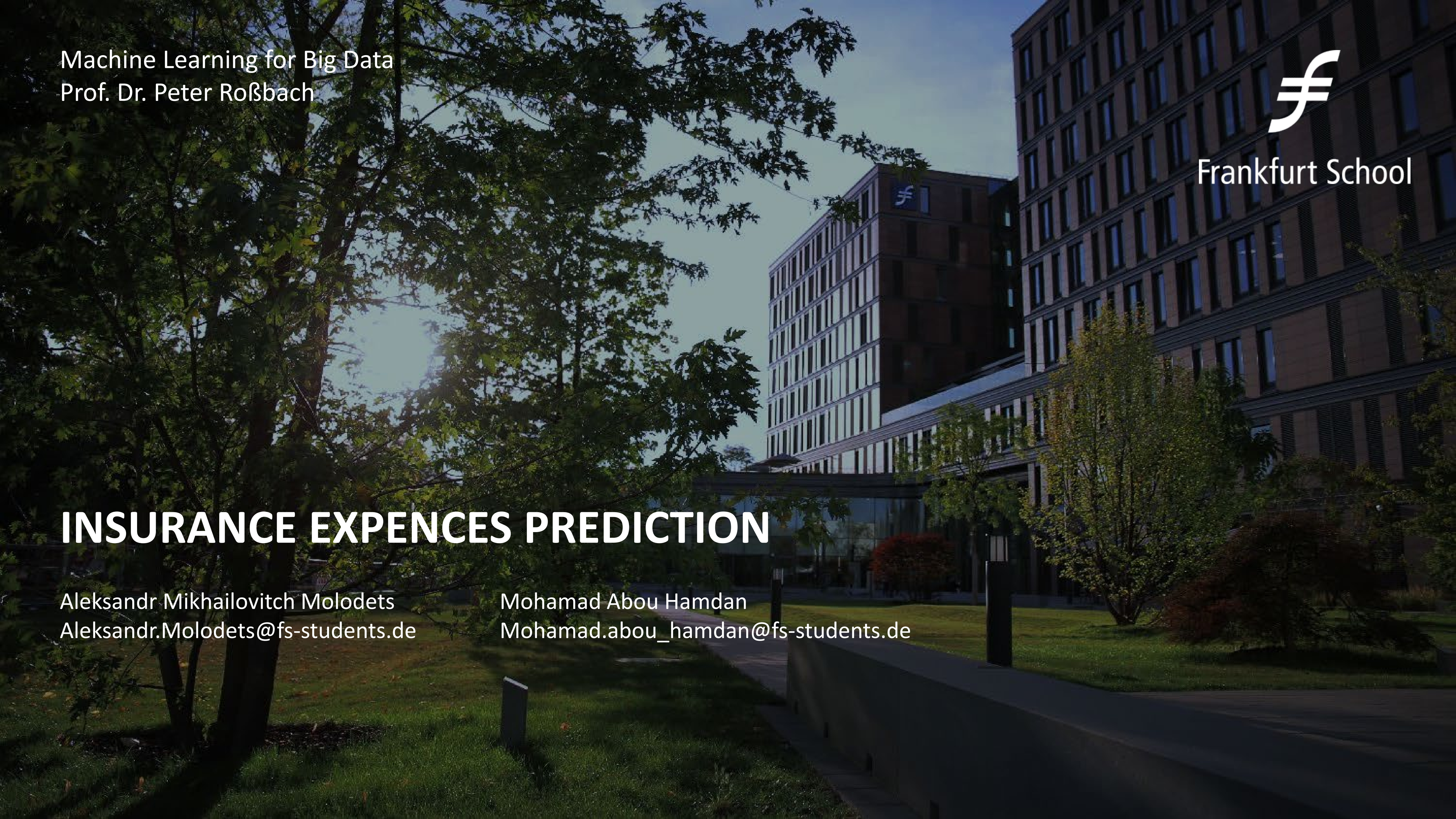


Machine Learning for Big Data
Prof. Dr. Peter Roßbach

Frankfurt School

INSURANCE EXPENCES PREDICTION

Aleksandr Mikhailovitch Molodets
Aleksandr.Molodets@fs-students.de

Mohamad Abou Hamdan
Mohamad.abou_hamdan@fs-students.de

Task description

Given the present dataset on the behaviour and personal characteristics of the clients, build an ML model best suited to estimate future medical expenditures

Coming next:

Dataset inspection and preparation: →

Original dataset inspection

1338 observations of 8 variables

	Characteristics						
	Independent variables						Dependent variable
Name of the variable	**age**	**sex**	**bmi**	**children**	**smoker**	**region**	**expenses**
Values	Numeric	**Male/female**	Numeric	Numeric	**Yes/No**	**Northeast/ northwest/ southeast/ southwest**	Numeric

