This is a typical multi-class classification supervised learning problem, there are numerous of algorithms could be used for this problem. Another complexity is that there are many different ways to train the model, considering different types of transformed data and different #of features to use. Below table shows the all combinations I had tried out to select the model.

Algorithms	Transformed data	#of feature used	Total # of combination
• **LinearDiscriminantAnalysis** • **LogisticRegression** • **K-nearest neighbors** • **CART(DecisionTreeClassifier)** • **SVM** • **RandomForest** • **ExtraTreesClassifier** • **AdaBoostClassifier** • **GradientBoostingClassifier**	• Original data • Standard Scaler • MinMax Scaler • Normalizer	• All features • Top 75% based on ExtraTree • Top 75% based on RandomForest • Top 75% based on RFE • Top 50% based on ExtraTree • Top 50% based on RandomForest • Top 50% based on RFE	**216** (9x4x6)

For each algorithm, I captured the combinations with top performance, below are the plot for to identify the best of from all algorithms, based on which we select the ExtraTreeClassifier + StadScaler Data + All feature which has the best performance on testing data.



2. Feature Engineering

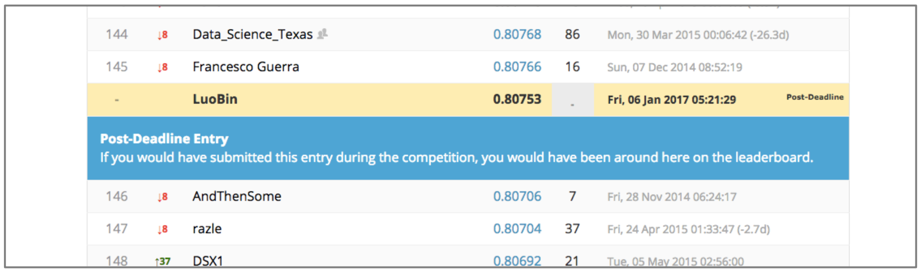
I used the selected train model to make the first submission to Kaggle, however the result is around 75.1% accuracy, which ranks at around 800 in the leaderboard. This is too far away from my goal. So, I started considering add more features to improve the performance.

The data has 5 features in meters, I have done some calculation and combination based on these attributes. Distance features can be divided into two types: vertical and horizontal. For all vertical distance features I combined them pair-wise with plus and minus operators, and the same for all horizontal distance features. Using horizontal distance and vertical distance to hydrology I calculated the Euclidean distance to hydrology. Below are the features I added.

['Ele_minus_VDtHyd'] = ['Elevation'] - ['Vertical_Distance_To_Hydrology']
['Ele_plus_VDtHyd'] = ['Elevation'] + ['Vertical_Distance_To_Hydrology']
['Distance_to_Hydrolody'] = (['Horizontal_Distance_To_Hydrology']**2 + ['Vertical_Distance_To_Hydrology']**2)**0.5
['Hydro_plus_Fire'] = ['Horizontal_Distance_To_Hydrology'] + ['Horizontal_Distance_To_Fire_Points']
['Hydro_minus_Fire'] = ['Horizontal_Distance_To_Hydrology'] - ['Horizontal_Distance_To_Fire_Points']
['Hydro_plus_Road'] = ['Horizontal_Distance_To_Hydrology'] + ['Horizontal_Distance_To_Roadways']

['Hydro_minus_Road'] = ['Horizontal_Distance_To_Hydrology'] - ['Horizontal_Distance_To_Roadways']
['Fire_plus_Road'] = ['Horizontal_Distance_To_Fire_Points'] + ['Horizontal_Distance_To_Roadways']
['Fire_minus_Road'] = ['Horizontal_Distance_To_Fire_Points'] - ['Horizontal_Distance_To_Roadways']

After adding the constructed new features to the dataset, I got a score of 0.80753 in 2nd Kaggle submission. This score is about 0.05 higher than the highest score mentioned above, which is a big improvement and allowed me to jump to 145th place on the leaderboard.



