# Assignment 1. Predicting cover of forest

[Report]

Yerlan Idelbayev
University of California San Diego
yidelbayev@ucsd.edu

## ABSTRACT

This paper is a report for CSE255 assignment #1. CoverType data set from UCI ML Archive, currently run as a competition on Kaggle, has been chosen and analysed. Results of this paper hugely outperform previous reported results in literature and coincides with current Kaggle leaderboard results.

## 1. DATASET DESCRIPTION

For this assignment I have chosen a kaggle competiion 'Forest Cover Type Prediction'[1]. This data is obtained from the US Geological Survey (USGS) and the US Forest Service (USFS) and includes four wilderness areas located in Roosevelt National Forest of northern Colorado, and provided by Machine Learning Laboratory of University of California Irvine[1]. Dataset contains 581k entries with 54 attributes each. However, there are only 12 real features because two of them are represented as a vector opposed to number notation. (see Table 1)

Each entry is observation on $30 \times 30$ m patch of forest land and goal of the competition is to predict cover type of this patch. Training set is chosen in so fashion that each class has the same number of observations.

### 1.1 Explanatory Analysis

First of all, let's take a look on how each feature is distributed, Fig 1. While most of the features are normally distributed across some mean, like *Slope or Hillshade*, the *Aspect* is a little bit different. The reason for such strange distribution is in physical meaning of this feature, since *Aspect* is measured in angles, either positive or negative angle from 0 must yield the same(almost) label for this path.

Another interesting dependencies are between Hillshade indexes, (Fig 3). As you can see, it seems like they depend in some quadratic fashion. All this observations will be useful for further feature engeneering to improve our model evaluations.

[1]http://www.kaggle.com/c/forest-cover-type-prediction/

## 2. PREDICTIVE TASK

This is a well studiet dataset: based on research of this dataset PhD dessertations and numerous papers were published. However, copmetetion on Kaggle is very challenging: given train set is much much less than test set: 15K observations in train set vs 565K observations in test set. However, task is pretty simple, based of observation you should predict *cover type* of this patch. Performance of each Kaggle submission is evaluated as multiclass accuaracy score, and all submissions are sorted by it. Ultimate goal of this paper is to be in top 10% on Kaggle Leaderboars.

We will start our model from initial simple logistic regression, and then will apply more advanced techniques like Support Vector Machines, Neural Networks and Extremely Randomized Decision Trees. Performance of our model will be evaluated on cross validation set generated by randomized KFold cross validation with $k = 5$, and then the best performing model will be selected and evaluation of its submission on Kaggle will be reported.

## 3. PREVIOUS RESULTS

Most of the publications citing this data set is from 2000-2001, and we should take it into account when comparing results. [5] reporst performance on the same test set, train set splits as 68.6% for LDA and 62% for Decision Trees. In PhD thesis [4] accuracy with NN classification is reported as 70%. However, current Kaggle results suggests that outperforming previous results will be fairly easy, and we need to aim for 80% accuracy.

## 4. FEATURE ENGINEERING

We are approaching the main and fundomental part of this assignment. Good features plays icredible role in model evaluation.We will be following an approach described in [7]. Namely, we can combine *Vertical distance to Hydrology* and *Horizontal distance to Hydrology*, and applying Pythogorean theorem yield us *Distance to Hydrology*. This feature carries much more information rather than simple horizontal and vertical notions, because obviously cover type of land should depend on direct distance to water source.

Another interesting feature could be obtained from *Vertical Hydrology distance*. If you have noticed from Fig 1 *Vertical Hydrology distance* contains some amount negative values.

