DASC Project Bmdataset Analysis

Alexander Bruckner and Michael Dinhof

3.11.2019

```
set.seed(5354818)
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------------------------------------

## <U+2713> ggplot2 3.2.1     <U+2713> purrr   0.3.3
## <U+2713> tibble  2.1.3     <U+2713> dplyr   0.8.3
## <U+2713> tidyr   1.0.0     <U+2713> stringr 1.4.0
## <U+2713> readr   1.3.1     <U+2713> forcats 0.4.0

## -- Conflicts ---------------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
```

Sidenote: Executing this notebook can take, depending on the hardware, up to 3 hours

Data

Here the bigmart dataset will be loaded and prepared for further operations.

Reading

Reading the data from a csv file with read_delim.

```
data <- read_delim("bigmart_train.csv", delim = ",")
```

```
## Parsed with column specification:
## cols(
##   Item_Identifier = col_character(),
##   Item_Weight = col_double(),
##   Item_Fat_Content = col_character(),
##   Item_Visibility = col_double(),
##   Item_Type = col_character(),
##   Item_MRP = col_double(),
##   Outlet_Identifier = col_character(),
##   Outlet_Establishment_Year = col_double(),
##   Outlet_Size = col_character(),
##   Outlet_Location_Type = col_character(),
##   Outlet_Type = col_character(),
##   Item_Outlet_Sales = col_double()
## )
```

Show Data and data types

```r
glimpse(data)
```

```
## Observations: 8,523
## Variables: 12
## $ Item_Identifier           <chr> "FDA15", "DRC01", "FDN15", "FDX07", "NCD19"…
## $ Item_Weight               <dbl> 9.300, 5.920, 17.500, 19.200, 8.930, 10.395…
## $ Item_Fat_Content          <chr> "Low Fat", "Regular", "Low Fat", "Regular",…
## $ Item_Visibility           <dbl> 0.016047301, 0.019278216, 0.016760075, 0.00…
## $ Item_Type                 <chr> "Dairy", "Soft Drinks", "Meat", "Fruits and…
## $ Item_MRP                  <dbl> 249.8092, 48.2692, 141.6180, 182.0950, 53.8…
## $ Outlet_Identifier         <chr> "OUT049", "OUT018", "OUT049", "OUT010", "OU…
## $ Outlet_Establishment_Year <dbl> 1999, 2009, 1999, 1998, 1987, 2009, 1987, 1…
## $ Outlet_Size               <chr> "Medium", "Medium", "Medium", NA, "High", "…
## $ Outlet_Location_Type      <chr> "Tier 1", "Tier 3", "Tier 1", "Tier 3", "Ti…
## $ Outlet_Type               <chr> "Supermarket Type1", "Supermarket Type2", "…
## $ Item_Outlet_Sales         <dbl> 3735.1380, 443.4228, 2097.2700, 732.3800, 9…
```

Structure

Changing character variables to factor

```r
data$Item_Fat_Content <- as.factor(data$Item_Fat_Content)
data$Item_Type <- as.factor(data$Item_Type)
data$Outlet_Size <- as.factor(data$Outlet_Size)
data$Outlet_Location_Type <- as.factor(data$Outlet_Location_Type)
data$Outlet_Type <- as.factor(data$Outlet_Type)
glimpse(data)
```

```
## Observations: 8,523
## Variables: 12
## $ Item_Identifier           <chr> "FDA15", "DRC01", "FDN15", "FDX07", "NCD19"…
## $ Item_Weight               <dbl> 9.300, 5.920, 17.500, 19.200, 8.930, 10.395…
## $ Item_Fat_Content          <fct> Low Fat, Regular, Low Fat, Regular, Low Fat…
## $ Item_Visibility           <dbl> 0.016047301, 0.019278216, 0.016760075, 0.00…
## $ Item_Type                 <fct> Dairy, Soft Drinks, Meat, Fruits and Vegeta…
## $ Item_MRP                  <dbl> 249.8092, 48.2692, 141.6180, 182.0950, 53.8…
## $ Outlet_Identifier         <chr> "OUT049", "OUT018", "OUT049", "OUT010", "OU…
## $ Outlet_Establishment_Year <dbl> 1999, 2009, 1999, 1998, 1987, 2009, 1987, 1…
## $ Outlet_Size               <fct> Medium, Medium, Medium, NA, High, Medium, H…
## $ Outlet_Location_Type      <fct> Tier 1, Tier 3, Tier 1, Tier 3, Tier 3, Tie…
## $ Outlet_Type               <fct> Supermarket Type1, Supermarket Type2, Super…
## $ Item_Outlet_Sales         <dbl> 3735.1380, 443.4228, 2097.2700, 732.3800, 9…
```

```r
summary(data)
```

```
##  Item_Identifier     Item_Weight     Item_Fat_Content Item_Visibility
##  Length:8523        Min.   : 4.555   LF     : 316     Min.   :0.00000
##  Class :character   1st Qu.: 8.774   low fat: 112     1st Qu.:0.02699
##  Mode  :character   Median :12.600   Low Fat:5089     Median :0.05393
##                      Mean   :12.858   reg    : 117     Mean   :0.06613
##                      3rd Qu.:16.850   Regular:2889     3rd Qu.:0.09459
##                      Max.   :21.350                    Max.   :0.32839
##                      NA's   :1463
##                   Item_Type        Item_MRP      Outlet_Identifier
##  Fruits and Vegetables:1232   Min.   : 31.29   Length:8523
##  Snack Foods          :1200   1st Qu.: 93.83   Class :character
##  Household            : 910   Median :143.01   Mode  :character
##  Frozen Foods         : 856   Mean   :140.99
##  Dairy                : 682   3rd Qu.:185.64
##  Canned               : 649   Max.   :266.89
##  (Other)              :2994
##  Outlet_Establishment_Year Outlet_Size   Outlet_Location_Type
##  Min.   :1985              High  : 932   Tier 1:2388
##  1st Qu.:1987              Medium:2793   Tier 2:2785
##  Median :1999              Small :2388   Tier 3:3350
##  Mean   :1998              NA's  :2410
##  3rd Qu.:2004
##  Max.   :2009
##
##             Outlet_Type   Item_Outlet_Sales
##  Grocery Store    :1083   Min.   :   33.29
##  Supermarket Type1:5577   1st Qu.:  834.25
##  Supermarket Type2: 928   Median : 1794.33
##  Supermarket Type3: 935   Mean   : 2181.29
##                           3rd Qu.: 3101.30
##                           Max.   :13086.97
##
```

Preperation

Reducing the levels of Item Type to make sure it does work well with nnet (the simplification has no influence on rpart or randomforest)

```
levels(data$Item_Type)
```

```
##  [1] "Baking Goods"          "Breads"                "Breakfast"
##  [4] "Canned"                "Dairy"                 "Frozen Foods"
##  [7] "Fruits and Vegetables" "Hard Drinks"           "Health and Hygiene"
## [10] "Household"             "Meat"                  "Others"
## [13] "Seafood"               "Snack Foods"           "Soft Drinks"
## [16] "Starchy Foods"
```

```
levels(data$Item_Type) <- list("Food"=c("Breads",
                                        "Breakfast",
                                        "Fruits and Vegetables",
                                        "Snack Foods",
                                        "Starchy Foods"),
                               "Non-Vegan-Food"=c("Meat",
                                                  "Seafood",
                                                  "Dairy"),
                               "Non-Fresh-Food"=c("Canned",
                                                  "Frozen Foods"),
                               "Drinks"=c("Hard Drinks", "Soft Drinks"),
                               "Others"=c("Baking Goods", "Health and Hygiene","Household","Others"))

levels(data$Item_Type)
```

```
## [1] "Food"           "Non-Vegan-Food" "Non-Fresh-Food" "Drinks"
## [5] "Others"
```

