IBM Watson Analytics - Marketing Data Analysis

Clemens Holzkorn

August 19, 2018

Before I begin with the Data Analysis and Prediction, I have to load the dependencies.

```
library(tidyverse)
library(xgboost)
library(FNN)
library(rpart)
library(cluster)
library(randomForest)
library(modelr)
library(caret)
library(rmarkdown)
library(knitr)
```

Data Analysis

Loading the data

The first step in any data project is, of course, loading the data.

```
df <- read_csv("C:/Users/Clem/OneDrive/Informatik_Lernen/Statistics/SampleProjects/Prediction_Marketing
```

```
## Parsed with column specification:
## cols(
##   MarketID = col_integer(),
##   MarketSize = col_character(),
##   LocationID = col_integer(),
##   AgeOfStore = col_integer(),
##   Promotion = col_integer(),
##   week = col_integer(),
##   SalesInThousands = col_double()
## )
```

```
df
```

```
## # A tibble: 548 x 7
##    MarketID MarketSize LocationID AgeOfStore Promotion  week
##       <int> <chr>           <int>      <int>     <int> <int>
##  1        1 Medium              1          4         3     1
##  2        1 Medium              1          4         3     2
##  3        1 Medium              1          4         3     3
##  4        1 Medium              1          4         3     4
##  5        1 Medium              2          5         2     1
##  6        1 Medium              2          5         2     2
##  7        1 Medium              2          5         2     3
##  8        1 Medium              2          5         2     4
##  9        1 Medium              3         12         1     1
## 10        1 Medium              3         12         1     2
## # ... with 538 more rows, and 1 more variable: SalesInThousands <dbl>
```

Data Tansformation

After successfully importing the data without producing NAs, I convert the variables in our dataset to meaningful types, which may be useful for further analysis.

MarketID and LocationID are arbitrary numbers and do not make a lot of sense as predictors. MarketSize, the type of promotion as well as the week make sense as factors, since week only takes on 4 discrete values.

```
unique(df$week)
```

```
## [1] 1 2 3 4
```

```
df$MarketSize <- as.factor(df$MarketSize)
df$Promotion <- as.factor(df$Promotion)
df$week <- as.factor(df$week)
```

AgeOfStore takes on many different values and therefore makes sense as a number.

```
length(unique(df$AgeOfStore))
```

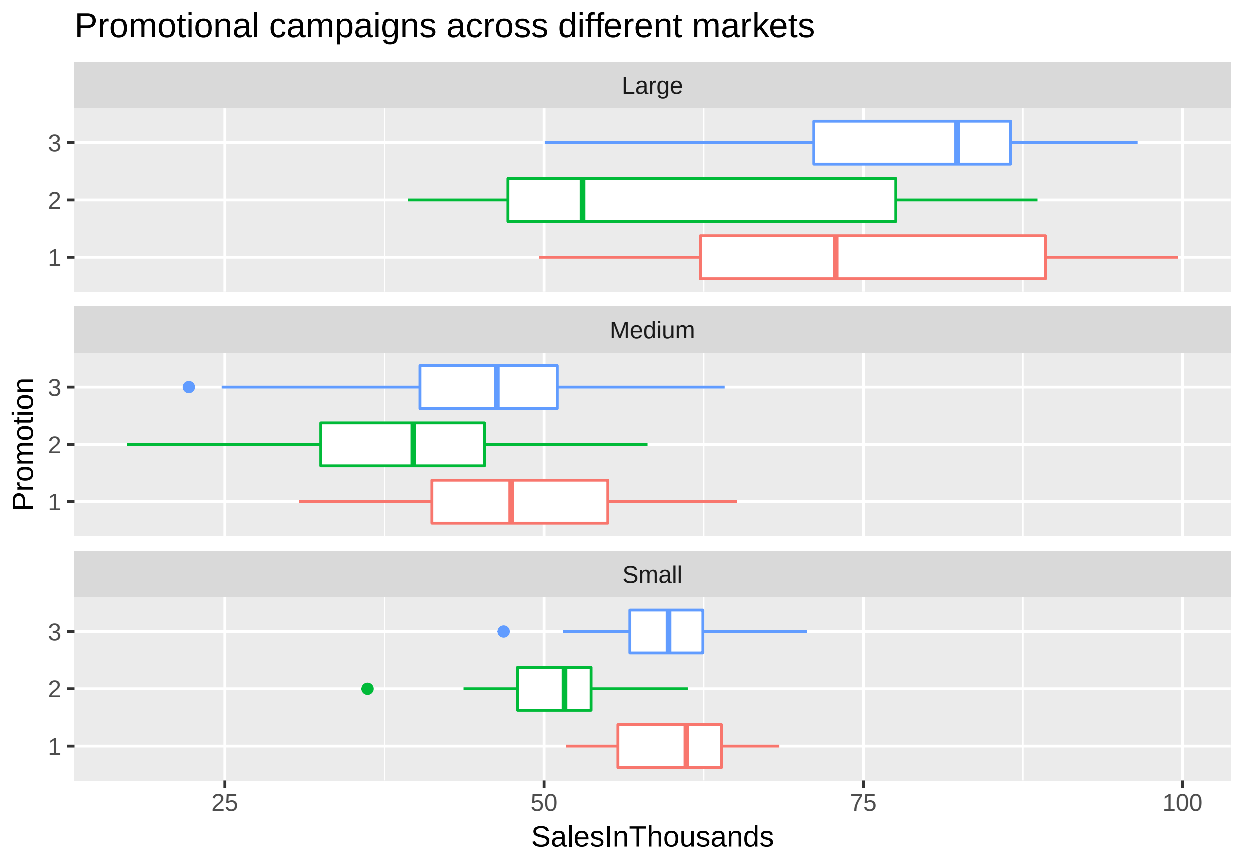
```
## [1] 25
```

Data Exploration

Since we are interested in the prediction of sales, we want to plot the different variables against the sales. To get a quick overview, we can use facets, which allow multiple plots on the same axes.

