Machine Learning Project

Earthquake magnitude prediction

Bușe-Dragomir Alexandru

CEN4S1A

17.01.2020

1. Dataset general information

The source for my dataset is represented by the website [www.kaggle.com](http://www.kaggle.com), one of the most used platforms of this kind together with Google Datasets, data.world and the UCI Machine Learning Repository.

I chose to use the dataset with the following name: **“Significant earthquakes, 1965-2016”.** As the name implies, it is a quite large set of data that contains relevant information about a large number of important (magnitude over 5.5 on Richter) earthquakes and their intrinsic and extrinsic properties (21 columns of data and over 20.000 records in total, although some rows have missing values).

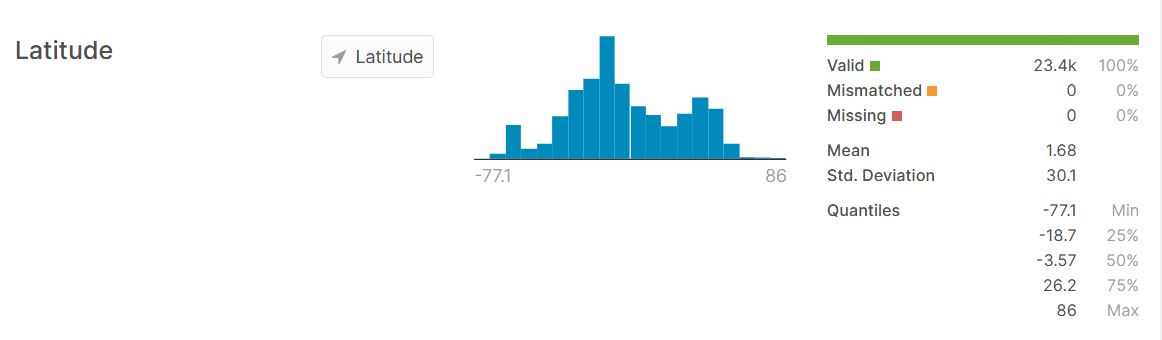
The source is represented by The National Earthquake Information Center which determines the location and size of all significant earthquakes that occur worldwide and disseminates this information immediately to national and international agencies, scientists, critical facilities, and the general public. The NEIC compiles and provides to scientists and to the public an extensive seismic database that serves as a foundation for scientific research through the operation of modern digital national and global seismograph networks and cooperative international agreements. The NEIC is the national data center and archive for earthquake information.

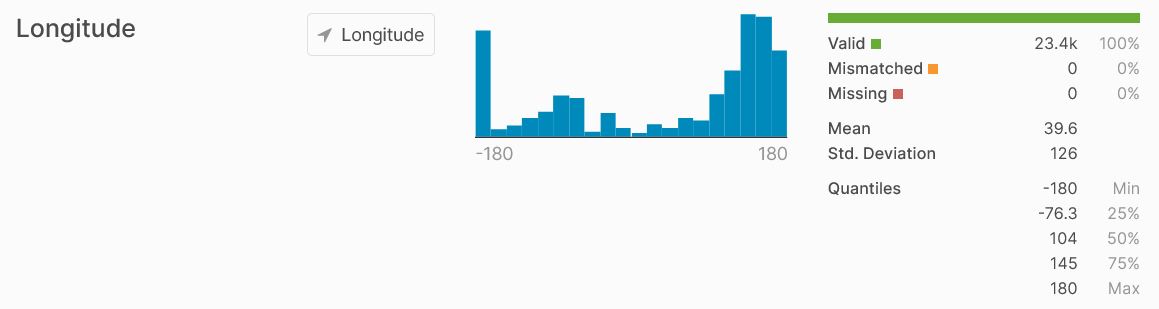
To conclude, this dataset includes a record of the date, time, location, depth, magnitude, and source of every earthquake with a reported magnitude 5.5 or higher since 1965.

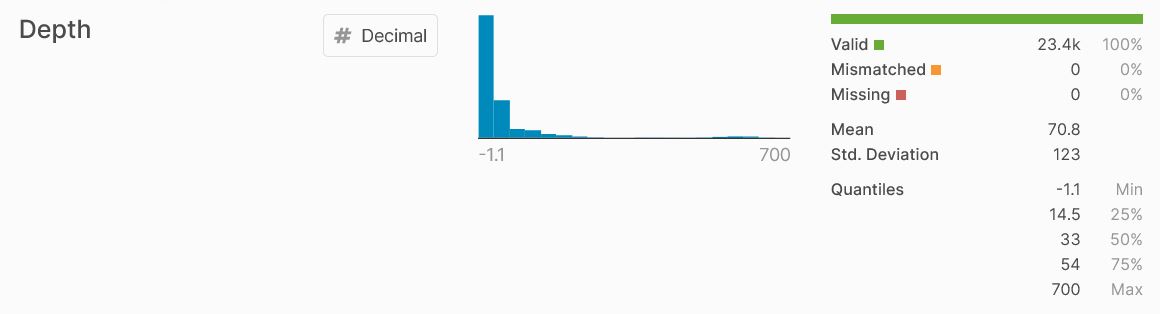
As some data had no clear relationship with the magnitude, the fields which have been kept for the final project are:

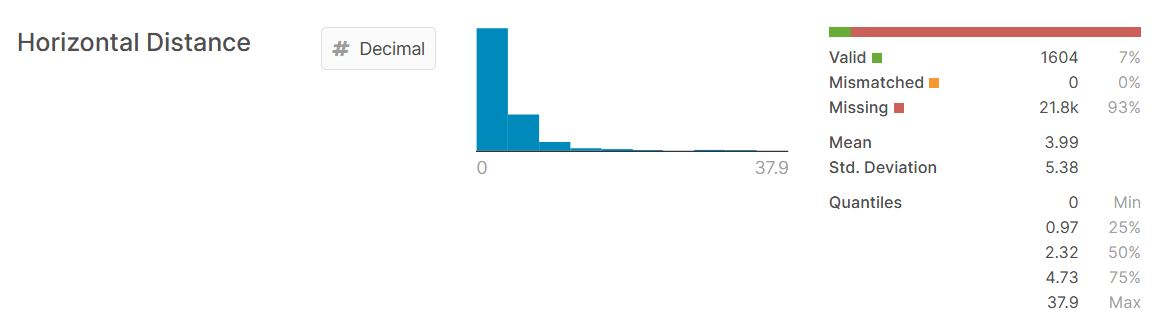
1. Latitude: [geographic coordinate](https://en.wikipedia.org/wiki/Geographic_coordinate_system) that specifies the [north](https://en.wikipedia.org/wiki/North)–[south](https://en.wikipedia.org/wiki/South) position of a point on the Earth's surface (the point that represents the epicenter of the earthquake).
2. Longitude: [geographic coordinate](https://en.wikipedia.org/wiki/Geographic_coordinate_system) that specifies the [east](https://en.wikipedia.org/wiki/East)–[west](https://en.wikipedia.org/wiki/West) position of a point on the [Earth](https://en.wikipedia.org/wiki/Earth)'s surface (the point that represents the epicenter of the earthquake).
3. Depth: the depth measured in meters at which the epicenter of the earthquake was measured to be.
4. Azimuthal gap: an azimuth is an [angular measurement](https://en.wikipedia.org/wiki/Angle#Measuring_angles) in a [spherical coordinate system](https://en.wikipedia.org/wiki/Spherical_coordinate_system). The [vector](https://en.wikipedia.org/wiki/Vector_space) from an observer ([origin](https://en.wikipedia.org/wiki/Origin_(mathematics))) to a point of interest is [projected](https://en.wikipedia.org/wiki/Graphical_projection) [perpendicularly](https://en.wikipedia.org/wiki/Perpendicular) onto a [reference plane](https://en.wikipedia.org/wiki/Reference_plane); the angle between the projected vector and a reference vector on the reference plane is called the azimuth.
5. Horizontal distance: horizontal distance from the epicenter to the nearest station (in degrees). 1 degree is approximately 111.2 kilometers. In general, the smaller this number, the more reliable is the calculated depth of the earthquake.
6. Root mean square: a post-stack attribute that computes the square root of the sum of squared amplitudes divided by the number of samples within the specified window used. With this [root mean square amplitude](https://en.wikipedia.org/wiki/Root_mean_square_amplitude), one can measure reflectivity in order to map direct hydrocarbon indicators in a zone of interest. However, RMS is sensitive to noise as it squares every value within the window.
7. Magnitude: a post-stack attribute that computes the maximum value of the absolute value of the amplitudes within a window. This can be used to map the strongest direct hydrocarbon indicator within a zone of interest.

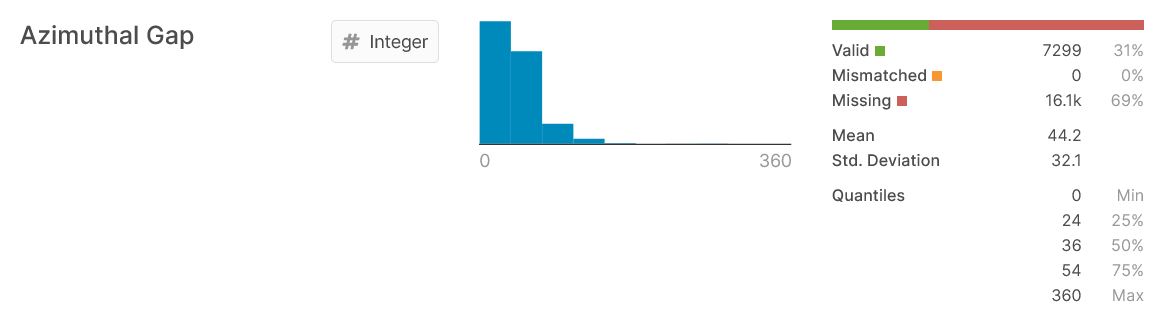
As the latitude, longitude, depth and magnitude are present for 99% of the records, these have been kept in an initial trial in order to have more than 21.000 sets of data for training. Results were disappointing, so all above columns have been kept for the final tests; the number of complete records for all 7 columns is 1604, but results are about 10 times better and some correlations can be made in this way.

