

Machine Learning Engineer

Nanodegree Capstone Project

Giovanni Mària February 2nd, 2018

I. Definition Project Overview

The classification problem that I am going to solve in this project is a kind of problem that it is possible to solve thanks to supervised learning techniques provided by machine learning field.

Regarding forest cover type background we all know which is the importance of country forest cover type from biological point of view. It is important to know how it is going to evolve in order to protect the biodiversity and the future of our species. This allow also to follow the deforestation closer and try to put some limitations in order to ensure and secure the future of forests.

This is a kind of classification problem that can be solved by using supervised learning algorithms provided by machine learning field. Other kind of classification problem similar to « cover forest type » are several. The mostly known are for example : Titanic survivors prediction¹ (maybe the most known on Kaggle learners community), Spam detection² and Random acts of pizza³

More specifically, I will use two datasets provided by the University of Californiaⁱ that include features (such as: elevation, aspects, slope, etc.) that describe the forest cover type in the Roosevelt National Forest and that will be used to predict an integer classification for the cover type itself.

I am going to use several algorithm models in order to get a high accuracy score and be able to have a very good prediction for every row of the “Cover_Type” feature in the test set.

¹ <https://www.kaggle.com/c/titanic>

² <https://www.kaggle.com/uciml/sms-spam-collection-dataset>

³ <https://www.kaggle.com/c/random-acts-of-pizza>

Problem Statement

Forest cover type prediction problem means to predict the predominant kind of tree cover from strictly cartographic variables. Independent variables were then derived from data obtained from the US Geological Survey and USFS. The data is in raw form (not scaled) and contains binary columns of data for qualitative independent variables such as wilderness areas and soil type⁴.

The goal is to find out a model that will be accurate for predicting the forest cover type for each row. In order to find out more I will follow the steps below :

- Upload both datasets
- Explore data in order to identify most valuable features to train our model
- Clean the datasets and specially avoid partially empty or missing values
- Fit the dataset to a multiple models and apply further the cross validation for it
- Run train test split on train datasets and plot the learning curve in order to evaluate how the model is performing compared to the number of samples
- Fit “Random Forest Classifier” and “hypertune” parameters (through cross-validation)

⁴ <https://www.kaggle.com/c/forest-cover-type-prediction>

- Recheck the most valuable features for applying random forest model
- Ensure again that this model doesn't show any underfitting/overfitting issue
- Readjust parameters if needed to obtain a better learning curve output
- Run the prediction on the test set
- Fit other valuable classification models for this kind of problem in order to obtain a benchmark
- Pick the most performing model to apply

Metrics

Since it concerns a binary classification problem and kaggle used "classification accuracy" to evaluate the output. I think that the most valuable metrics to use is accuracy because it is more likely correspond to Kaggle metric. Other than that I think that get an eye on precision, recall, f1 and support (occurrences of each class) could give us a wider landscape on how algorithm performance is.

Let's have a closer eye on each metric. Precision Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. (eg. What is the ration of good prediction on cover forest type class ?). Then recall is the ratio of correctly predicted positive observations to the all observations in actual class (how many cover type that are correctly predicted we labeled ?). Finally the F1 score is the weighted average of Precision

and Recall, that give to the output more consistency. That's why I chose these metrics. All formula are below⁵ :

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

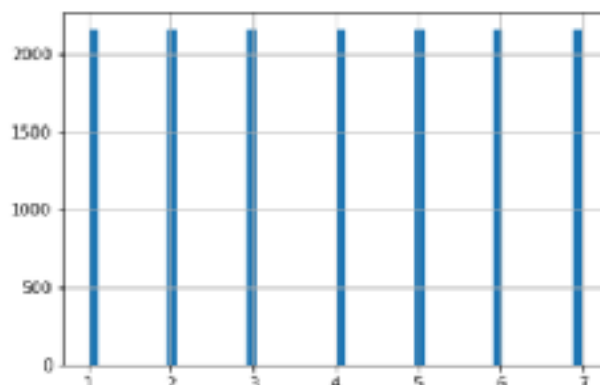
$$f1 = 2 \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

II. Analysis Data Exploration

Concerning data I am using for this problem the dataset includes four wilderness areas located in the Roosevelt National Forest of northern Colorado. It is a "CSV" file there are 15120 rows and 55 columns

```
[15120 rows x 55 columns]>
```

Then we plot « cover_types » feature with plots in order to have a better idea of the feature to predict.



⁵ <https://www.c>

In the « Training_set » cover type are perfectly distributed as we can see in the plot below.

This kind of dataset has no particular issues. When I print out the first two rows of the train set I see no string values or incomplete data (Image 1).

Image 1

	Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrology	\
Id					
1	2596	51	3	258	
2	2598	56	2	212	

	Vertical_Distance_To_Hydrology	Horizontal_Distance_To_Roadways	\
Id			
1	0	518	
2	-6	148	

	Hillshade_9am	Hillshade_Noon	Hillshade_3pm	\
Id				
1	224	232	148	
2	220	235	151	

	Horizontal_Distance_To_Fire_Points	Wilderness_Area1	Wilderness_Area2	\
Id				
1	6279	1	0	
2	6225	1	0	

	Wilderness_Area3	Wilderness_Area4	Soil_Type1	Soil_Type2	Soil_Type3	\
Id						
1	0	0	0	0	0	
2	0	0	0	0	0	

	Soil_Type4	Soil_Type5	Soil_Type6	...	Soil_Type22	Soil_Type23	\
Id				...			
1	0	0	0	...	0	0	
2	0	0	0	...	0	0	

	Soil_Type24	Soil_Type25	Soil_Type26	Soil_Type27	Soil_Type28	\
Id						
1	0	0	0	0	0	
2	0	0	0	0	0	

	Soil_Type29	Soil_Type30	Soil_Type31	Soil_Type32	Soil_Type33	\
Id						
1	1	0	0	0	0	
2	1	0	0	0	0	

There are no null values neither (Image 3). We run a fast and dirty random forest without any cross validation or train/test/split I get an accuracy score of 1 using 500 estimators (Image 2).