In our first facet, we see that promotions tend to work equally in different markets, but Promotion number 2 seems to perform worse across all markets.

```
ggplot(data = df) +
  geom_boxplot(mapping = aes(x = Promotion, y = SalesInThousands, color = Promotion)) +
  facet_wrap(~ MarketSize, nrow = nlevels(df$MarketSize)) +
  theme(legend.position="none") +
  coord_flip() +
  ggtitle("Promotional campaigns across different markets")
```



This pattern needs further investigation. To feed our suspicion with information, we perform a t-test for difference in means between promotional campaign 2 and the other campaigns.

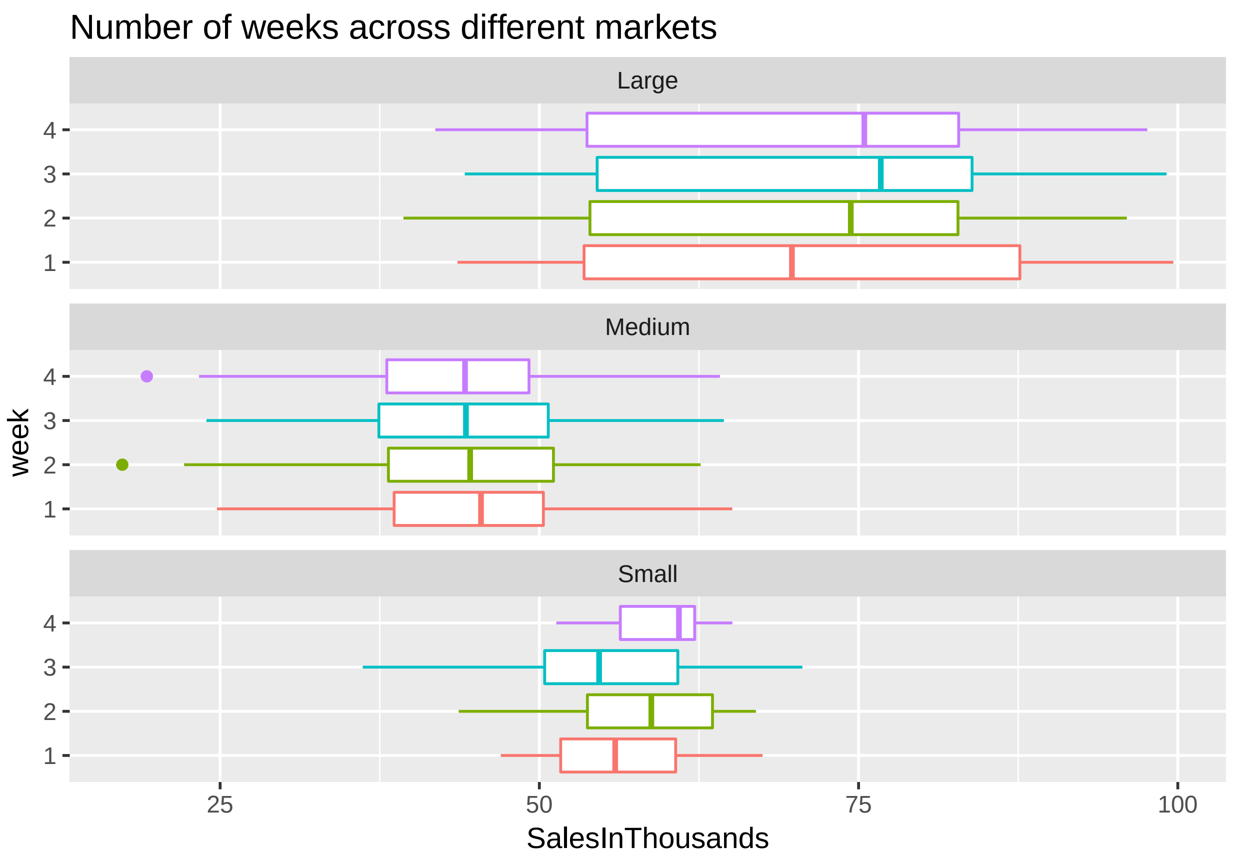
We may reject the null hypothesis of equal means on all practical levels of significance, concluding that campaign 2 does indeed perform worse than the other two campaigns.

```
grouped_promotion <- df$Promotion
grouped_promotion[grouped_promotion == "3"] <- "1"

prom_meantest <- as.tibble(cbind(df$SalesInThousands, grouped_promotion))
prom_meantest$grouped_promotion <- as.factor(prom_meantest$grouped_promotion)

t.test(V1 ~ grouped_promotion, data = prom_meantest)
```

```
##
##  Welch Two Sample t-test
##
## data:  V1 by grouped_promotion
## t = 6.6241, df = 413.92, p-value = 1.089e-10
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   6.56944 12.11367
## sample estimates:
## mean in group 1 mean in group 2
##        56.67097        47.32941
```

In the another facet, we see that the influence of weeks on Sales might be very small. Only the sales in small markets could be affected by the influence of weeks

