Classification of forest cover types in undisturbed forests

Sebastián Andrés Cajas Ordóñez

Kaggle Username: SEBASTIÁN CAJAS

NeptunID: RK9KJV

The idea behind this classification task is to identify 7 different types of forest cover types in undisturbed forests. During this analysis, first please run the EDA called entitled CAJAS_RK9KJV_EDA and afterwards, the code CAJAS_RK9KJV_MODEL. These models were created using Visual Studio Code with the plugin of Jupyter notebook. The first one, re-writes the training set with the data engineering features and the second deploys the different tested models, as descripted on the Models section below

I. Exploratory Data Analysis – EDA

Conducting EDA to understand our data, the first thing to do is to check whether there is or not missing data, for this I have used the command *isnull()* and was able to check that any of the data points over the 54 different attributes was missing. The data is complete or doesn't need any type of data interpolation since there is not data missing. This conclusion was also possible to check with the use of the library missingno, which let a visual inspection of whether or not there is data missing over the dataset. The goal of this project is to perform the multi-classification of 7 different types of forest cover types: 1) Spruce/Fir, 2) Lodgepole Pine, 3) Ponderosa Pine, 4) Cottonwood/Willow, 5) Aspen, 6) Douglas-fir, 7) Krummholz.

From the *Figure 1,* we can see that the cover type 1 and 2 are the most predominant features over the dataset, while labeling from 3 to 7 are much lower. From the Figure below, we can see that the dataset is not completely balanced, there are more counted classes over class 1 and 2 in comparison to classes 3 to 7.

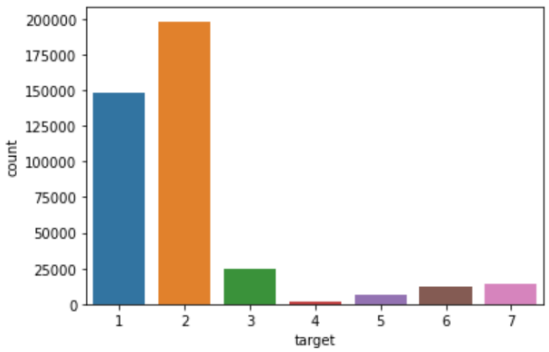
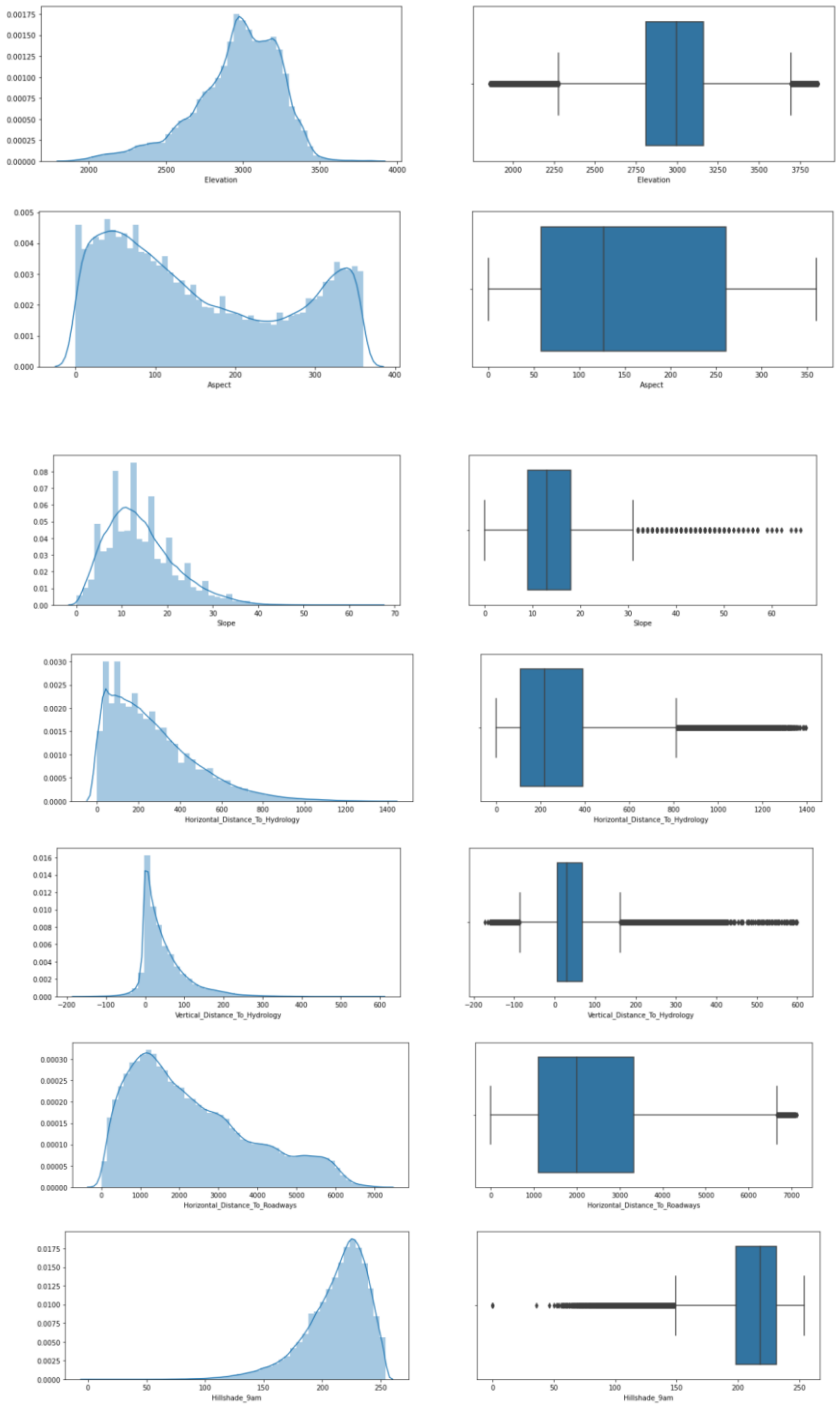


