MAP553

CLASSIFICATION PROJECT

08 Janvier 2023

Benjamin Lapostolle

TABLE DES MATIÈRES

1 INTRODUCTION

This project aims to classify the type of forest in a specific region of the United States based on more than 40 digital features. The goal is to accurately determine the forest type for each area within the region. To achieve optimal performance, I plan to follow the following steps in developing the model:

1. Feature engineering and Feature selection
2. Model Selection

2 FEATURES ENGINEERING AND FEATURES SELECTION

2.1 Features Engineering

In this section, we are attempting to identify new features that will enhance the performance of our algorithms. The initial analysis of the available features indicates that they do not all carry equal weight. For example, the feature *Elevation* seems to be particularly effective at distinguishing between different *Cover_Type* as shown in the boxplot1, which aligns with our expectations. In contrast, other features like *Sol_Type* appear to be less impactful. This may be because our training data contains insufficient examples of certain *Sol_Type*, preventing this variable from effectively discriminating between different *Cover_Type*.

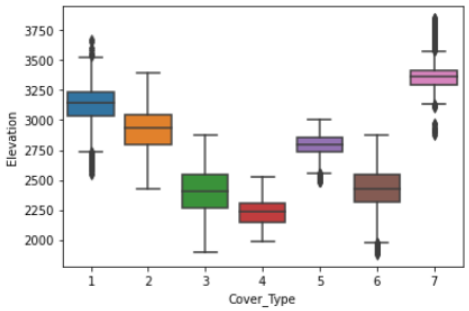

Figure 1: Boxplot showing the influence of the feature Elevation

To optimize our model's performance, we apply mathematical functions to the most useful variables. Specifically, we use the $log(1+p)$ function and polynomial functions. The resulting variables are:

1. Log1p of : *Elevation, Slope, Horizontal_Distance_To_Hydrology, Horizontal_Distance_To_Roadways, Hillshade_9am, Hillshade_Noon, Hillshade_3pm, Horizontal_Distance_To_Fire_Points*
2. *Square of Elevation*, to highlight the range of values taken by this variable

3. *Cube of Vertical_Distance_To_Hydrology*, to keep the negative values taken by this variable
4. Dividing the Hillshade_9am variable into 10 bins
5. *cos_Aspect* : calculating the cosine of Aspect since it is an angle
6. *Distance_To_Hydrology* : the distance to hydrology calculated using *Horizontal_Distance_To_Hydrology* and *Vertical_Distance_To_Hydrology*

In total, we have 67 variables after excluding *Id* and *Cover_Type*.

2.2 Features selection

As previously stated, some features carry more weight than others. To optimize computation time and prevent overfitting, which can hinder the performance of our algorithms, it is advisable to use feature selection techniques to reduce the number of features if necessary. There are several methods available for this purpose.

2.2.1 • Removing features with low variance

Through this technique, we remove variables that have a low variance and therefore do not contain enough information to discriminate the data and determine the *Cover_Type*. By applying this technique, we typically observe that the *Soil_Type15* variable has a variance of zero: this variable is always equal to 0. Therefore, we can immediately exclude it from the useful variables for our problem. This technique also shows other features that have a low variance, and these are mainly variables of the *Soil_Type* categories. This is not surprising, as we had already noticed that these variables were equal to 1 in few samples. I decided to keep them for now, having in mind that these variables may not be relevant in our problem. The table below1 lists the features and their variances when they were less than 0.001.

Type 7	**Type 8**	**Type 9**	**Type 15**	**Type 21**	**Type 25**	**Type 27**	**Type 28**	**Type 36**
6.61e-05	1.32e-04	2.64e-04	0.00	6.61e-04	3.97e-04	5.29e-04	4.63e-04	9.25e-04

Table 1: Variance of Soil_Type features

2.2.2 • Recursive Feature Elimination

Another class of feature selection method is the recursive elimination method. In particular, we use a method that calculates the importance of each variable — which is provided, for example, by the *feature_importance_* attribute of a *RandomForestClassifier* — and removes the least important variable. This process is repeated until the number of variables is equal to a given value. The list of these variables after this process for n_feature $\in [30, 35, 40, 45, 60, 67]$ is included in the appendix.

