Machine Learning - 1100-ML0ENG (Ćwiczenia informatyczne Z-23/24)

Google Trends and the Stock Market - Regression

We will use the Google trends and stock market dataset. We will show the application of a **neural network** to examine the associations between Google search trends and the daily marker index - Dow Jones Industrial Average.

```
google=read.csv("http://imul.math.uni.lodz.pl/~bartkiew/ml/data/google.csv",stringsA
sFactors =T)
str(google)
```

Variables

- **X** - number
- **Index**: Time Index of the Observation
- **Date**: Date of the observation (Format: YYYY-MM-DD)
- **Unemployment**: The Google Unemployment Index tracks queries related to “unemployment, social, social security, unemployment benefits” and so on.
- **Rental**: The Google Rental Index tracks queries related to “rent, apartments, for rent, rentals”, etc.
- **RealEstate**: The Google Real Estate Index tracks queries related to “real estate, mortgage, rent, apartments” and so on.

- **Mortgage**: The Google Mortgage Index tracks queries related to “mortgage, calculator, mortgage calculator, mortgage rates”.
- **Jobs**: The Google Jobs Index tracks queries related to “jobs, city, job, resume, career, monster” and so forth.
- **Investing**: The Google Investing Index tracks queries related to “stock, finance, capital, yahoo finance, stocks”, etc.
- **DJI_Index**: The Dow Jones Industrial (DJI) index. These data are interpolated from 5 records per week (Dow Jones stocks are traded on week-days only) to 7 days per week to match the constant 7-day records of the Google-Trends data.
- **StdDJI**: The standardized-DJI Index computed by: StdDJI = 3+(DJI-11091)/1501, where m=11091 and s=1501 are the approximate mean and standard-deviation of the DJI for the period (2005-2011).
- **30-Day Moving Average Data Columns**: The 8 variables below are the 30-day moving averages of the 8 corresponding (raw) variables above: *Unemployment30MA, Rental30MA, RealEstate30MA, Mortgage30MA, Jobs30MA, Investing30MA, DJI_Index30MA*, and *StdDJI_30MA*.
- **180-Day Moving Average Data Columns**: The 8 variables below are the 180-day moving averages of the 8 corresponding (raw) variables: *Unemployment180MA, Rental180MA, RealEstate180MA, Mortgage180MA, Jobs180MA, Investing180MA, DJI_Index180MA*, and *StdDJI_180MA*.

Note that the variables highlighted in red can be omitted. The first three are identifiers, the last are statistical parameters calculated from the previous columns.

```
google=google[,-1:-3]
google=google[,-9:-24]
```

Result of deleting unnecessary columns

```
> str(google)
'data.frame':   731 obs. of  8 variables:
 $ Unemployment: num  1.54 1.56 1.59 1.62 1.64 1.64 1.71 1.85 1.82 1.78 ...
 $ Rental      : num  0.88 0.9 0.92 0.92 0.94 0.96 0.99 1.02 1.02 1.01 ...
 $ RealEstate  : num  0.79 0.81 0.82 0.82 0.83 0.84 0.86 0.89 0.89 0.89 ...
 $ Mortgage    : num  1 1.05 1.07 1.08 1.1 1.11 1.15 1.22 1.23 1.24 ...
 $ Jobs        : num  0.99 1.05 1.1 1.14 1.17 1.2 1.3 1.41 1.43 1.44 ...
 $ Investing   : num  0.92 0.94 0.96 0.98 0.99 0.99 1.02 1.09 1.1 1.1 ...
 $ DJI_Index   : num  13044 13044 13057 12800 12827 ...
 $ StdDJI      : num  4.3 4.3 4.31 4.14 4.16 4.16 4.16 4 4.1 4.17 ...
>
```

Let us note that our data is not perfect. We should have even ranges of each variable.

```
summary(google)
```

```
normalize <- function(x) {
 return((x - min(x)) / (max(x) - min(x)))
}
```

```
google.norm<-as.data.frame(lapply(google, normalize))
```

Here we use the **RealEstate** our dependent variable. We check if Google Real Estate Index could be predicted by other variables in our google dataset.