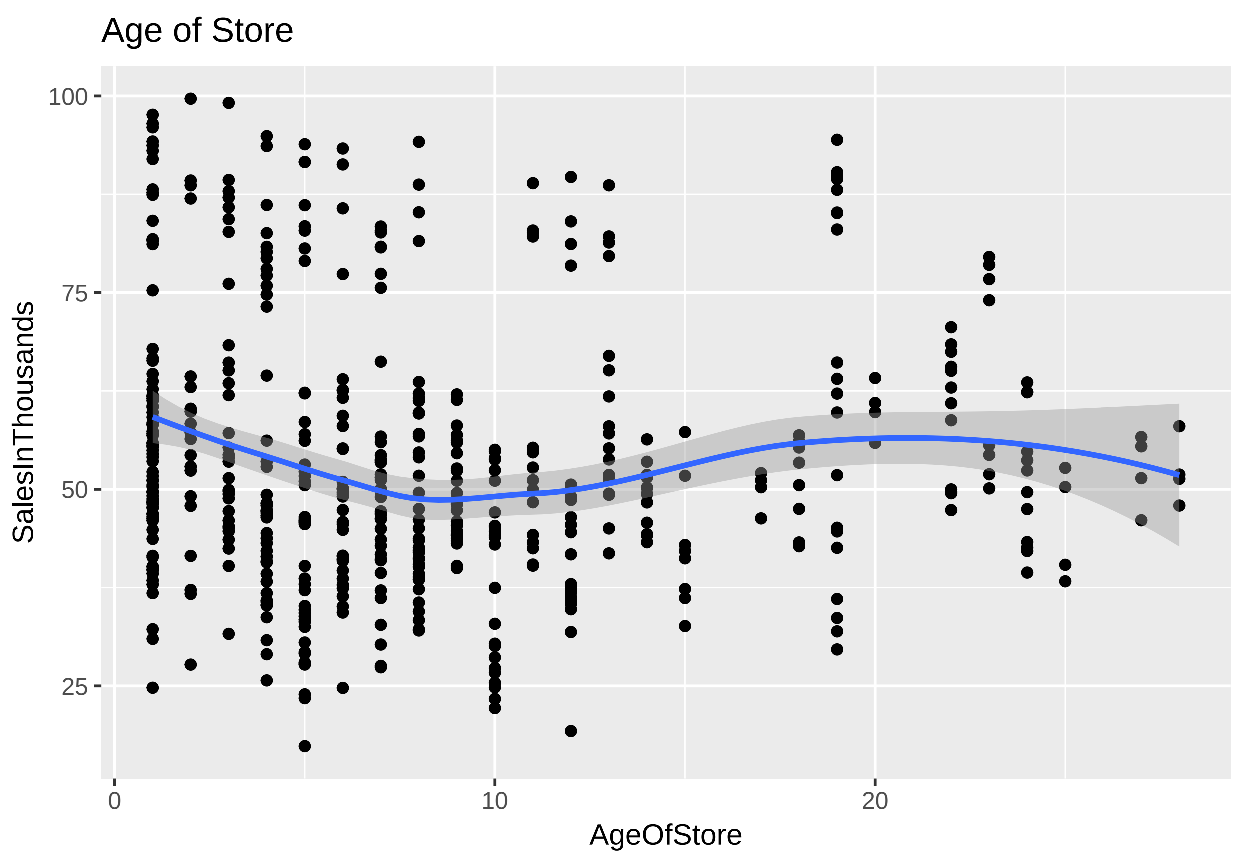
```
ggplot(data = df) +
  geom_boxplot(mapping = aes(x = week, y = SalesInThousands, color = week)) +
  facet_wrap(~ MarketSize, nrow = nlevels(df$MarketSize)) +
  theme(legend.position="none") +
  coord_flip() +
  ggtitle("Number of weeks across different markets")
```



In this next plot we can see that the age of the store does not have a clear impact on sales

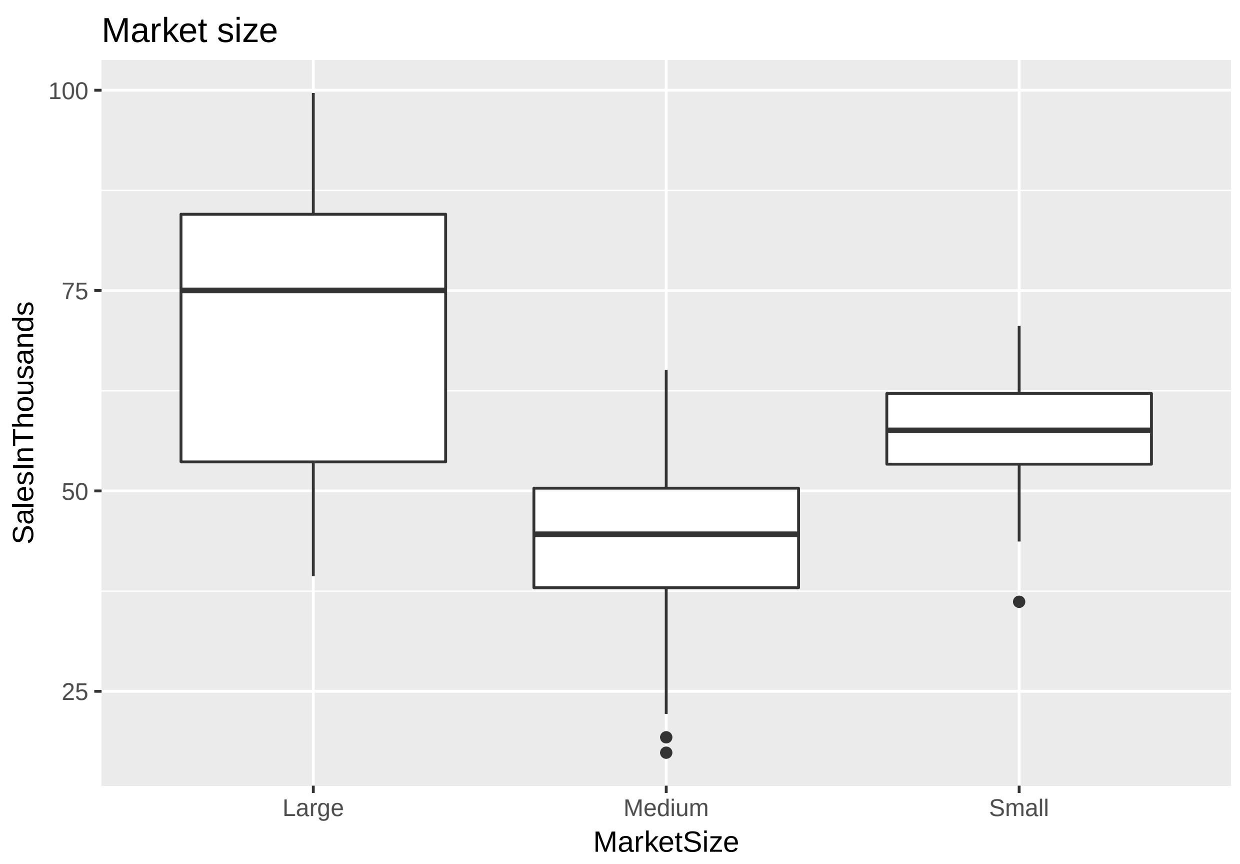
```
ggplot(data = df, aes(x = AgeOfStore, y = SalesInThousands)) +
  geom_point() +
  geom_smooth() +
  ggtitle("Age of Store")
```

```
## `geom_smooth()` using method = 'loess'
```