144	↓8	Data_Science_Texas	0.80768	86	Mon, 30 Mar 2015 00:06:42 (-26.3d)
145	↓8	Francesco Guerra	0.80766	16	Sun, 07 Dec 2014 08:52:19
-		**LuoBin**	**0.80753**	-	**Fri, 06 Jan 2017 05:21:29** Post-Deadline

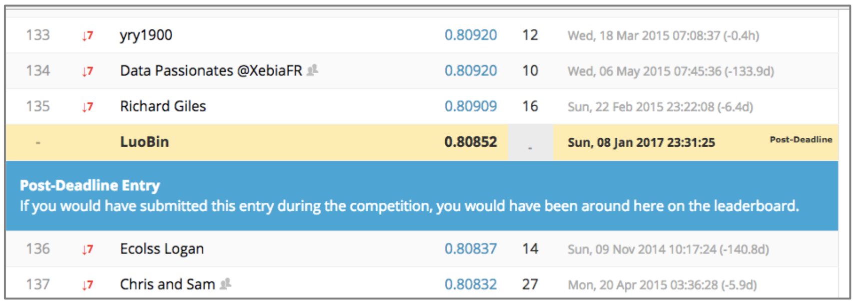
Post-Deadline Entry
If you would have submitted this entry during the competition, you would have been around here on the leaderboard.

146	↓8	AndThenSome	0.80706	7	Fri, 28 Nov 2014 06:24:17
147	↓8	razle	0.80704	37	Fri, 24 Apr 2015 01:33:47 (-2.7d)
148	↑37	DSX1	0.80692	21	Tue, 05 May 2015 02:56:00

Refinement

1. **Parameter tuning**
   Though there was a huge improvement by adding more features, however the performance was not met our target – 50th ranking, which was 0.81626 accuracy. Since it was not a big difference, so I think tuning the ExtraTreeClassifier with GridSearch and Cross Validation might help. Below are the parameters I was tuning:
   - K-fold: 10
   - 'n_estimators': [150],
   - 'max_features': ['auto',0.6,0.8],
   - 'min_samples_split': [2,4,8,16,32,64],
   - 'min_samples_leaf' : [1,10,25,50],
   - 'n_jobs': [-1],
   - 'random_state': [42]

   The best parameters from the GridSearch & CrossValidation are {'n_jobs': -1, 'min_samples_leaf': 1, 'n_estimators': 150, 'min_samples_split': 2, 'random_state': 42, 'max_features': 0.6}, by using the tuned model, I did a 3rd submission, this gave a leaderboard score of 0.80852, which took my ranking up to 135