Figure 1. Bar plot for class forest types

Plotting all the variables using the seaborn library, we can see depicted below the distribution of all the variables one by one, next to its boxplot. From this we can infer that most of the territories have an elevation of 3000 meters, a mean aspect of 120, a horizontal distance of roughly 200 m, a slope of 15 m, horizontal distance to hydrology of 210 m, vertical distance to hydrology of 30 m, a

horizontal distance to roadways of 2000 m, a hillshade at 9 am of 220m, a hillshade at noon of 125m, a hillshade at 3 pm of 140, a horizontal distance to fire points of 1800 m.



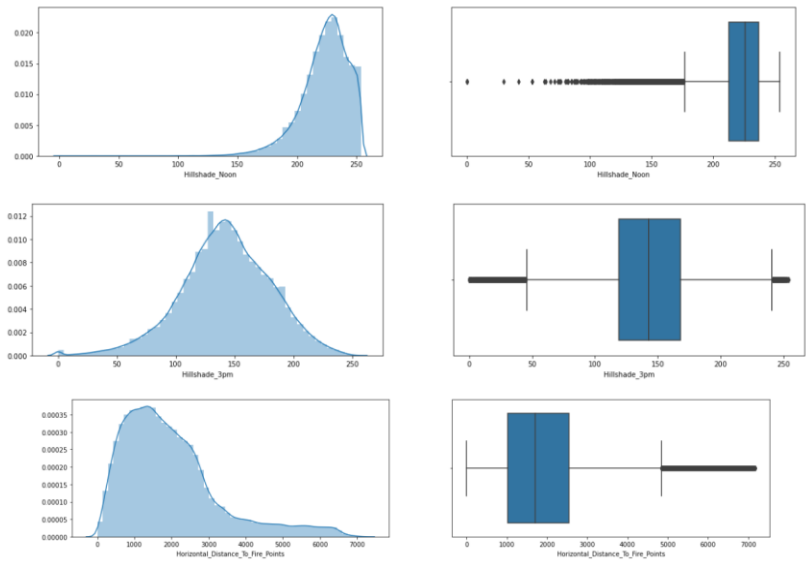


Figure 2. Histogram and boxplot distribution over attributes of dataset

Data labels are not completely balanced: class 1 and 2 are much more abundant over the dataset. Something that can easily be seen is that class 1 and 2 have a very large Elevation, Slope and aspect attributes, indicating that these features might have a strong correlation for classifying these classes. Also, the number of instances over the entire dataset is bigger, showing that the data is not fully balanced, so it would explain why these two classes are much strongly present. The only exception this problem would be on the number of occurrences over class 6 and 7, since the number of classes is similar as shown on class 1 and 2 over distances, elevation, aspect, slope, distance to hydrology and soil types.

Outliers analysis.

The quartiles divide the data into 4 equal regions. Instead of dividing by 100 in step 2, divide by 4, where the 2nd quartile is the same as the median. The 1st quartile is the 25th percentile, the 3rd quartile is the 75th percentile. [1]

Therefore, analyzed whether the data contains outliers with respect to the quartile ranges of the numeric variables. The idea of this was to detect the first and third quartile and interquartile range for a given column of a dataframe and then calculate the upper and lower limits to determine outliers. Then the upper/lower_limit was determined by the following expressions:

```
upper_limit = third_quartile+(3*IQR)
lower_limit = first_quartile-(3*IQR)
```

Then I would determine whether the value over the dataframe is under the lower limit or over the upper limit, if that was the case, then I would this outlier and its location. From this analysis, I extracted the the intervals within each attribute and found that the attribute called `Horizontal_Distance_To_Fire_Points` and `Horizontal_Distance_To_Roadways` were the ones that reported the biggest amount of outliers, therefore I used the selected upper/lower

limits and cut the signal. The result showed that 406701 rows while before there were 406708, so there was not too much change since the data seems to be without many outliers.

II. Data preprocessing

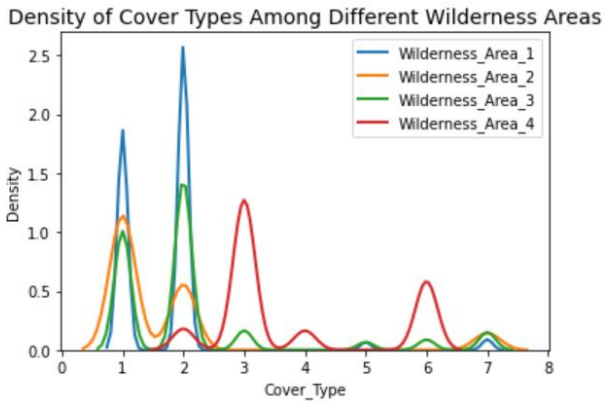


Figure 2. Density of Cover types over wilderness areas

a. Once it was verified that the columns of soil and wilderness contain only zeros and ones (hot-encoded format), the next step was to convert the soil type 1-40 from hot-encoded type, to interger type and merged them into one single column called soil_type.
b. Convert Wilderness areas 1-4 from binary type, to number type and merged them into one single column called wilderness_Area.
c. From *Figure 2,* the wilderness area categorical variables were sparsed for the Forest Cover types in order to analyze its distribution and its relationship with the wilderness area. For example, from this figure we can deduce that the cover type 1 is presented in wilderness areas 1, 2 and 3. The cover type 2 is presented mainly over the wilderness area 1 and less predominantly over the resting areas. The cover type 3 and 6 are presented mainly over the wilderness are 4 and the cover types 5 and 7 are presented mostly over wilderness area 3. This is an important graphic because it show us that the wilderness area attribute has a very important relevance into the classification of the forest cover types. Finally from this section, Convert the wilderness area values into integers 1-4 for easier analysis and computation, since later we would like to make some data transformation to see if it could improve our model performance

