Notes

# Basic Model

* **Common assumptions** 
  + **Open Date** used to estimate number of year from open date and to 2014 (guessed target year)
  + **City, City Group, Type** treated as **categorical variables** encoded but treated as numeric and not as factors
  + **P1-P37** treated as numeric variables
* **6-fold + bootstrap resampling**
* **Applied caret’s *nearZeroVar* function** 
  + Performed as **1,855,**273.26 on **leaderboard**
  + **Winner** model is **Enet\_Reg** (cross validation RMSE = **2,361**,006)
  + **Second** model is **RandomForest\_Reg** (cross validation RMSE = **2,362**,509)
  + **Third** model is **SVM\_Reg** (cross validation RMSE = **2,370**,142)
  + **Fourth** model is **PLS\_Reg** (cross validation RMSE = **2,371**,511)
  + **Fifth model** is **RobustLinearReg** (cross validation RMSE = **2,395**,484)
* **Removed only zero variance predictors[[1]](#footnote-1)**
  + Performed as **1,798,733.**58607on **leaderboard** (**improving 56,539.67394** the best performance of the winner model applying caret’s nearZeroVar function)
  + **Winner** model is **BaggedTree\_Reg** (cross validation RMSE = **2,361,869**)
  + **Second** model is **RandomForest\_Reg** (cross validation RMSE = **2,364,129**)
  + **Third** model is **Average** (cross validation RMSE = **2,450,247**)

Notice that **BaggedTree\_Reg** and **RandomForest\_Reg** don’t require feature scaling. Moreover it should be interesting **leaving all predictors** **un-scaled** except zero variance ones:

* **Winner** model is **BaggedTree\_Reg** (cross validation RMSE = **2,321,782**); performed as **1,907,954** on **leaderboard**
  + **Learnt that for BaggedTree\_Reg** **feature scaling is important**
* **re-do w/** removePredictorsMakingIllConditionedSquareMatrix = F, removeHighCorrelatedPredictors = F, featureScaling = T; bagged tree submission (mySub\_bagged\_tree.csv) performed on leaderboard as **1,865,466.33115; that should mean that using removePredictorsMakingIllConditionedSquareMatrix = T, removeHighCorrelatedPredictors = T is good thing ever!**
* **Second** model is **RandomForest\_Reg** (cross validation RMSE = **2,322,125**); performed as **1,789,447.28778** on **leaderboard**

# Basic Model improvement #1: boosting like & boosting

We can try to create a new feature as the prediction of the winner model both on the test set and the training set (6-folds-like procedure) and using the second model to predict on test set

* First “**tip**” feature has been build on **Enet\_Reg** applying caret’s *nearZeroVar* function both in test set and training set (6-folds-like procedure) while the predicting model is a **BaggedTree\_Reg** removing only zero variance predictors ( removePredictorsMakingIllConditionedSquareMatrix = F, removeHighCorrelatedPredictors = F, featureScaling = F)
* performed as **1,839**,573 on **leaderboard**
* **re-do w/** removePredictorsMakingIllConditionedSquareMatrix = T, removeHighCorrelatedPredictors = T, featureScaling = T; (mySub\_tip\_1.csv) performed on leaderboard as **1,865,326.24078**

I fatti dimostrano che, se non ci sono bugs, e’ piu’ efficace la tecnica di boosting invece che quella di tipping (**1,839**,573 vs **1,795,587 : +2.5%)**

**The traditional Gradient Boosting for Regression**

1. Select tree depth, D, and number of iterations, K
2. Compute the average response, y, and use this as the initial predicted value for each sample
3. for k = 1 to K do
4. **Compute the** **residual**, the difference between the observed value and the current predicted value, for each sample
5. **Fit a regression tree of depth, D, using the residuals as the response**
6. Predict each sample using the regression tree fit in the previous step
7. Update the predicted value of each sample by adding the previous iteration’s predicted value to the predicted value generated in the previous step
8. end