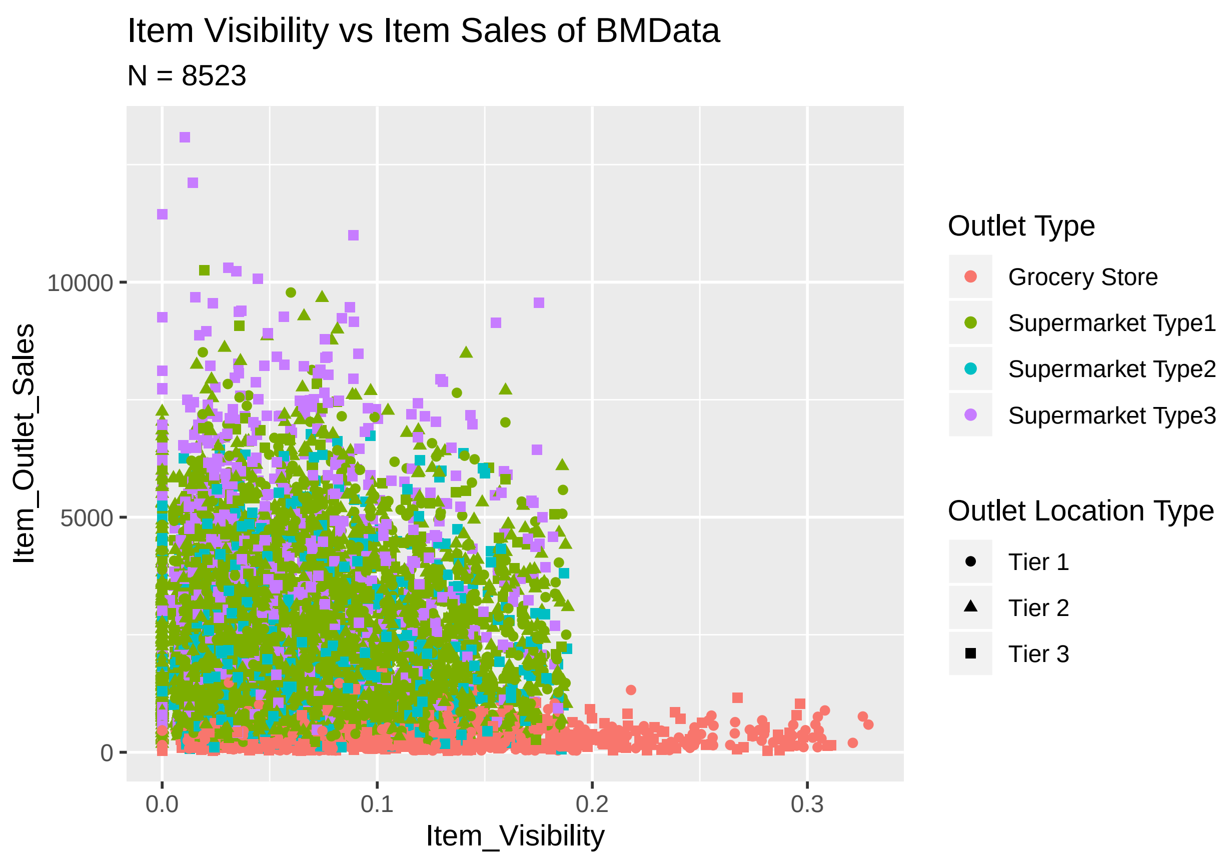
Visualization and Aggregation of the data

Visualization

Item Visibility

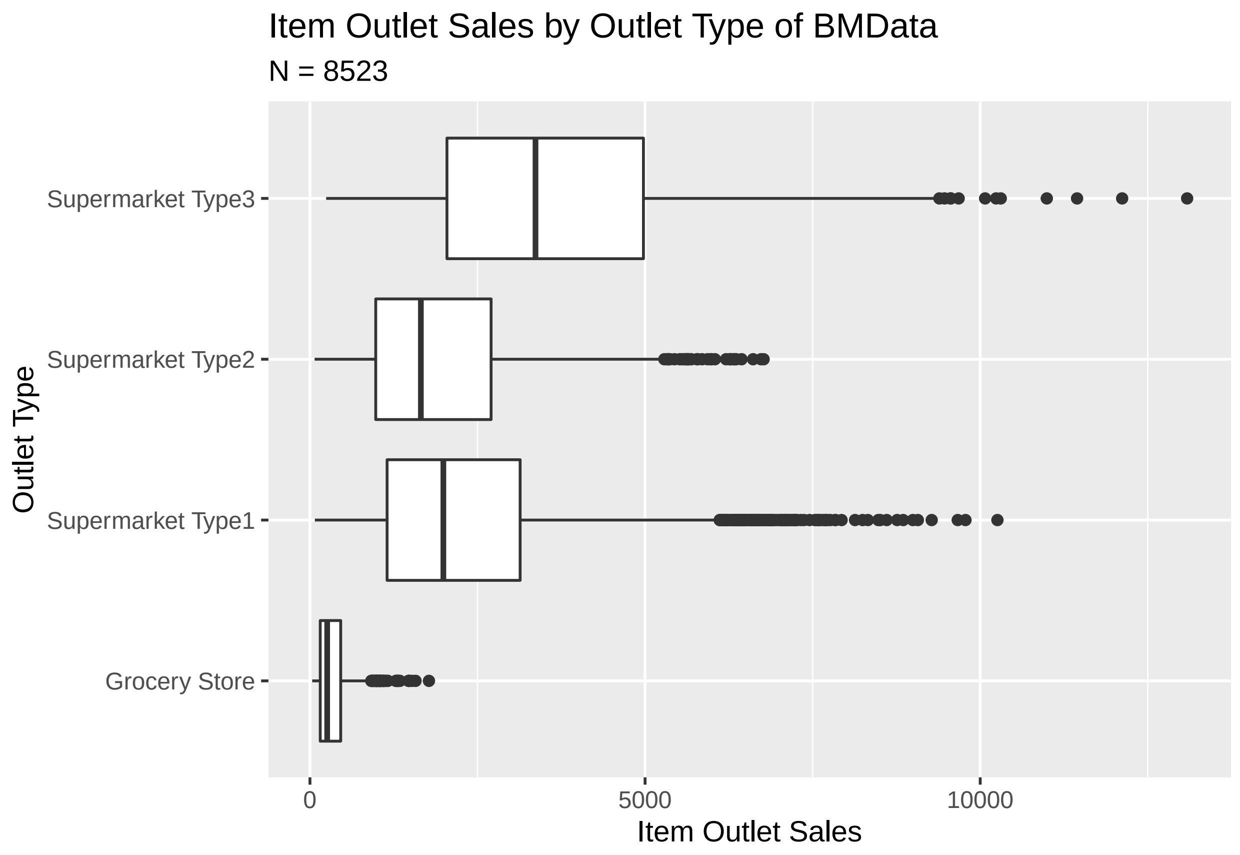
It might be interesting to relate visibility to the sales

```
g <- ggplot(data) +
  aes(x = Item_Visibility,y = Item_Outlet_Sales,shape=Outlet_Location_Type) +
  geom_point(aes(col=Outlet_Type))+
  ggtitle("Item Visibility vs Item Sales of BMData",subtitle = paste0("N = ",nrow(data))) +
  guides(col="legend") +
  scale_color_discrete(name="Outlet Type") +
  scale_shape(name="Outlet Location Type") +
  scale_fill_brewer(type = "qual",palette=6)
g
```



It is seems that the Outlet Type influences Item Outlet Sales. To further look at this a Boxplot is used

```
ggplot(data,aes(x=Outlet_Type,y=Item_Outlet_Sales))+
  geom_boxplot()+
  ggtitle("Item Outlet Sales by Outlet Type of BMData",subtitle = paste0("N = ",nrow(data))) +
  coord_flip()+
  ylab(label="Item Outlet Sales")+
  xlab(label = "Outlet Type")
```



Here it is visible that Supermarket Type 3 has a higher median and upper quartile than the other types, especially Grocery Store.