**Table 1: Dataset description**

| Data field | Description |
|---|---|
| elevation | Elevation in meters |
| aspect | Aspect in degrees azimuth |
| slope | Slope in degrees |
| horz_hydro | Horz Dist to nearest surface water features |
| vert_hydro | Vert Dist to nearest surface water features |
| horz_road | Horz Dist to nearest roadway |
| hillshade_9am | Hillshade index at 9am, summer solstice (0 to 255 index) |
| hillshade_noon | Hillshade index at noon, summer solstice (0 to 255 index) |
| hillshade_3pm | Hillshade index at 3pm, summer solstice (0 to 255 index) |
| horz_fire | Horz Dist to nearest wildfire ignition points |
| wild | Wilderness area designation (4 binary columns) |
| soil_type | Soil Type designation (40 binary columns) |

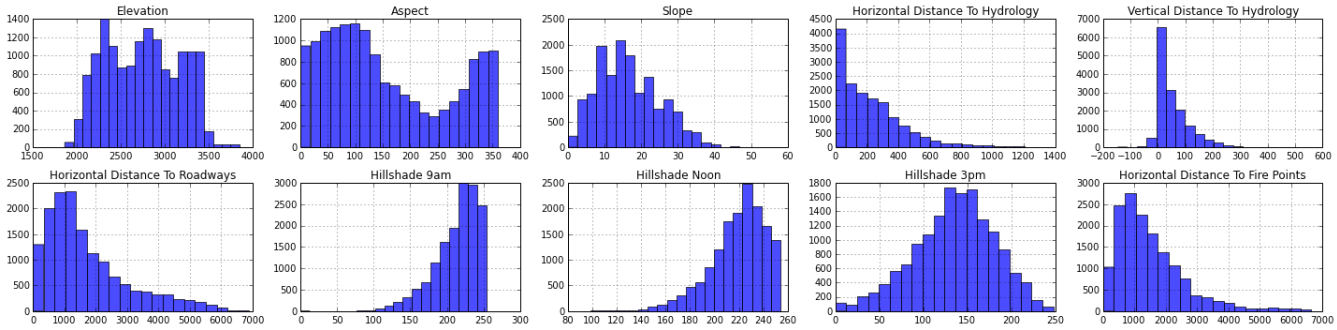


**Figure 1: Feature distribution of training set.**

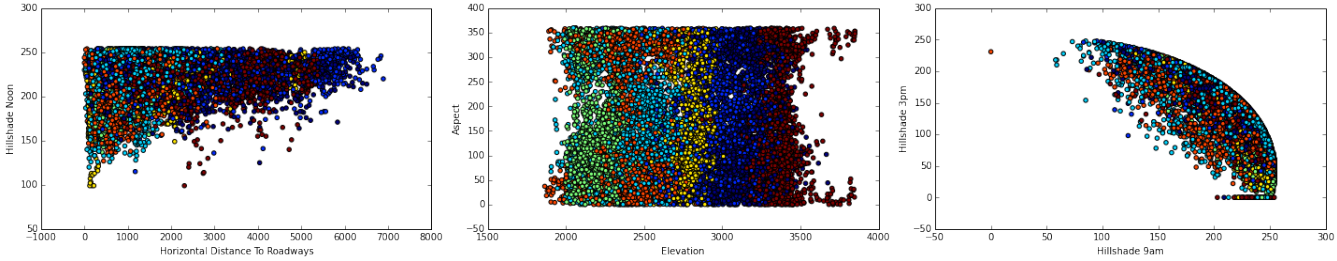


**Figure 2: Some feature relationships.**

**Table 2: Notation used in this report**

| Notation | Description |
|---|---|
| $x$ | 1 observation |
| $X$ | matrix of observations |
| $x_i$ | $i$-th feature of observation $x$ |
| $x^{(i)}$ | $i$-th observation |
| $y^{(i)}$ | label of observation $x^{(i)}$ |

For some cover types it might be good descriminator, so we can emphasise it introducing a new boolean feature *Is Water Source Below* which will contain *True* values for observations which distance to water source is negative.

Let's think about what else could define cover type. One of the characteristics of cover could is how jointly further it is from water resources and elevation. It makes sense because, cover of the area depends on multitude of factors, and joining them might be a good idea. Therefore, for each pair of features that represent distance, I introduced two features, their sum and absolute difference. Complete list of generated features is in Table 4(latest page).

# 5. MODELS

Notation described in this paper is described in Table 2.