Following this approach

* first we compute the response both in training set (6-folds-like procedure) and test set w/ **BaggedTree\_Reg** removing only zero variance predictors (removeOnlyZeroVariacePredictors=T, removePredictorsMakingIllConditionedSquareMatrix = F, removeHighCorrelatedPredictors = F, featureScaling = F)
* we compute the residual, i.e. the difference between the observed value and such prediction in train set;
* we fit an **Enet\_Reg** applying caret’s *nearZeroVar* function using the residuals as response;
* we predict the test set using the **Enet\_Reg;**
* we update the predicted value adding to the previous prediction (**BaggedTree\_Reg**) to the predicted value of residuals
* performed as **1,858**,328 on **leaderboard**
* **re-do w/ removeOnlyZeroVariacePredictors=T,** removePredictorsMakingIllConditionedSquareMatrix = T, removeHighCorrelatedPredictors = T, featureScaling = T; (mySub\_boost\_1.csv) performed on leaderboard as **1,755,088.64112**
* **re=do w/ RandomForest\_Reg instead of BaggedTree\_Reg:** performed on leaderboard as **1,788,992.76965**
* **re-do fitting also residuals residuals (y:RandomForest\_Reg,res: Enet\_Reg, res2:** **BaggedTree\_Reg):** performed as **1,795,587**.03120 on leaderboard
* **re-do fitting also residuals residuals** 
  + **y: BaggedTree\_Reg,res: Enet\_Reg, res2:** **RandomForest\_Reg (applying caret nearZeroVar function)**
  + performed as **1,861,892.03046**on leaderboard

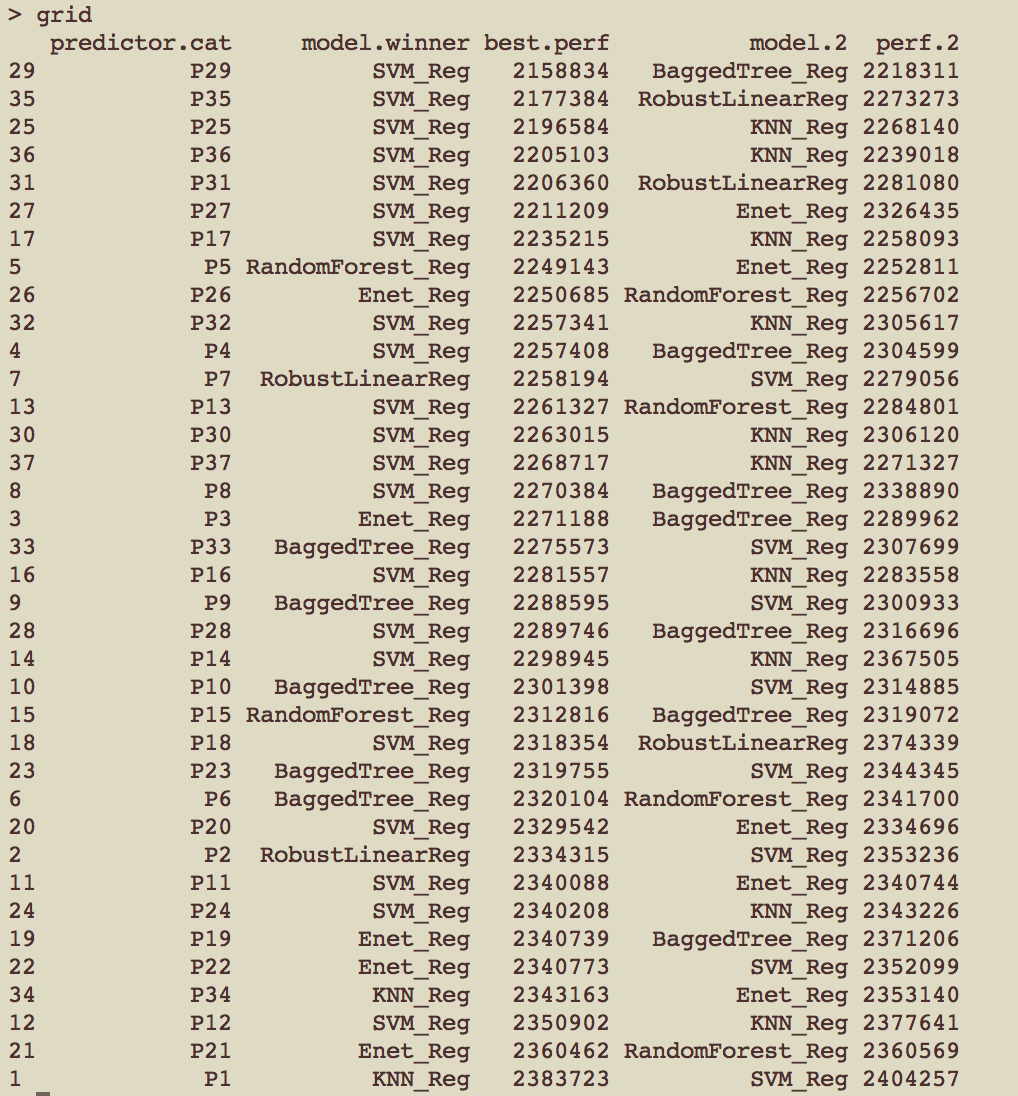
I fatti dimostrano che, se non ci sono bugs, il primo modello da usare e’ **BaggedTree\_Reg poi** I residui vanno futtati con **Enet\_Reg.**