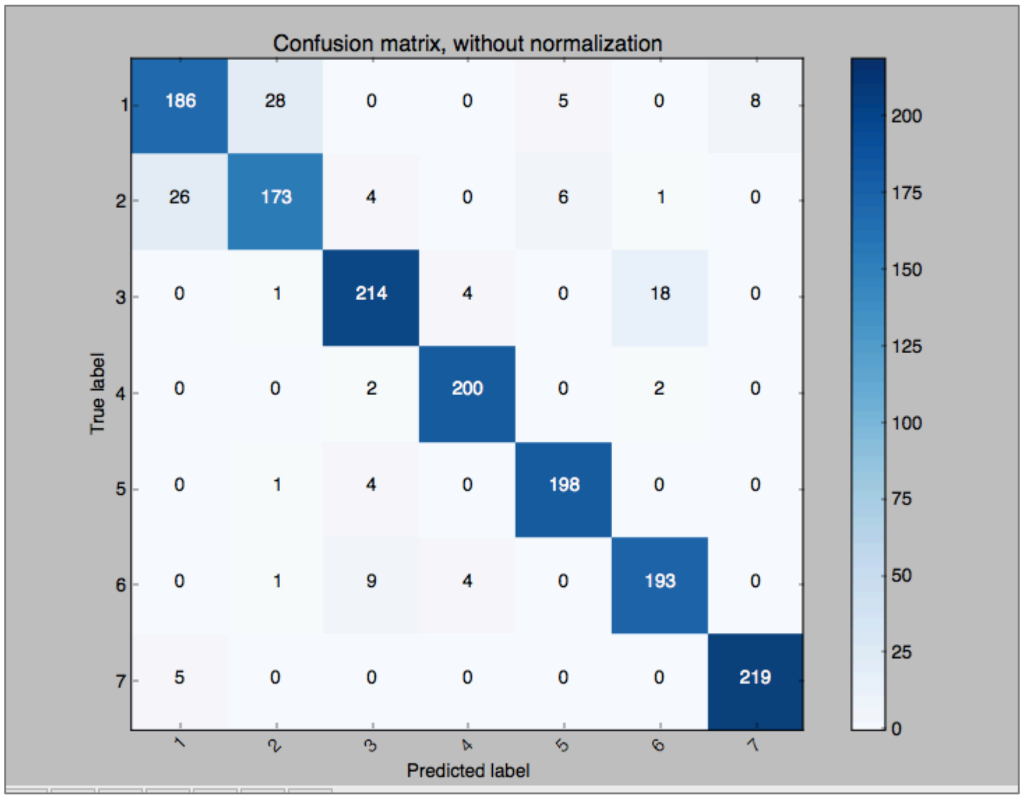


133	↓7	yry1900	0.80920	12	Wed, 18 Mar 2015 07:08:37 (-0.4h)
134	↓7	Data Passionates @XebiaFR	0.80920	10	Wed, 06 May 2015 07:45:36 (-133.9d)
135	↓7	Richard Giles	0.80909	16	Sun, 22 Feb 2015 23:22:08 (-6.4d)
-		**LuoBin**	**0.80852**	-	**Sun, 08 Jan 2017 23:31:25** Post-Deadline

Post-Deadline Entry
If you would have submitted this entry during the competition, you would have been around here on the leaderboard.

136	↓7	Ecolss Logan	0.80837	14	Sun, 09 Nov 2014 10:17:24 (-140.8d)
137	↓7	Chris and Sam	0.80832	27	Mon, 20 Apr 2015 03:36:28 (-5.9d)

2. **One-vs-All prediction**
   135 still has 0.7% distance from where my goal is. In order to find out how I can improve the performance, I did a confusion matrix to deep dive on what are classes I was mistakenly classifying.



Confusion matrix, without normalization

True label \ Predicted label	1	2	3	4	5	6	7
1	186	28	0	0	5	0	8
2	26	173	4	0	6	1	0
3	0	1	214	4	0	18	0
4	0	0	2	200	0	2	0
5	0	1	4	0	198	0	0
6	0	1	9	4	0	193	0
7	5	0	0	0	0	0	219

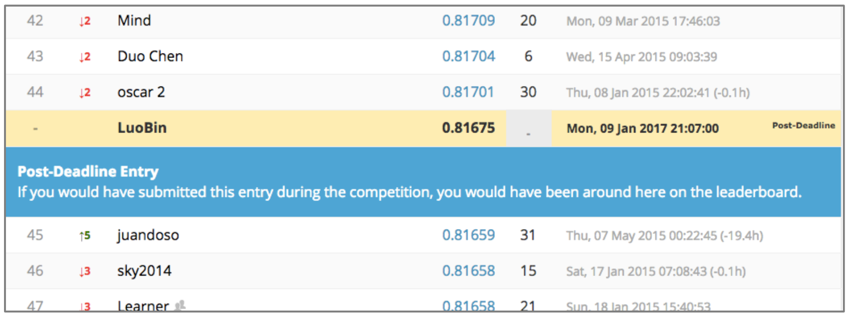
Above figure shows a confusion matrix that is produced by my Extra Trees model on 10% the testing data. It shows that type 1 and type 2 are where my model made most mistakes, some type 1s are also misclassified as type 5 and 7; some type 2s are misclassified as type 4, 5 or 6. But the prediction accuracy of types 7, 6, 5, 4 are high, so I decided to predict classes of 7,6,5,4 first by used an ordered One-vs-All approach for classes 7, 6, 5, 4 to prevent mis-classifications for types 1, 2, and 3. The prediction process I describe is as below:

1) In training dataset, except for class 7, label all other classes to be negative label, i.e. -1, train the model with all the training set, then predict all the testing set to get class 7 prediction.
2) To predict class 6, train the model again but excluding all the training instances of label 7, and then excluded the testing instances already been predicted as class 7 in step 1, used the model to predict the rest of test data to get class 6 prediction.
3) Repeat the above step 1 & 2 until to get all the test instance classified. Below table shows all the training & predict data set have been used for all the seven classes. *(x_i represent the train data with label as class i , y_i represent the testing set data are predicted as class i)*

Class (Cover type)	Class to predict	Training set	Test set
7	y_7	x_7 vs. $(x_1+x_2+x_3+x_4+x_5+x_6)$	y
6	y_6	X_6 vs. $(x_1+x_2+x_3+x_4+x_5)$	$y - y_7$
5	y_5	X_5 vs. $(x_1+x_2+x_3+x_4+x_5)$	$y - y_7 - y_6$
4	y_6	X_4 vs. $(x_1+x_2+x_3+x_4)$	$y - y_7 - y_6 - y_5$
3,2,1	$y_3, y_2, y_1,$	$x_1+x_2+x_3$	$y - y_7 - y_6 - y_5 - y_4$

4) Assembly all the predicted result and get the full predicted test data and make submission.

This approach gave me an accuracy of 0.81675 on another kaggle submission, which brought me to 45th ranking in the leaderboard!



42	↓2	Mind	0.81709	20	Mon, 09 Mar 2015 17:46:03	
43	↓2	Duo Chen	0.81704	6	Wed, 15 Apr 2015 09:03:39	
44	↓2	oscar 2	0.81701	30	Thu, 08 Jan 2015 22:02:41 (-0.1h)	
-		**LuoBin**	**0.81675**	-	**Mon, 09 Jan 2017 21:07:00**	**Post-Deadline**

Post-Deadline Entry
If you would have submitted this entry during the competition, you would have been around here on the leaderboard.

45	↑5	juandoso	0.81659	31	Thu, 07 May 2015 00:22:45 (-19.4h)
46	↓3	sky2014	0.81658	15	Sat, 17 Jan 2015 07:08:43 (-0.1h)
47	↓3	Learner	0.81658	21	Sun, 18 Jan 2015 15:40:53

Results

Model Evaluation and Validation

During development, a 10% of the training data was used as validation set to evaluate the models, together with different types of transformed data and different sets of features to make the prediction . The ExtraTrees+All features in StandardScaled data were chosen because they performed the best amongst the 216 tried combinations. To identify the best heperparameters, GridSearch was used to try out all the options for key parameter for Extra Trees model.

To verify the robustness of the final model, I use a cross validation technique (StratifiedShuffleSplit) on the dataset to ensure that the model generalizes well by using the entire dataset for both training and testing (10 folds data). The model consistently predicts the data with around 81% accuracy.