Image 2

Train set score: 1.00

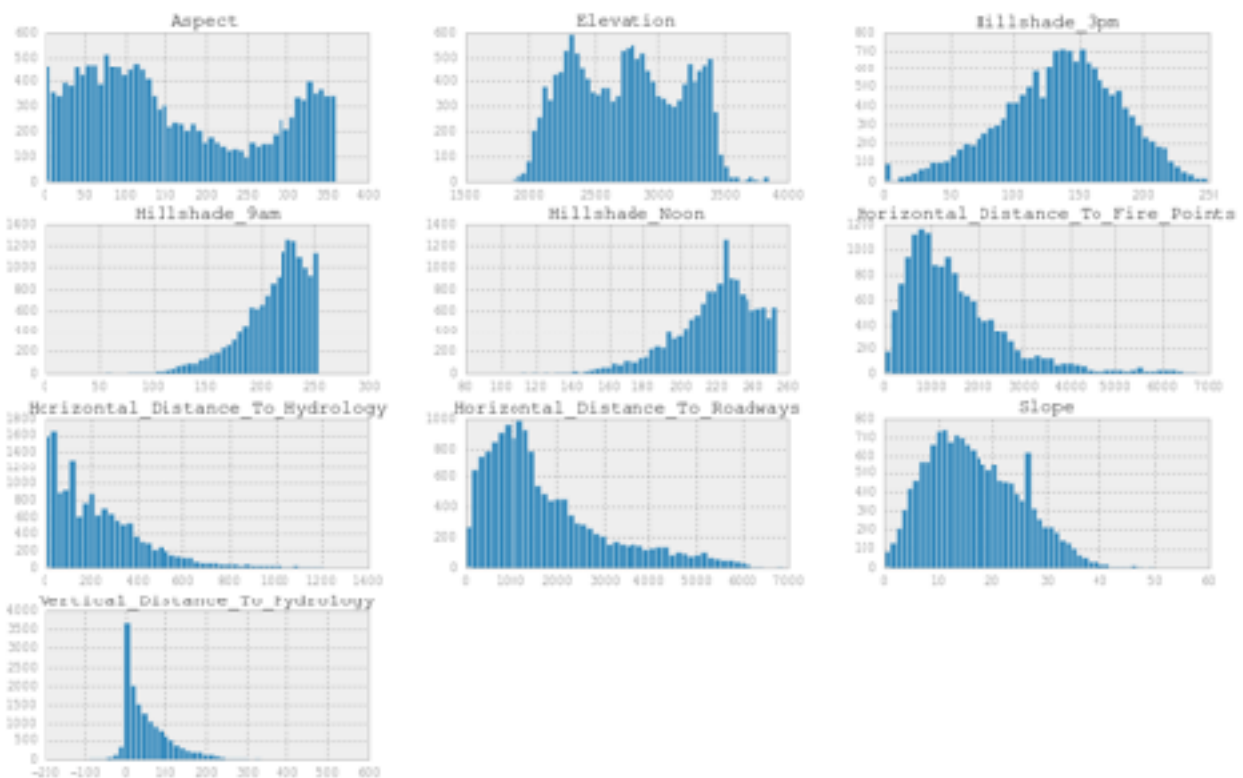
Image 3

```
Empty DataFrame
Columns: [Elevation, Aspect, Slope, Horizontal_Distance_To_Hydrology, Vertical_Distance_To_Hydrology, Horizontal_Distance_To_Roadways, Hillshade_9am, Hillshade_Noon, Hillshade_3pm, Horizontal_Distance_To_Fire_Points, Wilderness_Area1, Wilderness_Area2, Wilderness_Area3, Wilderness_Area4, Soil_Type1, Soil_Type2, Soil_Type3, Soil_Type4, Soil_Type5, Soil_Type6, Soil_Type7, Soil_Type8, Soil_Type9, Soil_Type10, Soil_Type11, Soil_Type12, Soil_Type13, Soil_Type14, Soil_Type15, Soil_Type16, Soil_Type17, Soil_Type18, Soil_Type19, Soil_Type20, Soil_Type21, Soil_Type22, Soil_Type23, Soil_Type24, Soil_Type25, Soil_Type26, Soil_Type27, Soil_Type28, Soil_Type29, Soil_Type30, Soil_Type31, Soil_Type32, Soil_Type33, Soil_Type34, Soil_Type35, Soil_Type36, Soil_Type37, Soil_Type38, Soil_Type39, Soil_Type40, Cover_Type]
Index: []

[4 rows x 55 columns]
```

There is a quite high possibility that this dataset would be ready for being used in order to create a trusty valuable model. Nevertheless we could improve this by choosing the best features for our model.

Also if I plot missing values (Image 4) I can notice missing values for Hillshade at 3pm.



Exploratory Visualization

In order to show up with some form of visualization I applied random forest in order to detect most important features to use (Image 4)

Image 5

Here upon we can notice that “Elevation” is the most important feature in the dataset that is more than two times higher than the 2nd most important feature. Thus, this would be a mandatory feature to look at.

Now that we know the importance of “Elevation” feature, I am interested to see which is the relationship between this feature and the others.

How we can see in the picture below we notice a correlation between “Elevation”, “HD_Roadways” and “HD_Hidrology”. That could be very interesting going forward for choosing the right features for our model. Tree based models make decisions based a criteria for each label and that’s why linearity is pretty important.