7IV's and 1 DV

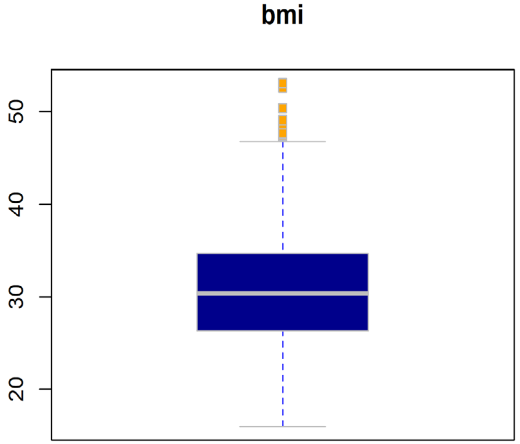
No N/A values

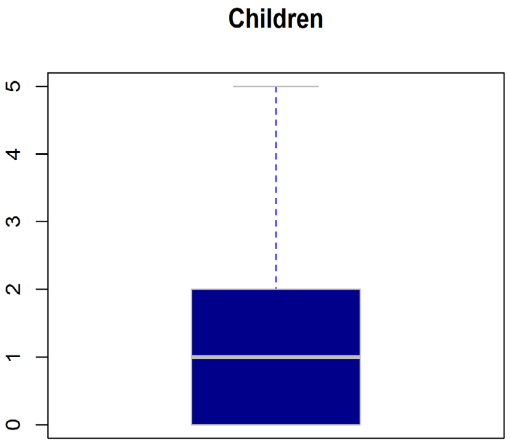
Factor variables: sex, smoker, region

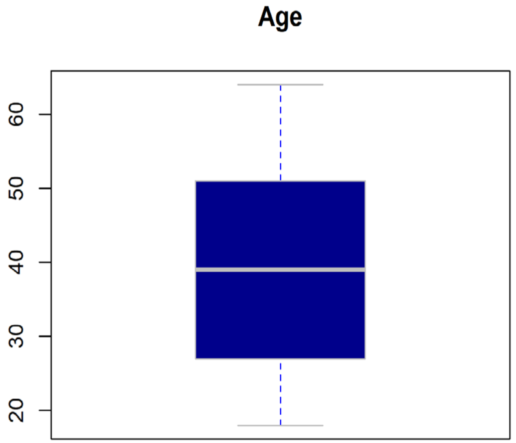
Creating dummy variables

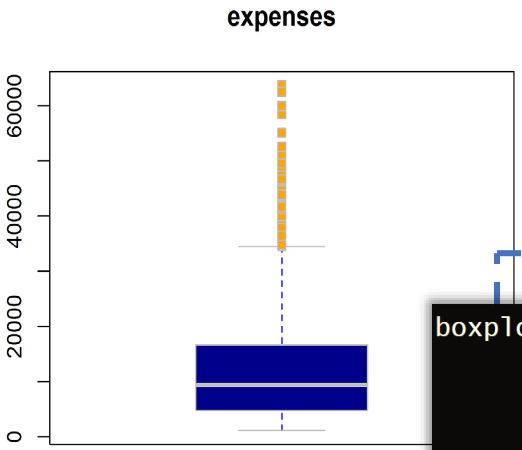
age	**sexmale**	**bmi**	**children**	**smokeryes**	**regionnorthwest**	**regionsoutheast**	**regionsouthwest**	**expenses**
Numeric	If male = 1 If not =0	Numeric	Numeric	If smoker = 1 If not = 0	If northwest = 1 If not = 0	If southeast = 1 If not = 0	If southwest = 1 If not = 0	Numeric

Outliers research for non-dummy variables









Mean: 30.66547
Median: 30.4
Max: 53.1
Min: 16

Mean: 1.094918
Median: 1
Max: 5
Min: 0

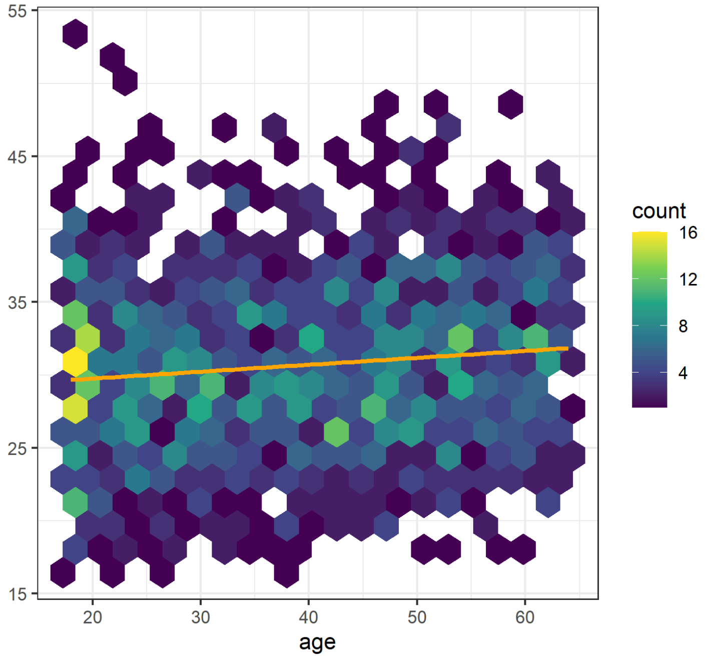
Mean: 39.20703
Median: 39
Max: 64
Min: 18

Mean: 13270.42
Median: 9382.03
Max: 63770.43
Min: 1121.87

```
boxplot(mydata_ori$bmi,
        horizontal = FALSE,
        lwd = 1,
        col = "darkblue",
        xlab = "",
        ylab = "",
        main = "bmi",
        notch = FALSE,
        border = "grey",
        outpch = 22,
        outbg = "orange",
        whiskcol = "blue",
        whisklty = 2,
        lty = 1)
mean(mydata_ori$bmi)
median(mydata_ori$bmi)
max(mydata_ori$bmi)
min(mydata_ori$bmi)
```

Outliers in bmi are within the scale of normality in US, in expenses are theoretically possible
There are no outliers requiring cleaning of a dataset

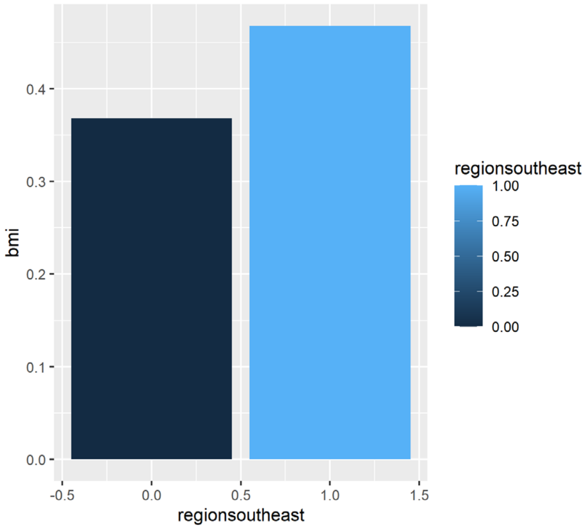
Biggest IV-correlations



Corr = 0.107457

Explainable by biological aspects

```
ggplot(mydata_v1, aes(x=age, y=bmi) )
+ geom_hex(bins = 20)
+ scale_fill_continuous(type = "viridis")
+ theme_bw()
+  geom_smooth(method = "lm", se = FALSE,col="orange")
```



Corr = 0.269927945

Explainable by regional cuisine: might be rational to retrieve more in-depth regional data