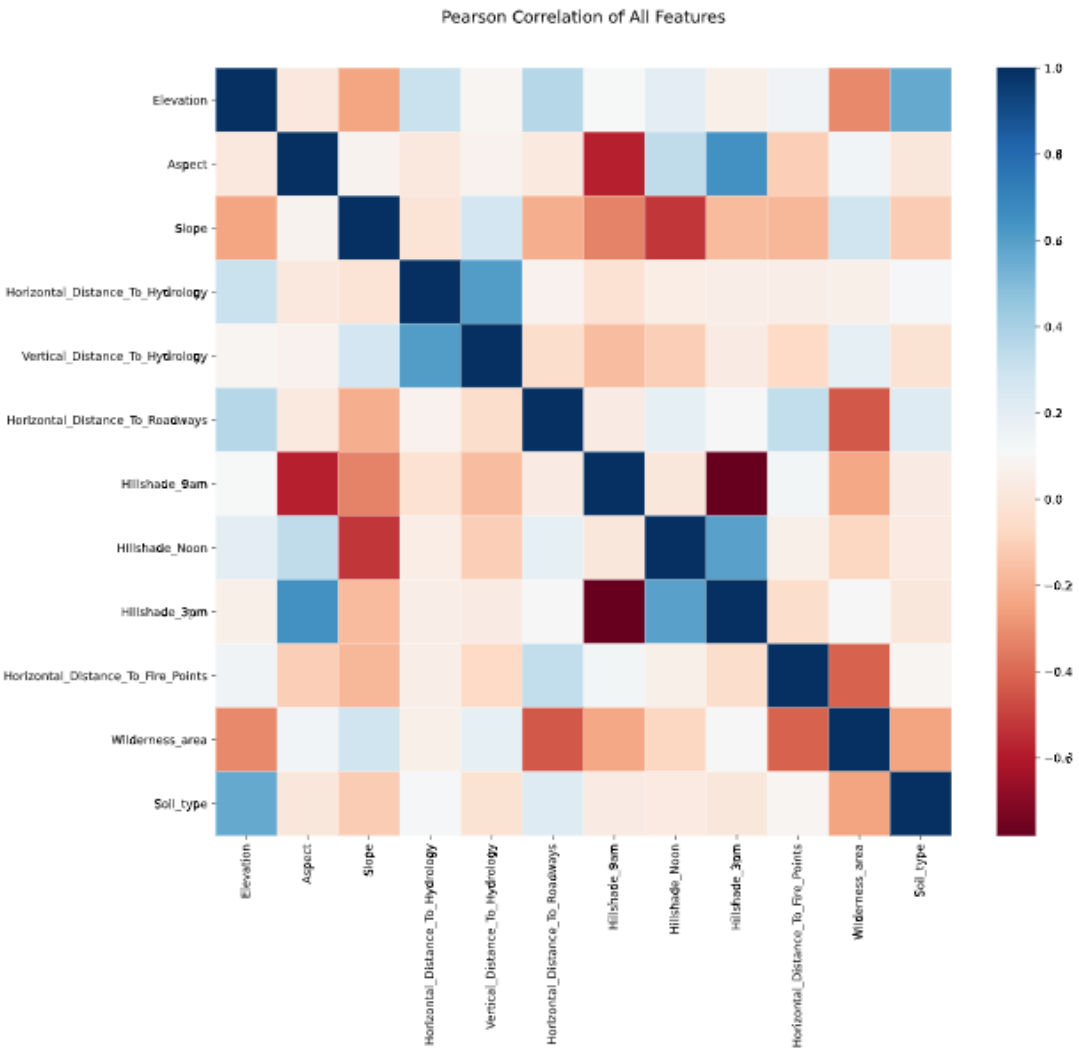


Figure 3. Correlation between attributes of the entire dataset.

d. From *Figure 3* we can see the Pearson correlation between our transformed variables, beforehad the same graph had been obtained using the hot-encoded format, however this plot did not yield as much information as this one, because here we can see:

- 1) there is a dependency between Elevation and Slope, and indeed this makes sense since the elevation is given in meters for the elevation of the territory, and the aspect is related to the declination in degrees on the same territory.
- 2) there is a dependency between elevation and wilderness area, which basically represents whether there is absence or presence of a wild area, meaning that the elevation is directly related to this flickering.
- 3) relationship between Elevation and Soil type.
- 4) relationship between Hillshade at 9 am and at 3 pm with Aspect, which reached a correlation of -0.58 and 0.65, respectively

- 5) Between slope and Hillshade at noon, which makes sense since the higher the slope, the higher the shade at this time of the day.
- 6) Horizontal distance to hydrology seems not to be related to any other variable, except to the vertical distance to hydrology, which are highly correlated with each other, yielding a correlation of 0.61.
- 7) the correlation between Hillshade at 9 am and the one at 3 pm presented a correlation of -0.78, which is a big dependence.
- 8) Horizontal distance to roadways and horizontal distance to fire points have a correlation with wilderness area.
- 9) Wilderness area and the soil type are also correlated with each other.

Data transformation: During the data transformation, the Euclidean distance to hydrology is calculated. The idea is to use the variables of the vertical and horizontal attributes to perform a data transformation, which is the distance from the selected point and the hydrologic section instead of the other variables. Additionally, other transformations were calculated as well, which considered the mean between the next attributes: Mean elevation to vertical-hydrology,Mean between horizontal distance to hydrology and firepoints, Mean between Mean_Distance_Hydrology_Roadways and Horizontal_Distance_To_Hydrology and finally the Mean between Horizontal_Distance_To_Fire_Points and roadways.

III. Algorithms

a. KNN Model: The KNN algorithm is a non-parametric method that is hugely used for classification and it is based in the distance between the different data points to conclude whether the data is contained within a specific class. The most important parameter is the number of neighbors, which was determined during the hyper tuning on the section IV. This was also the baseline algorithm that I used recursively to determine whether the data transformations and data analysis was improving the performance of the model and then once it showed improvements, I would apply this into the next models.

b. **Random forest model:** During this section I use the `feature_importances_` module, which depicted the feature ranking, from which I did a feature selection field vector and tested the Random forest and KNN models, to test its performance over feature selection.

```
Feature ranking:
                                    Elevation feature 0 (0.259108)
                                       Aspect feature 11 (0.141075)
                                        Slope feature 5 (0.123359)
          Horizontal_Distance_To_Hydrology feature 9 (0.118188)
            Vertical_Distance_To_Hydrology feature 3 (0.060067)
           Horizontal_Distance_To_Roadways feature 4 (0.056522)
                              Hillshade_9am feature 10 (0.047879)
                             Hillshade_Noon feature 1 (0.046245)
                              Hillshade_3pm feature 7 (0.041435)
       Horizontal_Distance_To_Fire_Points feature 6 (0.039023)
                            Wilderness_area feature 8 (0.037404)
                                  Soil_type feature 2 (0.029695)
```

c. Classification trees: As seen during the lectures, we learned about the gini criterum and entropy criterium, with the idea of seeing the importance of information gain to provide the maximal information about the classification. Then I tried the classification trees in this regard, by specifying the Entrope as main criterium first over the cleaned dataset 1 and secondly, focused on the gini index criterium,which

is focused on calculating the amount of probability of a specific feature that is classified incorrectly when selected randomly [2].

d. **XGBoost:** This model improves according to the gradient between the different variables and uses decision trees ensembles to make predictions. I chose it because it is easy to train and doesn't require many computational resources and usually works better with categorical and numeric features, which is this case.

e. **ExtraTreesClassifier:** This is an ensemble learning method based on decision trees over several subsamples and average them to improve the accuracy and over-fitting [3].