Cover Type Top 20 features

1. Elevation (0.225350)
2. Horizontal Distance To Roadways (0.092123)
3. Horizontal Distance To Fire Points (0.073993)
4. Horizontal Distance To Hydrology (0.063236)
5. Vertical Distance To Hydrology (0.053313)
6. Hillshade_9am (0.050844)
7. Aspect (0.049724)
8. Hillshade_3pm (0.046118)
9. Wilderness_Area4 (0.045877)
10. Hillshade_Noon (0.044778)
11. Slope (0.035979)
12. Soil_Type18 (0.022931)
13. Soil_Type38 (0.019515)
14. Soil_Type3 (0.018633)
15. Wilderness_Area1 (0.018276)
16. Soil_Type19 (0.018144)
17. Wilderness_Area1 (0.016282)
18. Soil_Type4 (0.012888)
19. Soil_Type40 (0.010858)
20. Soil_Type30 (0.007796)

Mean Feature Importance 0.018519

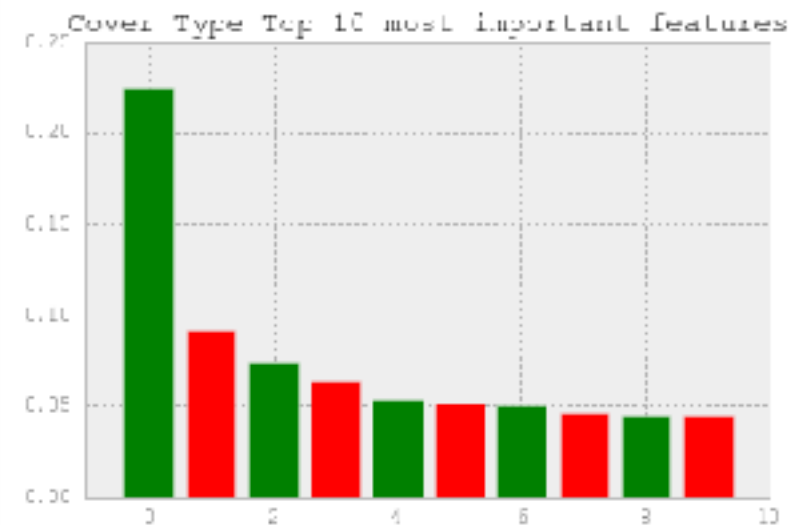
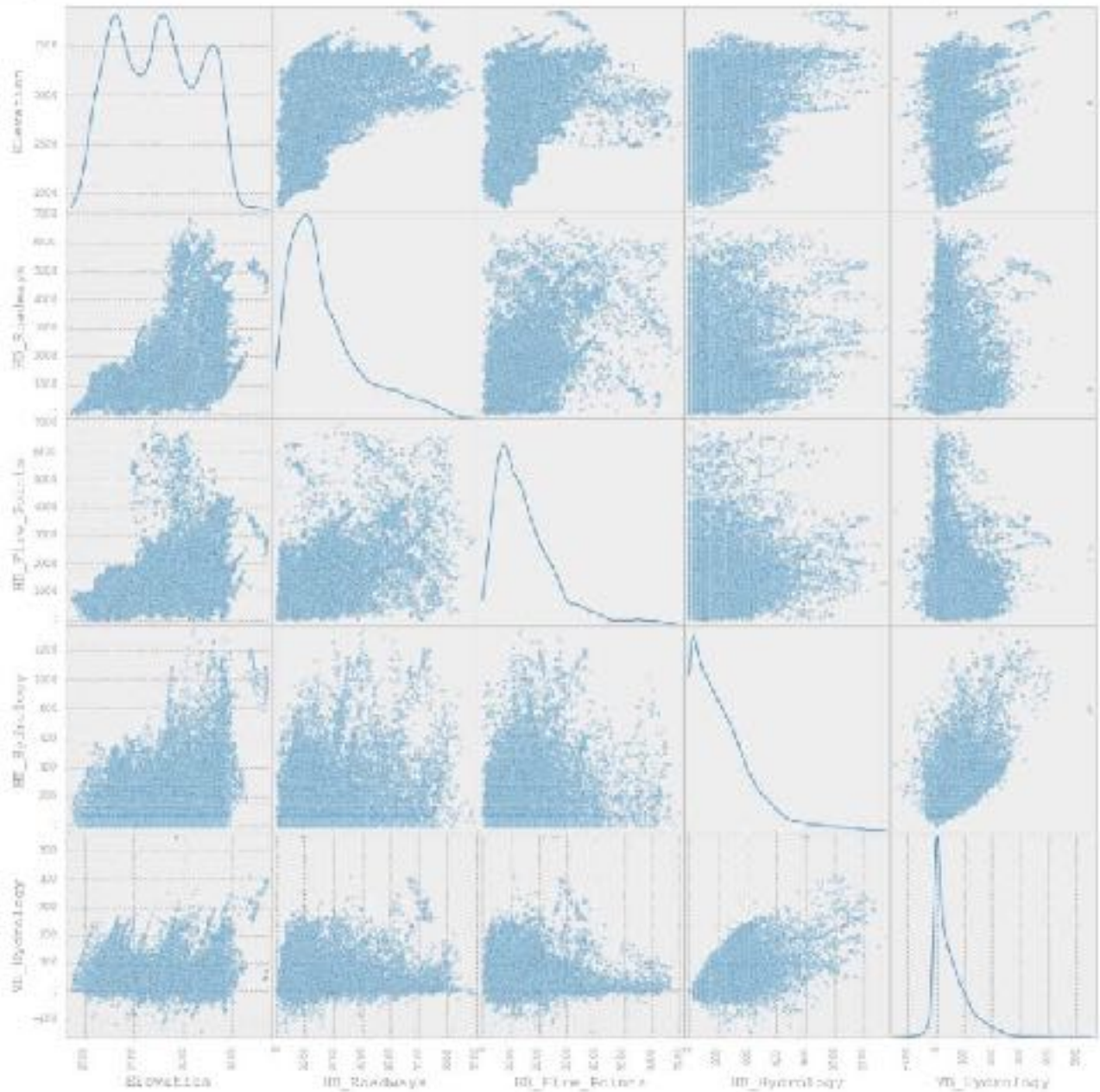


Image 6



We notice also that “wilderness_area” and “Soil_Type” are one hot encoded and we can convert back in order to do further analysis. Other than that all features are numerical and not categorical.

