*PREDICTING OPPORTUNITY VALUES FOR **SMEs*** - Capstone Project of HarvardX *Data Science Professional Certificate Course* - Harvard University

Miguel Ángel Boto Bravo, PhD in Linguistics

16/09/2020

1. INTRODUCTION

Predicting many of the sales variables involved in the preparation of an annual forecast in small, and / or recently created companies, is a major problem that affects the strategic planning of SME companies. Among the main variables, associated or not with competition in the market, we must highlight a low volume of historical data and the absence of data due to insufficient data collection.

The objective of the project that we present is to analyze the effectiveness of the application of some algorithm models in the prediction of sales values and, therefore, their use in the preparation of annual forecasts.

For this, we will use the data of sales opportunities of new clients of a small software company that contains, as we will see in the exploratory data analysis, limitations that affect the volume of data collected and the characteristics that make up the database.

2. EXPLORATORY ANALYSIS AND DATA VISUALIZATION.

2.1 Installing libraries and preparing the database:

Before starting the exploratory analysis of Sales Report, we must install and run the following libraries:

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("readxl", repos = "http://cran.us.r-project.org")
if(!require(readxl)) install.packages("readxl", repos = "http://cran.us.r-project.org")
if(!require(openxlsx)) install.packages("openxlsx", repos = "http://cran.us.r-project.org")


library(tidyverse)
library(caret)
library(data.table)
library(lubridate)
library(readxl)
library(openxlsx)
```

Next, we will download the Sales Report dataset with which we are going to work, format the dataset and create a training and validation sets:

```
#Import dataset

url <-"https://github.com/miguelboto/sales_report/blob/master/SR.xlsx?raw=true"

download.file(url, destfile = "SR.xlsx", mod="wb")

data <- read.xlsx("SR.xlsx")

#Converting date numeric rows to date format and including a new row with time in days (difference betw

sales_report <- data %>% mutate(open_date = as.Date(open_date, origin = "1899-12-30"),
                close_date = as.Date(close_date, origin = "1899-12-30"),
                last_date = as.Date(last_date, origin = "1899-12-30"),
                time = close_date - open_date)


# Validation: 10% of Sales Report data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`

test_index <- createDataPartition(y = sales_report$price, times = 1, p = 0.1, list = FALSE)
edx <- sales_report[-test_index,]
temp <- sales_report[test_index,]

# Opportunity_ID and count_id in validation set must be also in temp set

validation <- temp %>%
  semi_join(edx, by = "opportunity_name") %>%
  semi_join(edx, by = "count_id")

removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)
```

2.2 Exploratory analysis:

As we can see below, the database is composed of 730 rows and 12 columns:

```
dim(sales_report)
```

```
## [1] 730  12
```

```
head(sales_report)
```

```
##   opportunity_name                      opportunity_ID price close_date status
## 1           ABC001 3bdee373-6db1-4242-b68b-52fbb3829331 20536 2014-03-31    Won
## 2           ABC002 a50f14ba-5df9-fadf-05f5-52fd0e8f2a1b 55000 2014-03-26    Won
## 3           ABC003 7a6781c3-f1a4-df2a-1441-52fd014ce005 14012 2014-03-13    Won
## 4           ABC004 701031fe-6aad-c327-7f1c-52fd0bd990a5 17712 2014-02-21    Won
## 5           ABC005 557b43c8-dc64-77d5-0fef-53035ec153cf 17852 2014-09-26   Lost
## 6           ABC006 d7e71af2-a9c8-d964-7ce7-530361ea823f 20410 2014-05-30    Won
```