We will now prepare a training and test datasets.

```
# install.packages("caTools")
library(caTools)
set.seed(123)
split = sample.split(google.norm$RealEstate, SplitRatio = 0.7)
training_set = subset(google.norm, split == TRUE)
test_set = subset(google.norm, split == FALSE)
```

```
summary(google.norm$RealEstate)
summary(training_set$RealEstate)
```

Neural network model

```
# install.packages("neuralnet")
library(neuralnet)
google.nn1<-
neuralnet(RealEstate~Unemployment+Rental+Mortgage+Jobs+Investing+DJI_Index+StdDJI,
data=training_set)
```

```
google.nn1
```

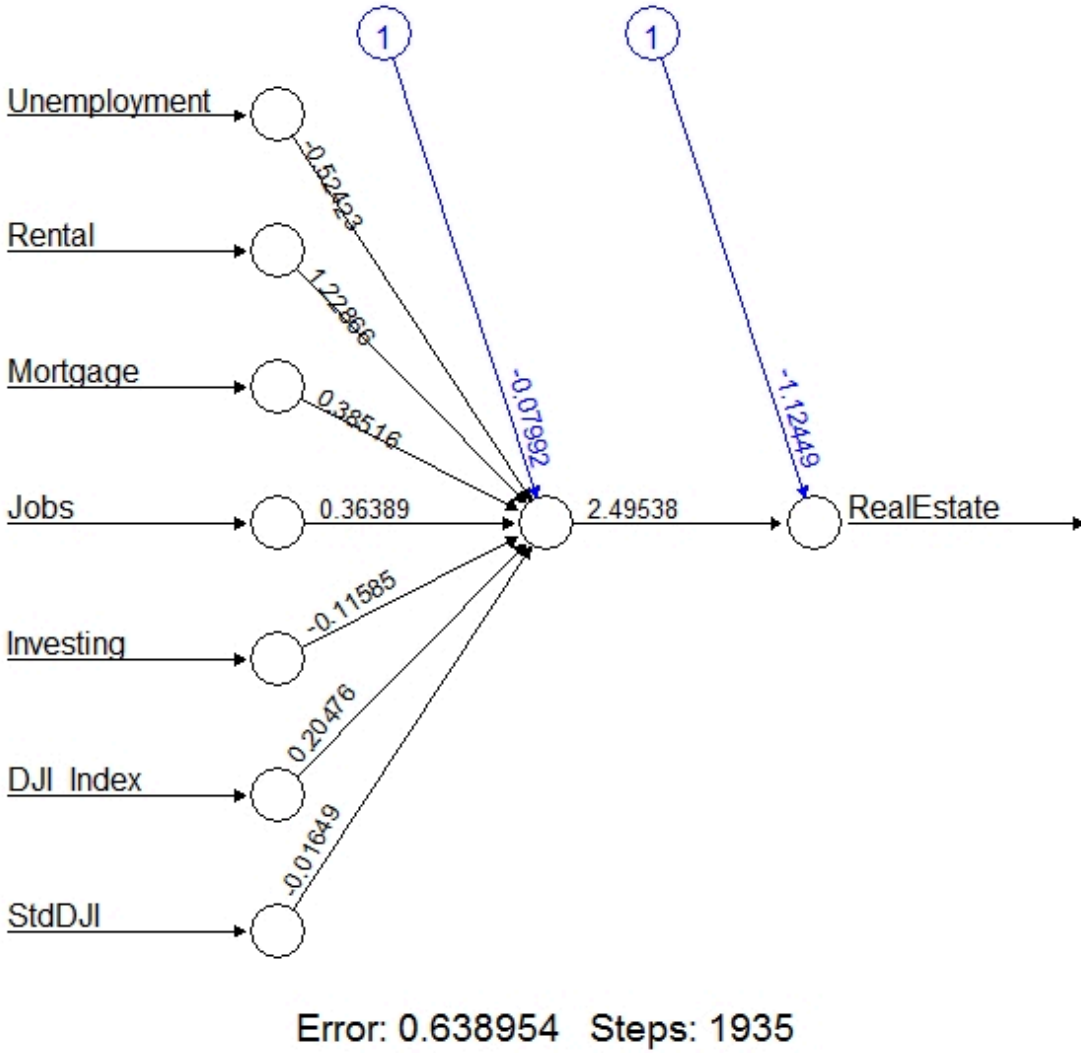
In neuralnet package we use the **function neuralnet** which returns a NN object containing:

- **call;** the matched call.
- **response;** extracted from the data argument.
- **covariate**; the variables extracted from the data argument.
- **model.list;** a list containing the covariates and the response variables extracted from the formula argument.
- **err.fct and act.fct;** the error and activation functions.
- **net.result;** a list containing the overall result of the neural network for every repetition.
- **weights;** a list containing the fitted weights of the neural network for every repetition.
- **result.matrix;** a matrix containing the reached threshold, needed steps, error, AIC, BIC, and weights for every repetition. Each column represents one repetition.

```
m <- neuralnet(target ~ predictors, data=mydata, hidden=1)
```

- **target:** variable we want to predict.
- ***predictors*: predictors we want to use. Note that we cannot use "." to denote all the variables in this function. We have to add all predictors one by one to the model.**
- **data:** training dataset.
- **hidden:** number of hidden nodes that we want to use in the model. By default, it is set to one.

```
plot(google.nn1)
```



The above graph shows that we have only one hidden node. **Error** represents the aggregate sum of squared errors and **Steps** is how many iterations the model went through. **Note that these outputs could be different when you run exact same codes twice because the weights are stochastically estimated.**

Also, *bias nodes* (blue singletons in the graph) may be added to feedforward neural networks acting like intermediate input nodes that produce constant values, e.g., 1. They are not connected to nodes in previous layers, yet they generate biased activation. Bias nodes are not required but are helpful in some neural networks as they allow offsetting activation functions.

Evaluating model

In the neuralnet package we use compute function instead of predict function.

```
google_pred<-compute(google.nn1, test_set)
pred_results<-google_pred$net.result
cor(pred_results, test_set$RealEstate)
r2(pred_results, test_set$RealEstate)
```

Since we are considering a regression task, we cannot use the confusion matrix. To evaluate the model, we can count the correlation, between the values obtained from the network model and those actually present in the dataset.

A **correlation** is a statistical measure of the relationship between two variables.

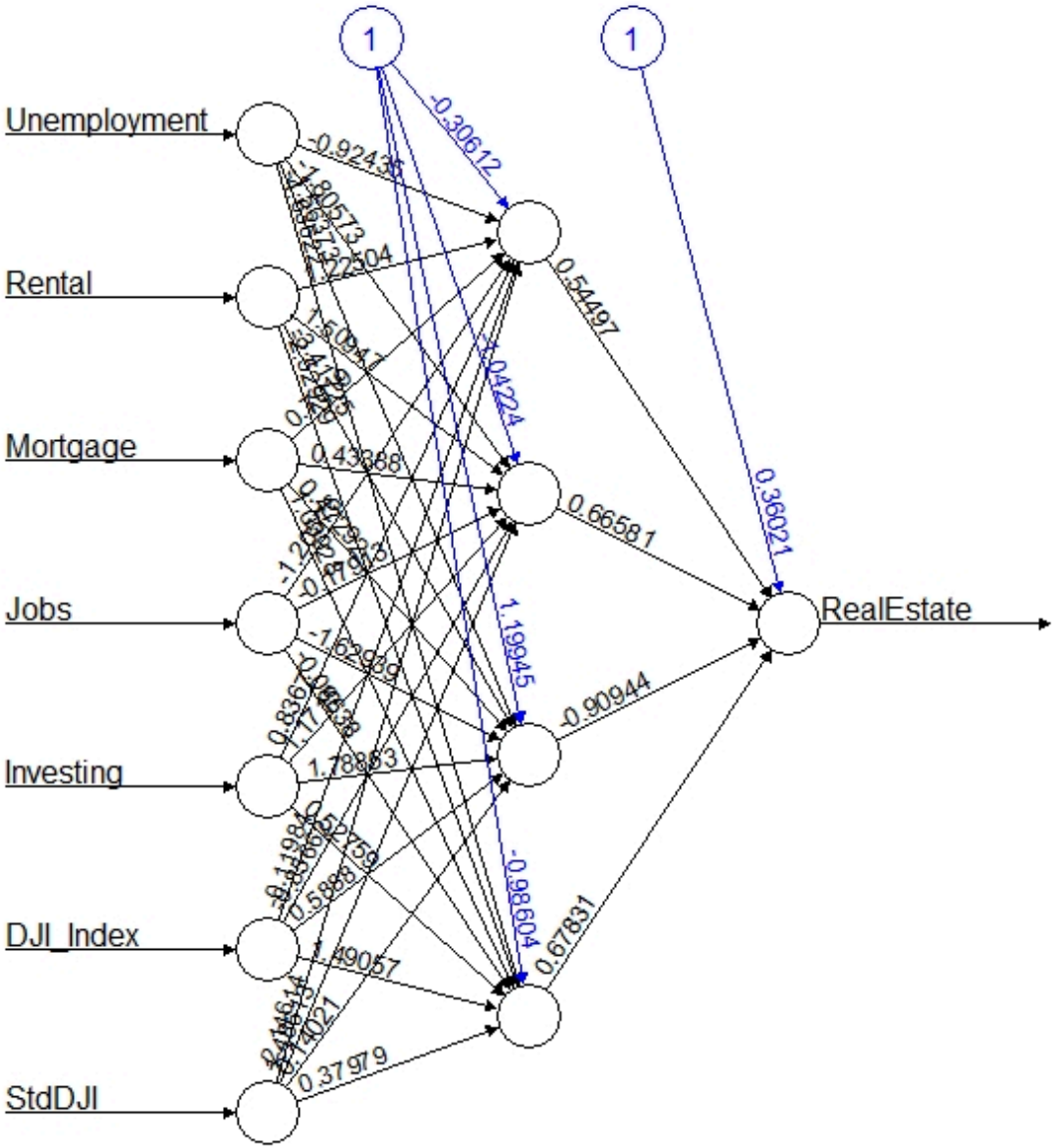
The **correlation coefficient** is a value that indicates the strength of the relationship between variables. The coefficient can take any values from -1 to 1. The interpretations of the values are:

- **-1:** Perfect negative correlation. The variables tend to move in opposite directions (i.e., when one variable increases, the other variable decreases).
- **0:** No correlation. The variables do not have a relationship with each other.
- **1:** Perfect positive correlation. The variables tend to move in the same direction (i.e., when one variable increases, the other variable also increases).

Improving model performance

Now we will put 4 hidden nodes in the NN model, so we will build more complicated model.

```
set.seed(123)
google.nn2<-
neuralnet(RealEstate~Unemployment+Rental+Mortgage+Jobs+Investing+DJI_Index+StdDJI,
data=training_set, hidden = 4)
plot(google.nn2)
google_pred2<-compute(google.nn2, test_set)
pred_results2<-google_pred2$net.result
cor(pred_results2, test_set$RealEstate)
r2(pred_results2, test_set$RealEstate)
```



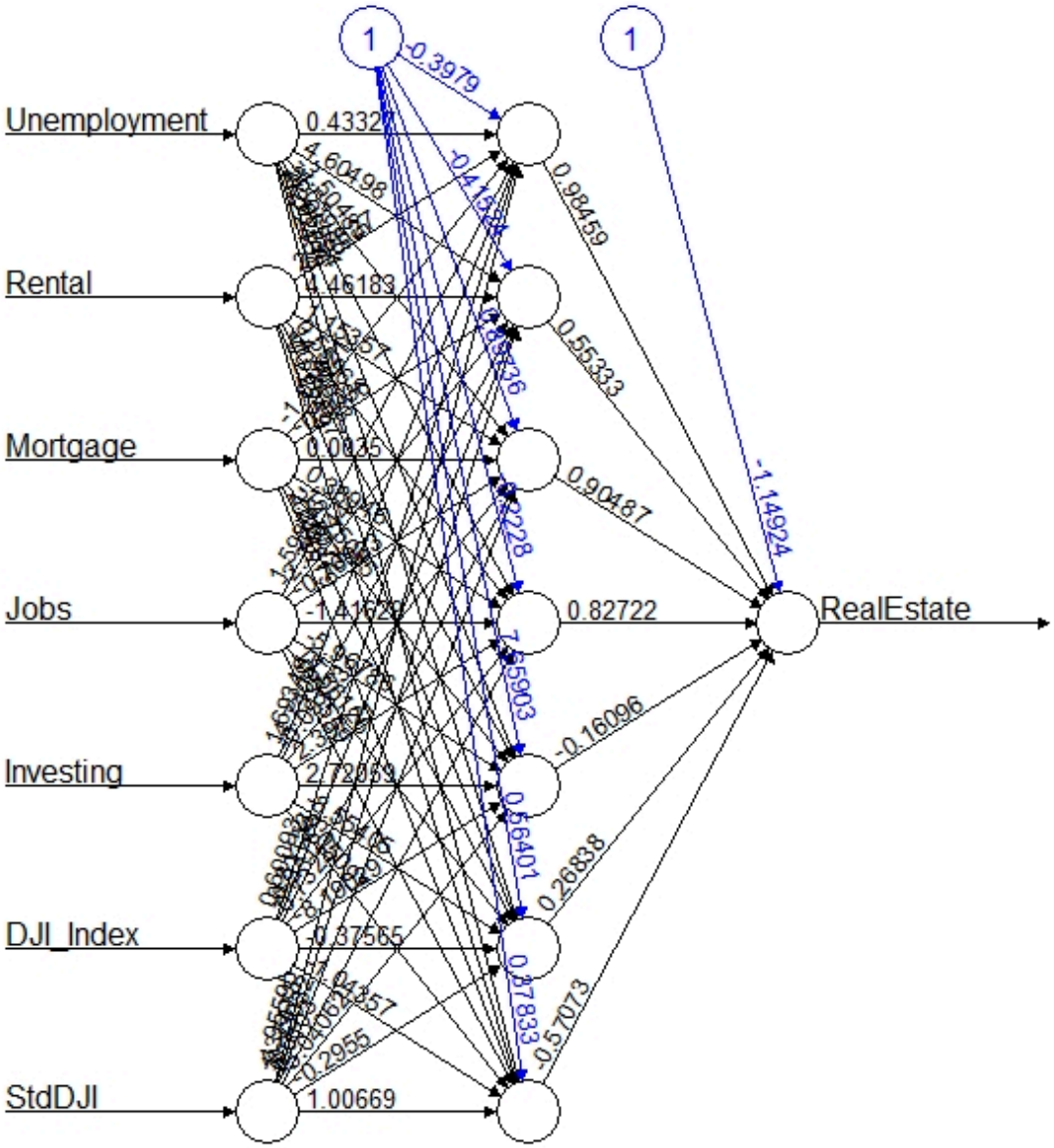
We have smaller **Error** i.e., sum of squared errors.

```
> cor(pred_results2, test_set$RealEstate)
          [,1]
[1,] 0.9852136
```

We get an even higher correlation. This is almost an ideal result! The predicted and observed **RealEstate** indices have a strong linear relationship.

Now we will put 7 hidden nodes in the NN model.

```
set.seed(123)
google.nn2<-
neuralnet(RealEstate~Unemployment+Rental+Mortgage+Jobs+Investing+DJI_Index+StdDJI,