**Inoltre, modelli a 3 livelli (BaggedTree\_Reg,res / Enet\_Re / RandomForest\_Reg ) sembrano meno performanti di quelli a 2 livelli**

# Basic Model improvement #2: guessing categorical variables among P1-P37

We can try to guess one at the time one of P1-P37 as categorical, encoding it, and looking at the performance (both cross validation and test set)

* grid 37 rows
* each row: who’s categorical among P1-P37, winner model, best.perf xval
* **applying caret’s *nearZeroVar* function and other defaults**
* **on the Pi w/ highest probability re-do w/ removeOnlyZeroVariacePredictors=T**



# Bugs

Attenzione alla riga

pred = ifelse(pred >= 1150 , pred , 1150)

> min(y)

[1] 1149870

subs

* mySub\_cat\_guess.csv (solo 1 var categorical con mod. vincente): 1766986.70794
* mySub\_boost\_2.csv (4 var categoriche con boosting a 3 modelli): **1,715,301.26597**
* mySub\_nu\_SVR.csv (4 var categoriche con nu svr): 1834839.58851

quindi:

* mySub\_boost\_2.csv (8 var categoriche con boosting a 3 modelli): **xxxxx** on ledearboard - 1815490.42594
* mySub\_boost\_3.csv (8 var categoriche con boost a 2 modelli): **xxxxx** on ledearboard - 1805490.42594
* mySub\_boost\_3.csv (4 var categoriche con boosting a 3 modelli): **xxxxx** on ledearboard - 1744372.89778

# Exploratory

4 var categoriche

redo xvalidation con 5 var categoriche

\*\*\*\*\*\* RMSE - mean \*\*\*\*\*\*

train.num Average Mode LinearReg RobustLinearReg PLS\_Reg Ridge\_Reg

1 137 2512316 2675786 4137292 2466014 2365618 2529620

Enet\_Reg KNN\_Reg SVM\_Reg BaggedTree\_Reg RandomForest\_Reg Cubist\_Reg best.perf