## 5.1 Logistic Regression

The very first model that was applied was a logistic regression in one vs. others mode. For each class $y^{(k)}$ we train regression on items that belong to class vs all other items. Logistic regression assumes that probability of observation $x$ belonging to class $y$ is given by sigmoid funciton [3]

$$P(x|y) = \sigma(x) = \frac{1}{1 + e^{-\theta \cdot x}}$$

Our goal is to maximize log likelihood $\mathcal{L}$ of train data set:

$$\begin{aligned}\mathcal{L}(data) &= \log P(data|\theta) \\ &= \sum_{i=1}^{N} (y^{(i)} \log \sigma(x^{(i)}) + (1 - y^{(i)}) \log(1 - \sigma(x^{(i)}))\end{aligned}$$

Maximizing it with respect to $\theta$ will give as classifier for class $y_k$ Then, bayes optimal classifier $h(x)$ for observation $x$ is:

$$h(x) = \arg\max_k P(x|y^{(k)})$$

Unfortunately, performance of Logistic regression on this train set is not satisfactorial. But more importantly, analyzing the learning curve suggest that even if we had more labeled data, it would not improved performance of this model. As you can see, score on training set slightly decreased when more training data was added.

However, one of the advantages of this models is it works very fast and actually for simple models it classifies with good accuracy, recall that we have 7 classes almost uniformely distributed, and random guessing gives as ~14% accuracy

## 5.2 Support Vector Machine

Second model that was applied was support vector machine. SVMs are recomended themselves as reliable classification


**Figure 3: Performance of logistic regression.**

tools, and before reinvention of Neural Networks(CNNs) was state of the art classification techniques.

SVM is more advanced model with simple idea: *large margin* classification [3]. Our assumption is there exists a hyperplane $\psi(x) = w^T x + b$ that separates classes:

$$\begin{cases}\psi(x^{(i)}) > 0, \text{if, } y^{(i)} = 1 \\ \psi(x^{(i)}) < 0, \text{if, } y^{(i)} = -1\end{cases}$$

And we want this margins $\rho(x^{(i)})$ between point and hyperplane be as much as possible :

$$\rho(x^{(i)}) = \frac{y^{(i)}\psi(x^{(i)})}{||w||}$$

So our objective $J$ is to maximize minimal distance:

$$J = \arg\max_{w,b}\{\frac{1}{||w||}\min(y_i\psi(x_i))\}$$

It turns out that maximization of $J$ is simply minimization of $||w||$.

However, since perfect separation is not always possible, we introduce slack variable:

$$\xi_i = |y^{(i)} - \psi(x^{(i)})|$$

This is a measure how far we are beyond hyperplane, and we want this overall score be as low as possible. Our optimization objective $J$ is:

$$J = c\sum_{i=1}^{N} \xi_i + ||w||$$

Running this SVM with linear kernel to train on our train set, unfortunately, failed. The reason is, up to this moment features were not normalized. After normalizing them to zero mean, variance one following results were obtained(Fig 6). As you can see, we have improved accuracy by 13% which means, our result is 20% better than it was with logistic regression.