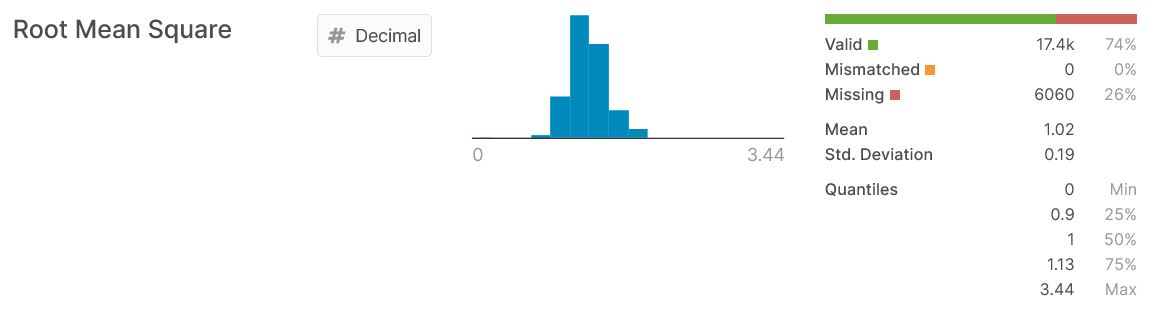














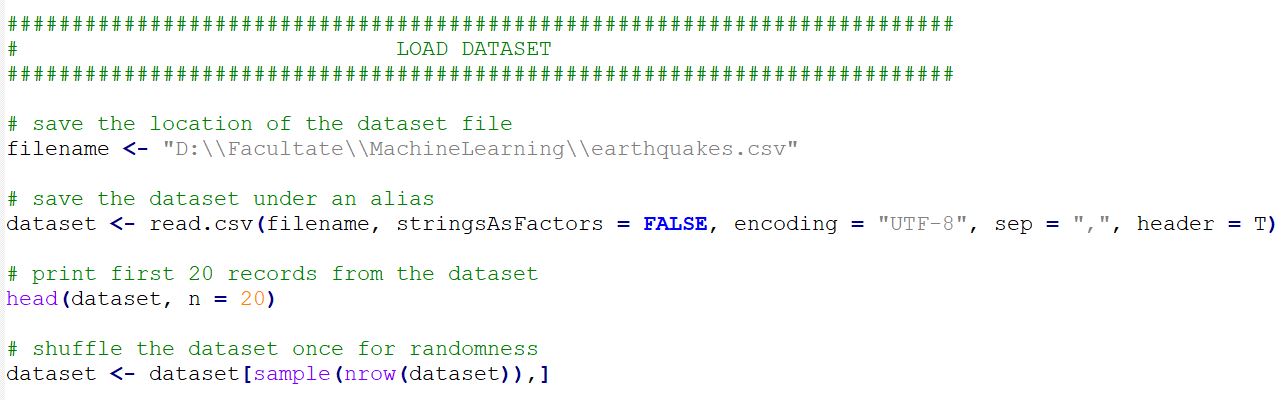
1. Libraries and loading the dataset

The following libraries have been used throughout the project:

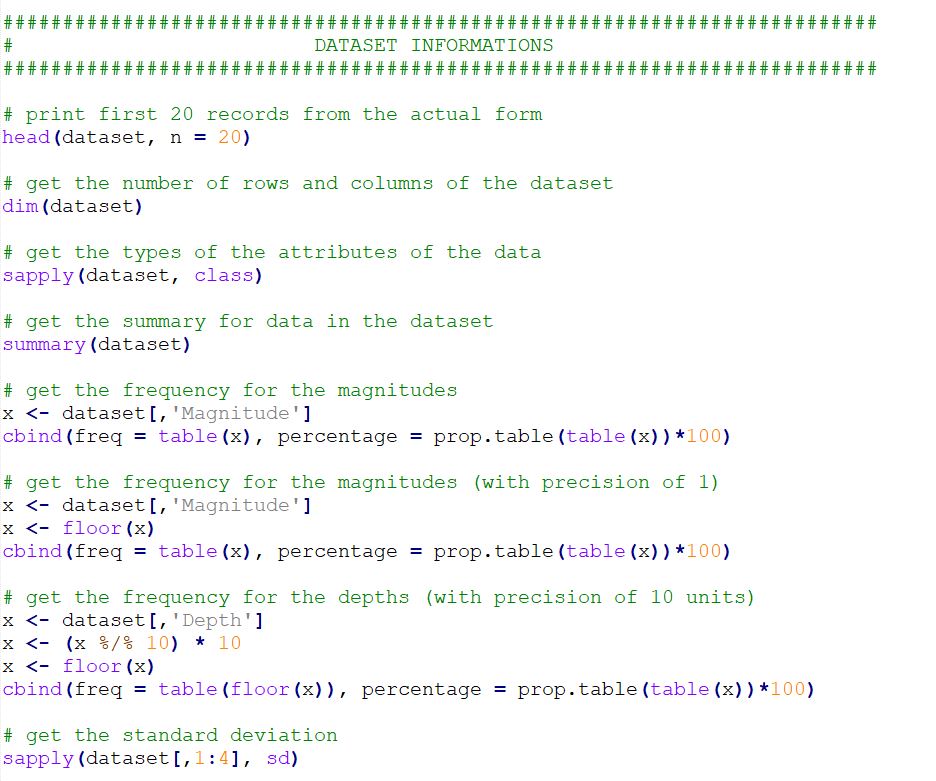
* *mlbench*: a collection of artificial and real-world machine learning benchmark problems, including, e.g., several data sets from the UCI repository.
* *e1071*: functions for latent class analysis, short time Fourier transform, fuzzy clustering, support vector machines, shortest path computation, bagged clustering, naive Bayes classifier etc.
* *lattice*: a powerful and elegant high-level data visualization system inspired by Trellis graphics, with an emphasis on multivariate data. Lattice is sufficient for typical graphics needs, and is also flexible enough to handle most nonstandard requirements.
* *corrplot*: a graphical display of a correlation matrix or general matrix. It also contains some algorithms to do matrix reordering. In addition, corrplot is good at details, including choosing color, text labels, color labels, layout, etc.
* *caret*: miscellaneous functions for training and plotting classification and regression models.
* *klaR*: miscellaneous functions for classification and visualization, e.g. regularized discriminant analysis, sknn() kernel-density naive Bayes, an interface to 'svmlight' and stepclass() wrapper variable selection for supervised classification, partimat() visualization of classification rules and shardsplot() of cluster results as well as kmodes() clustering for categorical data, corclust() variable clustering, variable extraction from different variable clustering models and weight of evidence preprocessing.



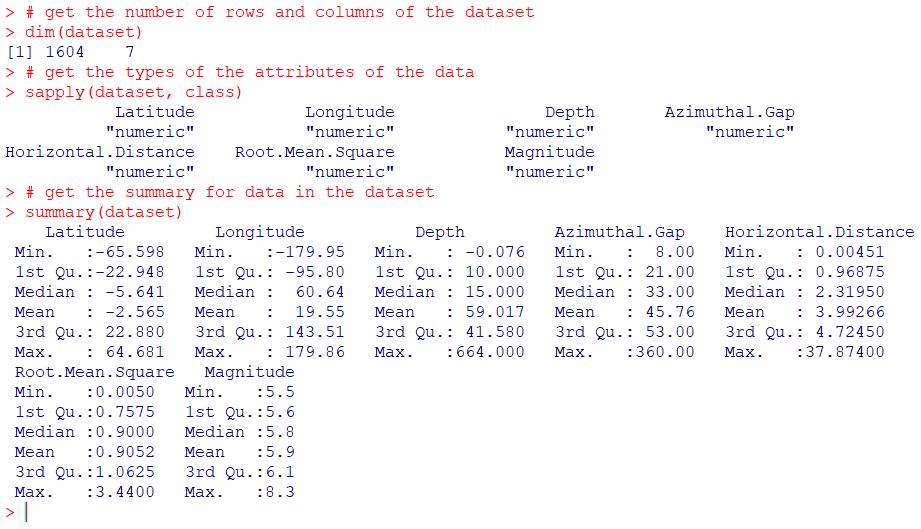
I saved the path towards the .csv file in a variable named filename and then used the read.csv function in order to save the data frame under the generic name “dataset” (also using UTF-8 as default encoding, specifying the separator to be a comma, strings not to be considered factors and first line to represent the title for the columns.

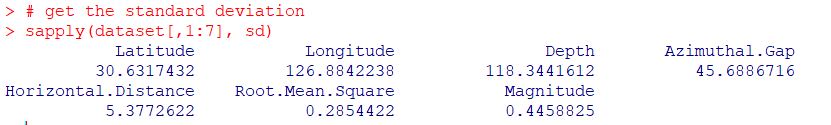


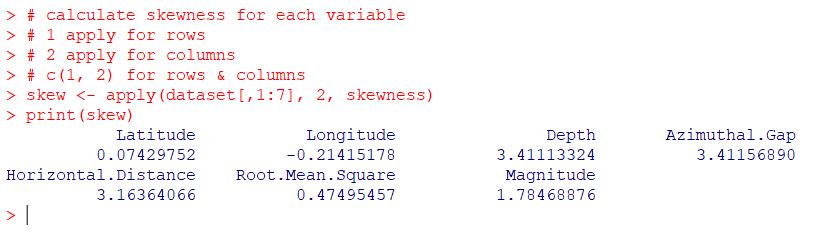
1. Data visualization and statistics

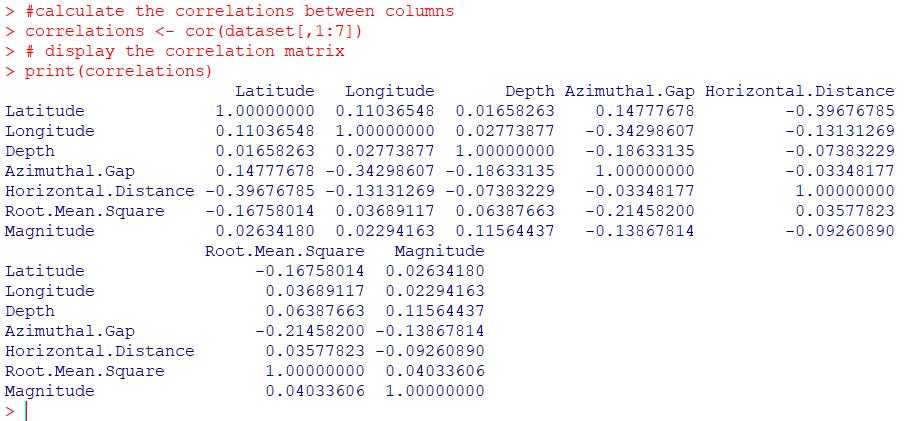


Below are some results from running the functions above:

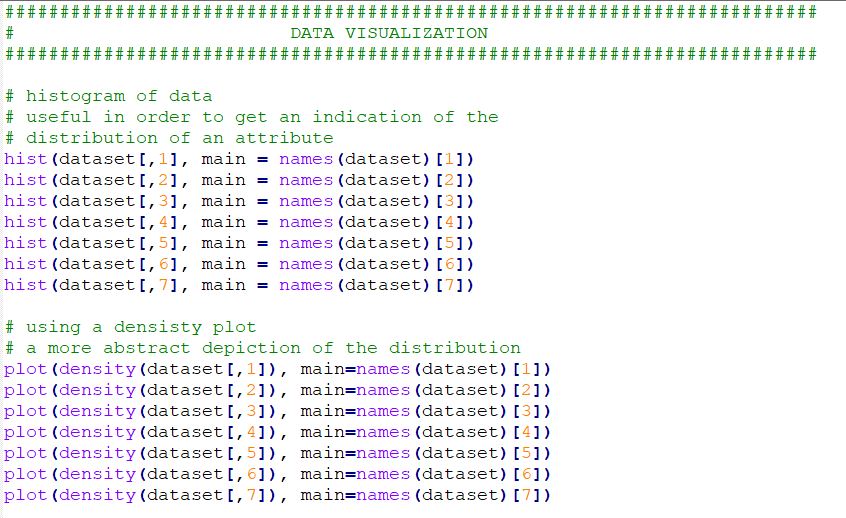




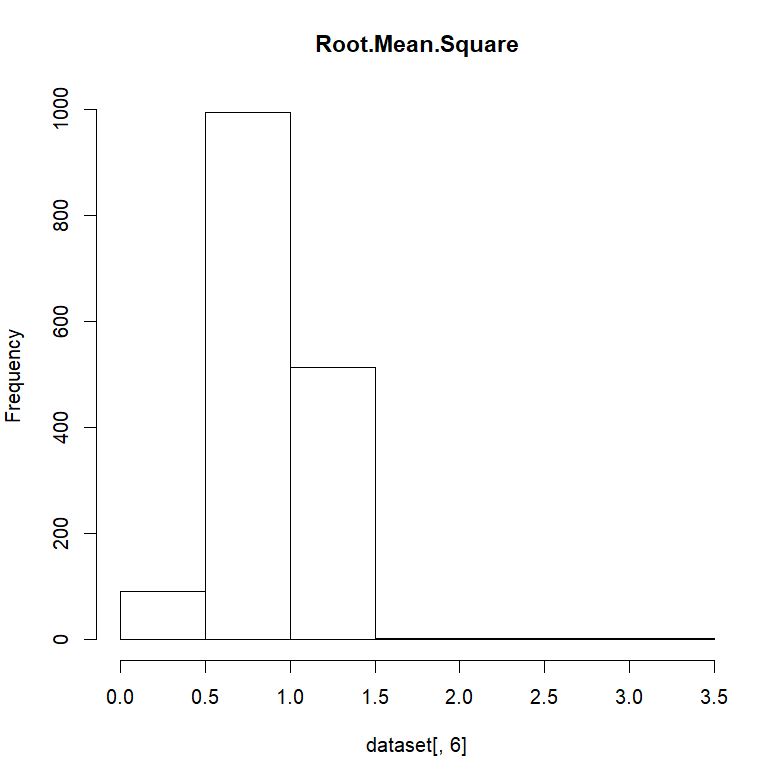


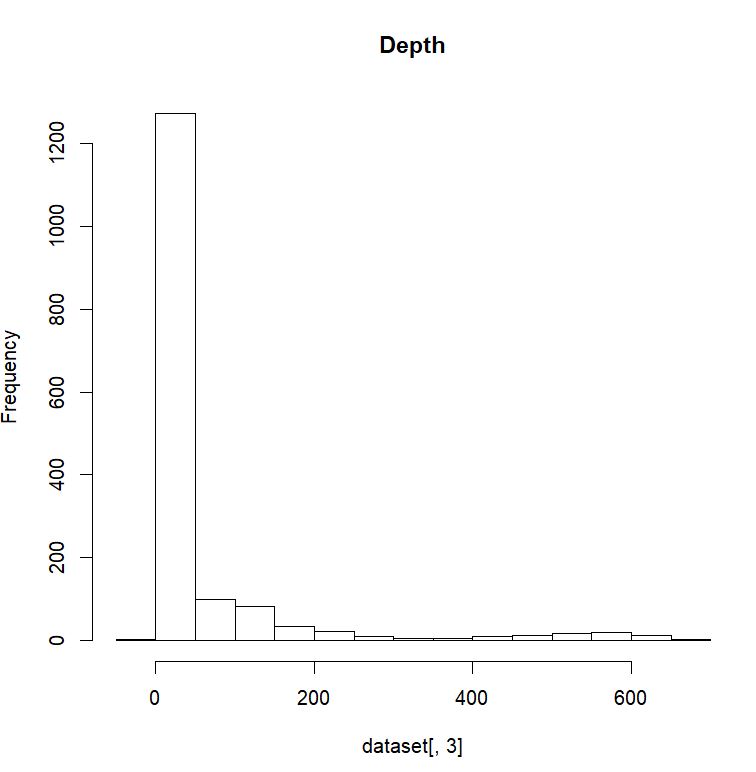


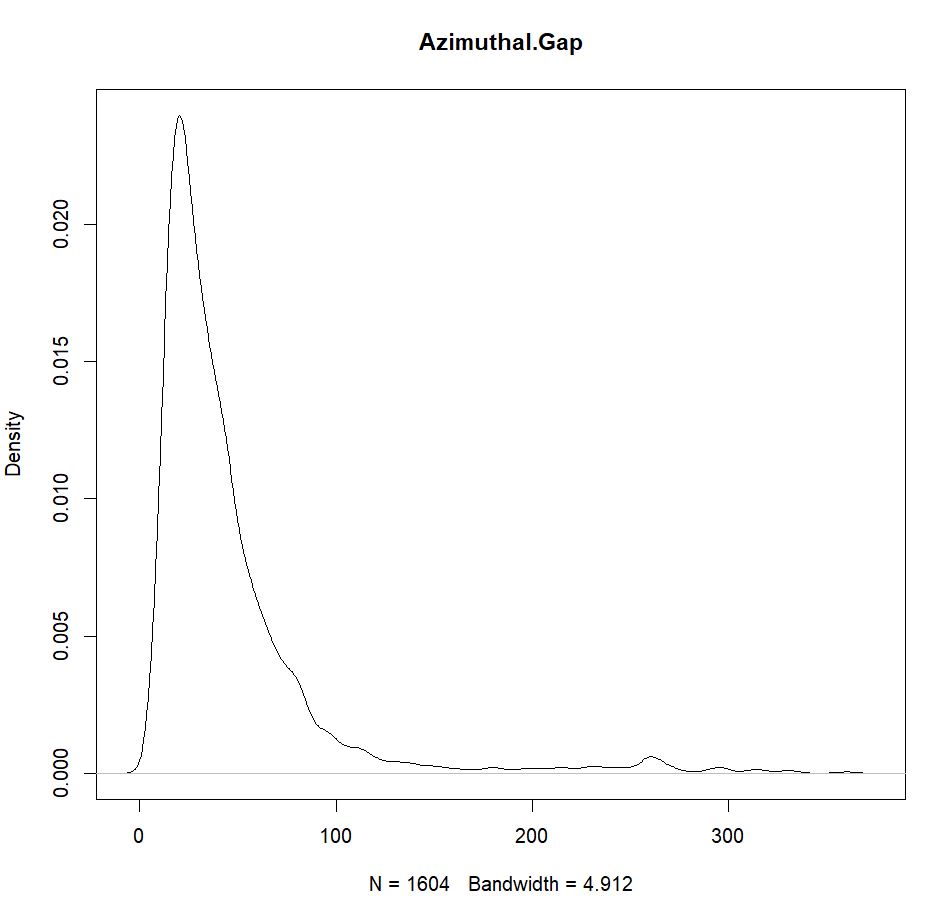
Below are some graphical representations for the data:

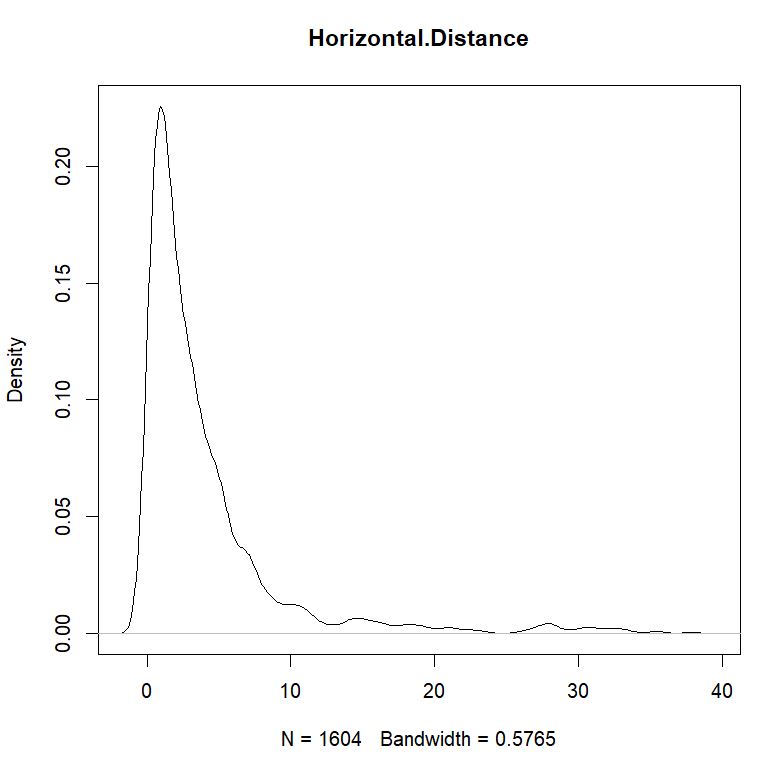


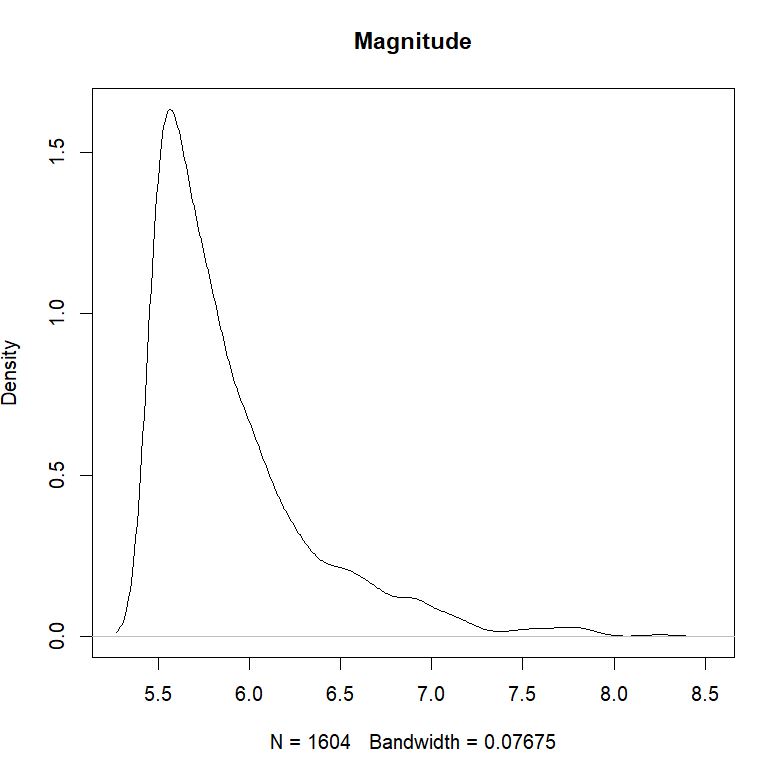


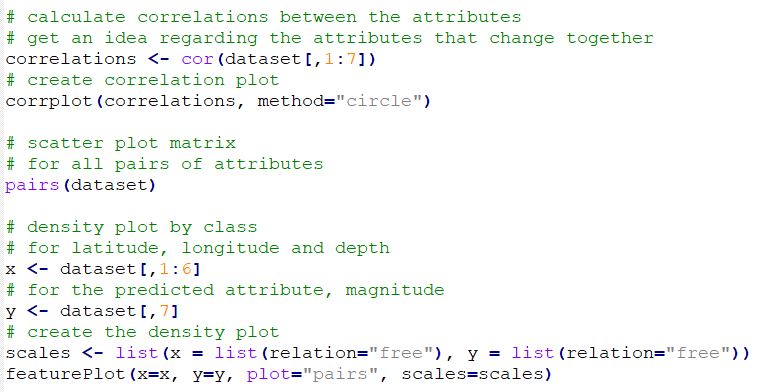


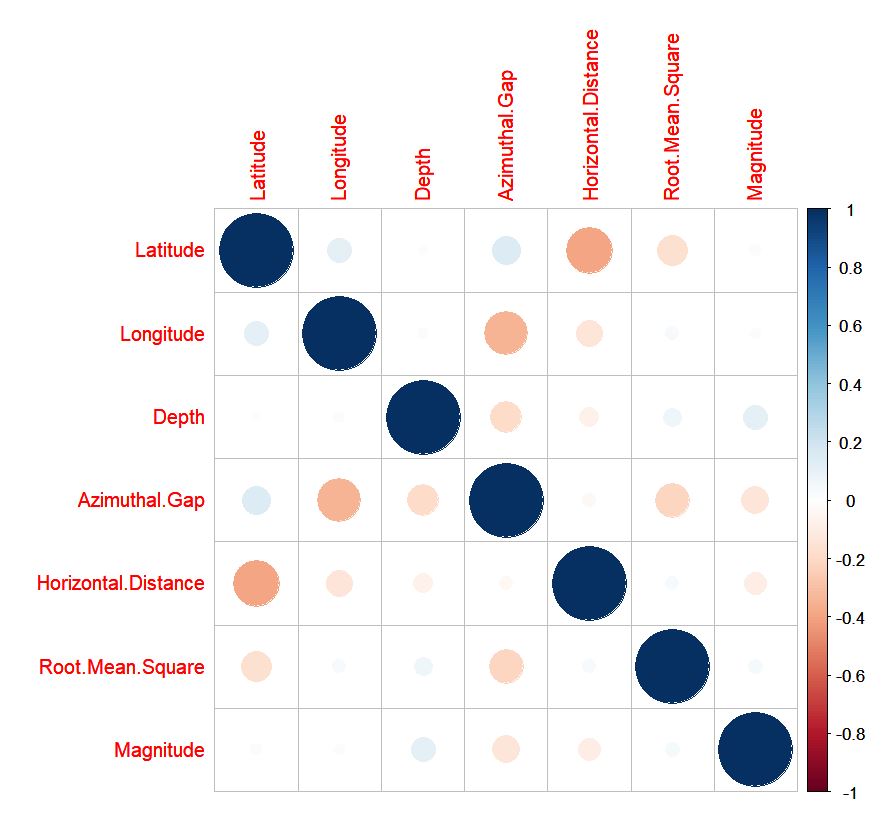


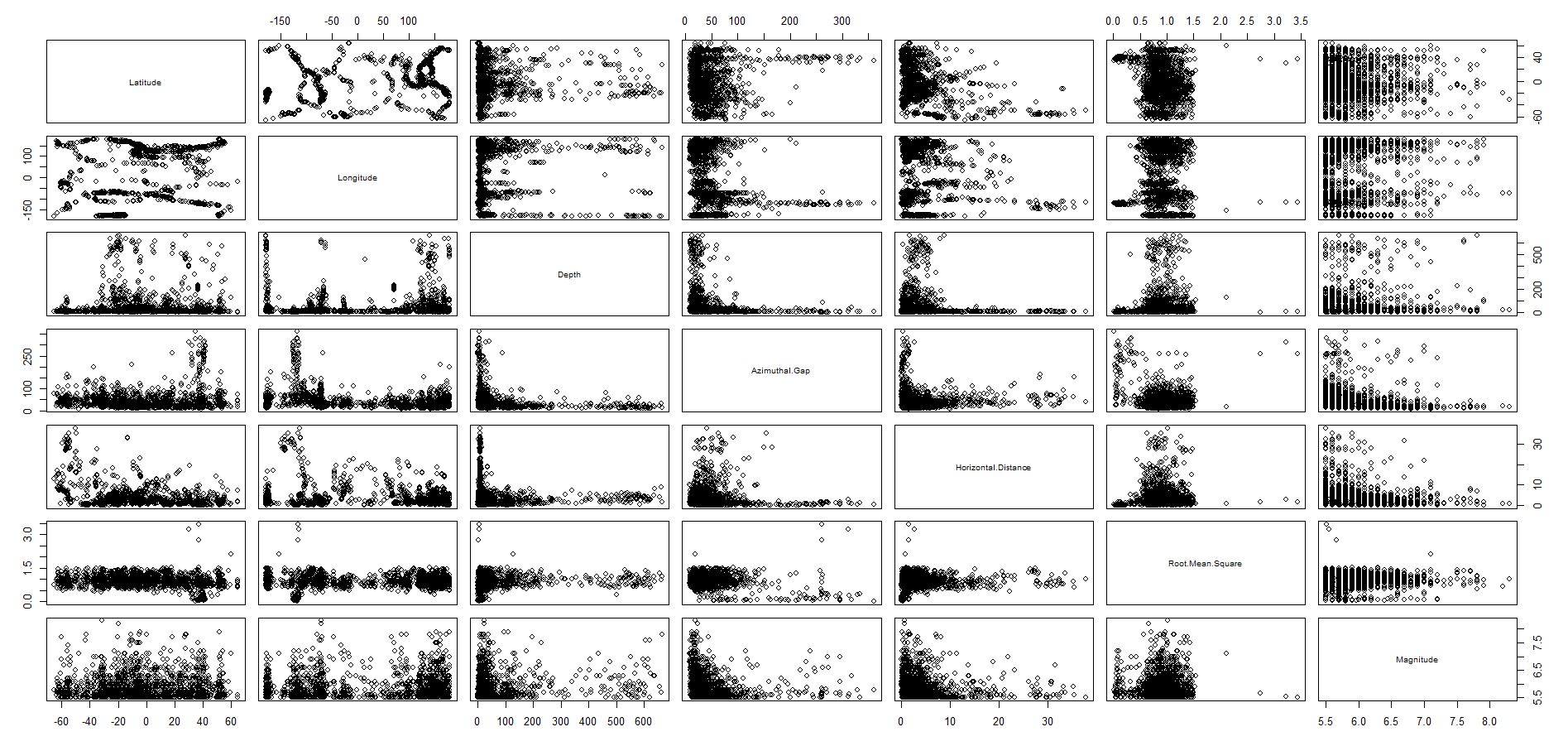


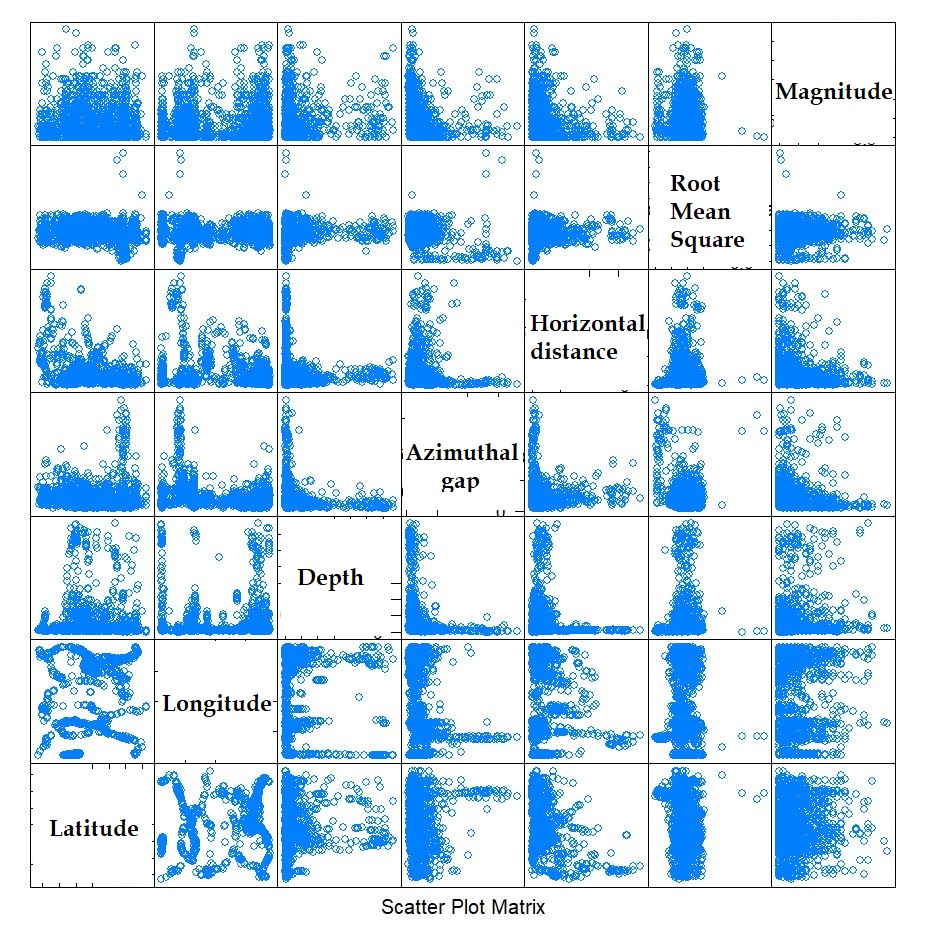








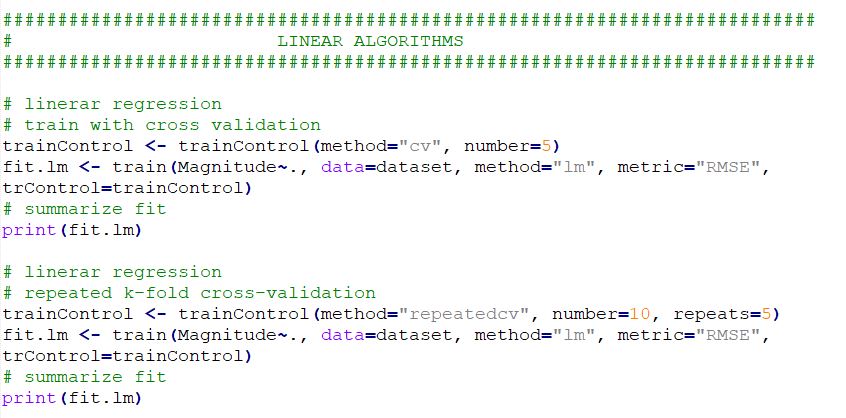


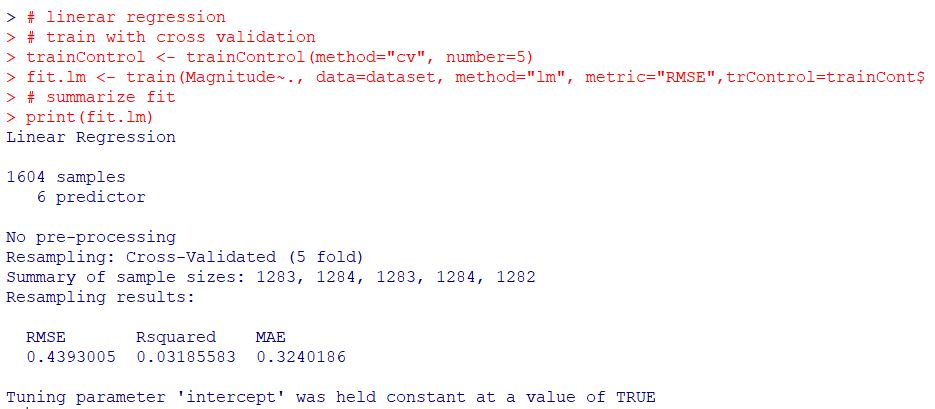


1. Models, machine learning   
   algorithms and training

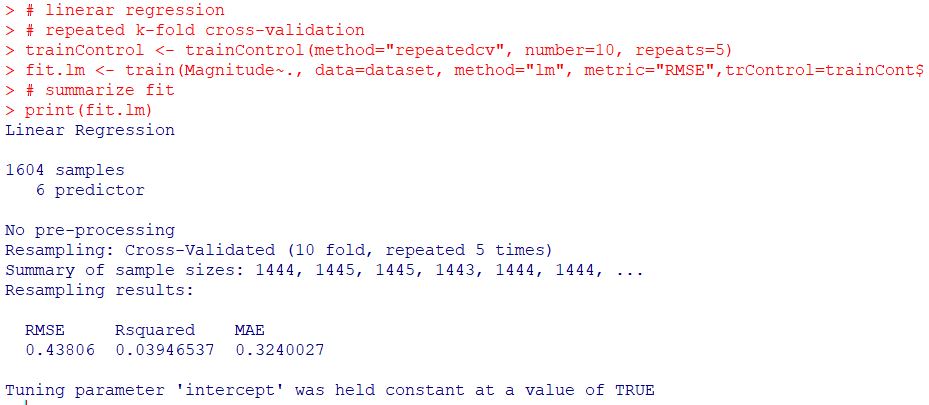