1 2337727 2341313 2297492 2437006 2407093 2377065 2297492

best.model

1 SVM\_Reg

>>>>>>>>>>>> The winner is ... SVM\_Reg

submission on leaderboard: **1,774,850**

redo with option removeOnlyZeroVariacePredictors=T,

\*\*\*\*\*\* RMSE - mean \*\*\*\*\*\*

train.num Average Mode LinearReg RobustLinearReg PLS\_Reg Ridge\_Reg

1 137 2440218 2915780 14941406 1e+09 2442329 1e+09

Enet\_Reg KNN\_Reg SVM\_Reg BaggedTree\_Reg RandomForest\_Reg Cubist\_Reg best.perf

1 1e+09 2421485 2452221 2285021 2302909 2463320 2285021

best.model

1 BaggedTree\_Reg

>>>>>>>>>>>> The winner is ... BaggedTree\_Reg

submission on leaderboard: **1,858,765**

**re-do with P27 as numeric and not as categorical**

\*\*\*\*\*\* RMSE - mean \*\*\*\*\*\*

train.num Average Mode LinearReg RobustLinearReg PLS\_Reg Ridge\_Reg

1 137 2477831 2874896 3780034 2359975 2305543 2531461

Enet\_Reg KNN\_Reg SVM\_Reg BaggedTree\_Reg RandomForest\_Reg Cubist\_Reg best.perf

1 2286716 2341806 2242088 2357560 2373399 2521279 2242088

best.model

1 SVM\_Reg

>>>>>>>>>>>> The winner is ... SVM\_Reg

submission on leaderboard: **1,753,086**

**boosting a 3 livelli (svm\_reg,enet,** **KNN\_Reg): 1,838,120**

**redo-basic 4 var with option correlationRhreshold = 0.10**

\*\*\*\*\*\* RMSE - mean \*\*\*\*\*\*

train.num Average Mode LinearReg RobustLinearReg PLS\_Reg Ridge\_Reg

1 137 2434974 2955695 4273455 2641816 2413738 2815284

Enet\_Reg KNN\_Reg SVM\_Reg BaggedTree\_Reg RandomForest\_Reg Cubist\_Reg best.perf

1 2569610 **2296753** 2397040 2410936 2385033 2558871 2296753

best.model

1 KNN\_Reg

>>>>>>>>>>>> The winner is ... KNN\_Reg

# Conclusions

So, the best model seems to be

* mySub\_boost\_2.csv (4 var categoriche con boosting a 3 modelli): **1,715,301.26597**

3 models are probably (in order):

* **BaggedTree\_Reg (**removeOnlyZeroVariacePredictors=T , …**)**
* **Enet\_Reg (…)**
* **BaggedTree\_Reg (..)**

So, we now will study this model in each step evaluating performances and possible improvements.