In order to determine the effect of the number of variables on the model's performance, we applied 6-fold cross-validation to a *RandomForestClassifier* and calculated the resulting score. The results

are depicted in the attached figure2. It appears that the optimal number of variables is 60, so I will continue using these 60 variables for the remainder of the analysis.

$n_{features}$	**30**	**35**	**40**	**45**	**60**	**67**
score	$0.86 \pm 6e-3$	$0.861 \pm 63e-3$	$0.864 \pm 72e-3$	$0.865 \pm 79e-3$	$0.869 \pm 6e-3$	$0.865 \pm 7e-3$

Table 2: Performance of a random forest classifier with respect to $n_{features}$

Another interesting test that could have been conducted is to calculate the algorithm's performance for each set of variables by removing a variable from the list of variables, and observe which removal of variable results in the least decrease in the algorithm's performance. This way, we can iteratively remove features. The disadvantage of this method is that it takes a long time to compute. As the results of the recursive feature elimination (RFE) seem satisfactory, I did not implement this technique.

Additionally, another technique for reducing the number of features while preserving their variance is to use a Principal Component Analysis (PCA). I tested the performance with 60 variables, and then with 50 variables obtained from a PCA of the 60. I found that the performance decreased — I obtained a score of 8.64 —, so I did not choose this solution. In conclusion, out of the 67 available variables, I kept only 60. In our problem, since the training dataset is relatively small, the algorithms run quickly and since that the performance increases with the number of variables until 60, I did not find it particularly necessary to significantly reduce the number of features. That is why I kept 60 of them.

3 MODEL SELECTION AND MODEL TUNING

This part is divided into two subcategories. First, we will test different models and observe their performance in order to select the best ones. Then, we will fine-tune the hyperparameters in order to improve these performances and achieve the best results.

3.1 Model Selection

We perform cross-validation for different models using the previously selected variables. The results obtained are presented in the figure below3. We can see that the performance of the algorithms is quite similar, however we can see that Random Forest and the LGBM algorithm show the best results. Therefore, I will use these two models to fine-tune the hyperparameters, especially since the LGBM algorithm is well-suited for hyperparameter tuning.

Further remarks :

- **K-Nearest Neighbors** : I wrote the best results obtained for $k = 3$ among $[3, 4, 5, 6, 7]$
- **Support Vector Machines** : The optimal performance was achieved with a third-degree polynomial kernel

Model	accuracy score
Random Forest	$0.869 \pm 0.6 * 10^{-2}$
Logistic Regression	$0.725 \pm 0.5 * 10^{-2}$
K-Nearest Neighbors	$0.804 \pm 0.55 * 10^{-2}$
Support Vector Machines	$0.75 \pm 0.34 * 10^{-2}$
LightGBM	$0.862 \pm 0.8 * 10^{-2}$
Neural Network	$0.79 \pm 0.11 * 10^{-1}$

Table 3: 6-fold cross-validation on different classification algorithms

Parameter	Value
Dropout	0.2
Learning rate	$1 * 10^{-4}$
Epochs	150
Loss function	Cross-Entropy
Optimizer	Stochastic Gradient Descent

Table 4: Parameters of the neural network

- **Scaling the data** : for **K-Nearest Neighbors**, **Support Vector Machines** and **Logistic Regression**, I first applied standardization to the data using the *StandardScaler* method from *scikit-learn*.
- **Neural Network** : the parameters are given in the table4.

3.2 Model Tuning

To carry out this task, I employed the grid search method from *Scikit-learn*, which easily allows for cross-validation of various parameters to be tested. For the Random Forest, I carried out a grid search on the parameters n_estimators and max_depth in $[100, 300, 400, 500, 600, 700, 800, 900, 1000]$ and $[15, 20, 25, 30, 35, 40, 45]$. The results were as follows: n_estimators = 900 and max_depth = 35 with a score of 8.77. The various scores obtained can be found in the annex.

As for the LGBM algorithm, many parameters can be optimized. To this end, I took several steps. First, I optimized the parameters num_leaves and max_depth, then the parameters n_estimators and learning_rate, all the while retaining the best estimators obtained previously. As a result, the following were determined to be the best estimators for LGBM: learning_rate = 0.05, max_depth = 12, n_estimators = 6000, num_leaves = 180, and I obtained a score of 8.81 in accuracy.