The main influence on sales seems to be the market size

```r
ggplot(data = df, aes(x = MarketSize, y = SalesInThousands)) +
  geom_boxplot() +
  ggtitle("Market size")
```



Accordingly, we calculate the means and medians of Sales, grouped by MarketSize and we see that the mean, as well as the median, differ greatly, with large markets being the most profitable on average.

```
MSize_median <- group_by(df, MarketSize) %>%
  summarize(median = median(SalesInThousands))

MSize_mean <- group_by(df, MarketSize) %>%
  summarize(mean = mean(SalesInThousands))

MSize_location <- as.tibble(cbind(MSize_median[,1], MSize_median[,2], MSize_mean[,2]))
MSize_location
```

```
## # A tibble: 3 x 3
##   MarketSize median  mean
##   <fct>       <dbl> <dbl>
## 1 Large        75.0  70.1
## 2 Medium       44.6  44.0
## 3 Small        57.6  57.4
```

With this information, we can move on to predicting sales.

Predictive Modeling

Choosing predictors, scaling and encoding

Following variables may be useful as predictors in the modeling process: - MarketSize - AgeOfStore - Promotion - Week

As a first step, functions for scaling and unscaling have to be created.

```
scale0_1 <- function(x){(x-min(x))/(max(x)-min(x))}
unscale <- function(a, x){a * (max(x) - min(x)) + min(x)} # where x is the original and a the scaled da
```

First, we build the model with factor variables for the linear regression the only variables that need to be changed are AgeOfStore and SalesInThousands (scaling).

```
predictors <- tibble(MarketSize = df$MarketSize,
                     Promotion = df$Promotion,
                     AgeOfStore = scale0_1(df$AgeOfStore),
                     week = df$week)
independent <- scale0_1(df$SalesInThousands)
mdf <- as.tibble(cbind(SalesInThousands = independent, predictors))
```

For the random forest and the boosted models we are going to build, we need to encode the factors, using "one hot encoding", meaning that the factors are being split into binary variables.

```
s_predictors <- as.tibble(cbind(model.matrix(~ MarketSize -1, data = df),
                                model.matrix(~ week -1, data = df),
                                model.matrix(~ Promotion -1, data = df),
                                AgeOfStore = scale0_1(df$AgeOfStore)))
```

We also scale the independent variable and combine everything to a dataframe that will be the basis of our modeling process.

```
s_sales <- scale0_1(df$SalesInThousands)

s_mdf <- as.tibble(cbind(SalesInThousands = s_sales, s_predictors))
```

Splitting

For the purpose of cross-validation, splitting the data into a training set and a test set is necessary.

```
# For the basic model

n = nrow(mdf)
trainIndex = sample(1:n, size = round(0.7*n), replace=FALSE)
train = mdf[trainIndex ,]
test = mdf[-trainIndex ,]

# For the scaled and encoded model

s_n = nrow(s_mdf)
s_trainIndex = sample(1:n, size = round(0.7*n), replace=FALSE)
s_train = s_mdf[trainIndex ,]
s_test = s_mdf[-trainIndex ,]
```