Below I will add print screen with the code and the results after running 2 linear algorithms: linear and regularized regression.

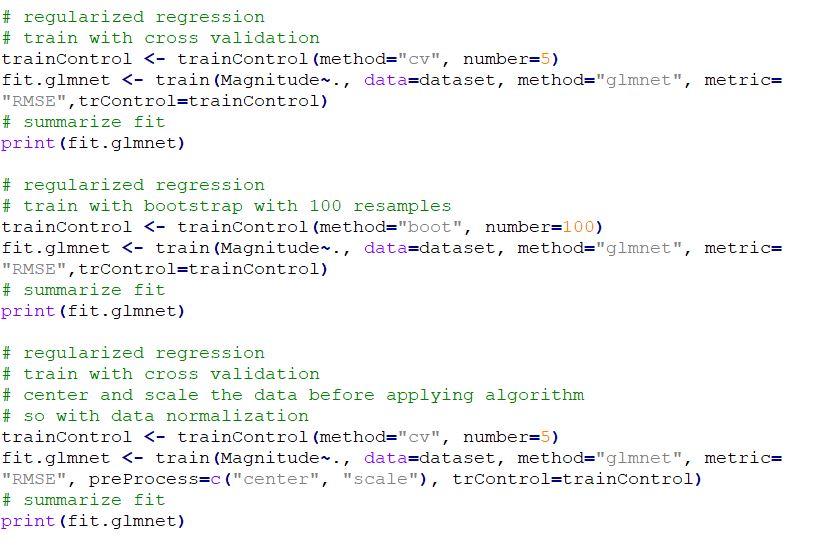
A number of different resampling methods have been used.

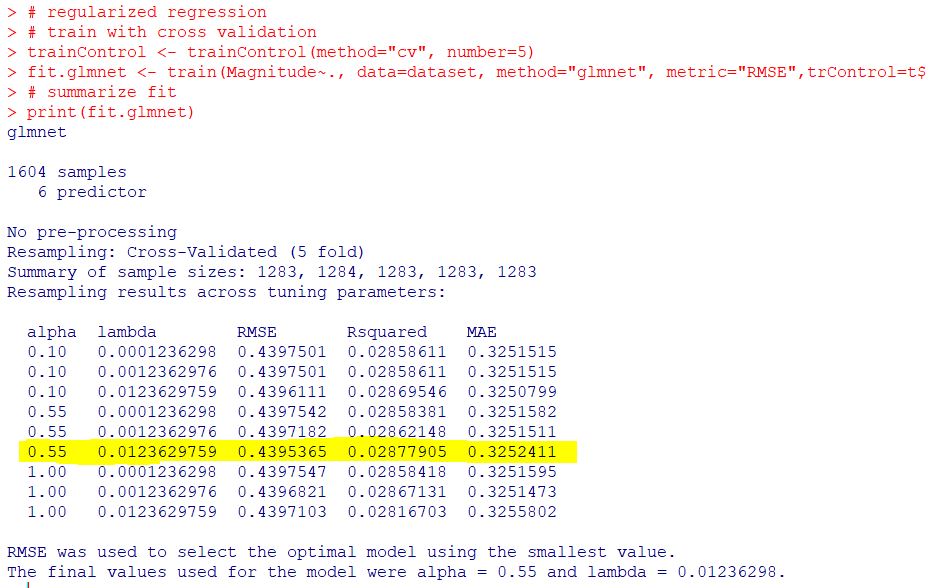


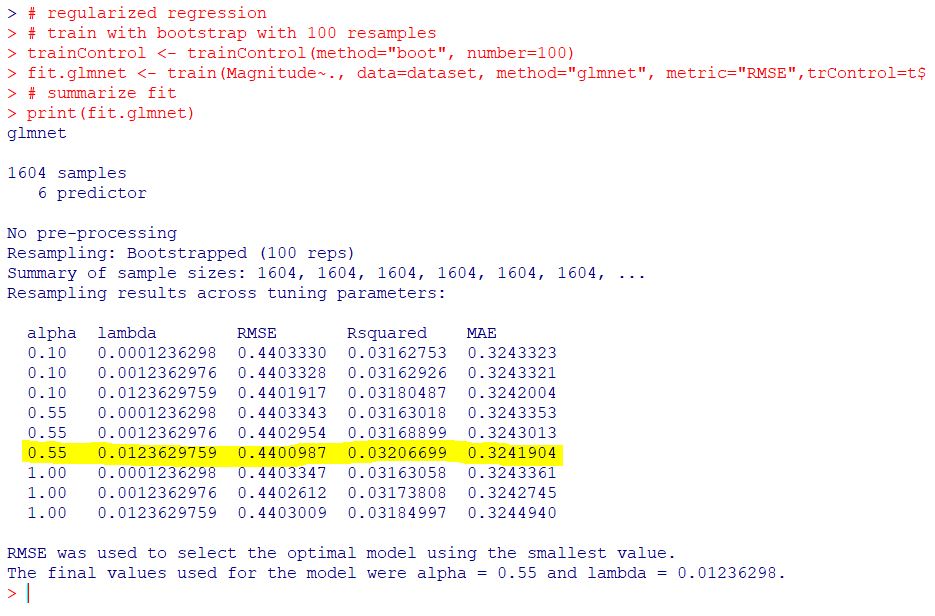


Although the cross validation with repeats (k-fold cross validation) works better with the simple regression model, the results are quite poor by using this linear and simple approach.