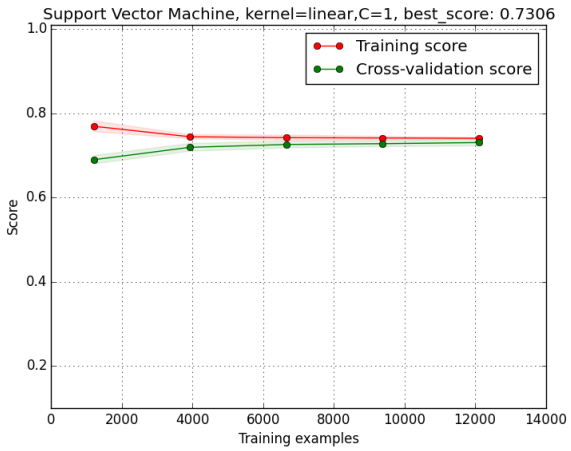


**Figure 4: Performance of linear svm.**

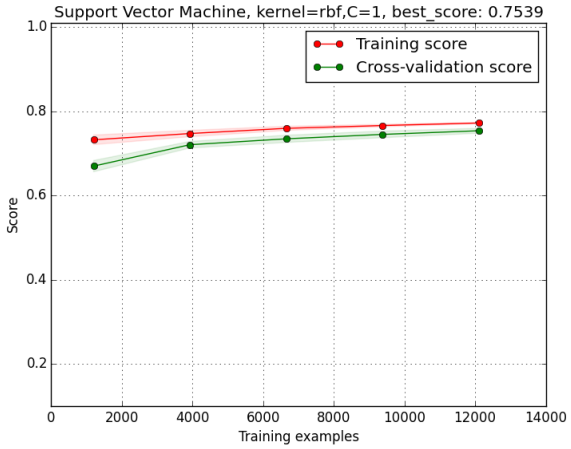


**Figure 5: Performance of svm with rbf kernel.**

However, this is not an end of story with SVM. For now, we have assumed that our boundary between classes in linear. However, using well known *kernel trick* we can embed larger dimensional features inside of our model almost for free. For this part I run SVM with *radial basis function* kernel, *rbf*. This made model not only faster in training, but yielded more accurate model. Fig 5

Comparing SVM with linear kernel versus SVM with rbf kernel, we clearly see that increasing number of training examples would help to rbf SVM, but for linear. Advantages of using SVM is better accuracy, large margin classification, wich means more generalizable solution at general.

## 5.3 Neural Networks

Since previously reported highest score on this data set was acquired by using Neural Networks this part of assignment was higly interested to me. For this part I used an Neural Network with one hidden layer as depcted on fig 8. Each circle on this figure called a neuron, and represents a simple function that gets some input and returns output. Depend-

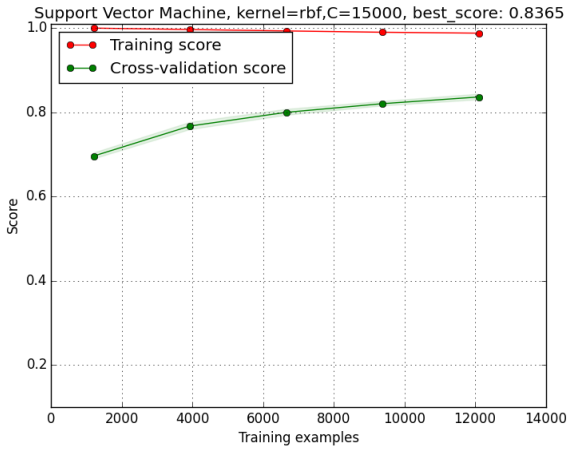


**Figure 6: Performance of svm with rbf kernel.**

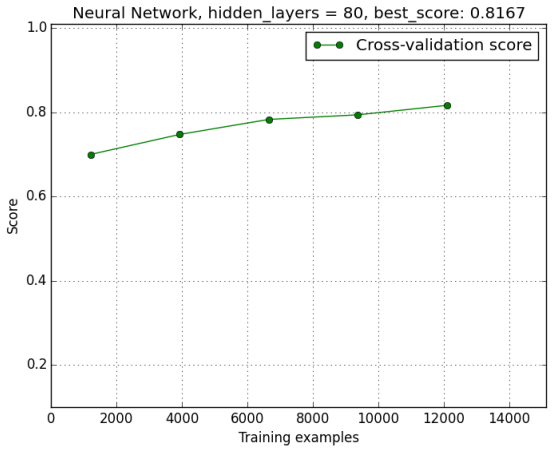


**Figure 7: Neural Networks.**

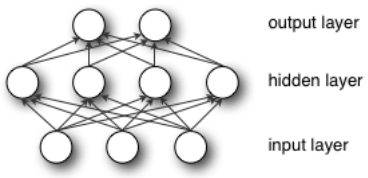


**Figure 8: Neural Network Topology**

ing on the level of cell it can have one or multiple inputs from previous layer. This network can be described by following function:

$$f(x) = G(b^{(2)} + W^{(2)}\sigma(b^{(1)} + W^{(1)}))$$

Where $\sigma$ is a well known *sigmoid* function, $b^{(i)}$ is a bias vector for each layer $i$, and $W^{(i)}$ is weight matrix of layer $i$. This function maps from high dimensional feature space into 7 dimensional labels space, and for each $x$ yields a vector $o \in R^7$. Then optimal classifier is $h(x)$:

$$o = f(x)$$
$$h(x) = \arg\max_i o_i$$

Objective of this model is following loss function[2]:

$$J(w, b) = \sum_{n=1}^{N} ||f(x^{(n)}) - y^{(n)}||^2$$

Optimization was done via *Stochastic gradient descent* with backpropagation of error in minibatches. For NN model the same set of normalized features from SVM model was taken, score on cross validation set can be seen on Fig 7 From the cross validation is clearly seen that adding more train examples will definitely help to our model performance. Even though training with SGD converges much faster than with usual gradient descent, training of this NN with much more training examples could be a problem. However, the same is true for SVM, that is why this coudn't be considered as real disadvantage. Most interesting property of this model is that it outperforms previously reported score by 15%. Submission on Kaggle with this results get my 112 place.

## 5.4 Extremely randomized trees

The latest model for this data set was Extremely randomized trees. The main difference of Extra-trees is that it splits nodes by chosing cut-points fully at random and it uses whole learning sample(instead of using bootstrap) to grow the trees.[6] There are three main parameters that affect performance of Extra trees: $K$ - number of random splits screened at each node, $n_{\min}$ - number of minimal splits required for splitting a node and $N_e$ - number of estimators(trees) in model.

Generally smaller $K$ is, the stronger randomization of trees. Similarly, larger values of $n_{\min}$ lead to smaller trees, higher bias and smaller variance. And finally, larger the number of esitmators $N_e$ is the better overall accuracy would be. For this dataset parameters were following: $K = 1$, $n_{\min} = 3$, $N_e = 500$

Surprisingly, this is the best performing model for this data set. It has the best cross validation score is 88%, see Fig 9 And more importantly, this model works amazingly fast. Comparing to SVM and NN, it runs in 10th of their time. By all means, in such challenging requirments of train set size, ExtraTrees is the best model for this dataset.

On this stage, after getting good model, we can evaluate importance of our features. Are introduced features helping

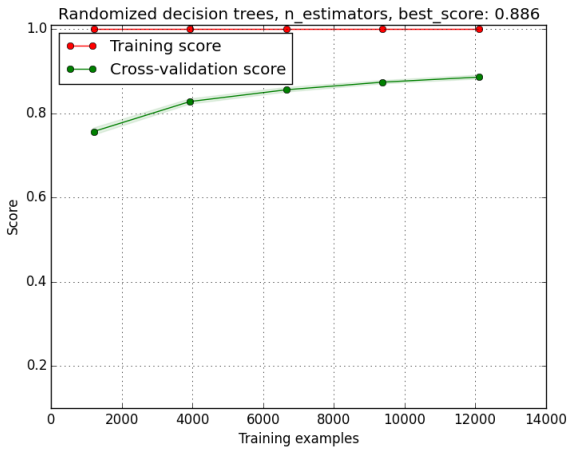


**Figure 9: Performance of Random Decision Trees.**

**Table 3: 15 most important features for ExtraTrees**

| Feature | Relative Importance |
|---|---|
| elev_hydro_vert_sum | 0.111532 |
| elevation | 0.107966 |
| elev_hydro_vert_diff | 0.080564 |
| elev_hydro_horz_sum | 0.063396 |
| elev_hydro_horz_diff | 0.053435 |
| fire_road_sum | 0.027266 |
| fire_road_sum_squared | 0.027224 |
| hydro_road_diff_squared | 0.026241 |
| hydro_road_diff | 0.026221 |
| hillshade_9am | 0.026044 |
| hydro_road_sum_squared | 0.026021 |
| horz_road | 0.025834 |
| hydro_road_sum | 0.025783 |
| aspect2 | 0.025008 |
| aspect | 0.024636 |

to our model, or not? Maybe feature engeenering was absolutely unnecessary? ExtraTrees implementation in scikit-learn package provides as a method to analyse feature importance. As you can see from Table 3 most of the relevant features are those that we engineered.