Model 1: Linear Regression

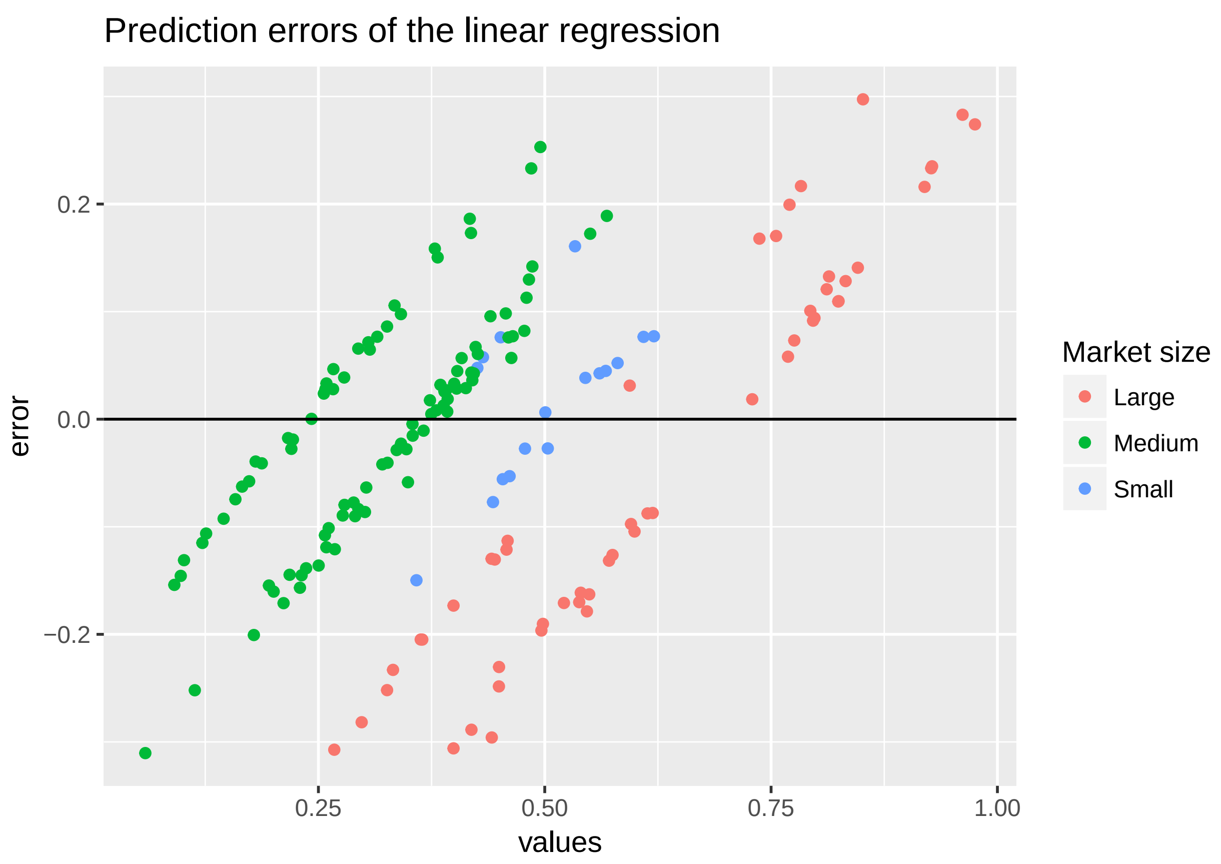
A linear regression can provide explanatory insight other "black box methods" do not have. First we build the model and look at the coefficients. We again see that the MarketSize as well as the promotional campaigns have significant influence.

```r
lmodel <- lm(mdf, data = train)
summary(lmodel)
```

```
##
## Call:
## lm(formula = mdf, data = train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.313862 -0.088773  0.009226  0.096178  0.305837
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.69275    0.02077  33.349  < 2e-16 ***
## MarketSizeMedium -0.33414    0.01526 -21.903  < 2e-16 ***
## MarketSizeSmall  -0.18981    0.02413  -7.865 3.96e-14 ***
## Promotion2       -0.13860    0.01666  -8.319 1.66e-15 ***
## Promotion3       -0.01426    0.01674  -0.852    0.395
## AgeOfStore        0.03816    0.02838   1.344    0.180
## week2             0.01216    0.01927   0.631    0.528
## week3             0.01086    0.01893   0.574    0.567
## week4             0.00838    0.01966   0.426    0.670
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1323 on 375 degrees of freedom
## Multiple R-squared:  0.5931, Adjusted R-squared:  0.5845
## F-statistic: 68.33 on 8 and 375 DF,  p-value: < 2.2e-16
```

Now we have a look at how good our predictions were.

```r
lm_pred <- predict(lmodel, test)

lm_test_pred <- tibble(values = test$SalesInThousands, predictions = lm_pred,
                       diff = test$SalesInThousands - lm_pred)

lm_diffplot <- tibble(values = lm_test_pred$values, error = lm_test_pred$diff,
                      cl = as.factor(test$MarketSize))

ggplot(lm_diffplot, aes(y = error, x = values, color = cl)) +
  geom_point() +
  geom_hline(yintercept = 0) +
  labs(color = "Market size") +
  ggtitle("Prediction errors of the linear regression")
```



As a final step, we compute the RMSE for comparison with the other models.

```
modelr::rmse(lmodel, data = df)
```

```
## [1] 55.26971
```