It might also be interesting to look at how Visibility influences the Sales

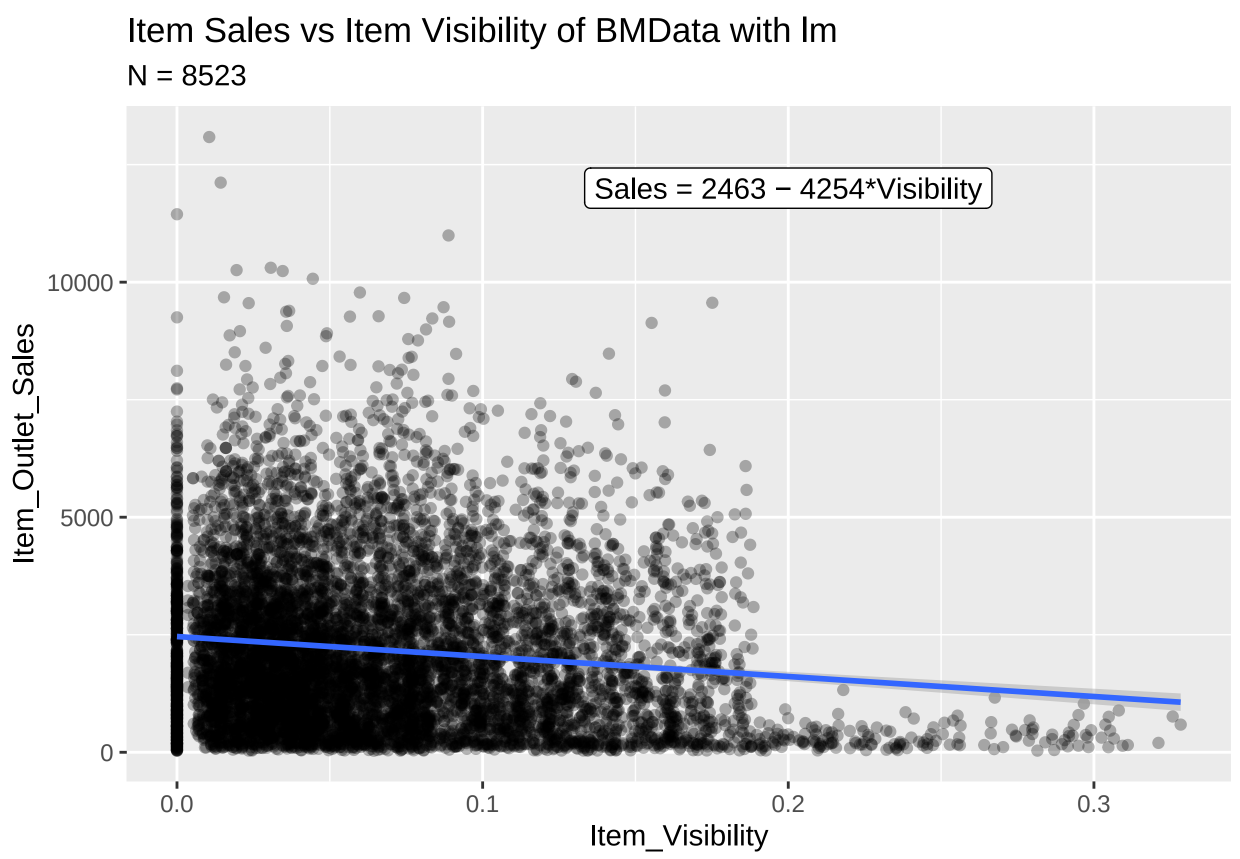
For that we calculate the linear Regression to write into the graph

```
lm(Item_Outlet_Sales ~ Item_Visibility, data=data)
```

```
##
## Call:
## lm(formula = Item_Outlet_Sales ~ Item_Visibility, data = data)
##
## Coefficients:
##     (Intercept)  Item_Visibility
##            2463            -4254
```

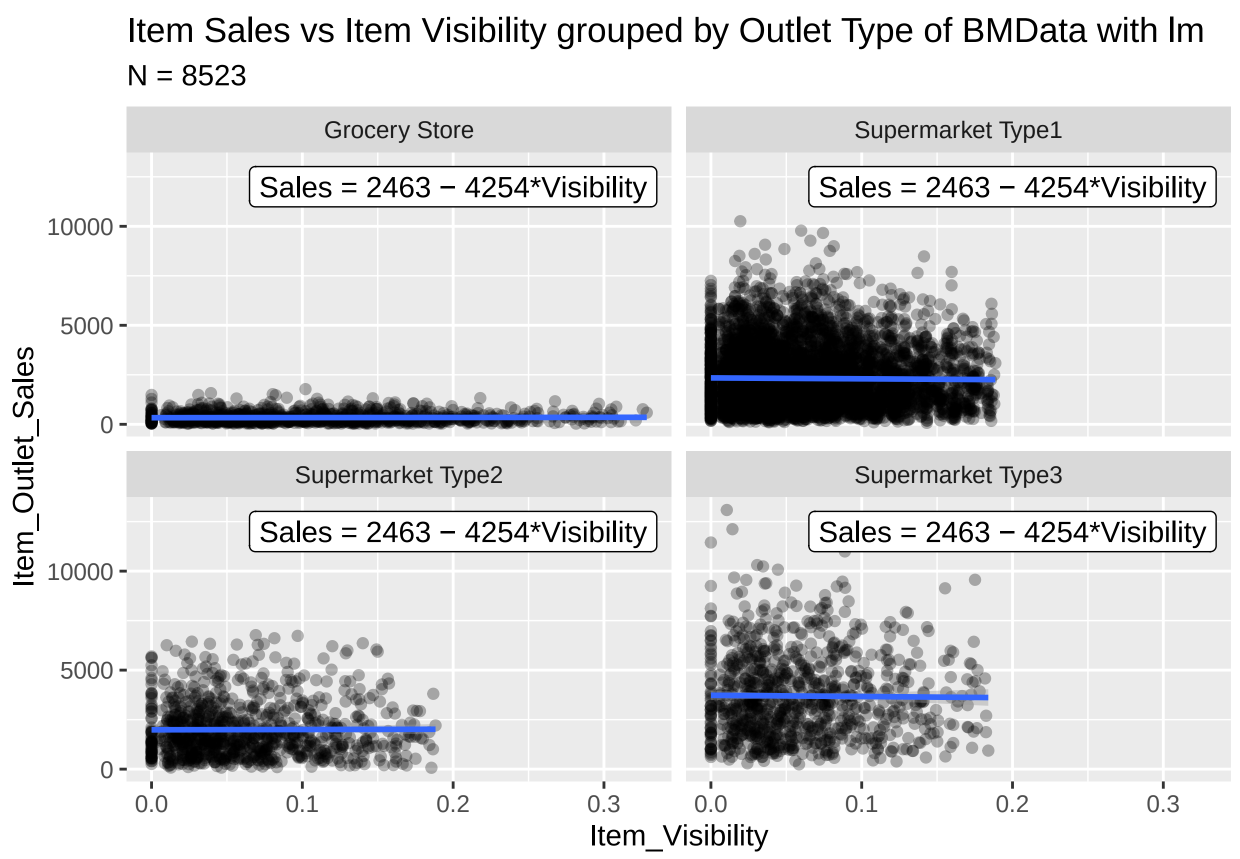
geom_point(alpha=0.3) is used to show via color where the concentration of points is higher

```
g2 <- ggplot(data,aes(x=Item_Visibility,y=Item_Outlet_Sales))+geom_point(alpha=0.3)+
  geom_smooth(method = lm) +
  annotate("label",x=0.2,y=12000,label="Sales = 2463 - 4254*Visibility") +
  ggtitle("Item Sales vs Item Visibility of BMData with lm",subtitle = paste0("N = ",nrow(data)))
g2
```