So, every level has the following parameters

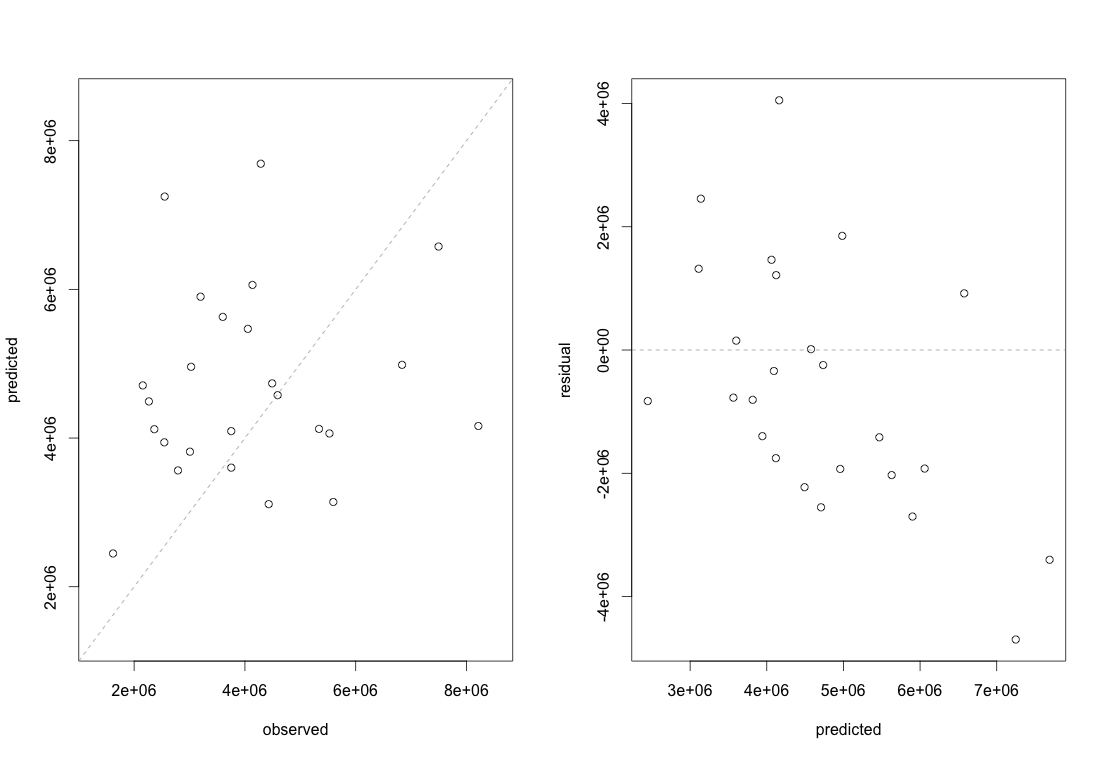
* Model (e.g. **BaggedTree\_Reg**)
* Data processing settings (e.g. removeOnlyZeroVariacePredictors=T)

## First level (BaggedTree\_Reg)

With 6-folds cross validation tried the following options

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Resampling | boot 200 | repeatedcv 10 30 | repeatedcv 10 30 | repeatedcv 10 30 |
| removeOnlyZeroVariacePredictors (on train set) | T | T | F | T |
| removePredictorsMakingIllConditionedSquareMatrix | T | T | T | F |
| correlationTreshold | NA | NA | NA | NA |
| removeHighCorrelatedPredictors | T | T | T | F |
| **6-fold RMSE - mean** | **2373970** | **2322744** | **2337519** | **2348031** |
| 6-fold RMSE - standard deviation | 868705 | 865,451 | 910,513 | 756967 |

So, using the best combination, this is the situation.



The **RMSE is 2.051.269** (not bad in comparison to the mean of 2.322.744), but it doesn’t seem to fit much. The highest error is 4,696,876 where for a observed value of 255,1252 we have a prediction of 7,248,128. Notice that the maximum value of the response variable is 19,696,939. So the prediction is not trivially wrong.

## Repeating the procedure considering P29 as numeric instead of categorical

****

**RMSE 1,707,604(better than 2.051.269)**

**Mean residuals = -525,489**

STD residuals = 1,658,241

🡪 **P29 is numeric**

## Second level (Enet\_Reg)

So, using the best settings of level 1, we

* estimate the residuals on training set (6-folds)
* find the best model fitting the residuals

**Residuals**

|  |  |  |
| --- | --- | --- |
| **ID** | **Mean** | **Standard Deviation** |
| 1 | -89,961 | 2,406,492 |
| 2 | -3,600 | 2,454,654 |
| 3 | -36,977 | 2,512,379 |
| 4 | -64,472 | 2,507,337 |

**Best Model - Residuals**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Resampling** | **removeOnlyZeroVariacePredictors (train)** | **correlationRhreshold** | **Best Model** | **6-fold RMSE** |
| 1 | repeatedcv 10 30 | F | NA | Cubist\_Reg | 2,269,758 |
| 2 | boot 200 | F | NA | Average | 2,378,161 |
| 3 | repeatedcv 10 30 | T | 0.10 |  |  |
| 4 | repeatedcv 10 30 | T | NA | Average | 2,360,618 |