Or even conduct a field research!

```
#For binary variables better using boxplot
ggplot(mydata_v3)
+ geom_bar(aes(regionsoutheast, bmi, fill = regionsoutheast),
           position = "dodge", stat = "summary", fun = "mean")
```

Default prediction and default RSS and trControl

- default.pred (Mean of expenses): 0.1987984
- Default residual sum of squares (training): 41.37523
- Default residual sum of squares (test): 8.613895

trControl: repeated cross-validation: 10-fold, 2 repeats

We encountered instable results during grid selection for neural networks, so decided to improve stability with repeats

```
#Mean of the dependent variable:
default.pred <- mean(mydata_v3.train$expenses)
default.pred

#ESS for training and training set compared to default
default.train.rss <- sum((mydata_v3.train$expenses-default.pred)^2)
default.train.rss
default.test.rss <- sum((mydata_v3.test$expenses-default.pred)^2)
default.test.rss
```

Coming next:
Creating and adjusting models: →

OLS

```r
set.seed(123)
ols <- train(expenses ~ ., data=mydata_v3.train,
             method="lm", trControl=TControl, metric="Rsquared")
ols
summary(ols)
ols$finalModel #Inspecting final model
```

Multiple R2: 0.753285471

	Values of coefficients
(Intercept)	-0.049353 (0.010757) ***
age	0.178726 (0.010023) ***
sexmale	-0.00346 (0.006042)
bmi	0.210286 (0.019317) ***
children	0.044712 (0.012472) ***
smokeryes	0.385820 (0.007342) ***
regionnorthwest	-0.00889 (0.008613)
regionsoutheast	-0.013671 (0.008647)
regionsouthwest	-0.015437 (0.008651)

*Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1*

R2 for training set: 0.7533

Pseudo R2 for test set: 0.7367

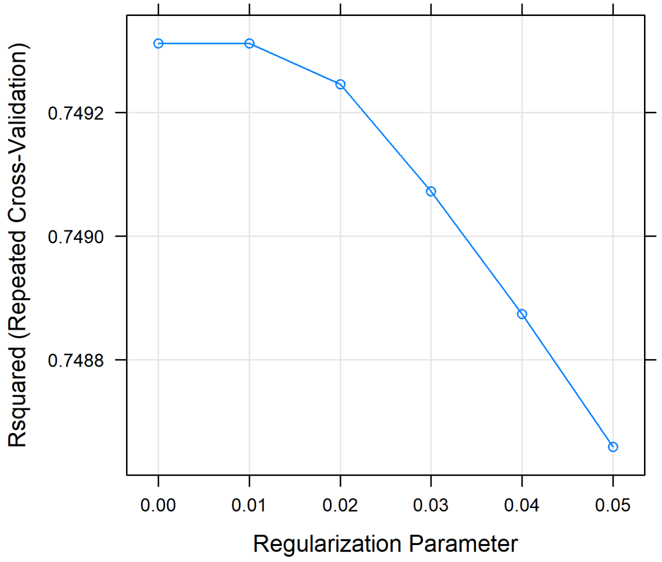
Not the best predictive quality: Highly likely that the relationship Is not linear.

Ridge regression

```
#Ridge
ridgeGrid <- expand.grid(alpha=0.00,
                         lambda=seq(from=0.0, to=0.05, by=0.001))
set.seed(123)
ridge <- train(expenses ~ .,
               data=mydata_v3.train,
               method="glmnet", tuneGrid=ridgeGrid,
               trControl=TControl, metric="Rsquared")
```

Tuning grid (Alpha = 0)

Lambda	RMSE	Rsquared	MAE
0	0.09908424	0.749312	0.07016327
0.01	**0.09908424**	**0.749312**	**0.07016327**
0.02	0.09954816	0.7492457	0.07081842
0.03	0.10084844	0.7490722	0.07248005
0.04	0.10242469	0.7488739	0.07421301
0.05	0.10417994	0.7486586	0.07595175



Radical drop for Rsquared on training is observed after lambda>0.01

Best tune:
Alpha=0
Lambda = 0.01

R2 for training set: 0.7507

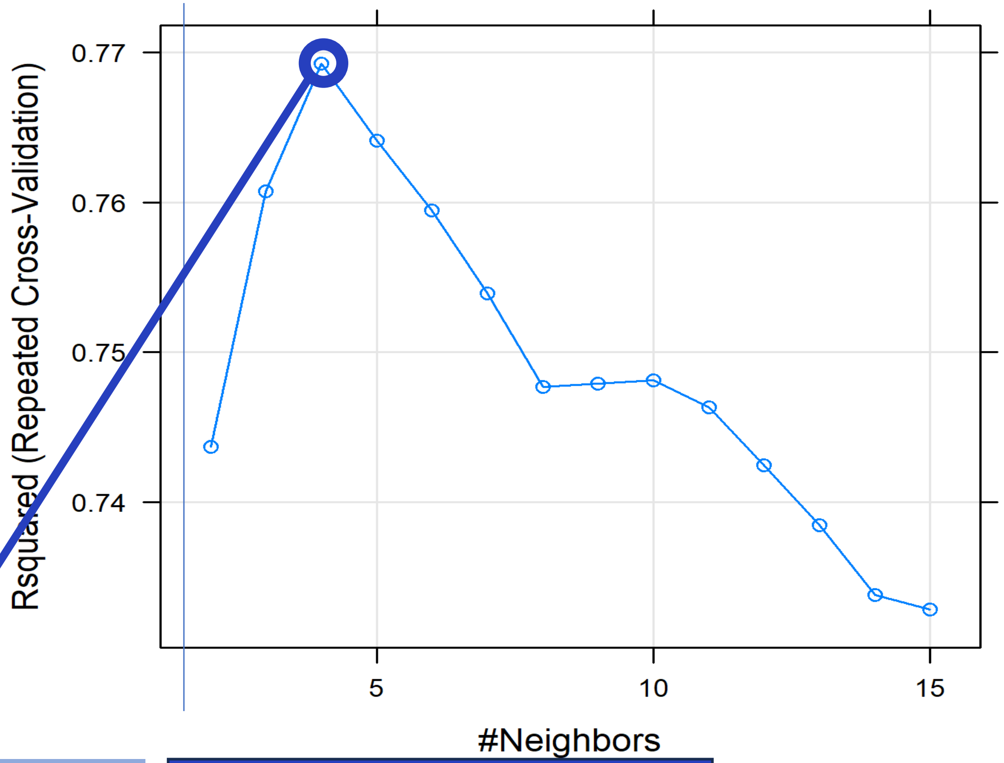
Pseudo R2 for test set: 0.7377

Did not improve the results radically, trying support vector regression.