According to the data: The less visible an item is the more Sales there are for this item. The reason behind this might be because items in a Supermarket are classified as less visible than items in a grocery store. It might be better to group it by the different Outlet Types

```
g2 +
  facet_wrap(~Outlet_Type) +
  ggtitle("Item Sales vs Item Visibility grouped by Outlet Type of BMData with lm",subtitle = paste0("N
```



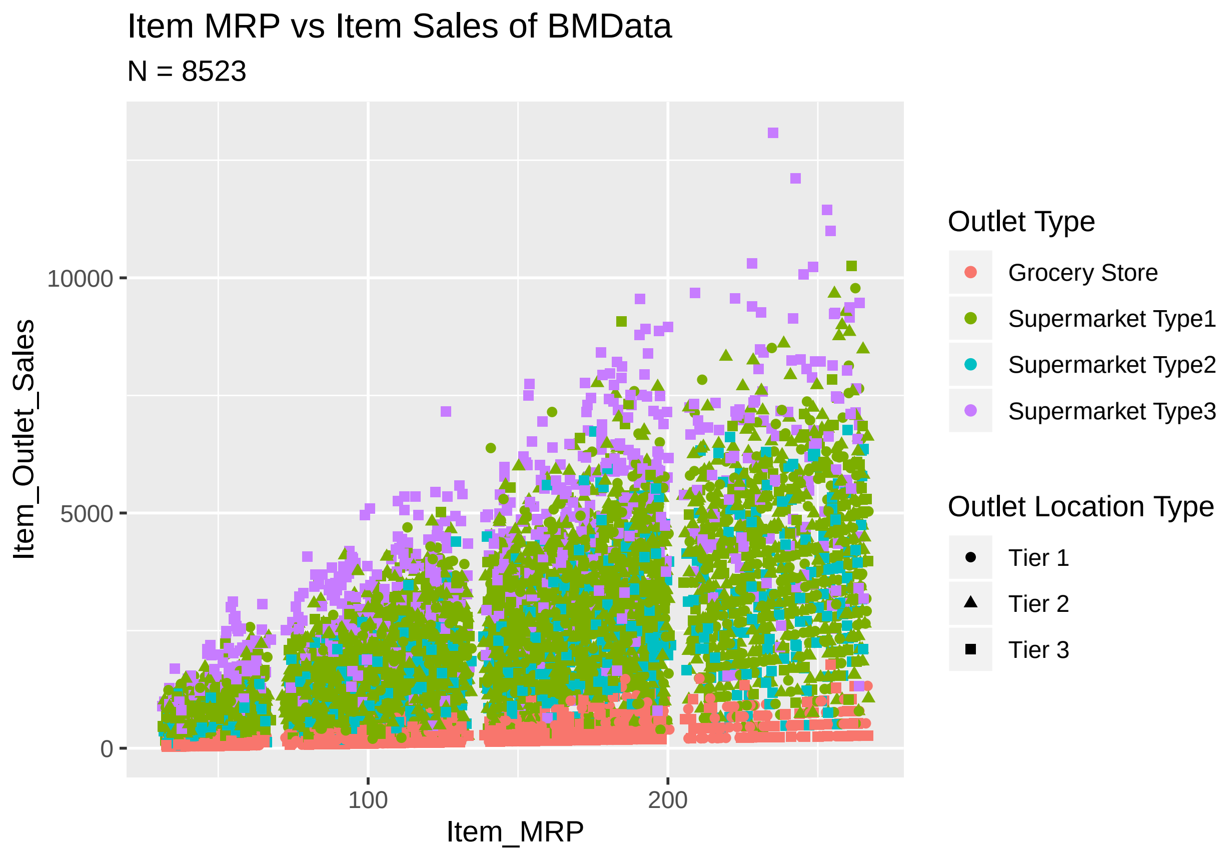
It seems that Visibility is generally a bit lower in the Supermarkets compared to the Grocery Stores: Only Grocery Stores have Items with Visibility > 0.2

It seems like Visibility does not have a strong Influence over the Sales, neither in general nor grouped by the Outlet Type

Item MRP

Now the same logic for MRP

```r
g3 <- ggplot(data) +
  aes(x = Item_MRP,y = Item_Outlet_Sales,shape=Outlet_Location_Type) +
  geom_point(aes(col=Outlet_Type))+
  ggtitle("Item MRP vs Item Sales of BMData",subtitle = paste0("N = ",nrow(data))) +
  guides(col="legend") +
  scale_color_discrete(name="Outlet Type") +
  scale_shape(name="Outlet Location Type") +
  scale_fill_brewer(type = "qual",palette=6)
g3
```