## Assuming the best model Enet\_Reg

(removeOnlyZeroVariacePredictors = F ,

performVarianceAnalysisOnTrainSetOnly = T ,

correlationRhreshold = NA)



**RMSE = 1,757,672 (better than 2.051.269)**

**Mean residuals = -546,489**

STD residuals = 1,705,005

## Repeating the procedure considering P29 as numeric instead of categorical

****

**RMSE = 2,522,002 (worst than 2.051.269)**

**Mean residuals = -** 53893.85

STD residuals = 2573420

It’s obvious that is only **1 outlier making the differences**

**Without the outlier**

**RMSE = 1,333,524**

### Let’s focus on such an outlier

Restaurants in Istanbul are 50 in the train set.

Restaurants in Istanbul with P1 == 5 there are 4

* 1,149,870
* 8,904,084
* 16,549,064
* 5,906,596

Restaurants in Istanbul with P1 == 5 there and train$P8 == 3

* **16,549,064**

That it’s our outlier. This suggest maybe a new feature **HighRevenuesCombinations** that is 1 when P8 == 3 and P1 == 5.

In the training set there’re is only one example, our outlier

> y[train$P1 == 5 & train$P8 == 3]

[1] 16,549,064

## Repeating the procedure considering P29 as numeric instead of categorical, P31 as categorical, P5 as categorical



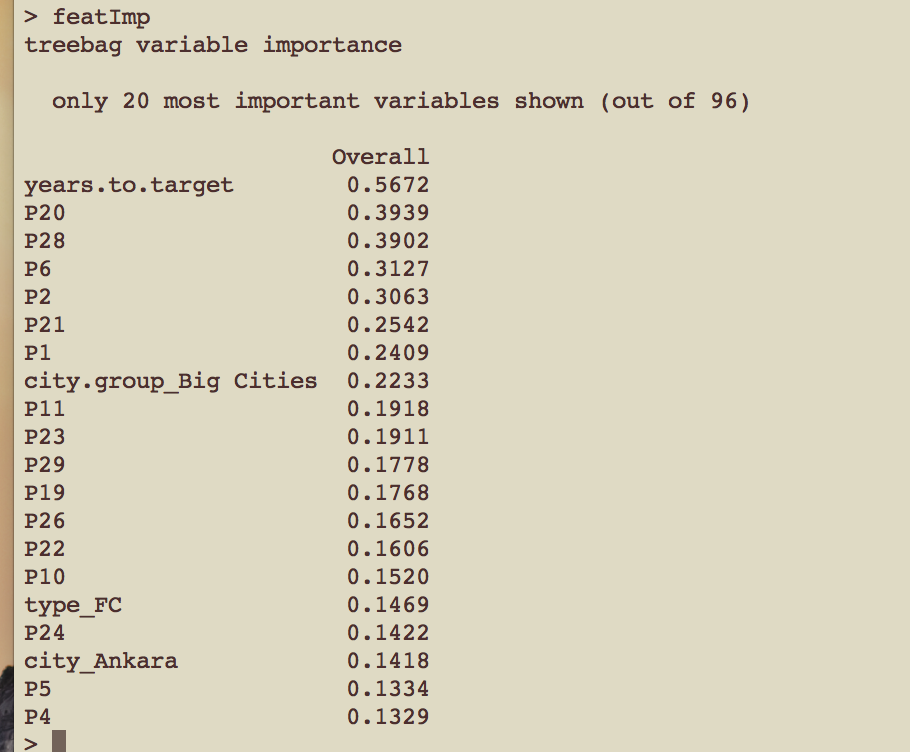
>> **RMSE = 2,439,904**

>> residuals - mean = -901324.7

>> residuals - sd = 2314075

🡪 Bad

# Predictor Importance BaggedTree\_Reg as a model





# Analysis on features values occurring in test set but not in train set

>>> City : vals[ 29 ] Aksaray Artvin Batman Bilecik Çanakkale Çankırı Çorum Düzce Erzincan Erzurum Giresun Hatay Kahramanmaraş Kars Kırıkkale Kırşehir Malatya Manisa Mardin Mersin Nevşehir Niğde Ordu Rize Siirt Sivas Tanımsız Yalova Zonguldak ...