K-Nearest neighbours

k	RMSE	Rsquared
2	0.100108	0.743676
3	0.096258	0.760743
4	0.094312	0.769267
5	0.095359	0.764131
6	0.096407	0.759458
7	0.097445	0.753921
8	0.098635	0.747705
9	0.098612	0.747886
10	0.098622	0.748135
11	0.099009	0.746331
12	0.099753	0.742467
13	0.100556	0.738475
14	0.101373	0.733812
15	0.101613	0.732828

k=seq(from=2, to=15, by=1)



R2 for training set: 0.8668

Pseudo R2 for test set: 0.7482

Best tune: 4-nearest neighbours model

```
#KNN
knnGrid <- expand.grid(k=seq(from=2, to=15, by=1))
set.seed(123)
knnmodel <- train(expenses ~ .,
                  data=mydata_v3.train, method="knn",
                  tuneGrid=knnGrid, trControl=TControl,
                  metric="Rsquared")
knnmodel
```

Kernel regressions

```r
#Kernel1 Linear+Radial
svlGrid <- expand.grid(C=seq(from=0.01, to=1, by=0.1))
eps <- 0.1
set.seed(123)
svr.linear <- train(expenses ~ .,
                    data=mydata_v3.train,
                    method="svmLinear", tuneGrid=svlGrid,
                    trControl=TControl, metric="Rsquared",
                    epsilon = eps)
```

```r
svrGrid <- expand.grid(sigma = c(.01, .015, 0.2), C = c(1, 10, 50, 100))
eps <- 0.1
set.seed(123)
svr.rbf <- train(expenses ~ .,
                 data=mydata_v3.train,
                 method="svmRadial", tuneGrid=svrGrid,
                 trControl=TControl, metric="Rsquared",
                 epsilon = eps)
```

Linear

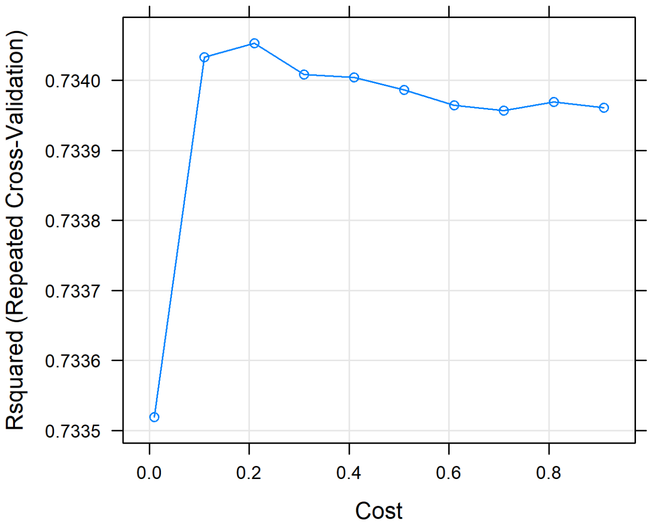
Best tune: c = 0.21

Tuning grid
(eps <- 0.1)

c
0.01
0.11
0.21
0.31
0.41
0.51
0.61
0.71
0.81
0.91

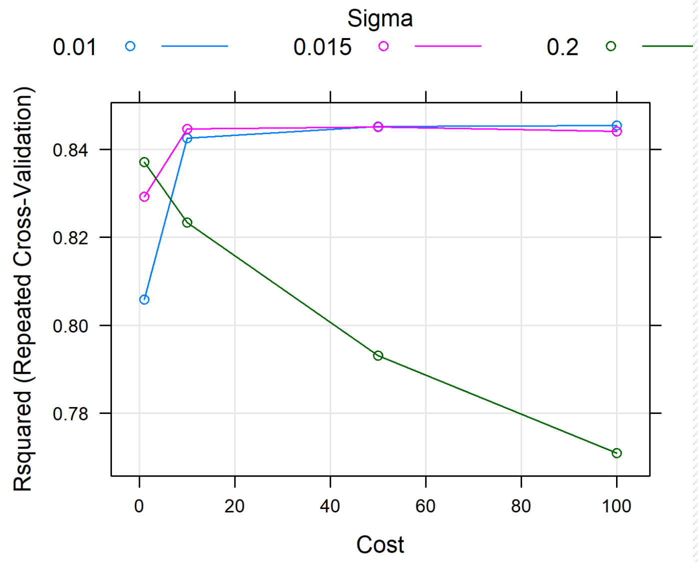
R2 for training set:
0.6999

Pseudo R2 for test set:
0.6685



Radial

sigma	C
0.01	1
0.015	1
0.2	1
0.01	10
0.015	10
0.2	10
0.01	50
0.015	50
0.2	50
0.01	100
0.015	100
0.2	100



Pseudo R2 for test set:
0.8307

Best tune:
sigma = 0.01
C = 100

R2 for training set:
0.8523

Conclusions so far

Radial Kernel regression gives the best results: it means that the relationship is not linear. Using neural network-based regressions might be useful

For this the authors use not only Caret package, but also "NeuralNetTools“, for the intermediary visualization of the trees

Coming next:

Neural network-based regressions: →

Neural network – 1 hidden layer

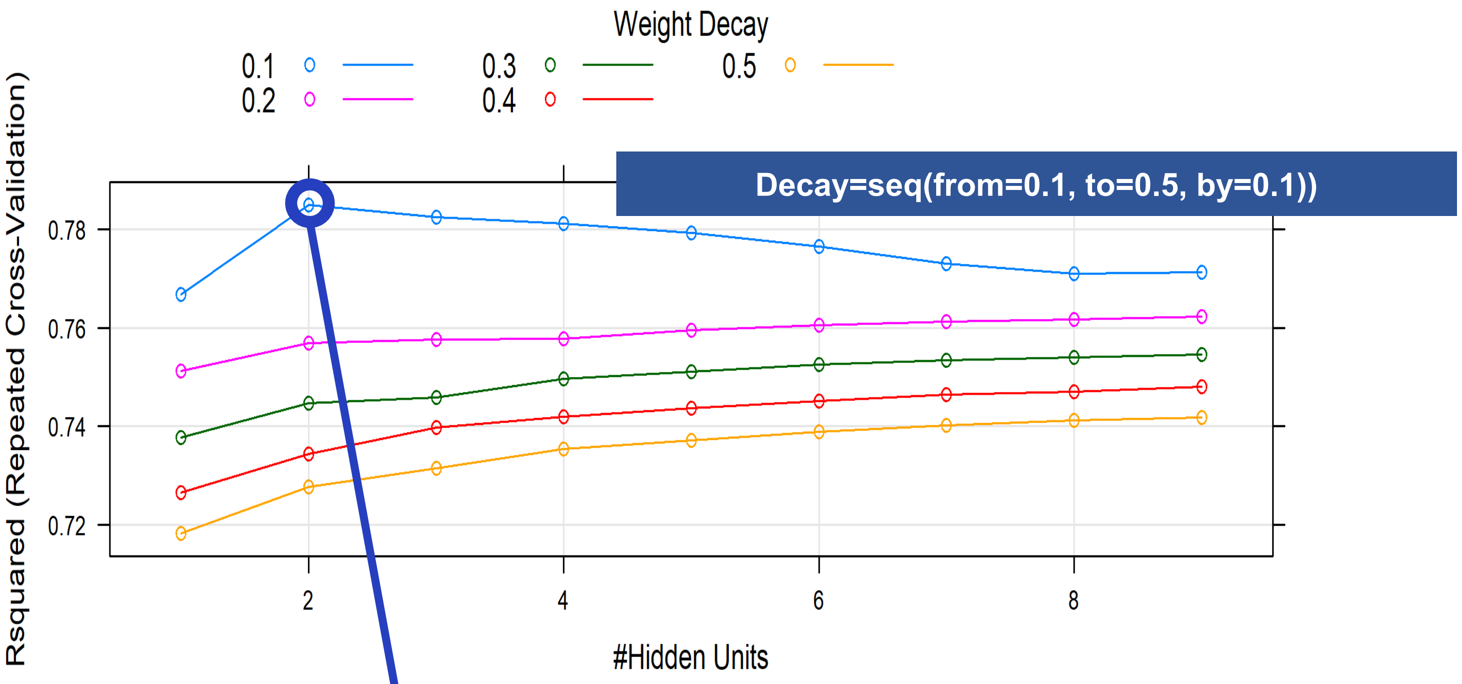
```
nnGrid <- expand.grid(size=1:8,
                      decay=seq(from=0.1, to=0.5, by=0.1))
```

Tuning grid:

Hidden layer size: from 1 to 8
Decay: 0.1,0.2,0.3,0.5

Proposal for adjustment:

To provide more detailed research on decay hyperparameter



R2 for training set: 0.7938

Pseudo R2 for test set: 0.7694

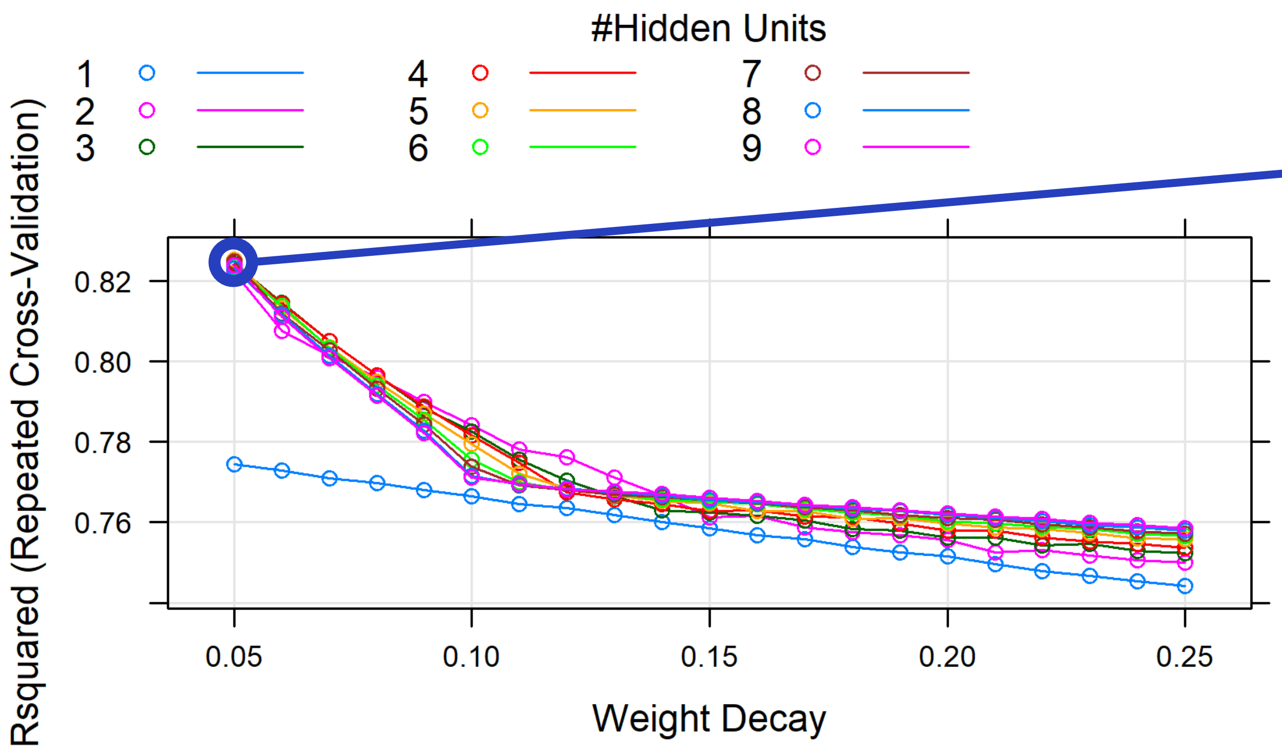
Best tune: (8)-2-(1) Neural network with decay = 0.1

Neural network – 1 hidden layer + decay adjusted

```
nnGrid_1 <- expand.grid(size=1:8,
                        decay=seq(from=0.05, to=0.25, by=0.01))
```

Adjusting to optimize decay

OLD: seq(from=0.1, to=0.5, by=0.1))	NEW: seq(from=0.05, to=0.25, by=0.01))



Neural Network 1	
0.7938406	0.7693969

R2 for training set: 0.8355

Pseudo R2 for test set: 0.8162

Significant Improvement!