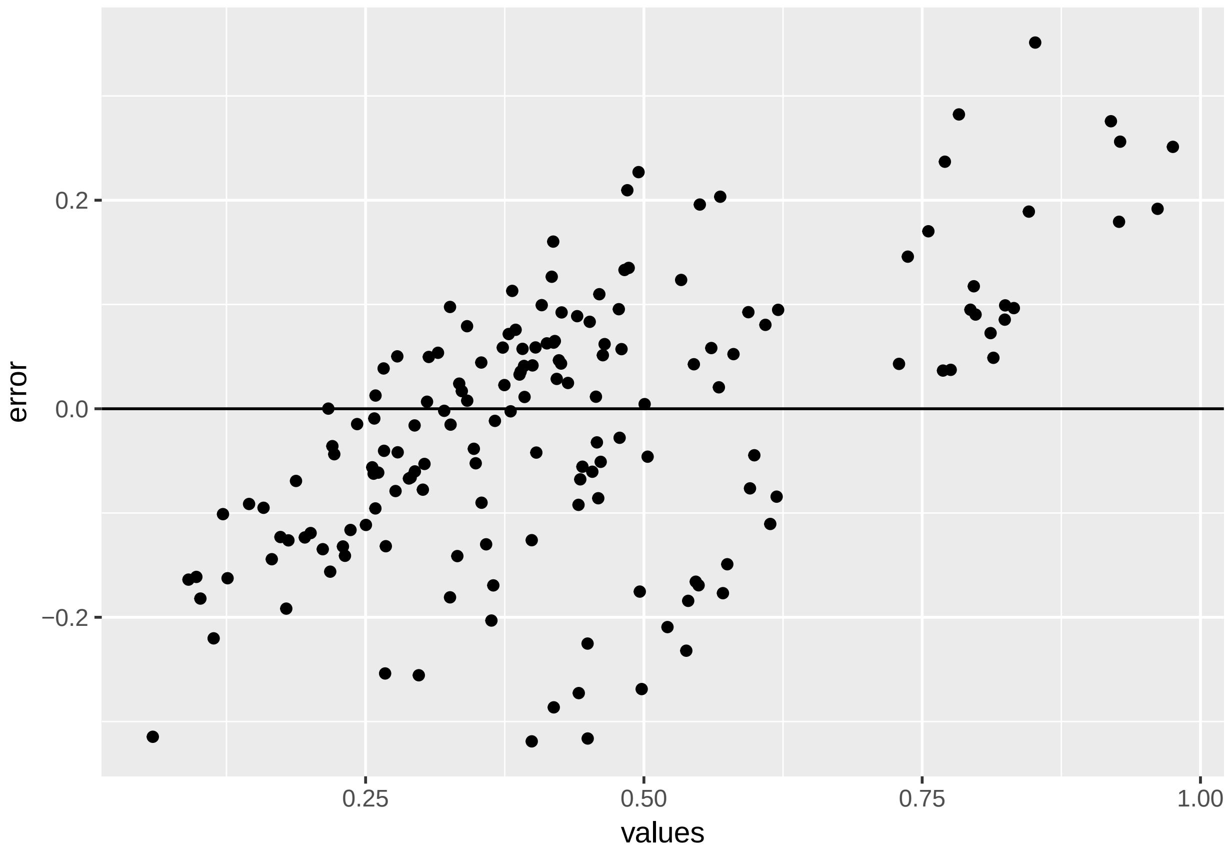
Model 2: Random Forest

Our next step up in predictive power will be computing a random forest. It will also provide us with information, which variables are important for our prediction.

```
rf <- randomForest(s_train$SalesInThousands ~ ., data = s_train, importance = TRUE)
```

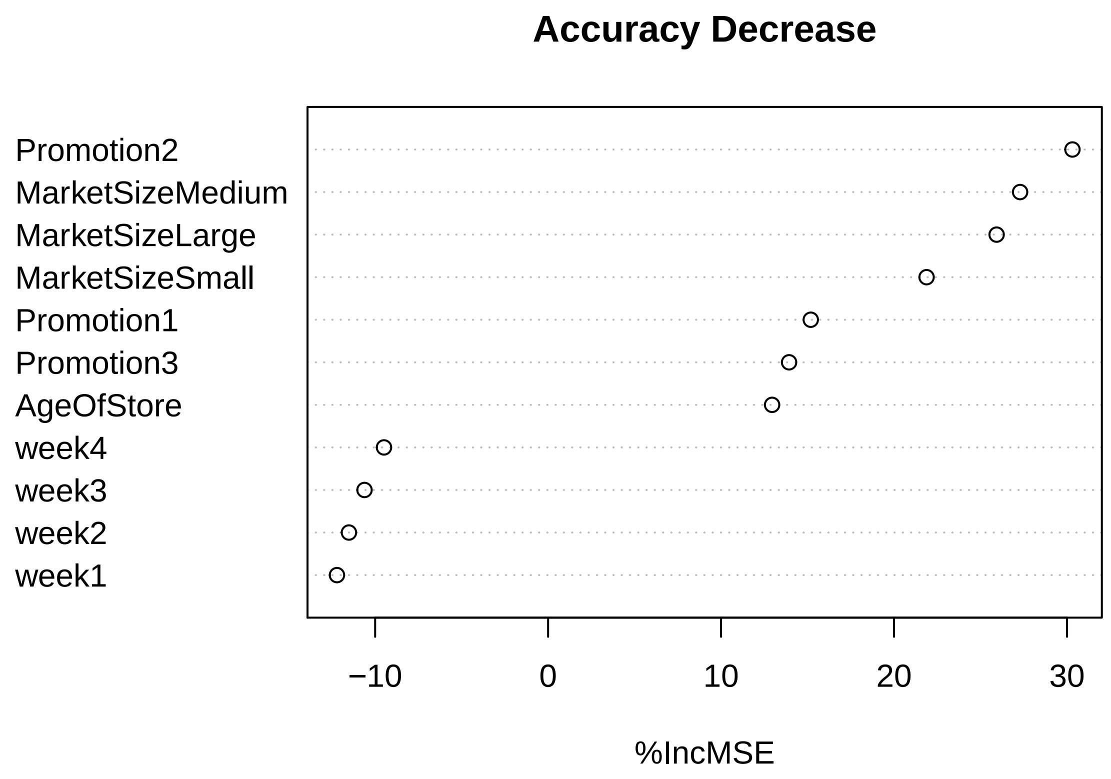
First we have a look at how good our predictions were: The errors are more evenly scattered around 0 than in the linear model, even though the random forest still does a mediocre job of predictions for large values.

```
rf_pred <- predict(rf, s_test)
rf_test_pred <- tibble(values = s_test$SalesInThousands, predictions = rf_pred,
                                 diff = s_test$SalesInThousands - rf_pred)

rf_diffplot <- tibble(values = rf_test_pred$values, error = rf_test_pred$diff)

ggplot(rf_diffplot, aes(y = error, x = values)) +
  geom_point() +
  geom_hline(yintercept = 0)
```