>>> City Group : vals[ 0 ] ...

>>> Type : vals[ 1 ] MB ...

>>> P1 : vals[ 1 ] 15 ...

>>> P2 : vals[ 1 ] 1.5 ...

>>> P3 : vals[ 0 ] ...

>>> P4 : vals[ 1 ] 2 ...

>>> P5 : vals[ 0 ] ...

>>> P6 : vals[ 0 ] ...

>>> P7 : vals[ 1 ] 6 ...

>>> P8 : vals[ 0 ] ...

>>> P9 : vals[ 1 ] 6 ...

>>> P10 : vals[ 0 ] ...

>>> P11 : vals[ 0 ] ...

>>> P12 : vals[ 0 ] ...

>>> P13 : vals[ 0 ] ...

>>> P14 : vals[ 0 ] ...

>>> P15 : vals[ 1 ] 6 ...

>>> P16 : vals[ 1 ] 6 ...

>>> P17 : vals[ 1 ] 12 ...

>>> P18 : vals[ 2 ] 2 15 ...

>>> P19 : vals[ 0 ] ...

>>> P20 : vals[ 0 ] ...

>>> P21 : vals[ 1 ] 12 ...

>>> P22 : vals[ 0 ] ...

>>> P23 : vals[ 0 ] ...

>>> P24 : vals[ 0 ] ...

>>> P25 : vals[ 1 ] 6 ...

>>> P26 : vals[ 0 ] ...

>>> P27 : vals[ 1 ] 7.5 ...

>>> P28 : vals[ 0 ] ...

>>> P29 : vals[ 1 ] 10 ...

>>> P30 : vals[ 1 ] 2 ...

>>> P31 : vals[ 0 ] ...

>>> P32 : vals[ 0 ] ...

>>> P33 : vals[ 1 ] 1 ...

>>> P34 : vals[ 3 ] 1 6 30 ...

>>> P35 : vals[ 0 ] ...

>>> P36 : vals[ 2 ] 1 8 ...

>>> P37 : vals[ 0 ] ...

>>> years.to.target : vals[ 3 ] 11 13 19 ...

So, remembering that mean **correct test output is 4,453,532.6** and that the mean of **my best submission is 4,576,304**

* there’re **26,664 cases** where prediction is done with **at least one not covered feature on the test set** (**26%**); mean = 4,499,436 – std. = 1,086,667
* there’re **3,901 cases** where prediction is done with **at least two not covered feature on the test set** (**3.9%**); mean = 4,526,733 – std. = 966,058.9
* there’re **667 cases** where prediction is done with **at least two not covered feature on the test set** (**0.7%);** mean = 4,587,734– std. = 904,305.9

So, the more the uncovered features the more, on average, the predicted revenues (bias?).

## Years.to.target not covered

On training set

> sort(unique(train.an$years.to.target))

[1] 0 1 2 3 4 5 6 7 8 9 10 12 14 15 16 17 18

while not covered values are 11 13 19

**There’re 2,783 cases – mean = 6,081,344 – std. = 661,393.5 (my best prediction)**

Focusing on the ones where years.to.target = 19 (not covered in train range), there’re 1,253 cases - mean = 6,379,378 – std. = 602,865.4