Image 7

```
Elevation          int64
Aspect             int64
Slope              int64
Horizontal_Distance_To_Hydrology int64
Vertical_Distance_To_Hydrology int64
Horizontal_Distance_To_Roadways int64
Hillshade_9am      int64
Hillshade_Noon     int64
Hillshade_3pm      int64
Horizontal_Distance_To_Fire_Points int64
Wilderness_Area1   int64
Wilderness_Area2   int64
...
Soil_Type30        int64
Soil_Type31        int64
Soil_Type32        int64
Soil_Type33        int64
Soil_Type34        int64
Soil_Type35        int64
Soil_Type36        int64
Soil_Type37        int64
Soil_Type38        int64
Soil_Type39        int64
Soil_Type40        int64
Cover_Type         int64
dtype: object
```

I have also noticed that two features as “Soil_Type15” and “Soil_Type7” have constant values (“0”) as you can see on the picture below.

Image 8

	Soil_Type7	Soil_Type8	Soil_Type9	Soil_Type10	Soil_Type11	\
count	15120.0	15120.000000	15120.000000	15120.000000	15120.000000	
mean	0.0	0.000000	0.000001	0.141667	0.026052	
std	0.0	0.000133	0.025710	0.340719	0.161656	
min	0.0	0.000000	0.000000	0.000000	0.000000	
25%	0.0	0.000000	0.000000	0.000000	0.000000	
50%	0.0	0.000000	0.000000	0.000000	0.000000	
75%	0.0	0.000000	0.000000	0.000000	0.000000	
max	0.0	1.000000	1.000000	1.000000	1.000000	

	Soil_Type12	Soil_Type13	Soil_Type14	Soil_Type15	Soil_Type16	\
count	15120.000000	15120.000000	15120.000000	15120.0	15120.000000	
mean	0.015013	0.011401	0.011177	0.0	0.007540	
std	0.121609	0.174621	0.105133	0.0	0.006506	
min	0.000000	0.000000	0.000000	0.0	0.000000	
25%	0.000000	0.000000	0.000000	0.0	0.000000	
50%	0.000000	0.000000	0.000000	0.0	0.000000	
75%	0.000000	0.000000	0.000000	0.0	0.000000	
max	1.000000	1.000000	1.000000	0.0	1.000000	

It is also interesting to notice the correlation between hillshade @ Noon and 9 am in order to detect zeros that will be excluded going forward in order to use a ready to use train set. Here below we can notice a clear positive correlation. Given that, we can fit a regressore to the train_set in order to predict the missing 3 pm hillshade values.

I have also printed out the number of zeros on a feature in order to understand which one I could drop to get a better scoring.

Image 9

0	14954	
1	849	
Name: Soil_Type4, dtype: int64		
0	14954	
1	100	
Name: Soil_Type5, dtype: int64		
0	14978	
1	678	
Name: Soil_Type6, dtype: int64		
0	15120	
Name: Soil_Type7, dtype: int64		
0	15110	
1	1	
Name: Soil_Type8, dtype: int64		
0	15110	
1	10	
Name: Soil_Type9, dtype: int64		
0	12490	
1	2140	
Name: Soil_Type10, dtype: int64		
0	14754	

Algorithms and Techniques

I think that ensemble learning methods are more appropriate for this kind of classification problem because it is specifically designed for binary problems such this one concerning the forest cover type prediction.

Among all methods available I pick up only two : ExtraTrees and random forest.

The first one I will use is the ExtraTrees Classifierⁱⁱⁱ (essentially a special type of random forest) that compared to random forest classifier use the same input training to train all trees and also it picks a node split very extremely (variable index and variable splitting value are chosen randomly).

I will use random forest first with 500 n estimators (I will use the same for extratrees) to have a first benchmark on kaggle on default dataset. I have chosen this algorithm because given that it is a part of ensemble methods. This method applies a procedure to build up a set of training example for each tree and it is implemented by a number of decision trees that generate as output a prediction of the trees. Other than that random forest find the best split (the optimal one by variable index and splitting value) in a subset of variables.

The reason why I have chosen only one algorithm (yes because I know that extra trees is very similar to random forest) is that those are perfectly designed for this kind of problem.

I will try to explain it better. Since the prediction concern to predict a over forest type in according with several features. There is a kind of decision tree that design which kind of species will be the next one. The fact that there is correlation within some features such « elevation » and « soil type » for example tell us more about the kind of algorithm is useful and should be used. That's why I didn't add SVM (that I have tried but it was very disappointing) and I have chosen to focus only on these two ones and explain why I have chosen them.


For decision tree issues those are among the best ones and also accuracy score I have obtained showed this. I hope that this explanation it is enough to explain all about my decision.


I am also going to use train test split in order to test the algorithms first and see if we could have adequate algorithms well-tailored for this kind of classification problem.


Benchmark

Since it is a old Kaggle competition I thought that I could use as a benchmark a first submission that I have done after applying a ExtraTrees classifier algorithm on the default dataset. The score I have obtained (categorization accuracy) on the platform is 0.34241 (Image 9) and the position I have obtained (on the private leaderboard since the competition is closed) is 1664 out of 1694 (Image 10)^{iv}. My goal is to improve this result.