Best tune: (8)-5-(1) Neural network with decay = 0.05

Neural network – multiple layers

```r
#NN type 2 - Multi-layered network
nn2Grid <- expand.grid(layer1=c(1,3,5,7,8),
                       layer2=c(0,1,3,5,7,8),
                       layer3=c(0,1,3,5,7,8))
set.seed(123)
nnmodel2 <- train(expenses ~ ., data=mydata_v3.train,
                  method="neuralnet",
                  tuneGrid=nn2Grid,
                  trControl=TControl,
                  metric="Rsquared")
```

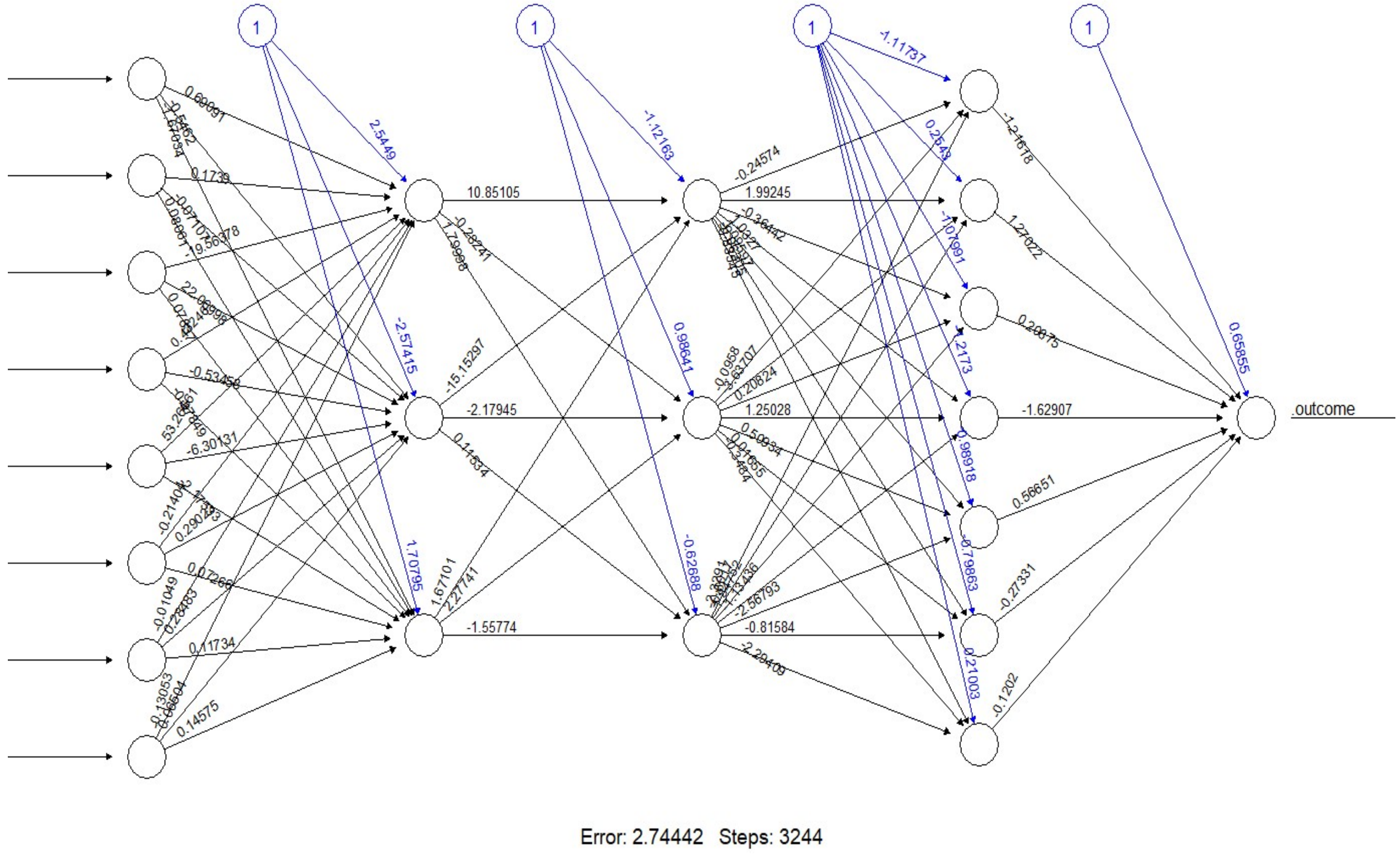
Tuning grid:

Layer 1: from 0 to 8 layers
Layer 2: from 1 to 8 layers
Layer 3: from 1 to 8 layers

R2 for training set:
0.8673

Pseudo R2 for test set:
0.8531

Best tune: (8)-3-3-8-(1) Neural network



```r
set.seed(123)
rfmodel <- train(expenses ~ .,
                 data=mydata_v3.train,
                 method="rf", trControl=TControl,
                 metric="Rsquared")
```

Random forest – without optimization

Tuning grid: Standard: mtry = 2,5,8

mtry	RMSE	Rsquared	MAE
2	0.084449	0.842333	0.057702
5	0.074729	0.852679	0.041999
8	0.076909	0.844876	0.043334

R2 for training set:
0.9573

Pseudo R2 for test set:
0.8481

Best tune: mtry = 5

Model is overfit.
Is there an opportunity to optimize the result?