> year.to.target.clusters

year.to.target rev.mean rev.sd zscore

1 0 2464944 1859796.2 1.3253842

2 1 2532287 1177020.1 2.1514389

3 2 3837859 1644521.8 2.3337236

4 3 4147879 1407195.9 2.9476200

5 4 4383878 1571571.4 2.7894868

6 5 4652457 2250819.2 2.0670062

7 6 4876962 1752458.0 2.7829264

8 7 5079333 2607745.9 1.9477868

9 8 4203972 2361841.7 1.7799548

10 9 6611118 6687625.3 0.9885599

11 10 3482435 763747.6 4.5596670

12 12 4991022 2533426.4 1.9700679

13 14 13596016 8628008.8 1.5757999

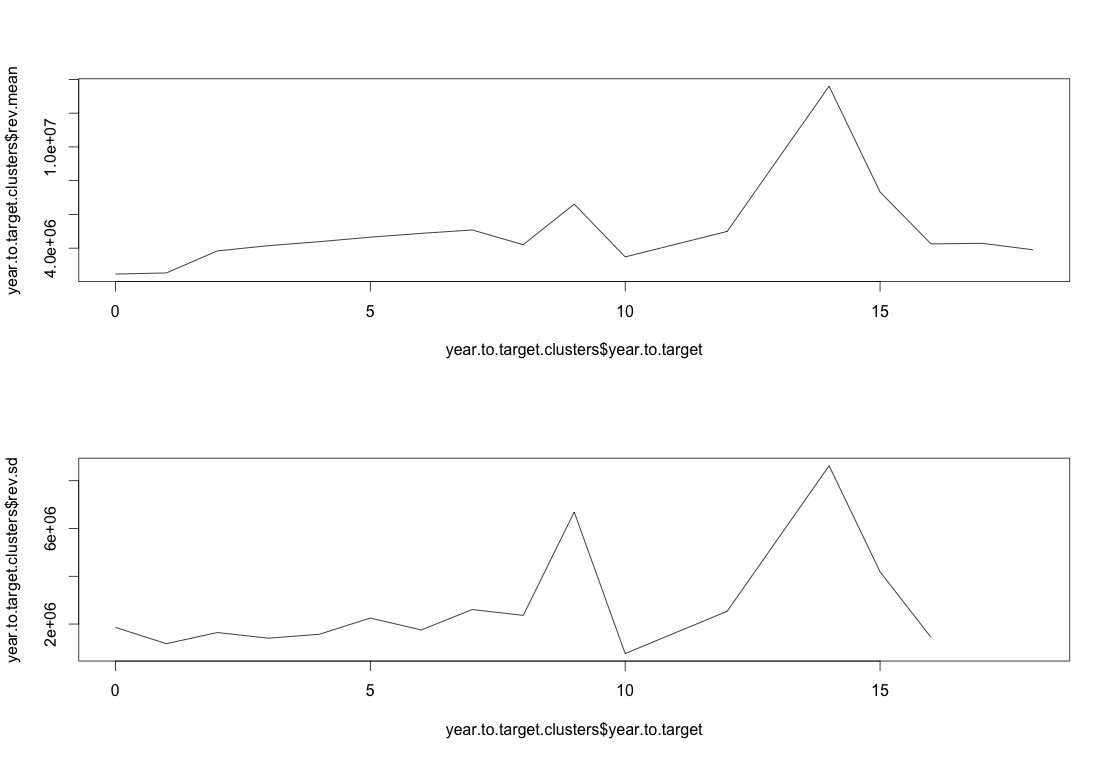
14 15 7329030 4186287.3 1.7507231

15 16 4251905 1444054.0 2.9444224

16 17 4286645 NA NA

17 18 3903884 NA NA

**very good z-score**



So, **we can think to create artificially 3 more entries in the training set**

* years.to.taget = 11 (mean / mode of 10 and 12 data )
* years.to.taget = 13 (mean / mode of 12 and 14 data )
* years.to.taget = 19 (mean / mode of 18 data)

1. removePredictorsMakingIllConditionedSquareMatrix = T, removeHighCorrelatedPredictors = T, featureScaling = T [↑](#footnote-ref-1)