An assignment for Generic drugs **Kraków, 18.12.2022**

Wiktoria Bieniek
DDD

Tasks:

1. To find and report top 10 best models for a given dataset.

2. To deploy and provide best model for the reference as either source code or binary object (i.e., Datafile)

3. To provide short written report on the modeling techniques and efforts taken in this exercise.

Eligibility criteria: At least one model with error on the testing dataset: RMSE <= 1.3

Dataset: individual train and test .TXT files. The dataset was introduced during the classes. The dependent variable is "Disintegration Time".

I obtained the set of data "**1**": **test_v1.txt** and **train_v1.txt** (I converted the data to the CSV files). For cubist I remained the txt files.

In my work below, I would like to present the results of my work using mainly: Cubist package of R, Excel, and Scikit-learn library of Python, which is one of the most useful libraries for machine learning, provides a range of supervised and unsupervised learning. Contains a lot of tools, but mostly I focused on regression.

Table 1. Summary of models with errors on the testing dataset.

Number of models	**Tool**	**Method**	**Value of RMSE**	**Condition (RMSE<=1.3)**
1	Python	Decision Tree	🟥 1,64	Bad
2	Python	Random Forest	🟩 1,15	Good
3	Python	Linear Regression	🟨 1,31	Medium
4	Python	Ridge Regression	🟩 1,18	Good
5	Python	Lasso Regression	🟩 1,02	Good
6	Python	ElasticNet Regression	🟩 1,2	Good
7	Python	Support Vector Machine	🟩 0,93	Good
8	Python	K-Nearest Neighbors	🟥 1,73	Bad
9	Python	Regression Multi-Layer Perceptron	🟩 0,42	Good
10	Python	H20	🟩 0,86	Good
11	Excel	Regression	🟥 1,71	Bad
12	R	Cubist	🟩 0,44	Good

Legend: 🟨 medium 🟥 bad 🟩 good

Result: I obtained among 12 models, **8 models** which fulfill the criteria condition **RMSE<=1.3,** (used: **RandomForest, LassoRegression, Support Vector Machine, Regression Multi-Layer Perceptron and H20 of AutoML**) other **1 models** results are quite close to 1.3 but it is not exactly what I should have obtained. The rest **3 models**, even if I tried change parameters and optimize, I got unsatisfactory result.

What metric was important to obtained?

RMSE - Root Mean Squared Error, it is a statistical measure which represent the measures the average of the residuals or error to check the model performance. One way to assess how "good" our models fit a given dataset is to calculate Root Mean Square Error (RMSE), which is a metric that tells us how apart our predicted values are from our observed values. The fact is that, if we have lower value of RMSE correlative and indicative of a "good" model.

Based on our result, I would like to describe method which I used to obtain the results:

Decision Tree

Decision Tree (DTs), supervised learning algorithm, also called as Classification and Regression Trees (CART). Our goal is to build a model which can predict the value of a given target variable by learning simple decision rules, which come from data features [1]. I created a CART, I used to the "DecisionTreeRegressor" class to build algorithm. Most important think is one of the lines to fit our model to appropriate given to us training data set. I used maximal depth, which means that I indicate maximum depth of the tree and minimum number of samples which require to be at a leaf node. We can say that the more value of the maximum depth of the tree, the more complex train we will obtain, it is also high risk to obtain bad accuracy. One the model is creating on the training data set; I was able to make the predictions and evaluation of metrics—RMSE on test data. My result is RMSE 1.64 (max depth = 3, but for max depth = 15 I obtained much higher value of RMSE, so I decided to put only for max depth = 3), to sum up: RMSE does not meet the condition (RMSE<=1.3)

Random Forest

Random forest supervised learning algorithm, called "Forest" – because there is a collection of decision trees. A forest is comprised of trees, we can say that the more trees it has, the more robust a forest is. Random forests create decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting [2]. I tried to select the random samples from a given dataset and the next construction a decision tree for each sample to obtain result of our prediction from each of tree. I tried to compare random forest to decision tree, but the random forest is more difficult to interpret, while a decision tree is easily to evaluate and faster. Decision Trees are useful, but the problem is that they often tend to overfit the training data. The first I instantiate the Random Forest Regression specifying "n_estimators" value, which indicates the exact number of trees in the forest. I fitted the model to the training data set and evaluate RMSE parameter. The above output shows that the RMSE = 1.15, means that meet the condition (RMSE<=1.3)

The next I tried to implement the following linear regression models.

Usually, we are use characteristics steps to evaluate for example regression models:

1. Import the necessary modules and required libraries.
2. Loading the data and performing if it necessary to train and test data or just basic check your data.
3. Create the arrays for you features and corresponding variables which are respond to your features.
4. Build, Predict and Evaluate the model.