Coming next:
Deep-Dark forest: →

Deep-Dark Forest: logic of optimization

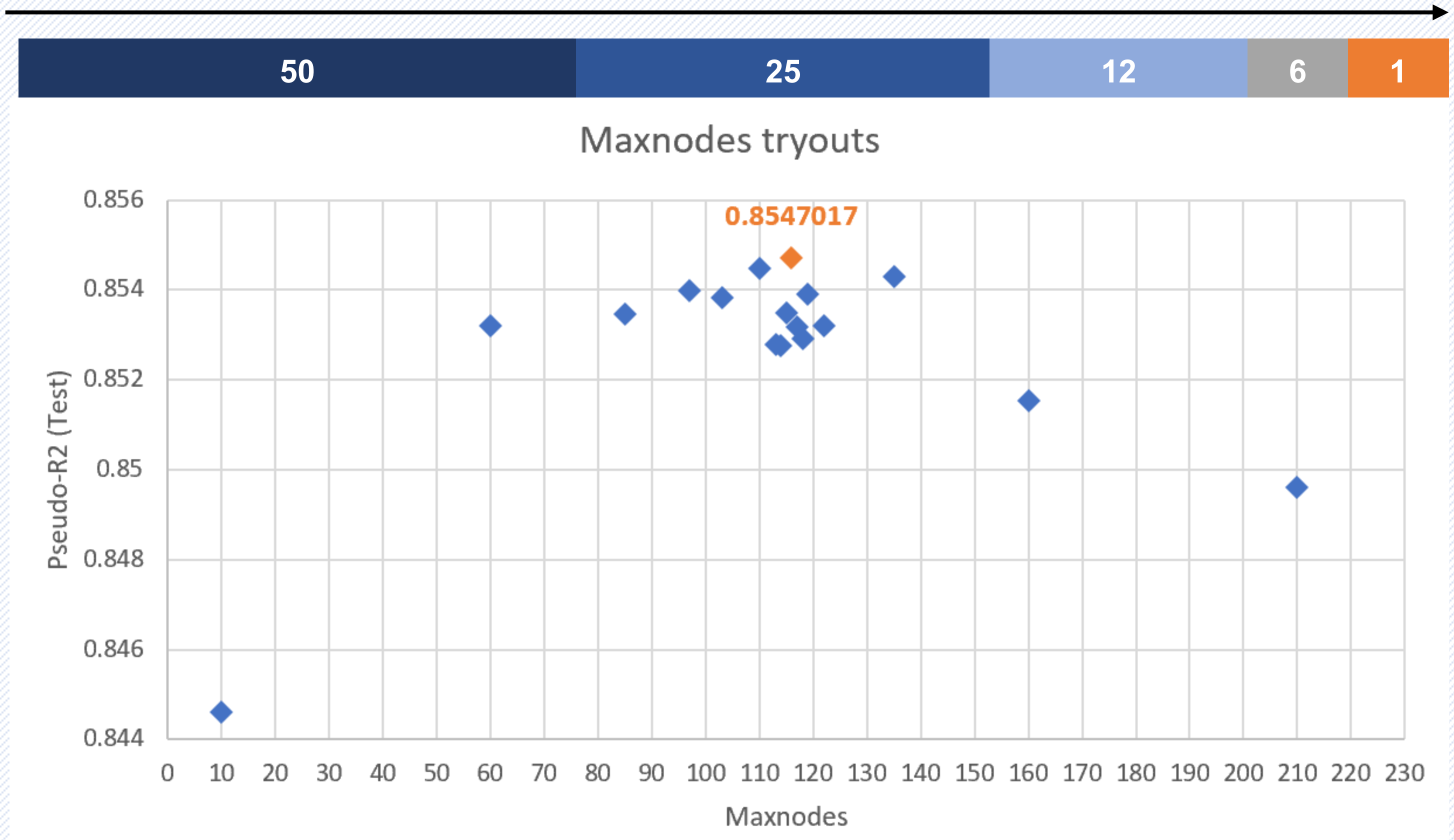
Usually to optimize Random Forest, the cycle is used to find optimal “maxnodes” hyperparameter.

However, usually steps which are used are large.

By using repeatedcv, we allowed ourselves **the stability** to make the cycle step smaller. But not **the time**.

So we thought of a logic of recursively decreasing steps to “scope in” to the optimal result of maxnodes.
(see on the right)