IV. **Tuning:** During this section, the tuning of the KNN model was mainly performed over several cases of study, considering its accuracy performance during several data transformations and data manipulation during the section II. Therefore, the section IV.a, which describes the tuning process for this algorithm, was used to obtain the best possible dataset, which was ultimately used by the other models (Models IV.b, IV.c, IV.d).

a. **KNN Model (baseline):** This model was initially tuned by iterating over a for-loop between 1 and 30 neighbors, showing its best performance with 3 neighbors. Afterwards, the leaf_size and the Minkowski distance formula (p) parameters were stored in a dictionary and added to GridSearchCV function, which optimizes by cross-validated grid-search over KNN model trained with the preprocessed dataset alongside with 10-fold cross validation. This showed that the best parameters were an euclidean distance (p) equal to 1, leafs = 1 and 3 neighbors, the results are in section V, table 1.

1. Then, from figure 3 we can see that there is a high correlation between Hillshade at 9 am and at 3 pm, so in this particular case I actually tested this features with the KNN model as a reference to hypertune with the above parameters and analyze whether these change on the variables were affecting my model performance, so I took the following attributes only: 'Elevation', 'Aspect', 'Slope', 'Horizontal_Distance_To_Hydrology', 'Vertical_Distance_To_Hydrology', 'Horizontal_Distance_To_Roadways', 'Hillshade_Noon', 'Hillshade_3pm', 'Horizontal_Distance_To_Fire_Points', 'Soil_type', 'Wilderness_area'. Since Hillshade at 9 am is highly correlated with Aspect, Slope and hillshade at 3 pm, I removed it to avoid redundancy and the resulting accuracy was 96.69 using 10-fold cross validation.
2. From the **data transformation in section II,** the correlation between the cover type and the different attibutes was calculated, this those features that had a correlation inverior to 0.02 were neglected, which encompassed the following:Horizontal_Distance_To_Hydrology, Euclidian_Distance_To_Hydrology, Aspect, sqrtAspect, sqrtVertical_Distance_To_Hydrology, Vertical_Distance_To_Hydrology. Using the data transformation described here, the baseline model from section IV.a yielded an accuracy of 96.5%.
3. Using the module StandardScaler to normalize the variables was also considered in this case, then the results were 10-fold cross validated with the entire dataset. This also yielded an accuracy of 96.5%.

b. **Random forest model:** I used 10-fold cross-validation with 10 Estimators as an initial check of the algorithm based on the trained model which was fed with the cleaned dataset. Afterwards and in order to find the best possible parameters for this model, I created a

function which receives the training data into consideration with the number of estimators as main parameter, consequently I for-looped heuristically this algorithm in an interval of 1-30, to check the best results in terms of accuracy.

c. **Classification trees:** For hyper tuning this model, I used gini index and Entropy criterium separately, testing them with its default parameters and analyzing which might have the best performance.

d. **XGBoost:** This algorithm used the data transformation from the section IV.a.1, which delivered the best performance over the KNN baseline model. In order to tune it, I used a for-loop to test the learning rate from 0.01 to 0.1, and the number of estimators from 500 to 5000. From this the best results were obtained when the learning rate was 0.1 and the number of estimators was 1000.

e. **ExtraTreesClassifier:** To tune this model, the number of estimators was calculated amongst a series of candidates between 50 and 1000; then the minimum number of samples required to split an internal node was tested in the interval [2, 3, 5, 7, 9]; the minimum number of samples required to be at a leaf node was tested on the interval [1, 2, 4, 6, 8];
The number of features to consider when looking for the best split was tested between ['auto', 'sqrt', 'log2', None]. All these intervals were saved in a grid, and used the RandomnizedSearchCV module to find the best value with 10-fold cross validation and a random state of 42, using the command *random_Cv.best_estimator.* The best results yielded 400 number of estimators, max_features = none, minimum sample leafs of 1, minimum samples split of 2. After training this model with datasets described in IV.a.1 and IV.a.2, the best results were obtained with IV.a.1, yielding an accuracy of 97.1 %.