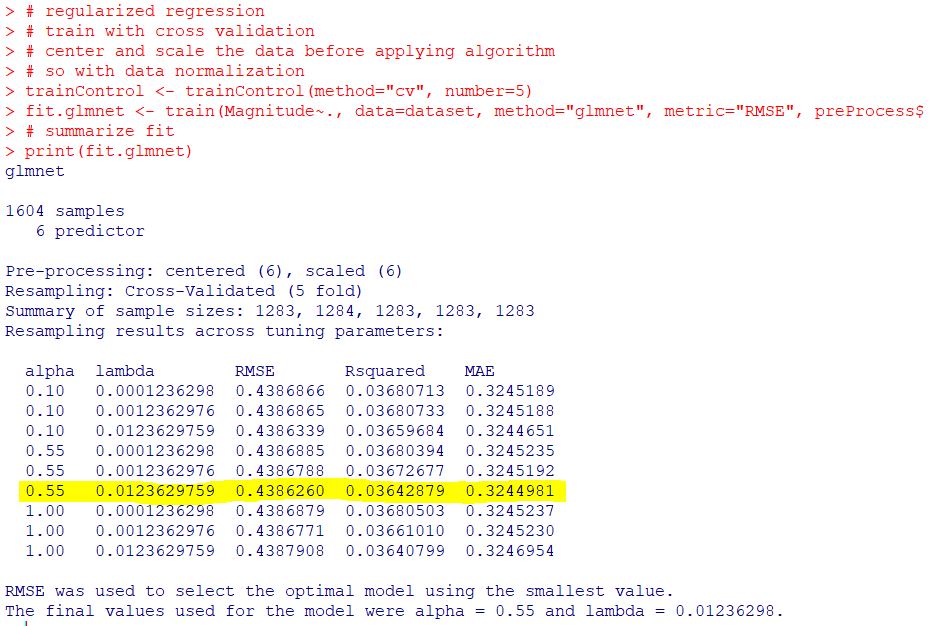


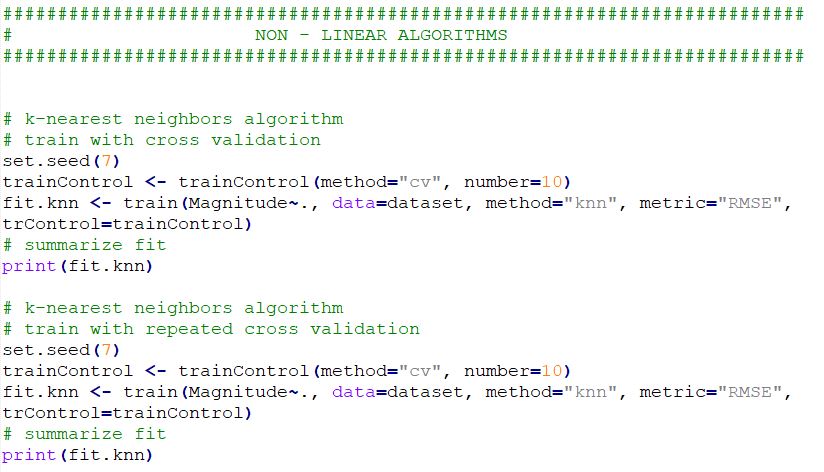


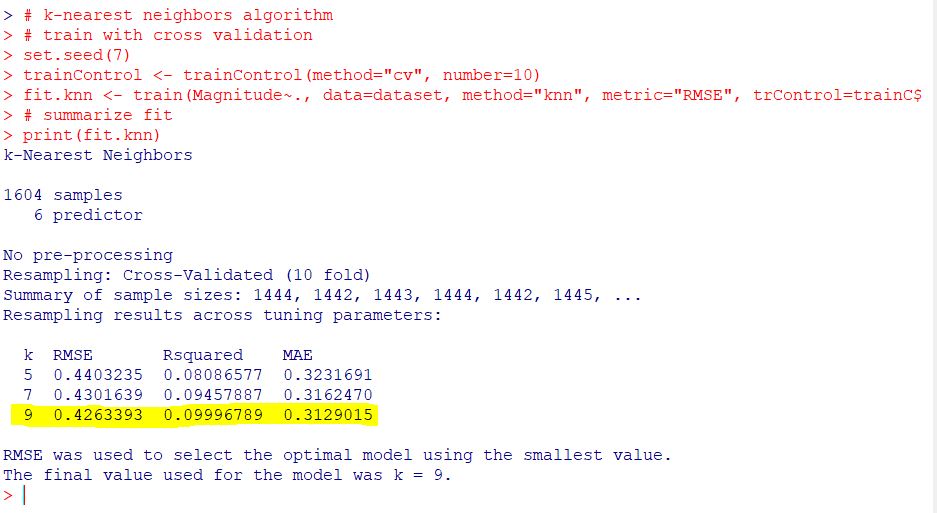
From the above pictures, it seems like the cross validation works better than the bootstrap, at least for regularized regression.

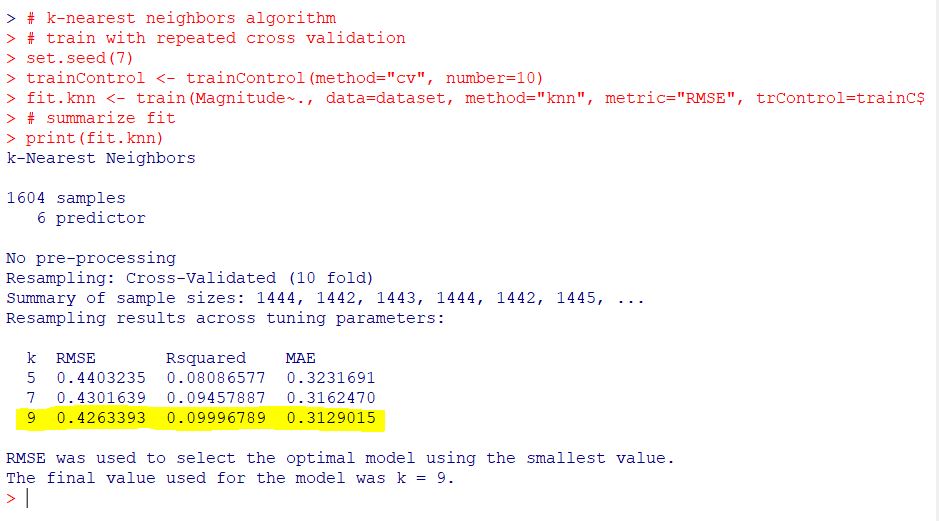
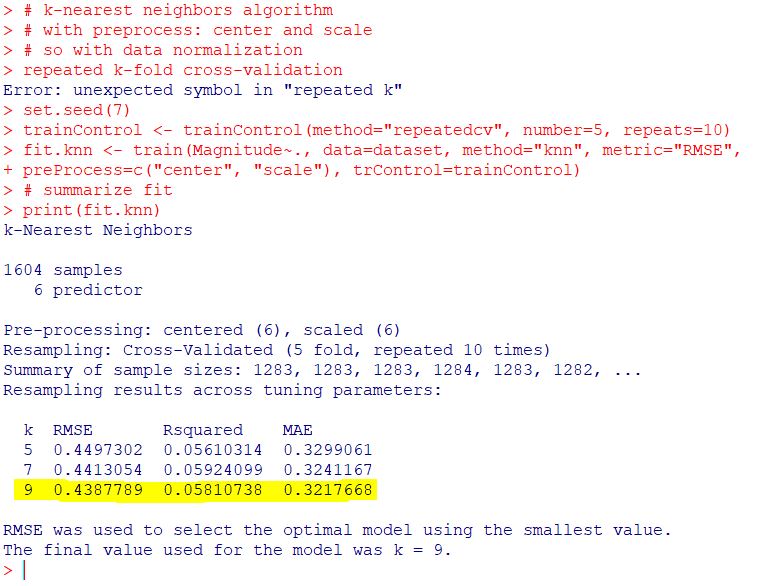
Normalizing the data by centering and scaling does improve the very low values of the R squared, but it increases the RMSE value slightly.

If we compare the linear regression and the regularized one, we can observe similar results, with some smaller values of RMSE in some cases, but with quite smaller R squared. Probably the repeated k-fold validation method applied together with linear regression would be the choice in the case of linear algorithms, but the dataset is way better suited for non-linear algorithms.









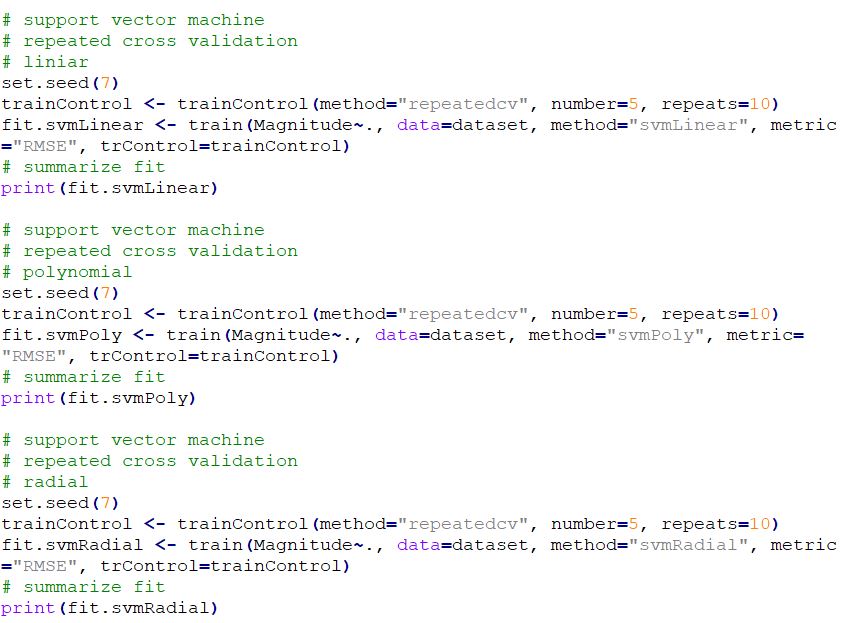
In the above cases we use the third algorithm, a non-linear one: k-nearest neighbor.

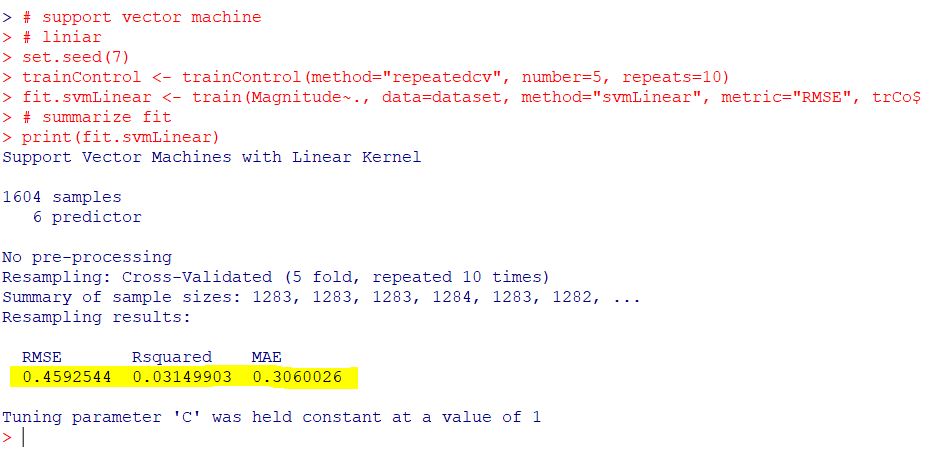
The first impression is that it behaves better for our dataset and when using cross validation and repeated cross validation, the results are very similar.