data=training_set, hidden = 7)
plot(google.nn2)
google_pred2<-compute(google.nn2, test_set)
pred_results2<-google_pred2$net.result
cor(pred_results2, test_set$RealEstate)
r2(pred_results2, test_set$RealEstate)
```



Error: 0.21736 Steps: 8089

```
> cor(pred_results2, test_set$RealEstate)
          [,1]
[1,] 0.9904685
>
```

Adding additional layers

```
set.seed(123)
google.nn43<-
neuralnet(RealEstate~Unemployment+Rental+Mortgage+Jobs+Investing+DJI_Index+StdDJI,
data=training_set, hidden = c(4,3,3))
plot(google.nn43)
google_pred43<-compute(google.nn43, test_set)
pred_results43<-google_pred43$net.result
cor(pred_results43, test_set$RealEstate)
r2(pred_results2, test_set$RealEstate)
```

However, this enhanced neural network may complicate the interpretation of the results (or may overfit the network to intrinsic noise in the data). We see that we have not obtained an improvement in the model, we may not complicate the network.

Last modified: piątek, 20 stycznia 2023, 5:37

Platformą administruje Komisja ds. Doskonalenia Dydaktyki wraz z Centrum Informatyki Uniwersytetu Łódzkiego Więcej

Na platformie jest wykorzystywana metoda logowania za pośrednictwem Centralnego Systemu Logowania.

Studentów i pracowników Uniwersytetu Łódzkiego obowiązuje nazwa użytkownika i hasło wykorzystywane podczas logowania się do systemu USOSweb.