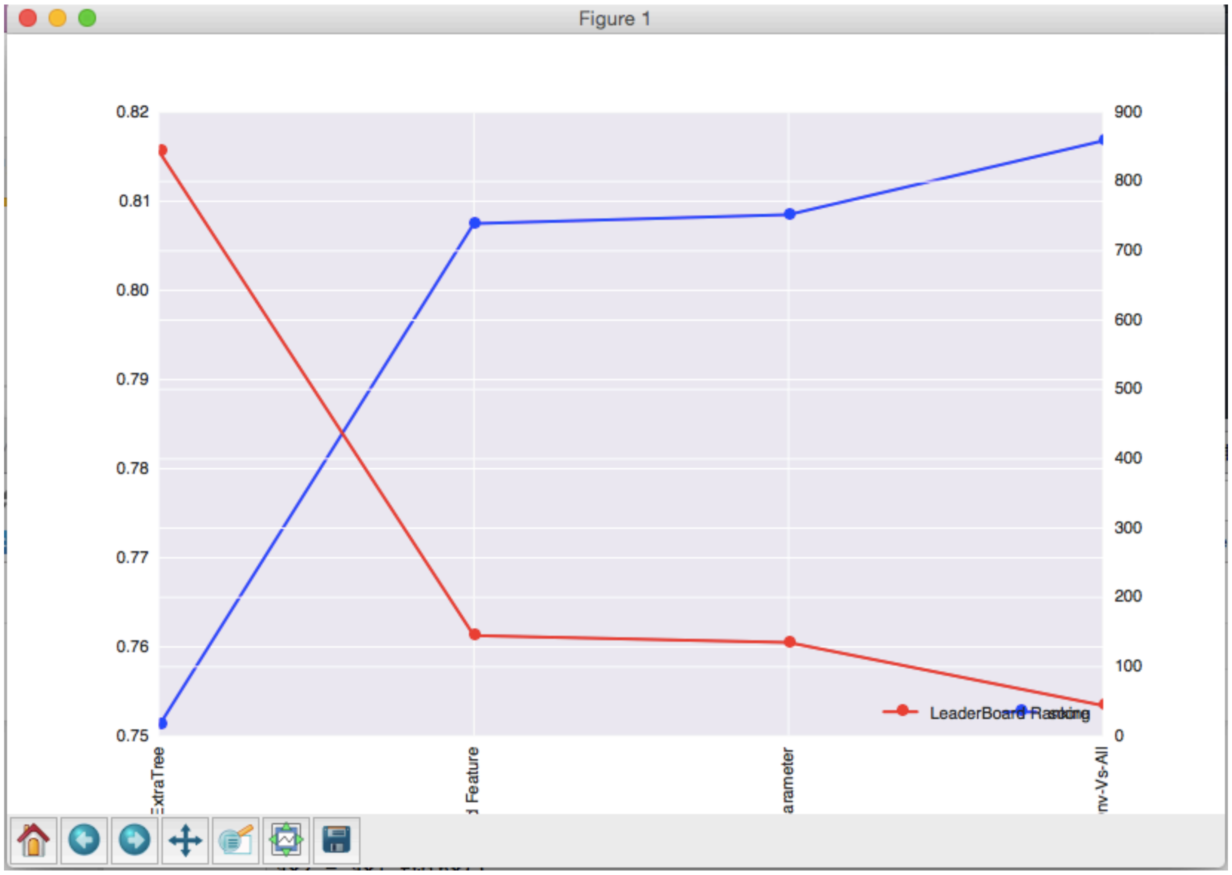
Justification

Applying the model training from above technique on the testing dataset downloaded from kaggle (565,892 instances), I got the following result: 81.675% accuracy with a final ranking of 45th which is better that our benchmark 50th. In the final prediction, we did not use the StandardScale data just because it would take more time to train, however since the performance meet our target so it could consider as an improvement opportunity.

Conclusion

Free-Form Visualization

To have a clear insight of the total process of this competition, I plotted 4representative scores and corresponding standings in Kaggle leaderboard in below figure to help me to summary and reflect on the end-to-end process.



The goal of this challenge is to predict forest cover type based on cartographical data which has 54 features. Different data transformation, preprocessing strategies, feature engineering techniques and classification models are used to make predictions. the Extra Trees model with Original data set and all features included was chosen as the final model. Adding new feature gave the most significant improvement. Lastly, the One-vs-All prediction strategy obtained the best result.

Reflection

1. Feature engineering is a great technique to improve significantly on the prediction performance, it required domain knowledge, so it would be good to consult Subject

Master Experts on the data we are using to make prediction.
2. Parameter tuning is necessary but don’t expect too much improvement, it could be done later phased in the project.
3. It is always good to start with model try different combination to identify the model you want to go with further, but also should considering data transformation and feature selection, because some model perform very differently on different transformed data.
4. Feature normalization and selection are good techniques to try out but don’t expect improvement every time.

References

[1] http://scikit-learn.org/stable/modules/feature_selection.html

[2] https://en.wikipedia.org/wiki/Bootstrap_aggregating

[3] http://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html

[4] http://machinelearningmastery.com/feature-selection-in-python-with-scikit-learn/

[5] https://www.kaggle.com/c/bnp-paribas-cardif-claims-management/forums/t/19841/extra-trees-classifier-getting-worse-feature-selection

[6] http://machinelearningmastery.com/an-introduction-to-feature-selection/

[7] http://scikit-learn.org/stable/modules/multiclass.html