Navigating by decreasing steps to find the optimized number of max nodes

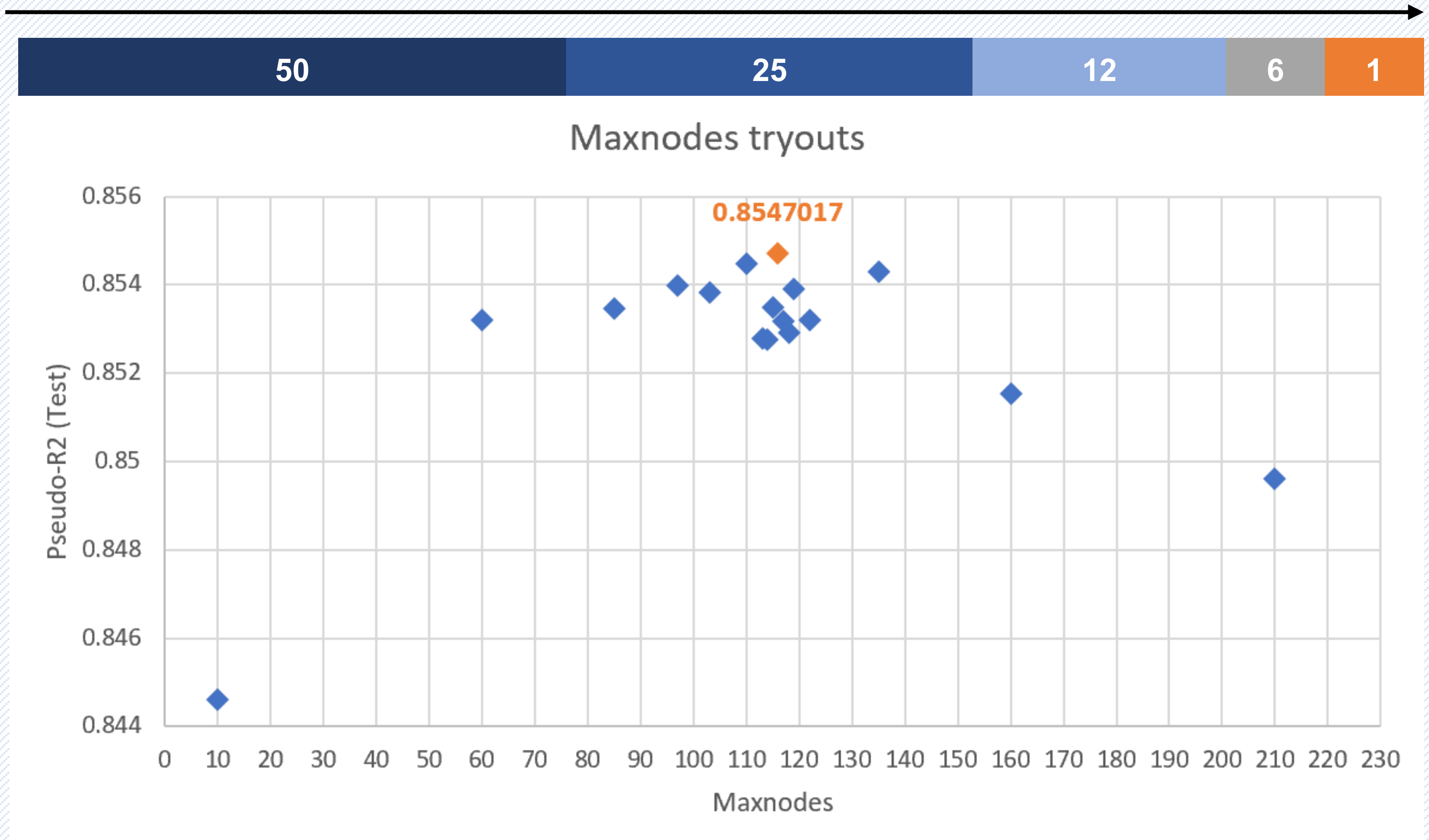


Deep-Dark Forest: logic of optimization

```
for (maxnodes in c(10,60,110,160,210)) {
  set.seed(123)
  rfmodel <- train(expenses ~ ., data=myda
```

```
for (maxnodes in c(60,85,110,135,160)) {
  set.seed(123)
  rfmodel <- train(expenses ~ ., data=myda
```

Navigating by decreasing steps to find the optimized number of max nodes



As a result, optimal number of maxnodes was found at maxnodes=116, which allowed to receive the following results:

R2 for training set: 0.9274

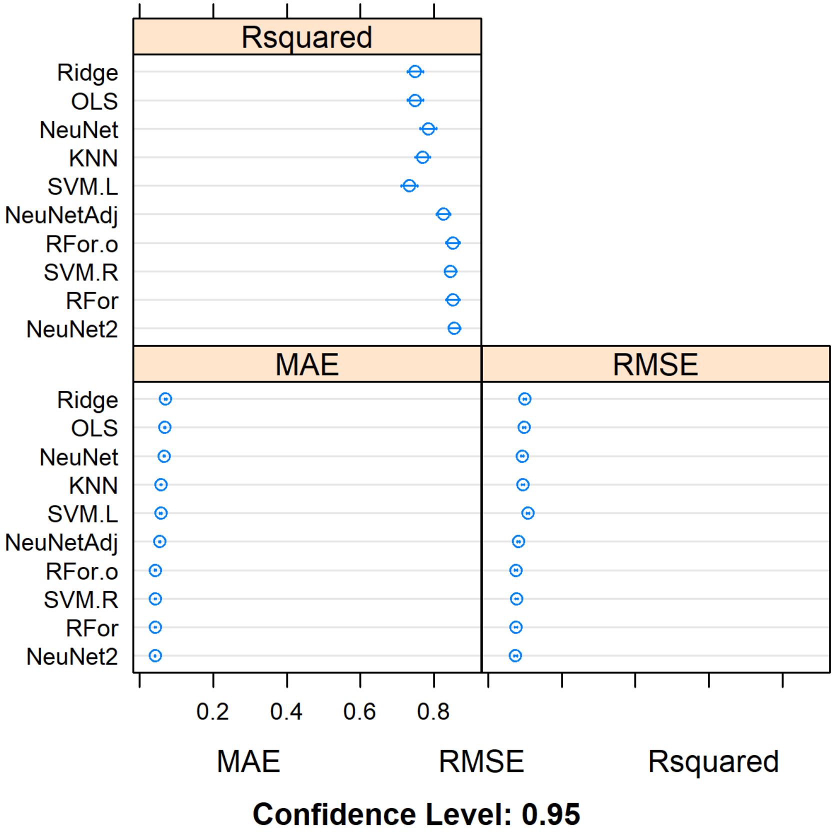
Pseudo R2 for test set: 0.8547

Best tune: maxnodes = 116

This result can be perceived as overfit, however the method used might be relevant for future optimization problems

Final results

- Optimized Random Forest seems overfit, however still performs on test set
- The optimal algorithm is thus multi-layered Neural network (neuralnet method) with (8)-3-3-8-(1) configuration



Model	R2.Train	R2.Test	Hyperparameters
Random Forest optimized	0.927427	0.854702	maxnodes = 116
Neural Network Multi-Layer	0.867340	0.853113	(8)-3-3-8-(1) Neuralnet
Random Forest	0.957386	0.848082	mtry=5
Radial SVR	0.852334	0.830719	sigma = 0.01, Cost = 100
Neural Network Single-Layer Adjusted Decay	0.835498	0.816199	(8)-5-(1) Nnet decay = 0.05
Neural Network Single-Layer	0.793841	0.769397	(8)-2-(1) Nnet decay = 0.1
k-NN	0.866813	0.748233	k=4
Linear Regression	0.753285	0.736734	-
Ridge Regression	0.750788	0.736660	Alpha=0, Lambda = 0.01
Linear SVR	0.699909	0.668502	c = 0.21

Thank you for the attention!

We are open for questions!