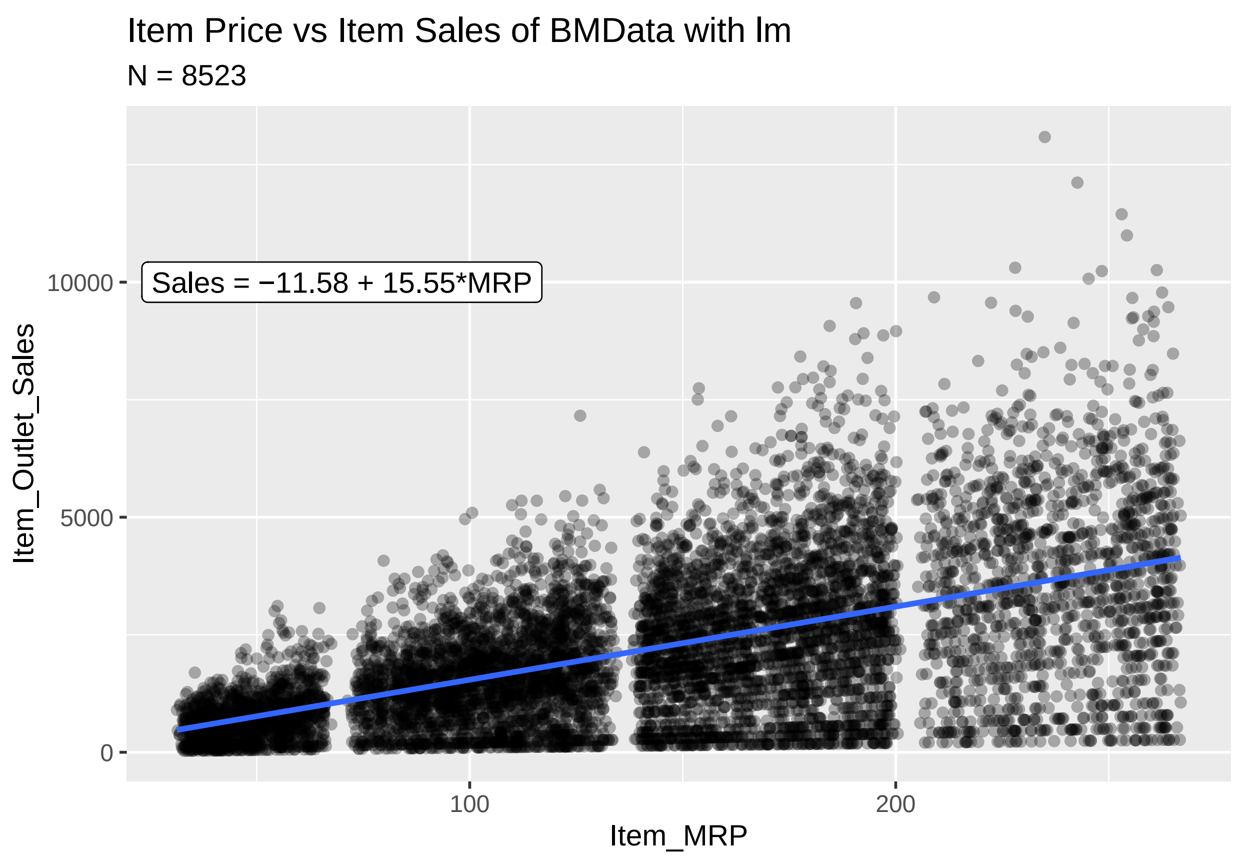
It seems that more expensive Items are sold better than cheaper items

```
lm(Item_Outlet_Sales ~ Item_MRP, data=data)
```

```
##
## Call:
## lm(formula = Item_Outlet_Sales ~ Item_MRP, data = data)
##
## Coefficients:
## (Intercept)     Item_MRP
##      -11.58        15.55
```

geom_point(alpha=0.3) is used to show via color where the concentration of points is higher

```
g4 <- ggplot(data,aes(x=Item_MRP,y=Item_Outlet_Sales))+geom_point(alpha=0.3)+
  geom_smooth(method = lm) +
  annotate("label",x=70,y=10000,label="Sales = -11.58 + 15.55*MRP") +
  ggtitle("Item Price vs Item Sales of BMData with lm",subtitle = paste0("N = ",nrow(data)))
g4
```



According to the data: The more expensive an item is the more Sales there are for this item. The reason behind this might be brand articles and the assumed quality of a higher price. Another reason might be, that there are more low-priced items than high-priced ones. So the sales can be more distributed in the cheaper section, compared to the more expensive one.

Aggregation of Data

```r
data %>%
  group_by(Outlet_Type) %>%
  summarise(avg_sales=mean(Item_Outlet_Sales),
            med_sales=median(Item_Outlet_Sales),
            max_sales=max(Item_Outlet_Sales),
            min_sales = min(Item_Outlet_Sales),
            avg_mrp = mean(Item_MRP),
            med_mrp = median(Item_MRP),
            N=n())
```

```
## # A tibble: 4 x 8
##   Outlet_Type      avg_sales med_sales max_sales min_sales avg_mrp med_mrp     N
##   <fct>                <dbl>     <dbl>     <dbl>     <dbl>   <dbl>   <dbl> <int>
## 1 Grocery Store         340.      257.     1776.      33.3    140.    144.  1083
## 2 Supermarket Typ…     2316.     1991.    10257.      73.2    141.    143.  5577
## 3 Supermarket Typ…     1995.     1655.     6769.      69.2    142.    141.   928
## 4 Supermarket Typ…     3694.     3365.    13087.     242.     140.    144.   935
```

It seems that the Outlet Type has influence on the Number of Sales, especially Grocery Store has a lower average and median. There is practically no difference in MRP.

The most observations are from Supermarket Type1

```r
data %>%
  group_by(Outlet_Location_Type) %>%
  summarise(avg_sales=mean(Item_Outlet_Sales),
            med_sales=median(Item_Outlet_Sales),
            max_sales=max(Item_Outlet_Sales),
            min_sales = min(Item_Outlet_Sales),
            avg_mrp = mean(Item_MRP),
            med_mrp = median(Item_MRP),
            N=n())
```

```
## # A tibble: 3 x 8
##   Outlet_Location… avg_sales med_sales max_sales min_sales avg_mrp med_mrp     N
##   <fct>                <dbl>     <dbl>     <dbl>     <dbl>   <dbl>   <dbl> <int>
## 1 Tier 1               1877.     1487.     9780.      34.0    141.    143.  2388
## 2 Tier 2               2324.     2004.     9665.      99.9    141.    143.  2785
## 3 Tier 3               2280.     1812.    13087.      33.3    141.    142.  3350
```

Outlet Location Type: Tier 1 has a lower average and median than the other two, while Tier 3 has a much higher maximum than the other 2. There is practically no difference in MRP.

```r
data %>%
  group_by(Outlet_Type,Outlet_Location_Type) %>%
  summarise(avg_sales=mean(Item_Outlet_Sales),
            med_sales=median(Item_Outlet_Sales),
            max_sales=max(Item_Outlet_Sales),
            min_sales = min(Item_Outlet_Sales),
            avg_mrp = mean(Item_MRP),
            med_mrp = median(Item_MRP),
            N=n())
```

```
## # A tibble: 7 x 9
## # Groups:   Outlet_Type [4]
##   Outlet_Type Outlet_Location… avg_sales med_sales max_sales min_sales avg_mrp
##   <fct>       <fct>                <dbl>     <dbl>     <dbl>     <dbl>   <dbl>
## 1 Grocery St… Tier 1                340.      265.     1482.      34.0    140.
## 2 Grocery St… Tier 3                339.      250.     1776.      33.3    141.
## 3 Supermarke… Tier 1               2313.     1959.     9780.     102.     141.
## 4 Supermarke… Tier 2               2324.     2004.     9665.      99.9    141.
## 5 Supermarke… Tier 3               2299.     2051.    10257.      73.2    141.
## 6 Supermarke… Tier 3               1995.     1655.     6769.      69.2    142.
## 7 Supermarke… Tier 3               3694.     3365.    13087.     242.     140.
## # … with 2 more variables: med_mrp <dbl>, N <int>
```

Outlet Type seems to be a stronger influence on Sales compared to Outlet Location Type. There is practically no difference in MRP.

Machine Learning

Loading necessary libraries

```
library(randomForest)
library("nnet")
library("keras")
library(class)
library("caret")
library("e1071")
library("rpart")
```

In this section we attempt to predict the value for Item Outlet Sales. As Input the model uses most of the other values (except the IDs).

Different Models (RandomForest, RPart, NNet) are tuned and the MSE is calculated and compared. The best model will be used for further excersises.