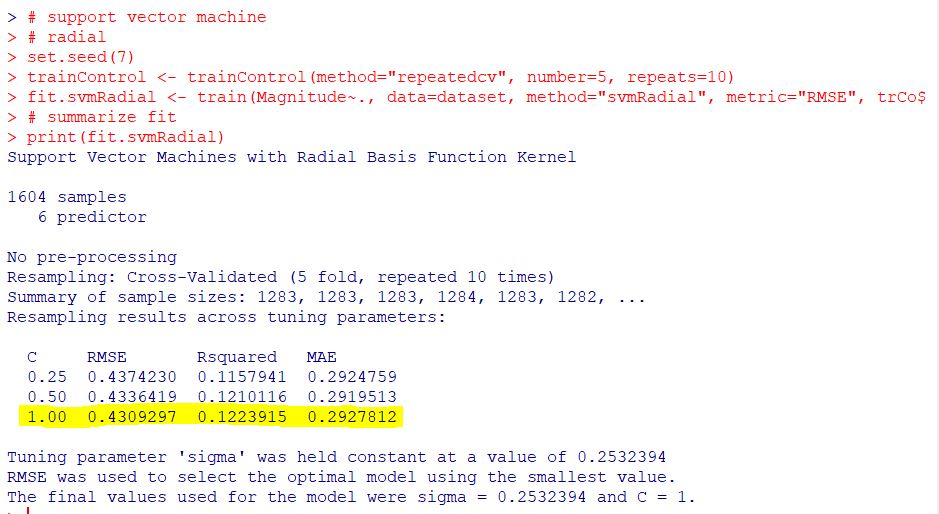
RMSE is slightly lower, but R squared is about 3 times greater than in the case of linear algorithms.

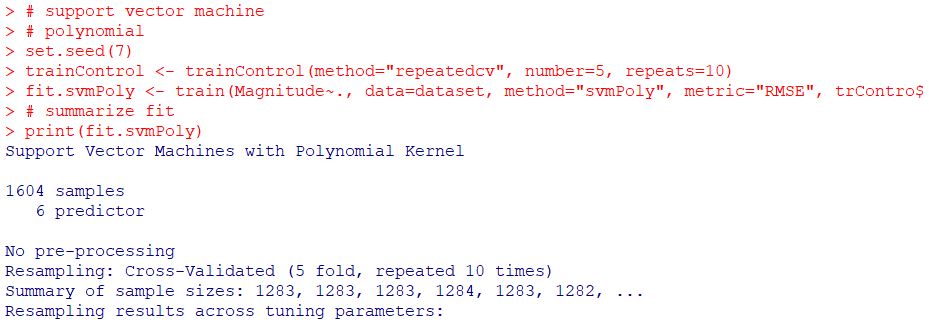
Pre-processing and normalizing data (center and scale) seem to be inappropriate in this context.

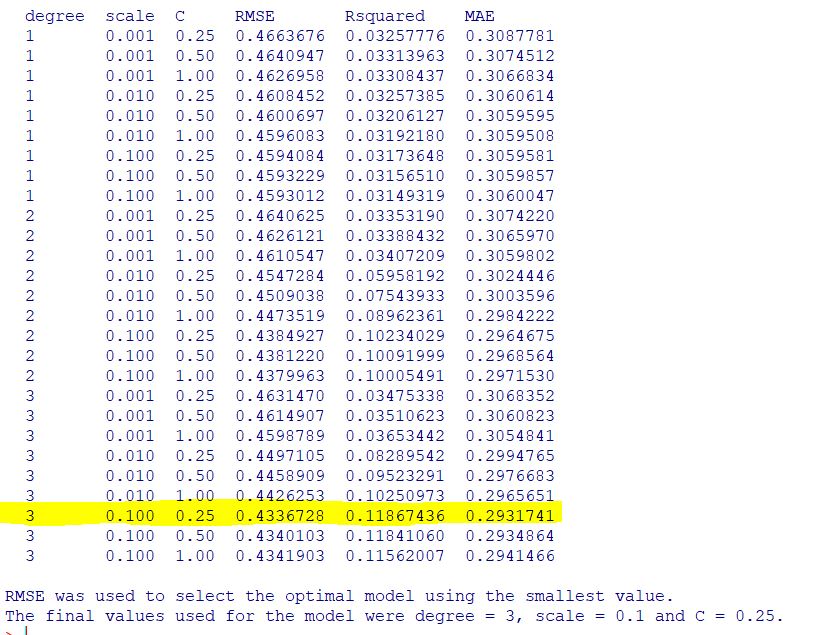
Next we will present the support vector machine versions with repeated cross validation and leave one out cross validation.

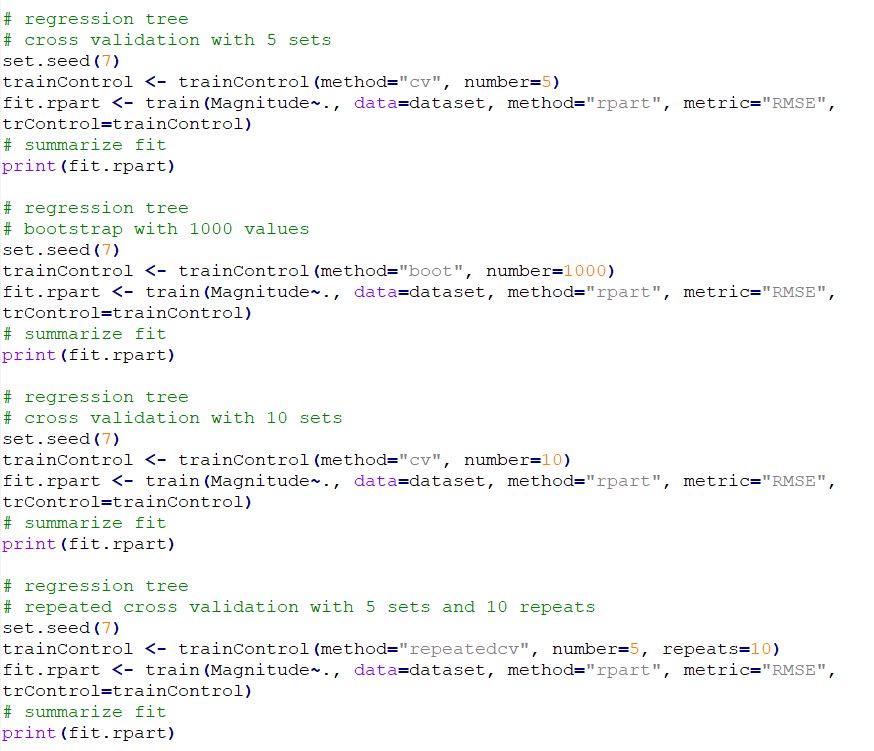


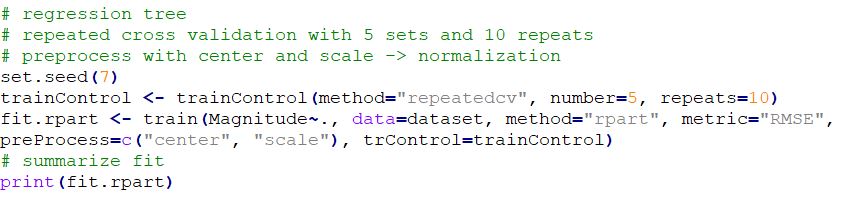










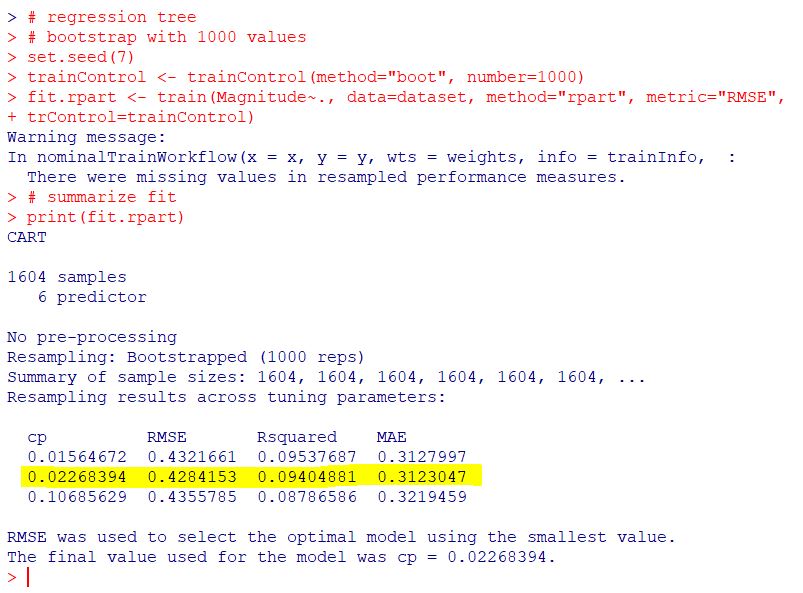


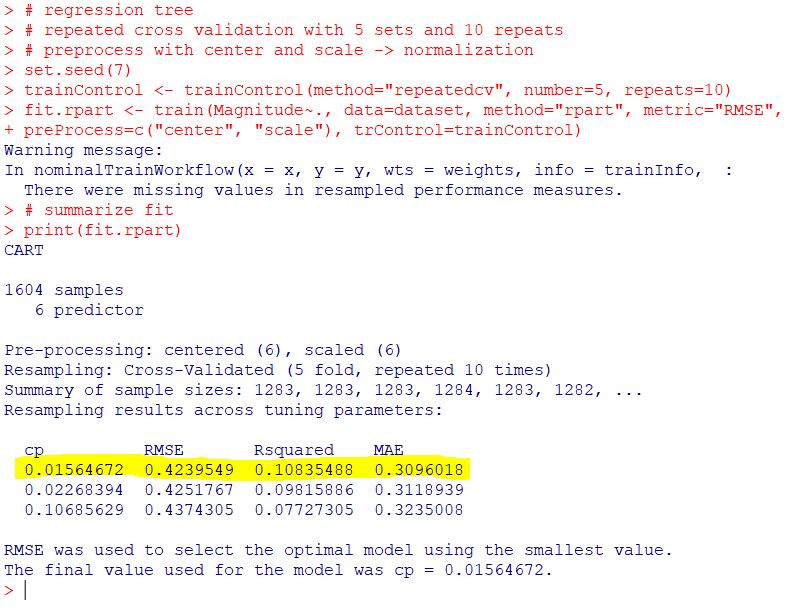
In the case of the regression trees, we obtain results that tend to have a better RMSE (~4.2 vs ~4.3), but the R square averages around 0.09-0.1.