V. Results

a. Using the KNN model the accuracy improved to 95.4554 %, when considering the pre-preprocessed data. Changing number of neighbors to n=3, the accuracy improved to 96.0553%. Then the hypertuning using the GridSearchCV library was very useful, which improved the model hugely to 96.55%. Then when performing data transformation over the dataset, the model reached a high performance with the data selection presented in IV.a.1, which yielded an accuracy of 96.9%.

b. 10-fold Cross-validation using the model_selection mododule from sklearn and using the original dataset, yielded a precision on the leaderboard of 0.96989. Another detail about this analysis was that I used the the entire dataset, that is, since I had already tried splitting it using split at 70% of the data (for keeping the validationset appart), I tried with the whole dataset to check whether more details with the entire data were possible to detect using it as a whole.

c. The random forest classifier with best accuracy was using 30 number of estimators and with 10-fold cross valudation reached 95.00%

d. I extracted the ranking features using the feature importance module from random forest and selected the following features:
fields = ['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'], afterwards I compared this with Random forest and KNN models, where KNN showed the best results.

e. The best results were obtained using the ExtraTrees Classifier with the dataset IV.a.1, meaning that the contribution for cleaning this dataset were better in terms of the data transformation explained above. This model was tuned several times and also is computationally expensive, for details about the tuning, see section IV.e

Model	**Accuracy – Using splitting from Sklearn**	**Accuracy – leaderboard (50% of testing data)**
KNN model hypertuned withGridSearch, with 3 neighbors and over the initial preprocessed dataset	96.25%	96.25%
KNN model hypertuned withGridSearch, with CV=10 and 3 neighbors and over the original dataset.	96.91%	96.91
KNN model with 3 neighbors and feature selection as specified in III.b	96.08%	-
KNN model with 3 neighbors and feature selection as specified in IV.a.1	96.69%	-
KNN model with 3 neighbors, feature selection and data transformation described in section IV.a.2	96.5%	-
KNN model with 3 neighbors, feature selection and data transformation described in section IV.a.3	96.5%	
Random forest classifier using 30 estimators reached and using 10-fold cross validation	95.00%	95.00%
Random forest classifier using 30 estimators with features selection as specified in III.B	93.30%	-
Decision trees classifier with gini criterium over 10-fold cross validation	92.98%	-
Decision trees classifier with Entropy criterium over 10-fold cross validation	67.93%	-
ExtraTrees with hypertuning depicted in IV.e with with dataset described in IV.a.1	97.1%	97.1%

ExtraTrees with hypertuning depicted in IV.e with with dataset described in IV.a.2	97.02%	-

f. **Overview of relevant results**

a. It must be highlighted that the K-neighrest neighbor (KNN) was the baseline algorithm for the entire project, I performed an iterative process of analyzing the dataset with different encoded and decoded forms of the data, also including many data transformations in several combinations (such as ratio, summation and square root between different attributes) and was very useful to obtain rapid analysis on whether the data transformations and data engineering was affecting the model or not. I concluded that this is an important algorithm to recognize the forest cover types because of the graphs of the correlograms presented above, since seems like the distances and also the wilderness areas are somehow related in terms of statistical measurements such as the Euclidean distance and the square root of certain variables. Finally, I also re-used the obtained dataset (cleaned dataset) from where this model perform the best, to feed the resting algorithms.

g. **Conclusions**

a. Using quartile range analysis for detecting outliers can be very useful during continuous data pre-processing, however, despite of the performed analysis on the **outlier's section**, the number of outliers were non-significant and had no contribution or improvements over the presented models.

b. The best results were obtained when the wilderness areas (1 to 4) and soil types were translated from hot-encoded to integer, with either KNN or the Extra class classifier. Despite of the feature engineering performed over the dataset (calculating Euclidean distances and determining the mean between the floating variables), too much noise and redundancy was produced and due to this the tested algorithms didn't show the best results with these transformations.

c.

h. **References**

[1] https://people.richland.edu/james/lecture/m170/ch03-pos.html

[2] https://medium.com/analytics-steps/understanding-the-gini-index-and-information-gain-in-decision-trees-ab4720518ba8

[3] Optical Character Recognition using Ensemble of SVM, MLP and Extra Trees Classifier: Ref: **https://ieeexplore.ieee.org/abstract/document/9154050/**