Structuring Data

Removing Item and Store Identifier since these are IDs and would not be beneficial for the Machine Learning Algorithms

And removing NA-Rows, NNet can not work with NAs

```
head(data)
```

```
## # A tibble: 6 x 12
##   Item_Identifier Item_Weight Item_Fat_Content Item_Visibility Item_Type
##   <chr>                 <dbl> <fct>                      <dbl> <fct>
## 1 FDA15                  9.3  Low Fat                   0.0160 Non-Vega…
## 2 DRC01                  5.92 Regular                   0.0193 Drinks
## 3 FDN15                 17.5  Low Fat                   0.0168 Non-Vega…
## 4 FDX07                 19.2  Regular                   0      Food
## 5 NCD19                  8.93 Low Fat                   0      Others
## 6 FDP36                 10.4  Regular                   0      Others
## # … with 7 more variables: Item_MRP <dbl>, Outlet_Identifier <chr>,
## #   Outlet_Establishment_Year <dbl>, Outlet_Size <fct>,
## #   Outlet_Location_Type <fct>, Outlet_Type <fct>, Item_Outlet_Sales <dbl>
```

```
mlData <- select(data, -Item_Identifier, -Outlet_Identifier)
mlData <- na.omit(mlData)
nrow(mlData)
```

```
## [1] 4650
```

Preparing test and train data

1. Taking random 2/3 for train-data and 1/3 for test-data

2. Scaling the numerical data

3. Encoding data in seperate variable for tensorflow

```
N = nrow(mlData)
train_ind = sample(1:N, size = N*2/3)
train = mlData[train_ind,] #zufällige 2/3 fürs trainieren abspeichern (erst ab hier skalieren, nicht vo
test = mlData[-train_ind,] #zufällige 1/3 fürs testen abspeichern
nrow(train)
```

```
## [1] 3100
```

```
pp = preProcess(train, method = c("center","scale"))

train_scaled = predict(pp, train)
test_scaled = predict(pp, test)
str(test_scaled)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    1550 obs. of  10 variables:
##  $ Item_Weight              : num  -0.77 0.163 1.203 0.474 -0.835 ...
##  $ Item_Fat_Content         : Factor w/ 5 levels "LF","low fat",..: 3 5 5 5 5 3 5 5 3 3 ...
##  $ Item_Visibility          : num  -0.995 -1.069 -0.335 0.888 0.194 ...
##  $ Item_Type                : Factor w/ 5 levels "Food","Non-Vegan-Food",..: 2 1 2 1 1 5 1 3 5 3 ...
##  $ Item_MRP                 : num  1.7311 -1.3351 0.0444 0.0663 -1.3558 ...
##  $ Outlet_Establishment_Year: num  -0.028 -1.654 -0.299 -0.028 -0.299 ...
##  $ Outlet_Size              : Factor w/ 3 levels "High","Medium",..: 2 1 3 2 3 2 2 3 2 2 ...
##  $ Outlet_Location_Type     : Factor w/ 3 levels "Tier 1","Tier 2",..: 1 3 1 1 1 3 1 1 1 3 ...
##  $ Outlet_Type              : Factor w/ 4 levels "Grocery Store",..: 2 2 2 2 2 3 2 2 2 3 ...
##  $ Item_Outlet_Sales        : num  0.9865 -1.2749 -0.0456 -0.4443 -0.4723 ...
##  - attr(*, "na.action")= 'omit' Named int  4 8 9 10 19 22 24 26 29 30 ...
##   ..- attr(*, "names")= chr  "4" "8" "9" "10" ...
```

```
encoder = dummyVars(~Item_Fat_Content+Item_Type+Outlet_Size+Outlet_Location_Type+Outlet_Type,data=train_

encoded_train = as.data.frame(predict(encoder,train_scaled))
encoded_train = cbind(encoded_train,train_scaled[,c(1,3,5,6)])

encoded_test = as.data.frame(predict(encoder,test_scaled))
encoded_test = cbind(encoded_test,test_scaled[,c(1,3,5,6)])

str(encoded_train)
```

```
## 'data.frame':    3100 obs. of  19 variables:
##  $ Item_Fat_Content.low fat     : num  0 0 0 0 1 0 0 0 0 0 ...
##  $ Item_Fat_Content.Low Fat     : num  0 0 0 0 0 1 1 1 1 1 ...
##  $ Item_Fat_Content.reg         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Item_Fat_Content.Regular     : num  0 1 1 1 0 0 0 0 0 0 ...
##  $ Item_Type.Non-Vegan-Food     : num  0 1 0 0 0 0 0 1 1 0 ...
##  $ Item_Type.Non-Fresh-Food     : num  0 0 1 0 0 0 0 0 0 0 ...
##  $ Item_Type.Drinks             : num  0 0 0 0 1 0 0 0 0 0 ...
##  $ Item_Type.Others             : num  0 0 0 1 0 0 0 0 0 1 ...
##  $ Outlet_Size.Medium           : num  1 0 0 1 1 1 0 1 0 0 ...
##  $ Outlet_Size.Small            : num  0 1 0 0 0 0 1 0 1 1 ...
##  $ Outlet_Location_Type.Tier 2  : num  0 1 0 0 0 0 0 0 0 0 ...
##  $ Outlet_Location_Type.Tier 3  : num  1 0 1 1 1 0 0 0 0 0 ...
##  $ Outlet_Type.Supermarket Type1: num  0 1 1 0 0 1 1 1 1 1 ...
##  $ Outlet_Type.Supermarket Type2: num  1 0 0 1 1 0 0 0 0 0 ...
##  $ Outlet_Type.Supermarket Type3: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Item_Weight                  : num  -0.492 -1.495 -0.556 -1.471 -1.449 ...
##  $ Item_Visibility              : num  -1.105 0.808 -0.233 0.126 1.235 ...
##  $ Item_MRP                     : num  -1.572 -1.556 -0.977 0.74 0.316 ...
##  $ Outlet_Establishment_Year    : num  1.33 0.65 -1.65 1.33 1.33 ...
```

```
str(encoded_test)
```

```
## 'data.frame':    1550 obs. of  19 variables:
##  $ Item_Fat_Content.low fat     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Item_Fat_Content.Low Fat     : num  1 0 0 0 0 1 0 0 1 1 ...
##  $ Item_Fat_Content.reg         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Item_Fat_Content.Regular     : num  0 1 1 1 1 0 1 1 0 0 ...
##  $ Item_Type.Non-Vegan-Food     : num  1 0 1 0 0 0 0 0 0 0 ...
##  $ Item_Type.Non-Fresh-Food     : num  0 0 0 0 0 0 0 1 0 1 ...
##  $ Item_Type.Drinks             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Item_Type.Others             : num  0 0 0 0 0 1 0 0 1 0 ...
##  $ Outlet_Size.Medium           : num  1 0 0 1 0 1 1 0 1 1 ...
##  $ Outlet_Size.Small            : num  0 0 1 0 1 0 0 1 0 0 ...
##  $ Outlet_Location_Type.Tier 2  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Outlet_Location_Type.Tier 3  : num  0 1 0 0 0 1 0 0 0 1 ...
##  $ Outlet_Type.Supermarket Type1: num  1 1 1 1 1 0 1 1 1 0 ...
##  $ Outlet_Type.Supermarket Type2: num  0 0 0 0 0 1 0 0 0 1 ...
##  $ Outlet_Type.Supermarket Type3: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Item_Weight                  : num  -0.77 0.163 1.203 0.474 -0.835 ...
##  $ Item_Visibility              : num  -0.995 -1.069 -0.335 0.888 0.194 ...
##  $ Item_MRP                     : num  1.7311 -1.3351 0.0444 0.0663 -1.3558 ...
##  $ Outlet_Establishment_Year    : num  -0.028 -1.654 -0.299 -0.028 -0.299 ...
```