Linear Regression

Linear regression is one of the most probably popular simplest regression technique, due to easiest way to interpreting results. We try to implement the dependent variable (y) based on the set of independent variables (x)[3]. We assume that the predictors have a linear correlation with our target variable and predictors do not correlate to each other. Based on this information and our following step, I build the model on the training set and made the predictions. The output shows that the RMSE = 1.31 (almost within the range RMSE<=1.3)

Ridge Regression

Going further using the regression models, I tried to used Ridge regression, which is kind of the extension of the linear regression, in most cases uses when we have a multicollinearity in multiple regression data. When we have a set of data which contains a higher value of the predictor variables than the value of observations [3]. I constructed the regression model based on Ridge class, fitted the model to training data and evaluated the metrics RMSE, which is equal to 1.18.

Lasso Regression

Another modification of linear regression is Lasso regression (Least Absolute Shrinkage and Selection Operator), I used to this regression as a for the selection of the subset of variables. It provides greater prediction accuracy as compared to other regression models. Lasso help us to increase model interpretation. [4] I constructed lasso regression model by using the Lasso class. I was repeated the same steps as previous and calculate the RMSE = 1.02

ElasticNet Regression

ElasticNet combines the properties of both Ridge and Lasso regression, and it is a last regression model which is a combination and extension of linear regression. I repeated the same steps and previous, output showed me that the RMSE = 1.2

Support Vector Machine

Support Vector Machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis [5]. Given a set of training examples, each marked as belonging to one of the categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other.

K-Nearest Neighbors

K-Nearest Neighbors (KNN) - Supervised neighbors-based learning comes in two flavors: classification for data with discrete labels, and regression for data with continuous labels. KNN classifier returns the mode of the nearest K neighbors, the KNN regressor returns the mean of the nearest K neighbors. [6] I used in my model regression for data with continuous data labels. The label assigned to a query point is computed based on the mean of the labels of its nearest neighbors. As we increase the number of neighbors, the model starts to generalize well, but increasing the value too much would again drop the performance.[7]

Multi-Layer Perceptron

MLP Regressor implements a multi-layer perceptron (MLP) that trains using backpropagation with no activation function in the output layer. MLP or multi-layer perceptron is an artificial neural network (ANN), which consists of a minimum of three layers: an input layer, one or more hidden layers and an output layer.

AutoML H20

H20 AutoML it is a fully automated, supervised learning and highly scalable, algorithm which automates the process of training a large selection of candidate models, result of the AutoML run is a "leaderboard": a ranked list of models, which can be ranked by numerous model performance metrics. In general, the AutoML algorithm is based on training H20 machine learning algorithms to obtain many models in a short period of time [8].

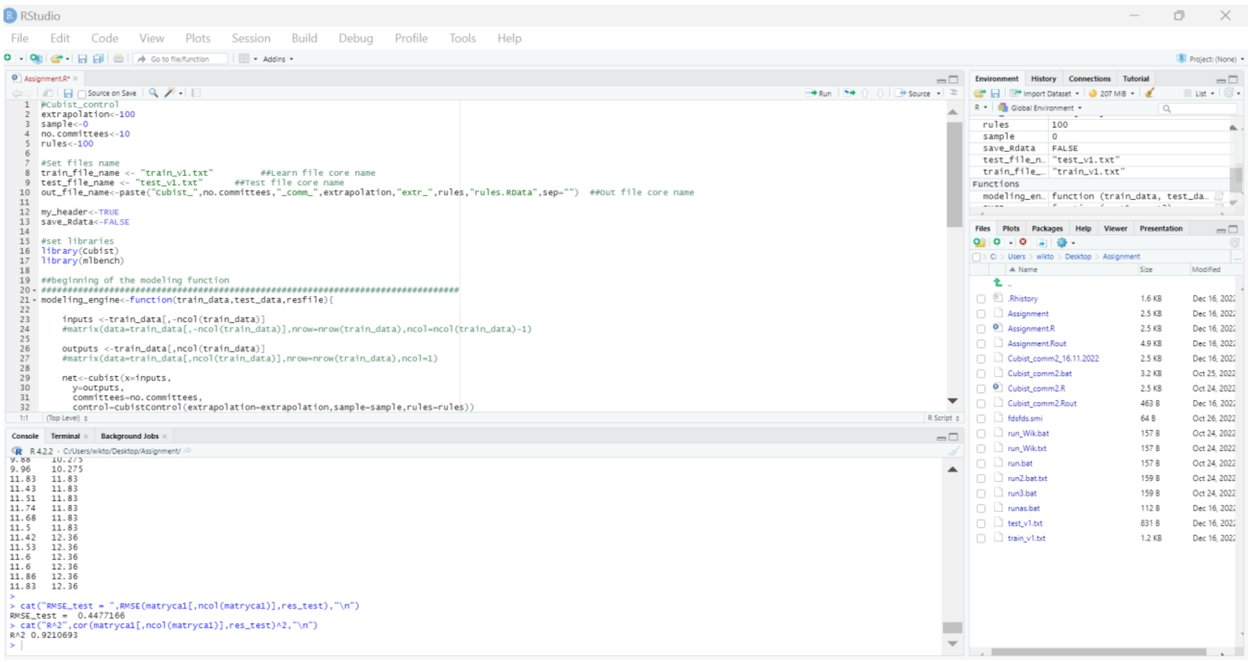
Excel

Mainly to understand the relationship between a predictor variable (x) and respond variable (y).