Image 10



Competitions
Datasets
Kernels
Discussion
Jobs
...




Forest Cover Type Prediction

Use cartographic variables to classify forest categories

1,691 teams · 3 years ago

Overview
Data
Kernels
Discussion
Leaderboard
Rules
Team
My Submissions
Late Submission

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
ETC0.csv	a few seconds ago	1 seconds	5 seconds	0.34211

Complete

[Jump to your position on the leaderboard](#)

You can select up to 2 submissions to be used to calculate your final leaderboard score. If 2 submissions are not selected, they will be chosen based on your best submission scores on the public leaderboard.

Your final score may not be based on the same exact subset of data as the public leaderboard, but rather a different private data subset of your full submission — your public score is only a rough indication of what your final score is.

You should thus choose submissions that will most likely be best overall, and not necessarily on the public subset.






















3 submissions for **Giovanni Maria**

Sort by: **Most recent**

All
Successful
Selected

Submission and Description	Public Score	Use for Final Score
<div>ETC0.csv</div> <div>a few seconds ago by Giovanni Maria</div>	<u>0.34211</u>	<input type="checkbox"/>

Image 11

Overview	Data	Kernels	Discussion	Leaderboard	Rules	Team	My Submissions	Late Submission
1656	—	Jaustin					 0.37053	1 3y
1657	—	RITESH VARYANI 2					 0.37053	1 3y
1658	—	Joseph Finnegan					 0.37053	2 3y
1659	—	Robert ?					 0.37053	1 3y
1660	—	iosentinel					 0.37053	1 3y
1661	—	mari					 0.36600	1 3y
1662	—	jwa17					 0.35166	10 3y
1663	—	Yeqing Zhang					 0.35107	3 3y
1664	—	Rui Kobe Zhu					 0.34929	2 3y
1665	—	Destiny Anyalwa					 0.34214	5 3y
1666	—	ninjasoul					 0.32570	1 3y
1667	—	Peter Prettenhofer					 0.32570	1 3y
1668	—	priyank					 0.32089	1 3y
1669	—	Riksmir					 0.31456	7 3y
1670	—	fernandoh1411					 0.31110	1 3y
1671	—	afterhours					 0.30472	1 3y
1672	—	akshay rao					 0.29232	1 4y
1673	—	2Ducks					 0.28657	4 3y
1674	—	RaviMudgal					 0.28175	1 4y
1675	—	minkyblink					 0.28100	1 3y
1676	—	Jonathan Scanlan					 0.27802	3 3y

III. Methodology

(approx. 3-5 pages)

Data Preprocessing

First of all in according with the data exploration I did previously we can start with:

- Removing “Soil_Type7” and “Soil_Type15” because are constants and then useless (they have only “0” values)

- Remove also other two features that contain too many zero's "Soil_Type8" and "Soil_Type25"
- I also normalize the features in order to standardize the range if independent features in the data
- The data are one hot encoded because it better for classification issues. That means that for example for soil_type I have a binary classification for a « soil_typeN » and that simplify the features engineering and for applying several algorithms.

Implementation

First of all I import all libraries I need for going forward in data exploration, features engineering and also for applying and scoring the algorithms I choose. Then I visualize how the data is structured in the train set and also I tried to understand if there is any correlation between features themselves.

Before optimizing the dataset itself I have ran a extraTrees classifier algorithm in order to have an idea of how it could perform on default dataset and also to have some benchmark to challenge successfully. I have scored the prediction created by extra trees classifier and I have applied this to the test set in order to be able to do the Kaggle submission very useful in order to feed my benchmark.

After these first steps, I thought to understand if there is any « null » values or constant values that could be interesting to pull out before running my algorithms.

I have removed both "Soil_Type7" and "Soil_Type15". In order to do this I have used the same dataset variable name used to upload the csv and I drop both columns by using the following code `variable_name.drop(["Soil_Type7","Soil_Type15","Soil_Type8","Soil_Type25"], axis=1).`

Then I applied a new algorithm such as Random Forest. Afterwards, I have scored the quality of the algorithm before applying it to the test set. Finally, and I have submitted the prediction to Kaggle and

hopefully I have obtained a better score than the first submission that means that I have found a better algorithm and that features engineering have worked.