Tuned models

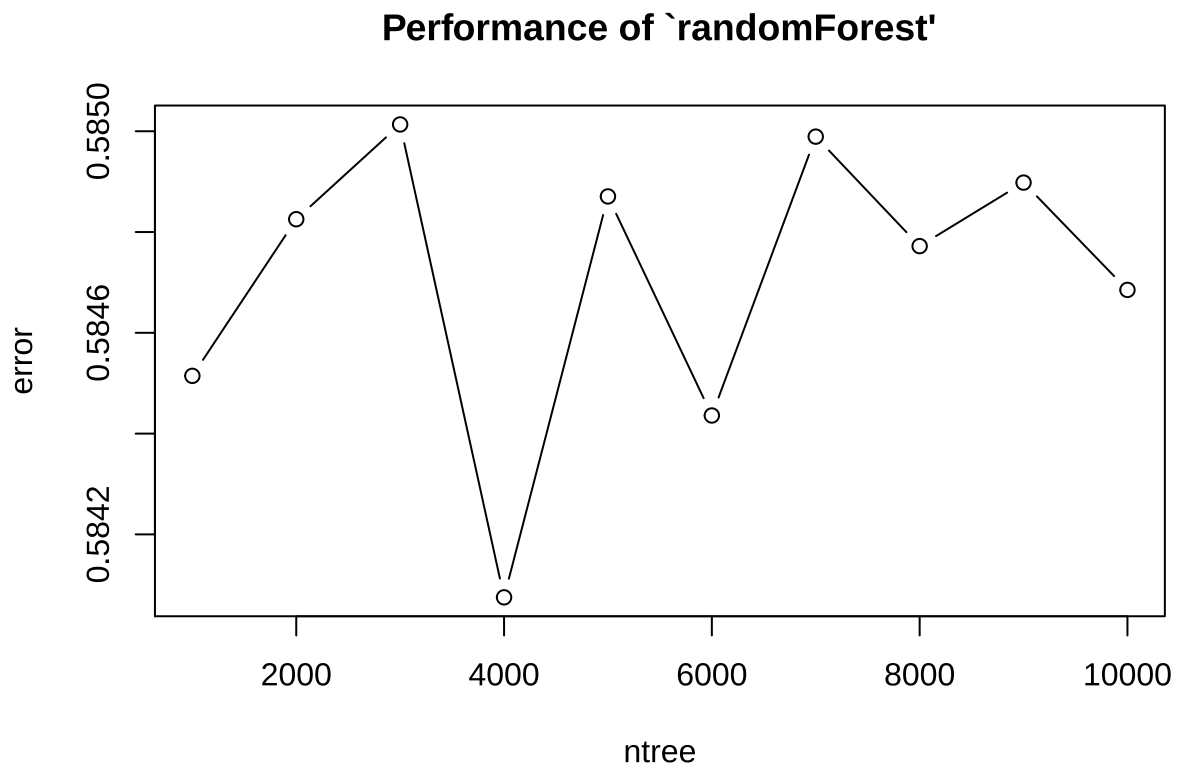
RandomForest

Tuning Randomforest and showing result

```
rforest_tuned = tune.randomForest(Item_Outlet_Sales ~., data = train_scaled, ntree = 1:10*1000 ,tunecon
rforest_tuned
```

```
##
## Parameter tuning of 'randomForest':
##
## - sampling method: fixed training/validation set
##
## - best parameters:
##  ntree
##   4000
##
## - best performance: 0.5840751
```

```r
plot(rforest_tuned)
```



RandomForest is best at 4000 Trees

Nnet

Tuning NNet

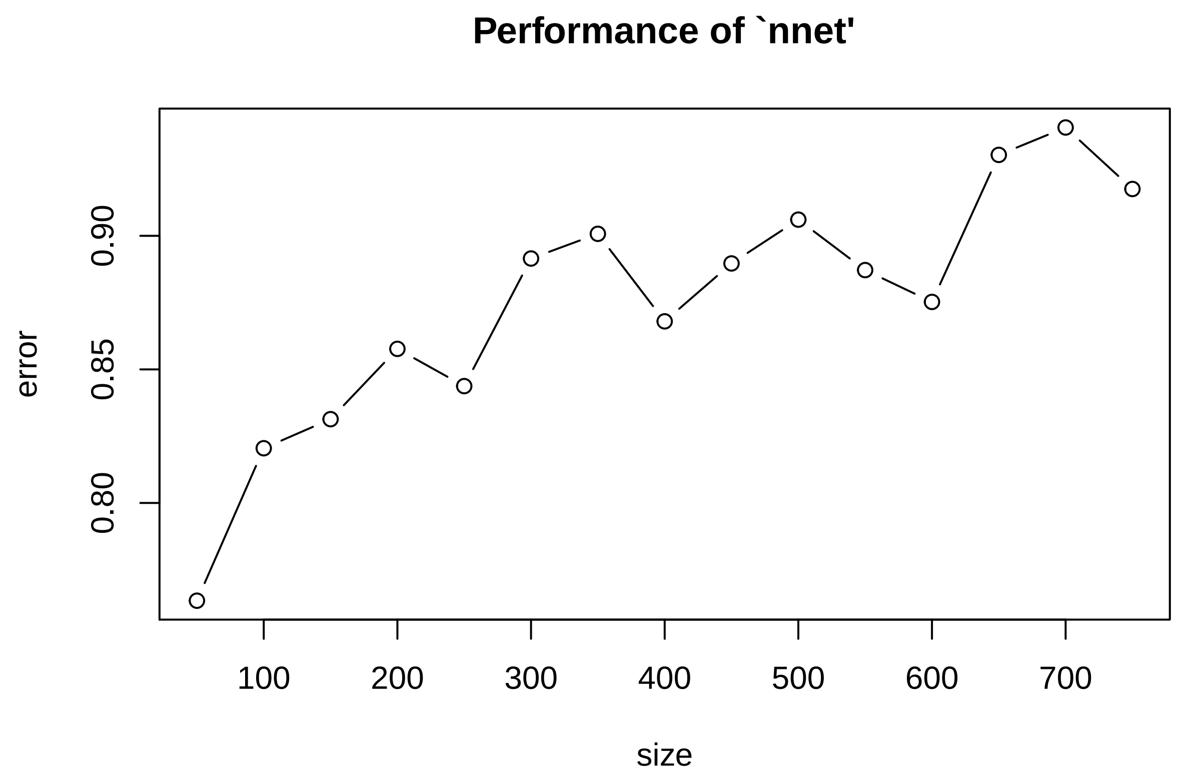
```r
nnet_tuned = tune.nnet(Item_Outlet_Sales ~ ., data = train_scaled, size = 1:15*50, tunecontrol = tune.c
nnet_tuned
```

```
##
## Parameter tuning of 'nnet':
##
## - sampling method: fixed training/validation set
##
## - best parameters:
##  size
##    50
##
## - best performance: 0.7634187
```

```r
nnet_tuned$best.model
```

```
## a 19-50-1 network with 1051 weights
## inputs: Item_Weight Item_Fat_Contentlow fat Item_Fat_ContentLow Fat Item_Fat_Contentreg Item_Fat_Con
## output(s): Item_Outlet_Sales
## options were - linear output units
```

```
plot(nnet_tuned)
```