Based on that I tried to conduct regression analysis, I ended up a model which tell me about the predicted value for the response variable based on the value of the predictor variable. I followed running regression analysis using Data Analysis "ToolPak" plugin and Regression option/

Cubist

Cubist is a rule-based model that is an extension of Quinlan's M5 model tree. A tree is grown where the terminal leaves contain linear regression models.[9]



```r
#Cubist_control
extrapolation<-100
sample<-0
no.committees<-10
rules<-100

#Set files name
train_file_name <- "train_v1.txt"         ##Learn file core name
test_file_name <- "test_v1.txt"      ##Test file core name
out_file_name<-paste("Cubist_",no.committees,"_comm_",extrapolation,"extr_",rules,"rules.RData",sep="")  ##Out file core name

my_header<-TRUE
save_Rdata<-FALSE

#set libraries
library(Cubist)
library(mlbench)

##beginning of the modeling function
##############################################################################
modeling_engine<-function(train_data,test_data,resfile){

    inputs <-train_data[,-ncol(train_data)]
    #matrix(data=train_data[,-ncol(train_data)],nrow=nrow(train_data),ncol=ncol(train_data)-1)

    outputs <-train_data[,ncol(train_data)]
    #matrix(data=train_data[,ncol(train_data)],nrow=nrow(train_data),ncol=1)

    net<-cubist(x=inputs,
      y=outputs,
      committees=no.committees,
      control=cubistControl(extrapolation=extrapolation,sample=sample,rules=rules))
```

```
9.88    10.275
9.96    10.275
11.83   11.83
11.43   11.83
11.51   11.83
11.74   11.83
11.68   11.83
11.5    11.83
11.42   12.36
11.53   12.36
11.6    12.36
11.6    12.36
11.86   12.36
11.83   12.36
>
> cat("RMSE_test = ",RMSE(matryca1[,ncol(matryca1)],res_test),"\n")
RMSE_test =  0.4477166
> cat("R^2",cor(matryca1[,ncol(matryca1)],res_test)^2,"\n")
R^2 0.9210693
> 
```

rules	100
sample	0
save_Rdata	FALSE
test_file_n…	"test_v1.txt"
train_file_…	"train_v1.txt"
Functions	
modeling_en…	function (train_data, test_da…

C: > Users > wikto > Desktop > Assignment

Name	Size	Modified
..		
.Rhistory	1.6 KB	Dec 16, 202
Assignment	2.5 KB	Dec 16, 202
Assignment.R	2.5 KB	Dec 16, 202
Assignment.Rout	4.9 KB	Dec 16, 202
Cubist_comm2_16.11.2022	2.5 KB	Dec 16, 202
Cubist_comm2.bat	3.2 KB	Oct 25, 202
Cubist_comm2.R	2.5 KB	Oct 24, 202
Cubist_comm2.Rout	463 B	Dec 16, 202
fdsfds.smi	64 B	Oct 26, 202
run_Wik.bat	157 B	Oct 24, 202
run_Wik.txt	157 B	Oct 24, 202
run.bat	157 B	Oct 24, 202
run2.bat.txt	159 B	Oct 24, 202
run3.bat	159 B	Oct 24, 202
runas.bat	112 B	Dec 16, 202
test_v1.txt	831 B	Dec 16, 202
train_v1.txt	1.2 KB	Dec 16, 202

Summary: To better prediction of model, parameter RMSE it is good use feature importance and reduction of input vector to know what exactly input data (x) influence significant on output data (y). All models allow me compare RMSE parameter between of them.

Bibliography:

[1] https://scikit-learn.org/stable/modules/tree.html

[2] https://www.datacamp.com/tutorial/random-forests-classifier-python

[3] https://www.pluralsight.com/guides/linear-lasso-ridge-regression-scikit-learn

[4] https://www.jigsawacademy.com/blogs/ai-ml/lasso-regression

[5]https://towardsdatascience.com/unlocking-the-true-power-of-support-vector-regression-847fd123a4a0

[6] https://towardsdatascience.com/k-nearest-neighbors-94395f445221

[7] https://scikit-learn.org/stable/modules/neighbors.html

[8]https://www.semanticscholar.org/paper/H2O-AutoML%3A-Scalable-Automatic-Machine-Learning-LeDell/22cba8f244258e0bba7ff4bb70c4e5b5ac3e2382

[9] https://rpubs.com/hwborchers/701464