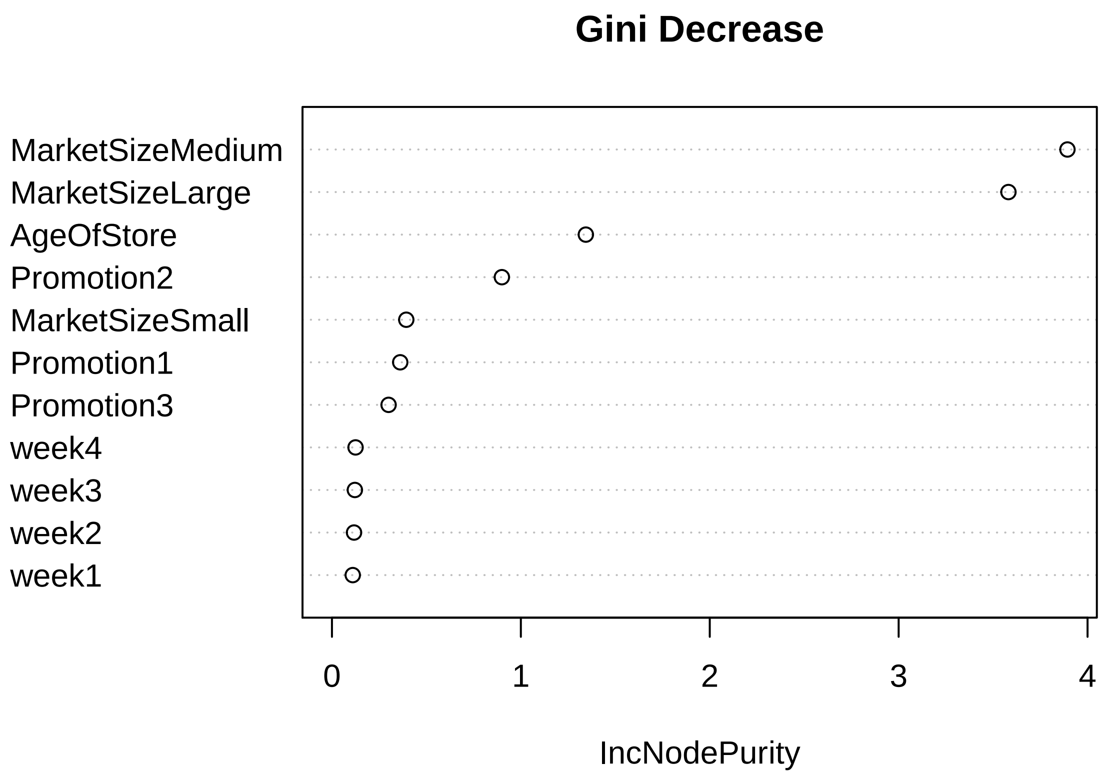


The random Forest also computed the variable importance. This mainly confirms what we have seen in the Exploratory Analysis. The most important variables are MarketSize and Promotion (Promotional campaign #2 in particular). The age of the store is only of importance in regards to the Gini Decrease, but not regarding Accuracy.

```r
varImpPlot(rf, type = 1, main = "Accuracy Decrease")
```



```
varImpPlot(rf, type = 2, main = "Gini Decrease")
```



Finally we compute the RMSE. As expected, the Random Forest outperforms the simple linear model by far.

```r
mean(sqrt(rf$mse))
```

```
## [1] 0.1343121
```

Model 3: Boosted Trees

Finally, we try to get even better predictions with boosting. For that, we try a few different models with different settings for the shrinkage factor (eta) and the maximum depth of the tree (max_depth).

First we set up the folds and the parameter list:

```r
ms_predictors <- data.matrix(s_predictors)
ms_sales <- as.numeric(s_sales)

N <- nrow(ms_predictors)
fold_number <- sample(1:5, N, replace = TRUE)
params <- data.frame(eta = rep(c(.1, .5, .9), 3),
                     max_depth = rep(c(3, 6, 12), rep(3,3)))
```

Now we apply the preceding algorithm to compute the error for each model and each fold using five folds.

```r
error <- matrix(0, nrow = 9, ncol =5)
for(i in 1:nrow(params)){
  for (k in 1:5){
    fold_idx <- (1:N)[fold_number == k]
    xgb <- xgboost(data = ms_predictors, label = ms_sales,
                    params = list(eta = params[i, "eta"],
                                  max_depth = params[i, "max_depth"]),
                    objective = "reg:linear", nrounds = 100, verbose = 0)
    pred <- predict(xgb, ms_predictors)
    error[i, k] <- mean(ms_sales - pred)
  }
}

avg_error <- 100 * rowMeans(error)
xgb_mdls_errors <- as.tibble(cbind(params, avg_error))
xgb_mdls_errors_abs <- as.tibble(cbind(params, avg_error_abs = abs(avg_error)))
```

We get the smallest error for eta = 0.9 and max_depth = 6.

```
xgb_winner <- arrange(xgb_mdls_errors_abs, avg_error_abs)[1,]
xgb_winner
```