Refinement

The algorithm I tested is the ExtraTreesClassifier. Then I settled all parameters by using train_test_split with a test size of 30% (0.3) and three estimators (20,30,100) different values for max_depth (5,10,15) and also three kind of max features (0.1, 0.2, 0.3). Then I test it with best parameters and grid scores. Then I test the accuracy and I sort out the best parameters that is 100 estimators, 0.2 as max features and 15 as max tree depth.

```
Out[37]:
mean: 0.52121, std: 0.02348, params: {'n_estimators': 20, 'max_features': 0.1, 'max_depth': 5},
mean: 0.60134, std: 0.01892, params: {'n_estimators': 20, 'max_features': 0.1, 'max_depth': 5},
mean: 0.62094, std: 0.02116, params: {'max_features': 0.1, 'n_estimators': 100, 'max_depth': 5},
mean: 0.62144, std: 0.02144, params: {'n_estimators': 20, 'max_features': 0.2, 'max_depth': 5},
mean: 0.58104, std: 0.01132, params: {'n_estimators': 20, 'max_features': 0.2, 'max_depth': 5},
mean: 0.54439, std: 0.01173, params: {'max_features': 0.2, 'n_estimators': 100, 'max_depth': 5},
mean: 0.52901, std: 0.01532, params: {'n_estimators': 20, 'max_features': 0.3, 'max_depth': 5},
mean: 0.53745, std: 0.01545, params: {'n_estimators': 20, 'max_features': 0.3, 'max_depth': 5},
mean: 0.55164, std: 0.01543, params: {'n_estimators': 100, 'max_features': 0.3, 'max_depth': 5},
mean: 0.62101, std: 0.01471, params: {'max_features': 0.1, 'n_estimators': 20, 'max_depth': 10},
mean: 0.67104, std: 0.01432, params: {'n_estimators': 20, 'max_features': 0.1, 'max_depth': 10},
mean: 0.69113, std: 0.01037, params: {'n_estimators': 100, 'max_features': 0.1, 'max_depth': 10},
mean: 0.68005, std: 0.01442, params: {'max_features': 0.2, 'n_estimators': 20, 'max_depth': 10},
mean: 0.68431, std: 0.01734, params: {'n_estimators': 20, 'max_features': 0.2, 'max_depth': 10},
mean: 0.69495, std: 0.01346, params: {'max_features': 0.2, 'n_estimators': 100, 'max_depth': 10},
mean: 0.69189, std: 0.01414, params: {'n_estimators': 20, 'max_features': 0.3, 'max_depth': 10},
mean: 0.63178, std: 0.01422, params: {'n_estimators': 20, 'max_features': 0.3, 'max_depth': 10},
mean: 0.67147, std: 0.01432, params: {'n_estimators': 100, 'max_features': 0.3, 'max_depth': 10},
mean: 0.73604, std: 0.01242, params: {'n_estimators': 20, 'max_features': 0.1, 'max_depth': 15},
mean: 0.73626, std: 0.01616, params: {'max_features': 0.1, 'n_estimators': 20, 'max_depth': 15},
mean: 0.74184, std: 0.01437, params: {'n_estimators': 100, 'max_features': 0.1, 'max_depth': 15},
mean: 0.73609, std: 0.01202, params: {'n_estimators': 20, 'max_features': 0.2, 'max_depth': 15},
mean: 0.74151, std: 0.01602, params: {'n_estimators': 20, 'max_features': 0.2, 'max_depth': 15},
mean: 0.73561, std: 0.01502, params: {'n_estimators': 100, 'max_features': 0.2, 'max_depth': 15},
mean: 0.73609, std: 0.01514, params: {'n_estimators': 20, 'max_features': 0.3, 'max_depth': 15},
mean: 0.73771, std: 0.01589, params: {'max_features': 0.3, 'n_estimators': 20, 'max_depth': 15},
mean: 0.73526, std: 0.01317, params: {'n_estimators': 100, 'max_features': 0.3, 'max_depth': 15}]
```

Best accuracy obtained: 0.706609342480629

Parameters:

```
n_estimators:100
max_features:0.2
max_depth:15
```

Finally I apply a classification report in order to obtain a better score.

Then I tried random forest classifier using the same data of ExtraTrees Classifier. I have also noticed a score of 100% (that could be little overfitting) but when I ran the classification report and I compare this to the first algorithm I notice that is slightly better. Also when I submit this on kaggle I obtained 0.67573 that is better than ExtraTree Classifier (0.63218) and almost double the the first attempt I did.

	precision	recall	f1-score	support
class1	0.64	0.61	0.63	646
class2	0.59	0.62	0.60	644
class3	0.76	0.74	0.75	659
class4	0.88	0.97	0.92	642
class5	0.73	0.80	0.76	622
class6	0.77	0.68	0.72	671
class7	0.93	0.88	0.91	652
avg / total	0.76	0.76	0.76	4536

	precision	recall	f1-score	support
class1	0.72	0.67	0.69	646
class2	0.68	0.66	0.67	644
class3	0.78	0.79	0.79	659
class4	0.90	0.96	0.93	642
class5	0.87	0.90	0.88	622
class6	0.80	0.79	0.79	671
class7	0.93	0.94	0.93	652
avg / total	0.81	0.81	0.81	4536

[Overview](#)
[Data](#)
[Kernels](#)
[Discussion](#)
[Leaderboard](#)
[Rules](#)
[Team](#)
[My Submissions](#)
[Late Submission](#)

⚠ This competition has completed

You should thus choose submissions that will most likely be best overall, and not necessarily on the public subset.

3 submissions for [Giovanni Maria](#)

Sort by Name

All Successful Selected

Submission and Description	PublicScore	Use for Final Score
ETC2.csv 30 minutes ago by Giovanni Maria add submission details	0.46014	<input type="checkbox"/>
ETC3.csv 17 minutes ago by Giovanni Maria add submission details	0.34241	<input type="checkbox"/>
RF_second.csv 3 hours ago by Giovanni Maria add submission details	0.67573	<input type="checkbox"/>

IV. Results

Model Evaluation and Validation

I have chosen all features except for constants features and those that are equal to zero.

I have chosen to use ExtraTrees classifier and Random Forest and I have noticed that the second one have a better score compare to first Kaggle submission. For n_estimators I have chosen 500 because nothing has changed by tuning this to 1000 and I thought to keep the algorithm more efficient.