Upon plotting the confusion matrix for predictions made by the LightGBM algorithm, we obtain the following figure2. It can be seen from this figure that classes 1 and 2 are particularly poorly predicted and are frequently confused. One potential way for improving performance would be to identify discriminatory variables between category 1 and category 2.

NB : While studying the datasets, I noticed that the training data was included in the test set. As a result, after producing predictions on the test set, I perform a final step where I copy the labels

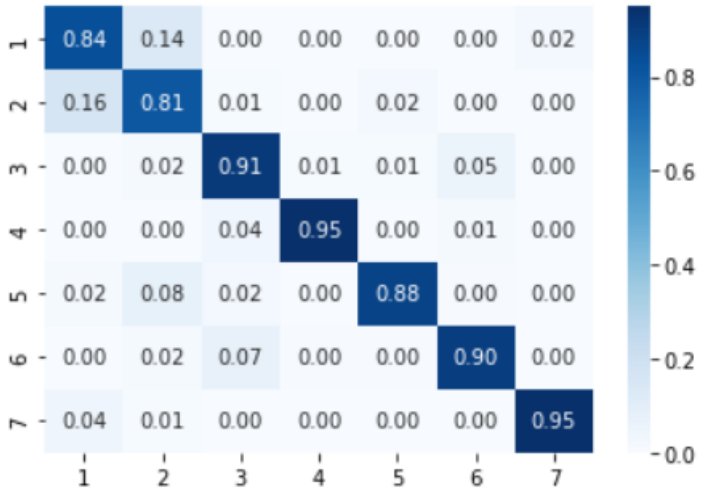

Figure 2: *Correlation matrix with Kendall method*

from the training set and paste them onto the predictions.

4 CONCLUSION

In conclusion, in order to achieve the best precision score in this exercise, I first created new variables from those that, according to a preliminary study, seemed to be important for classification. Then, to improve computing time and to avoid a large number of variables that could decrease performance by for example overfit my training data set, I used feature selection methods to keep the most relevant features. Next, I compared different models on the pre-processed data using K-fold cross-validation, selected the two best models — **Random Forest** and **LightGBM** — and fine-tuned their hyperparameters to further increase performance.

5 ANNEXES

5.1 Correlation Matrix

5.2 List of Selected Features

$n_{feature} = 30$: *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_I Wilderness_Area1, Wilderness_Area3, Wilderness_Area4, Soil_Type3, Soil_Type10, Soil_Type38, Soil_Type39,*