Why does ExtraTrees yield by all means the suprior results? One intuition is that classification of cover types is done by scintist in similar manner. We call this patch covered by Pine if this and that conditions are held, and Aspen if that and another conditions held. The decision trees do the same, we are creating rules and classifying by rules. ExtraTrees just do this job more optimally, selects more pivot variables and that is why yields better results.

## 5.5 Getting more

So far, we have obtained 88% score on CV set, it is pretty good, it gives 75 place on Kaggle leaderboard, but is that all? For now we have used only first 11 features, and features engeenered from it, however there are still two binary feature vectors: *Wilderness area* and *Soil Type*. After including all the features into training set, ExtraTrees yield 90% score on

**Table 4: New features**

| | |
|---|---|
| aspect2 | (aspect+180) mod 360 |
| aspect3 | sin(aspect) |
| elev_hydro_vert_sum | elevation + hydro_vert |
| elev_hydro_vert_diff | abs(elevation - hydro_vert) |
| elev_hydro_horz_sum | elevation + hydro_horz |
| elev_hydro_horz_diff | abs(elevation - hydro_horz) |
| dist_to_hydrology | root of squared sum of Vert. and Horz dist to hydro |
| dist_to_hydrology_squared | dist3_To_Hydrology squared |
| hydrology_lower | bolean, True when vert. hydro is negarive |
| horz_hydro_fire_sum | horz_hydro + horz_fire |
| horz_hydro_fire_sum_squared | horz_hydro_Fire_sum squared |
| horz_hydro_fire_diff | abs(horz_hydro - horz_fire) |
| horz_fydro_fire_diff | horz_hydro_fire_diff squared |
| hydro_road_sum | horz_hydro + horz_road |
| hydro_road_sum_squared | horz_road_sum squared |
| hydro_road_diff | abs(horz_hydro - horz_road) |
| hydro_road_diff_squared | hydro_road_diff squared |
| fire_road_sum | horz_fire + horz_road |
| fire_road_sum_squared | fire_road_sum squared |
| fire_road_diff | abs(horz_fire - horz_road) |
| fire_road_diff_squared | fire_road_diff squared |

CV set, and it is 45 place on Kaggle.

# 6. CONCLUSIONS

The goal of this paper is to predict cover of forest based on cartographical data, and obtain competative submission on Kaggle. Previous reported scores are compared to results of this paper and significant improvement noticed. Moreover, during this work 4 different classification techniques were applied and their advantages and disadvantages are discussed. Overall, best results are obtained using Extremely Random decision trees, which gave 100% accuracy for train set, 90% accuracy for cross validation set, and 82% accuracy on test set. Main challenge of this work was discrepancy between sizes of train set and test set, (15K vs 565K). The best performming model evaluated to be on 75th place on Kaggle[2].

# 7. REFERENCES


[1] D.J. Newman A. Asuncion. UCI machine learning repository. `https://archive.ics.uci.edu/ml/datasets/Covertype`, 2007.

[2] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[4] Jock A. Blackard. *Comparison of Neural Networks and Discriminant Analysis in Predicting Forest Cover Types*. PhD thesis, Fort Collins, CO, USA, 1998. AAI9921979.

[5] Joao Gama, Ricardo Rocha, Pedro Medas, and xxx cx. Accurate decision trees for mining high-speed data streams. In *KDD*, page 523, 2003.

[6] Pierre Geurts. Extremely randomized trees. In *MACHINE LEARNING*, page 2006, 2003.

[7] Alexander Guschin. Feature engeenering benchmark. `"http://nbviewer.ipython.org/github/aguschin/kaggle/blob/master/forestCoverType_featuresEngineering.ipynb`, 2014.


[2]Results can be seen on `http://www.kaggle.com/users/114661/yerlan-idelbayev`