I have tried also different kind of random_state on train test split function to put the model robustness into the test.

for example I tried to tune this parameter to 35 and I have obtained the output below

Extratrees classifier

	precision	recall	f1-score	support
class1	0.79	0.76	0.77	682
class2	0.77	0.64	0.70	640
class3	0.86	0.79	0.82	658
class4	0.93	0.98	0.95	624
class5	0.80	0.94	0.86	619
class6	0.83	0.86	0.84	657
class7	0.93	0.96	0.95	656
avg / total	0.84	0.84	0.84	4536

Accuracy score: 84.4576719577%

Random Forest classifier

	precision	recall	f1-score	support
class1	0.81	0.75	0.78	682
class2	0.79	0.71	0.75	640
class3	0.86	0.84	0.85	658
class4	0.93	0.97	0.95	624
class5	0.90	0.95	0.92	619
class6	0.86	0.88	0.87	657
class7	0.91	0.98	0.94	656
avg / total	0.86	0.87	0.86	4536

Accuracy score: 86.5961199295%

Than I tried to tune the random_state classifier to 136 and I have obtained optimal results such as:

Extratrees classifier

	precision	recall	f1-score	support
class1	0.87	0.88	0.88	650
class2	0.88	0.77	0.82	673
class3	0.95	0.92	0.93	667
class4	0.97	1.00	0.98	627
class5	0.85	0.97	0.90	646
class6	0.92	0.93	0.93	641
class7	0.98	0.97	0.98	632
avg / total	0.92	0.92	0.92	4536

Accuracy score: 91.6666666667%

Random Forest classier

	precision	recall	f1-score	support
class1	0.93	0.93	0.93	650
class2	0.95	0.90	0.92	673
class3	0.96	0.96	0.96	667
class4	0.97	1.00	0.98	627
class5	0.97	0.99	0.98	646
class6	0.96	0.96	0.96	641
class7	0.97	0.99	0.98	632
avg / total	0.96	0.96	0.96	4536

Accuracy score: 95.9656084656%

as you can see my target algorithm worked largely better and that's why I have chosen to tune differently this parameter and pick the best one.

Of course, this could be used on other kind of problem similar to cover forest where we need to predict an output with a label and classification problem that is binary (that's why random forest is so performing).

Justification

The final result is most valuable than the first I reported thanks to the change of algorithm. I think that this solution is not either innovative but it is interesting to notice how a different kind of algorithm even if it part of the same "ensemble methods" could have such an important impact on the score.

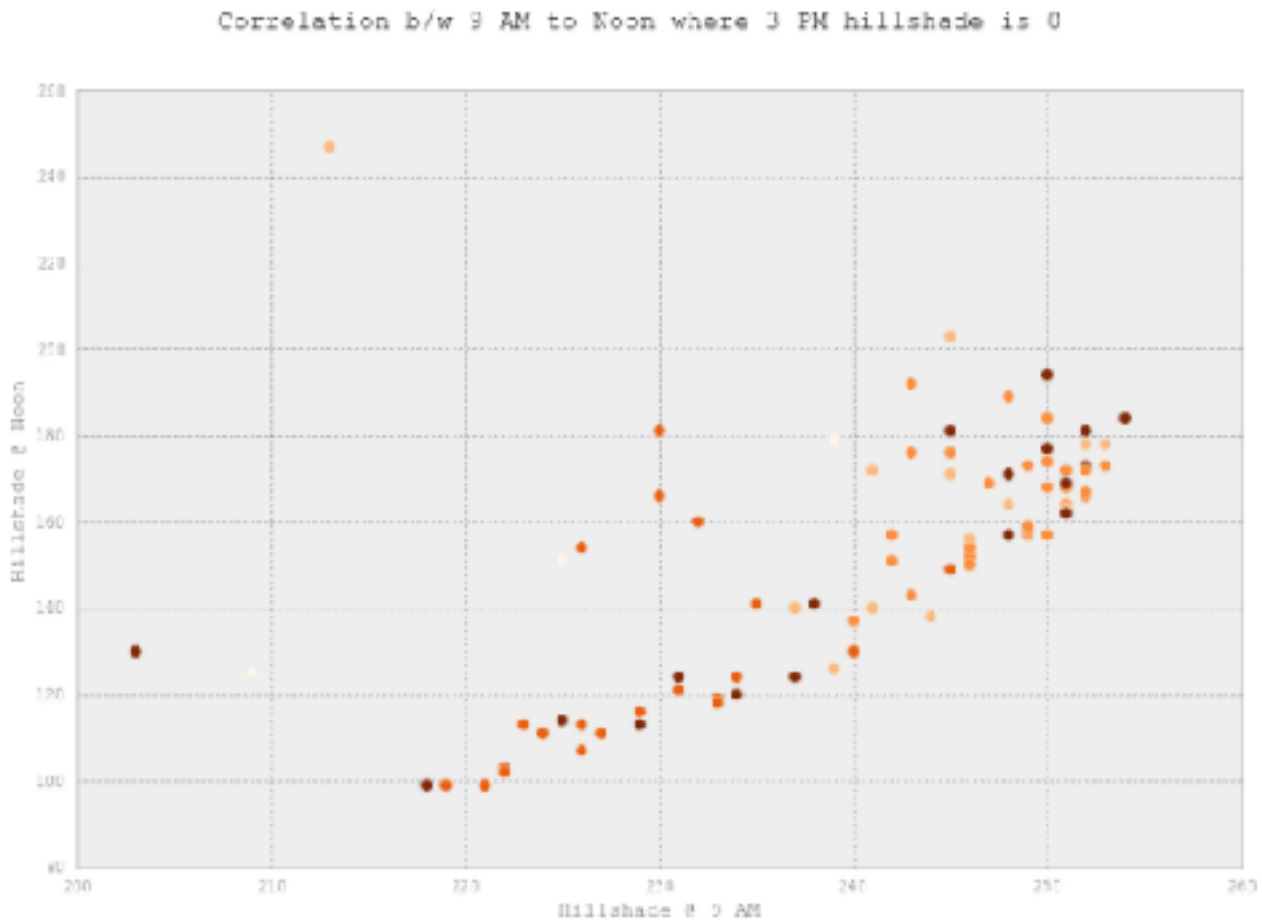
V. Conclusion

(approx. 1-2 pages)

Free-Form Visualization

Concerning the dataset I found interesting how the correlation between 9 AM to Noon where 3 PM hillshade is zero is interesting. That means that zeros have an important weight on this dataset that I have discussed previously.