NNet is best at 50 Nodes

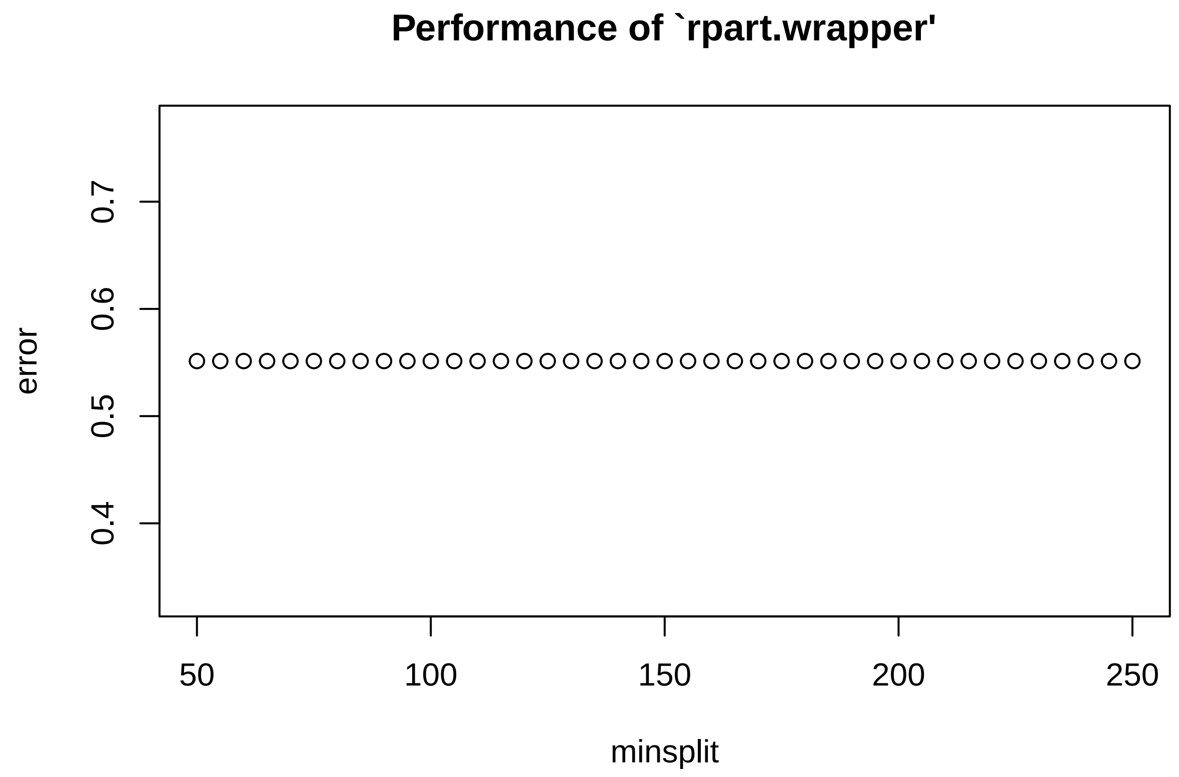
RPart

Tuning RPart

```
rpart_tuned = tune.rpart(Item_Outlet_Sales ~ ., data = train_scaled, minsplit = 10:50*5, tunecontrol = 
rpart_tuned
```

```
##
## Parameter tuning of 'rpart.wrapper':
##
## - sampling method: fixed training/validation set
##
## - best parameters:
##  minsplit
##        50
##
## - best performance: 0.5513611
```

```r
plot(rpart_tuned)
```



Changing Minsplit does not worsen or improve the process

Predict Test Data

Predicting test data

```r
fitted_rf = predict(rforest_tuned$best.model, test_scaled)
fitted_nnet = predict(nnet_tuned$best.model, test_scaled)
fitted_rpart = predict(rpart_tuned$best.model, test_scaled)
```

MSE with scaled test data

Calculating Mean Squared Error with a function to show the error of the models and find the best one

```r
mse <- function(real,pred) {
  return(mean((real-pred)^2))
}

cm_rf <- mse(fitted_rf, test_scaled$Item_Outlet_Sales)
cm_rpart <- mse(fitted_rpart, test_scaled$Item_Outlet_Sales)
cm_nnet <- mse(fitted_nnet, test_scaled$Item_Outlet_Sales)
```

MSE of Random Forest

```r
cm_rf
```

```
## [1] 0.5682681
```

MSE of NNet

```r
cm_nnet
```

```
## [1] 0.706763
```

MSE of RPart

```r
cm_rpart
```

```
## [1] 0.5866074
```

Model to choose

The Mean squared error of RandomForest is the lowest with 0.57, NNet has a msq of 0.99 and RPart has 0.59. The best model is RandomForest.

```r
importance(rforest_tuned$best.model)
```

```
##                            IncNodePurity
## Item_Weight                    309.22428
## Item_Fat_Content                75.47039
## Item_Visibility                324.23787
## Item_Type                      116.56007
## Item_MRP                      1668.40833
## Outlet_Establishment_Year       49.63577
## Outlet_Size                     35.72257
## Outlet_Location_Type            36.79250
## Outlet_Type                     13.34441
```

According to RandomForest the most important variable is Item_MRP (Price) while the Outlet Type is not as important.

Predicting some values of the testdata, to do this the predicted values need to be put in the same dataframe as the testdata was and unscaled (this procedure is also used in api server.R). Then it can be compared to unscaled real values.

```
unPreProc <- function(preProc, data){
  stopifnot(class(preProc) == "preProcess")
  stopifnot(class(data) == "data.frame")
  for(i in names(preProc$mean)){
    tmp <- data[, i] * preProc$std[[i]] + preProc$mean[[i]]
    data[, i] <- tmp
  }
  return(data)
}

predictions <- cbind(test_scaled[1:5,1:9],Item_Outlet_Sales=predict(rforest_tuned$best.model, test_scal

comparison <- cbind(real=head(test["Item_Outlet_Sales"],5),predicted=unPreProc(pp,predictions)["Item_Ou
colnames(comparison)<-c("real","pred","abs_diff")
comparison
```

```
##        real      pred abs_diff
## 1 3735.1380 3474.1673 260.9707
## 2  343.5528  945.2412 601.6884
## 3 2187.1530 2083.0434 104.1096
## 4 1589.2646 2383.6667 794.4021
## 5 1547.3192  851.8368 695.4824
```

As can be seen by these examples the difference is quite high. This is to be expected because of the high MSE Value

Deploying the best Model as API-Function

Saving the RandomForest model, the PreProcess for scaling new data and the scaled data to use in an API

PreProcess is necessary to scale the data given to the API

The Scaled Data is necessary to ensure that the data given to the API has the same Levels as the test data, otherwise it does not work

```
model <- rforest_tuned$best.model
save(model,file="model.rda")
save(pp,file="preprocess.rda")
save(train_scaled,file="df_for_levels.rda")
```

The API Logic is in the server.R file. The Client for testing in client.R.

The client sends a dataframe as JSON String to the Server. The server does the following:

1. creates out of that a dataframe again
2. scales it
3. predicts the value for Item Sales
4. unscales it
5. sends it back to the client as String

It is important that the client adds a value for Sales Output to the data he sends, because it is needed for the predict function, as can be seen in the importance-function above, it is not used in the model.

Special Feature

As a special feature shiny is used to create an interactive Visualization of the data. The files are in the folder "shiny".

In this Interface the different visualizations can be shown with a chosen number of random observations to see a less busy graphic than in the ones in this report.

Files

Dashboard - Presentation of the project, needs various jpg files that are included in the dashboard

shiny - The files in the shiny directory are the server and the ui file for shiny

API - The files in the API directory are the server, the client and files that need to be loaded by the server to work, these files and server.R need to be in the working directory in which the server is executed, or the paths in the server.R and start_server.R files need to be adapted

project_v3.Rmd/pdf - The main project