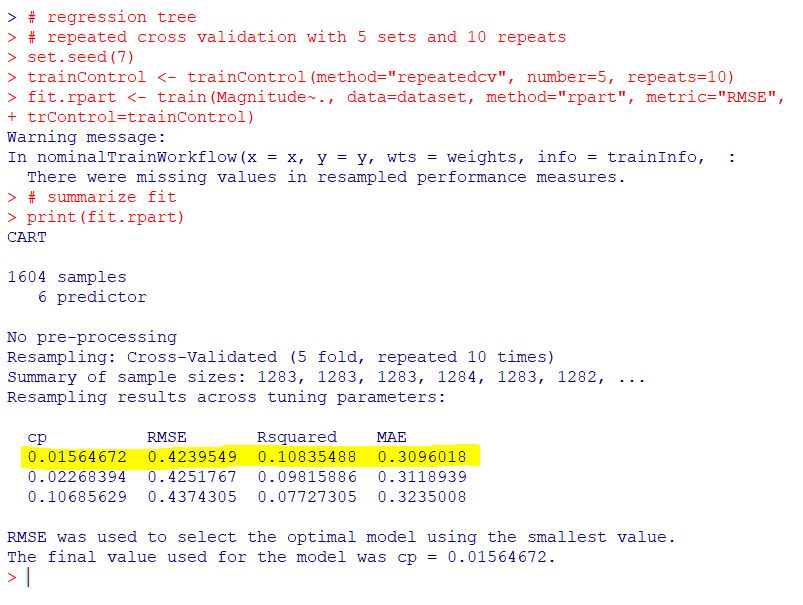
Again, the best version is the one that uses repeated cross validation; a similar result is obtained after centering and scaling the data (after normalization).

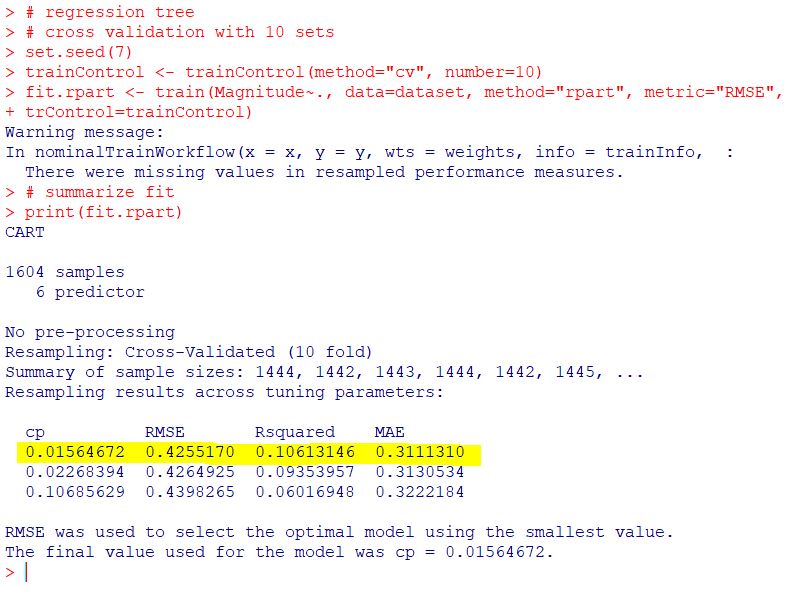
So, in general, the best results are obtained by using the support vector machine in the radial or polynomial version or regression trees.

Regression trees obtain better RMSE, while support vector machine implementations tend to get ahead regarding the R squared metric.









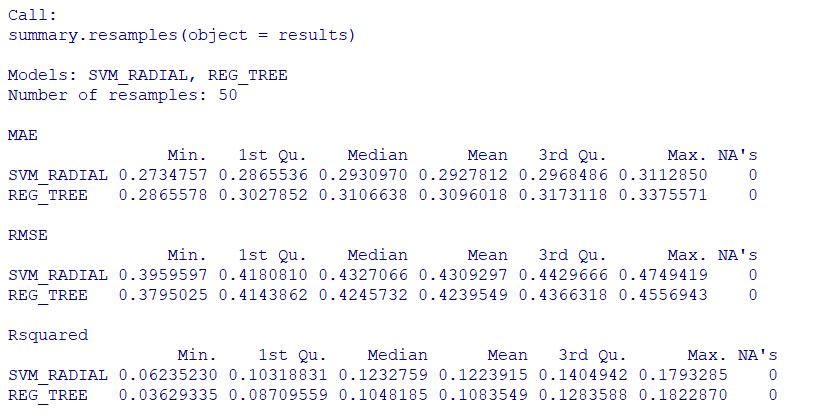
1. Compare best two models

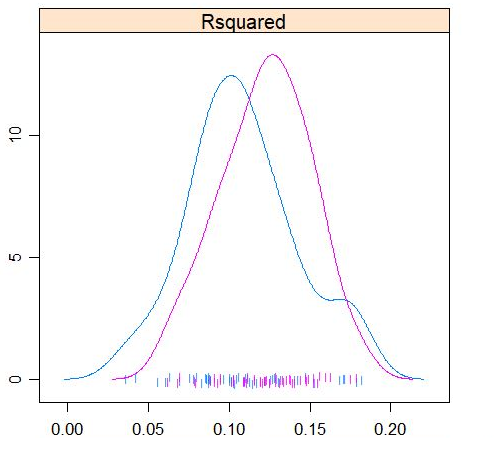


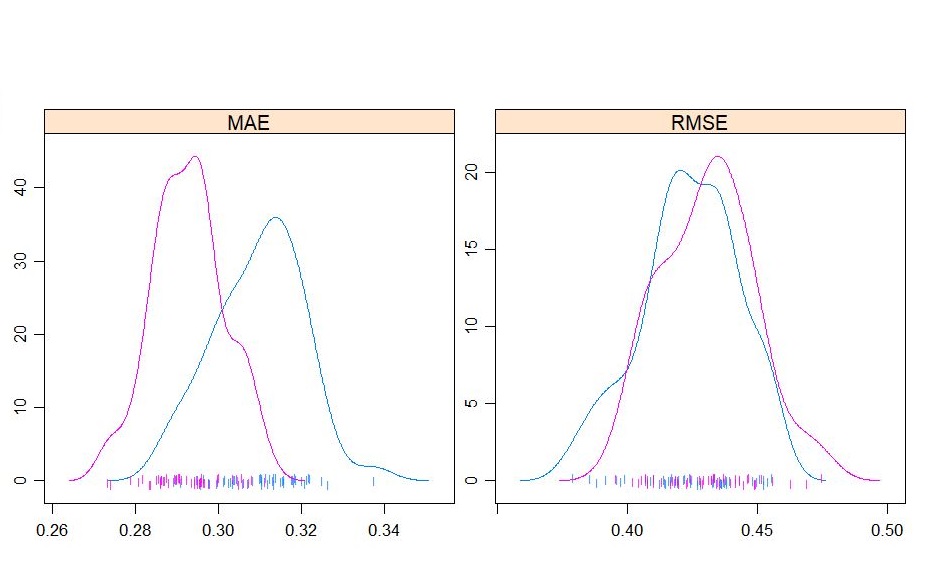
In this section I will compare:

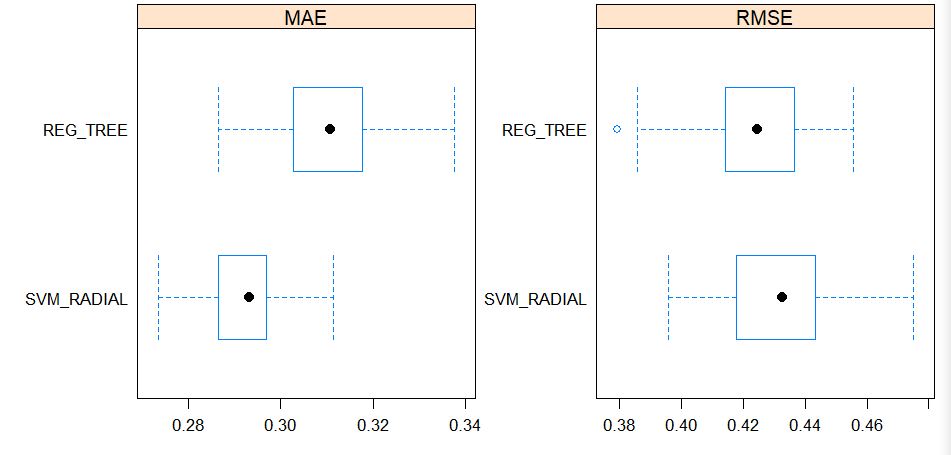
1. Regression tree, repeated cross validation (5 sets, 10 repeats)
2. Support vector machine radial, repeated cross validation (5 sets, 10 repeats)

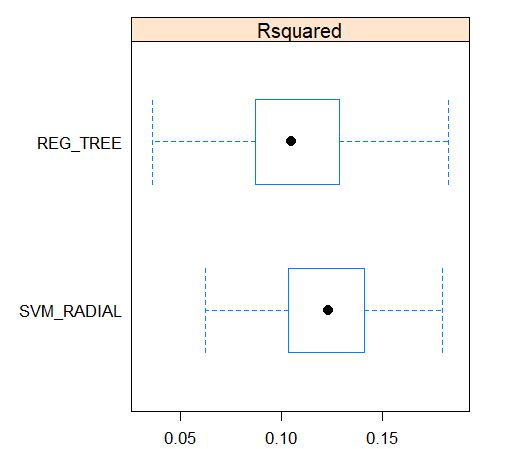
These two were the algorithms trough which I obtained, in the case of the regression tree, the smallest RMSE and in the case of the support vector machine (radial and polynomial) the largest R squared.

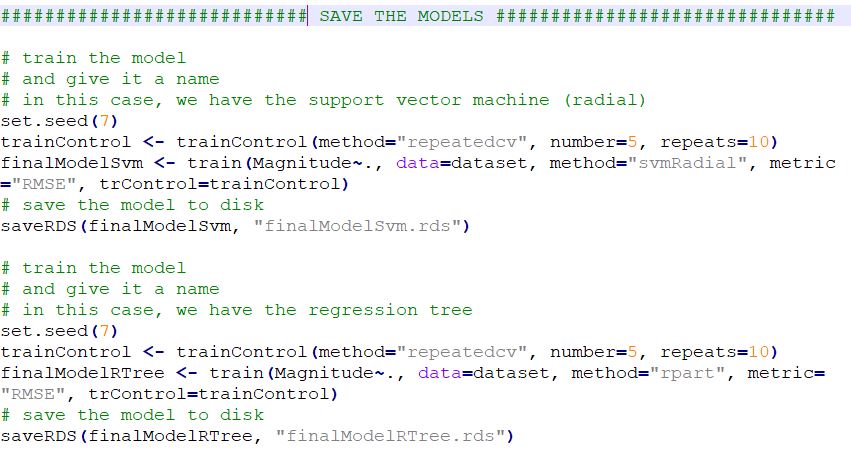










The final step is to save the two selected models for later use.