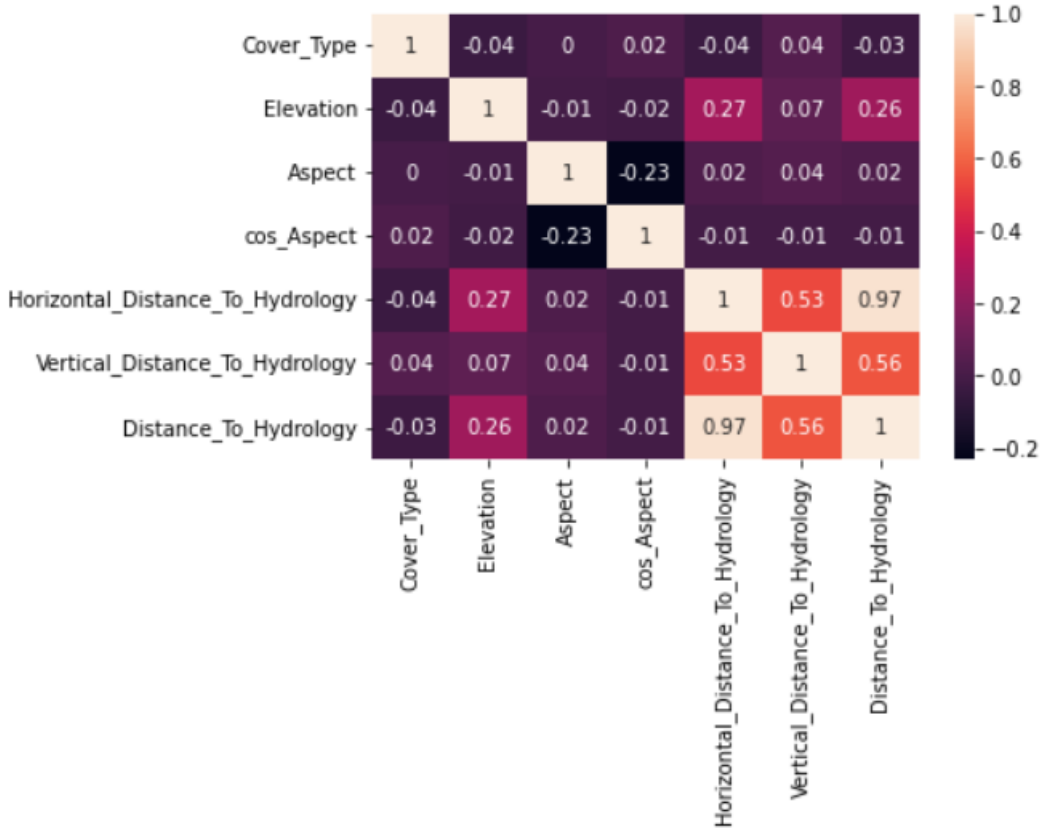

Figure 3: *Correlation matrix with Kendall method*

log1p_Elevation, log1p_Slope, log1p_Horizontal_Distance_To_Hydrology, log1p_Horizontal_Distance_To_Roadways, log1p_Hillshade_9am, log1p_Hillshade_3pm, log1p_Horizontal_Distance_To_Fire_Points, cos_Aspect, Distance_To_Hydrology, Elevation_Square, Horizontal_Distance_To_Fire_Points_over_Elevation, Hillshade_9am_binning cube_Vertical_Distance_To_Hydrology

$n_{feature} = 35$: *Soil_Type2, Soil_Type4, Soil_Type12, Soil_Type30, Soil_Type40*

$n_{feature} = 40$: *Wilderness_Area2, Soil_Type13, Soil_Type17, Soil_Type29, Soil_Type32*

$n_{feature} = 45$: *Soil_Type6, Soil_Type22, Soil_Type23, Soil_Type31, Soil_Type33*

$n_{feature} = 60$: *Soil_Type1, Soil_Type5, Soil_Type11, Soil_Type14, Soil_Type16, Soil_Type18, Soil_Type19, Soil_Type20, Soil_Type24, Soil_Type26, Soil_Type28, Soil_Type34, Soil_Type35, Soil_Type36, Soil_Type37*

$n_{feature} = 67$: *Soil_Type7, Soil_Type8, Soil_Type9, Soil_Type15, Soil_Type21, Soil_Type25, Soil_Type27*

5.3 Grid Search Results

	100	300	400	500	600	700	800	\
15	0.848214	0.851521	0.850397	0.849934	0.851124	0.851984	0.851323	
20	0.860450	0.864418	0.862831	0.863757	0.865476	0.864286	0.863823	
25	0.863624	0.865476	0.865741	0.865675	0.866534	0.865476	0.866005	
30	0.863029	0.866534	0.865410	0.866138	0.866468	0.866138	0.866270	
35	0.865013	0.866997	0.866402	0.867262	0.867460	0.866601	0.867394	
40	0.862302	0.865939	0.865873	0.864881	0.866005	0.867328	0.866402	
45	0.864484	0.866336	0.863823	0.866799	0.864286	0.865542	0.866468	

	900	1000
15	0.850992	0.852315
20	0.864616	0.863955
25	0.866931	0.865939
30	0.865741	0.866601
35	0.866931	0.867130
40	0.865807	0.867593
45	0.866534	0.866799

Figure 4: *Grid Search of the parameters n_estimators and max_depth in Random Forest*

	20	100	200	400	800	1500	3000
3	0.781812	0.781812	0.781812	0.781812	0.781812	0.781812	0.781812
6	0.835185	0.839484	0.839484	0.839484	0.839484	0.839484	0.839484
9	0.847884	0.860053	0.860979	0.860979	0.860979	0.860979	0.860979
12	0.847024	0.866799	0.868783	0.868717	0.868717	0.868717	0.868717

Figure 5: *Grid Search of the parameters num_leaves and max_depth in LGBM*