This is important to notice that correlations between different features could be the key to improve the model by pick just most important features based on correlation between each other. We notice few outliers and most part of them are very well distributed. that's one of most clear example I found in order to explain the importance of finding a correlation between features in this kind of dataset and classification problem such as cover forest type. Anyway, feature correlation is thus important on several classification problem and not only this one. This is very useful to make the algorithm better and focus only on features that matter and that will help us most in order to apply the optimal model and predict the target feature correctly with a high precision.



Also in this picture below is interesting to see the correlation between hillshade Noon and 3 pm. I think that it was also interesting to notice about the weight of those features and how they have contributed to the final score.

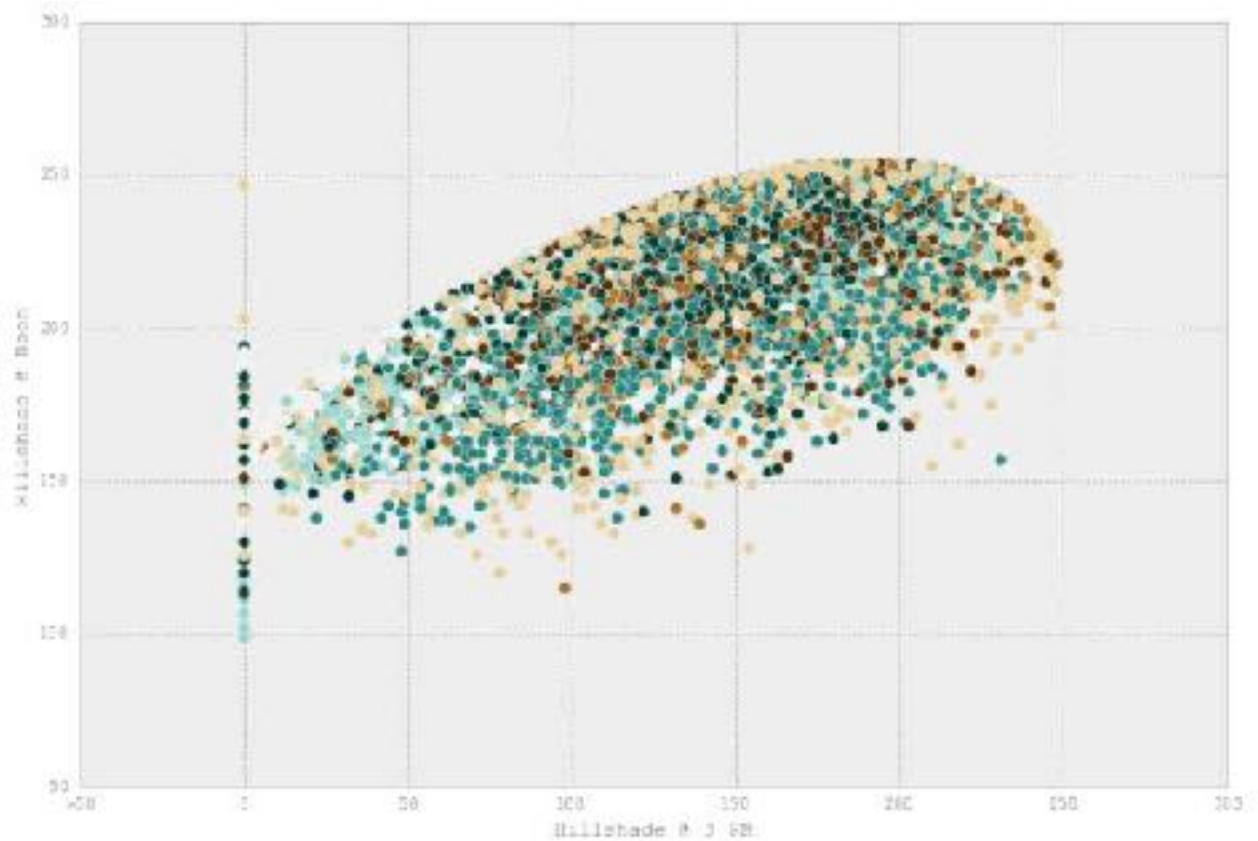
Reflection

In this project I predicted the forest cover type for 4 wild areas in USA.

I initially used ExtraTrees classifier on the default dataset and I submitted this to the kaggle project in order to have a benchmark going forward.

I have then clean up the dataset by deleting useless features that could impact the algorithm I have chosen in order to get a better score. Then I have ran the new algorithm (random forest) on the dataset and I have succefully obtained a better result even if the number of estimators was the same for both algorithms.

It was an interesting project because I have learned how to approach a supervised machine learning problem and all steps necessary to take advantage of all skills I acquired thanks to Udacity. Especially the to explore the data that in this case was very particular because I also found one hot encoded features (eg. Soil type) and how deal



with it. Then I have found a way to clean up the dataset and reuse it to put it into the test before to use it on the test dataset and submit this to kaggle. In fact also that way to submit and use kaggle was very interesting. I think that initially was a mystery how to handle a kaggle project but I am ready today to dive into a new machine learning challenge.

Improvement

Honestly I am far to have done a perfect job and this can be improved by someone that is not a rookie like me.

First of all, it would be possible to have a better feature engineering and rework the dataset in order to have an higher score on Kaggle. For example, adapt the dataset for a SVM algorithm could be a good improvement (of course it is necessary a minmaxScaler and Normalization

of data). Of course also the models could be improved by testing more `n_estimators` or tuning other parameters.

ⁱ <http://archive.ics.uci.edu/ml/index.php>

ⁱⁱ http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html

ⁱⁱⁱ <http://zhangchuxu.net/courses/536.pdf>

^{iv} <https://www.kaggle.com/c/forest-cover-type-prediction/leaderboard>