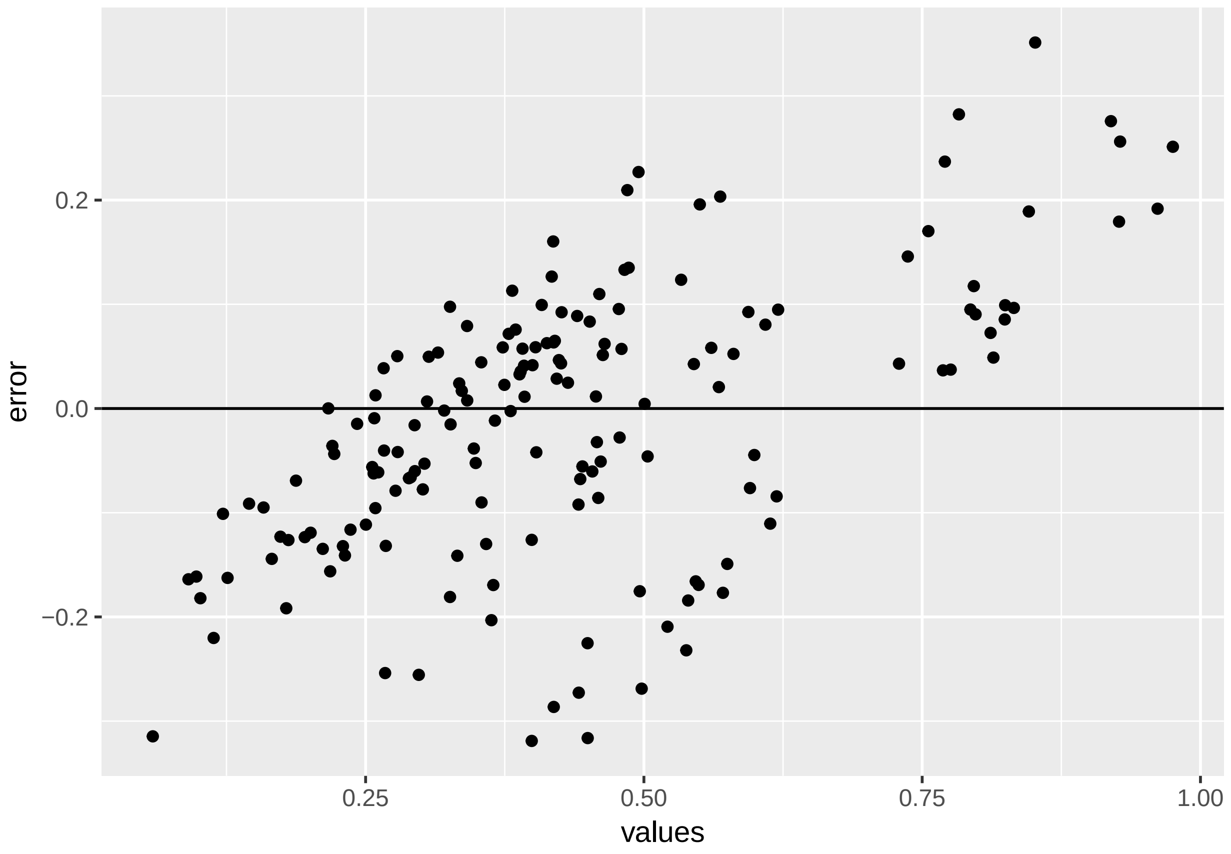
```
## # A tibble: 1 x 3
##     eta max_depth avg_error_abs
##   <dbl>     <dbl>         <dbl>
## 1   0.9         6    0.00000299
```

Therefore we now build this model and compute the rmse. Cross-Validation already took place, since the model was fitted on 5 different folds.

```
xgb_finalmodel <- xgboost(data = ms_predictors, label = ms_sales,
                          objective = "reg:linear", nrounds = 100,
                          eta = xgb_winner$eta, xgb_winner$max_depth,
                          verbose = 0)
```

I run the boosted model on the same test data the other models were tested on, so that we can compare the three models visually. In comparison to the other two models, the errors are smaller and spread out more evenly.

```
ms_test <- data.matrix(s_test[,-1])

xgb_pred <- predict(xgb_finalmodel, ms_test)

xgb_test_pred <- tibble(values = s_test$SalesInThousands, predictions = xgb_pred,
                        diff = s_test$SalesInThousands - xgb_pred)

xgb_diffplot <- tibble(values = xgb_test_pred$values, error = xgb_test_pred$diff)

ggplot(rf_diffplot, aes(y = error, x = values)) +
  geom_point() +
  geom_hline(yintercept = 0)
```



Finally, we compute the RMSE of the boosted model. This final model is recommended when trying to predict sales for new data.

```
min(xgb_finalmodel$evaluation_log$train_rmse)
```

```
## [1] 0.084955
```

Further ideas

A final step we could take is weighting the data to achieve more evenly scattered errors and an even smaller RMSE. The values between 0.25 and 0.6 are evenly scattered and well-predicted. As we have seen throughout the analysis, the errors left and right of this interval are mainly driven by the MarketSize (Medium and Large).

Weighting, however, will not solve this issue, since the observations Medium and Large markets already more numerous than those of Small ones. The "over-representation" of Medium Markets does not lead to a "predictive shift" in their favor.

```
nrow(df[df$MarketSize == "Large",])
```

```
## [1] 168
```

```
nrow(df[df$MarketSize == "Small",])
```

```
## [1] 60
```

```
nrow(df[df$MarketSize == "Medium",])
```

```
## [1] 320
```

Conclusions

The two largest influences on sales throughout the explanatory analysis and the models are the size of the market, as well as the promotional campaign. Promotional campaign number 2 performs worse than the other campaigns across all markets. Medium-sized markets do not achieve per-market sales as high as Small or Large markets. This fact needs further business investigation, since Medium markets are the most numerous by far.

As anticipated, a gradient-boosted tree model outperformed linear regression and the random forest and should therefore be chosen for prediction. Due to the many categorical variables and small number of variables and observations, however, computed predictions on new data should be used with caution.