```
##   count_id market_class     sale_class  vendor  open_date  last_date     time
## 1       A1            H  Word of Mouth vendor1 2014-02-12 2014-04-08  47 days
## 2       A2            H Self Generated vendor1 2014-02-13 2014-10-31  41 days
## 3       A3           TA  Word of Mouth vendor1 2014-02-13 2014-10-01  28 days
## 4       A4            H  Word of Mouth vendor1 2014-02-13 2014-04-30   8 days
## 5       A5            H Self Generated vendor1 2014-02-18 2015-06-04 220 days
## 6       A6            H  Word of Mouth vendor1 2014-02-18 2016-07-07 101 days
```

Each line corresponds to a business opportunity with new potential clients. Since it is a provision of services under the SaaS model, it is not a sale of the finished product upon delivery, but the value expressed in the *price* column indicates the annual turnover of each new customer gained or, failing that, the potency of each lost customer.

Therefore, the key variable for our analysis is in the *price* field, which shows the annual value in $ of each opportunity.

Likewise, the *status* variable indicates if the opportunity was *won* or *lost*, *open_date* the date of opening the negotiations, *close_date* the date of the end of the opening and *time* the time elapsed between the opening and closing.

For its part, we can use two more variables of interest for the analysis: the origin of the potential customer (*sale_class*), that is, the sales channel through which the business opportunity was generated, and the type of customer (*market_class*).

The *sale_class* category contains 7 variables: website, self generated, campaign, word of mouth, bidding, partner and trade show. For its part, *Market_class*, has 5 variables: H (hospitals), CE (clinical engineering), O (others), M (manufacturers), TA (technical assistance), which correspond to the five markets that our software company can cater with your product.

Next we show the summary of the data contained in the dataset:

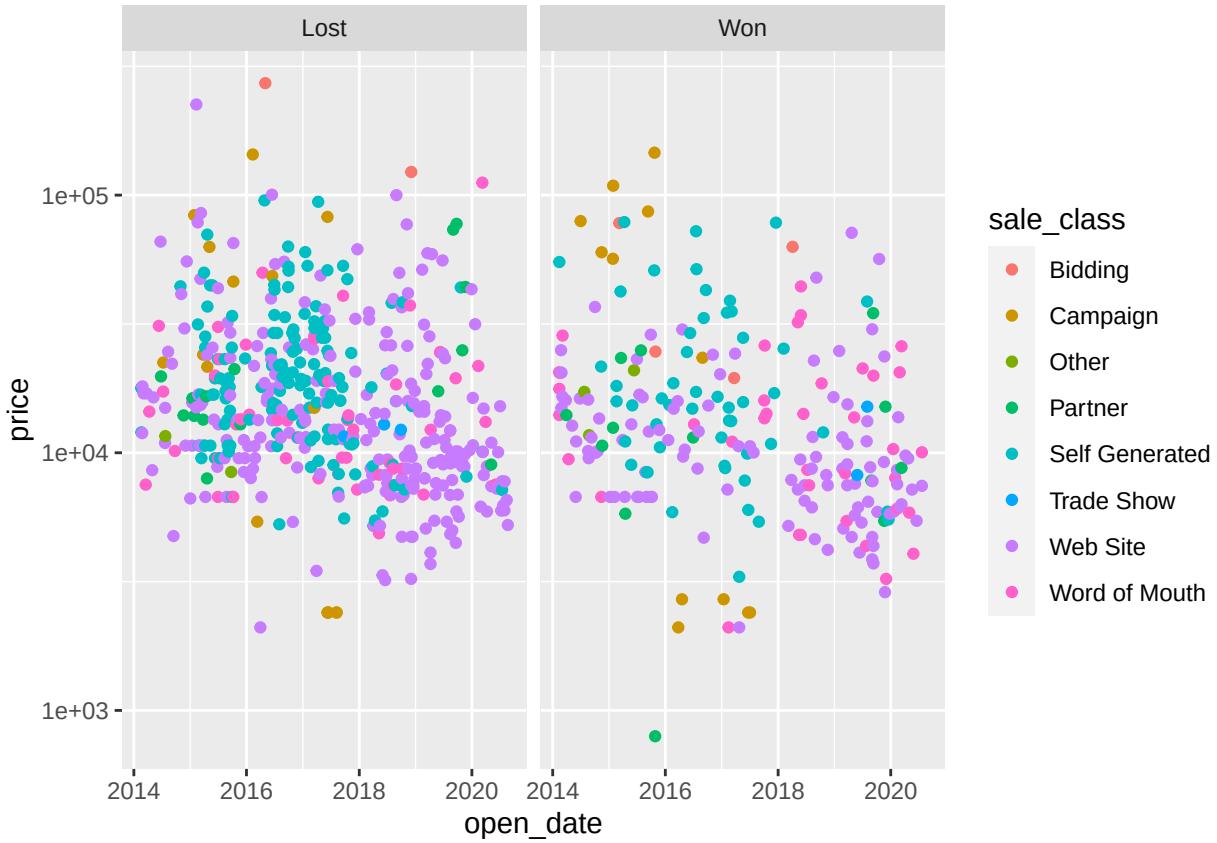
```
summary(sales_report)
```

```
##  opportunity_name   opportunity_ID         price            close_date
##  Length:730         Length:730         Min.   :   793.8   Min.   :2014-02-21
##  Class :character   Class :character   1st Qu.:  8562.7   1st Qu.:2016-08-31
##  Mode  :character   Mode  :character   Median : 13389.0   Median :2018-06-28
##                                        Mean   : 19849.6   Mean   :2018-02-10
##                                        3rd Qu.: 22409.5   3rd Qu.:2019-07-17
##                                        Max.   :272000.0   Max.   :2020-10-31
##     status            count_id         market_class        sale_class
##  Length:730         Length:730         Length:730         Length:730
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##     vendor            open_date            last_date             time
##  Length:730         Min.   :2014-02-12   Min.   :2014-03-13   Length:730
##  Class :character   1st Qu.:2015-10-27   1st Qu.:2016-08-31   Class :difftime
##  Mode  :character   Median :2017-04-22   Median :2018-06-28   Mode  :numeric
##                     Mean   :2017-05-16   Mean   :2018-02-24
##                     3rd Qu.:2018-11-12   3rd Qu.:2019-07-31
##                     Max.   :2020-08-21   Max.   :2020-08-31
```

The data was collected between February 2014 (the start date of the company's activities) and September 2020. In the first plot we can see the distribution of opportunities by time, value and status. In the second one, the distribution per origin and status:

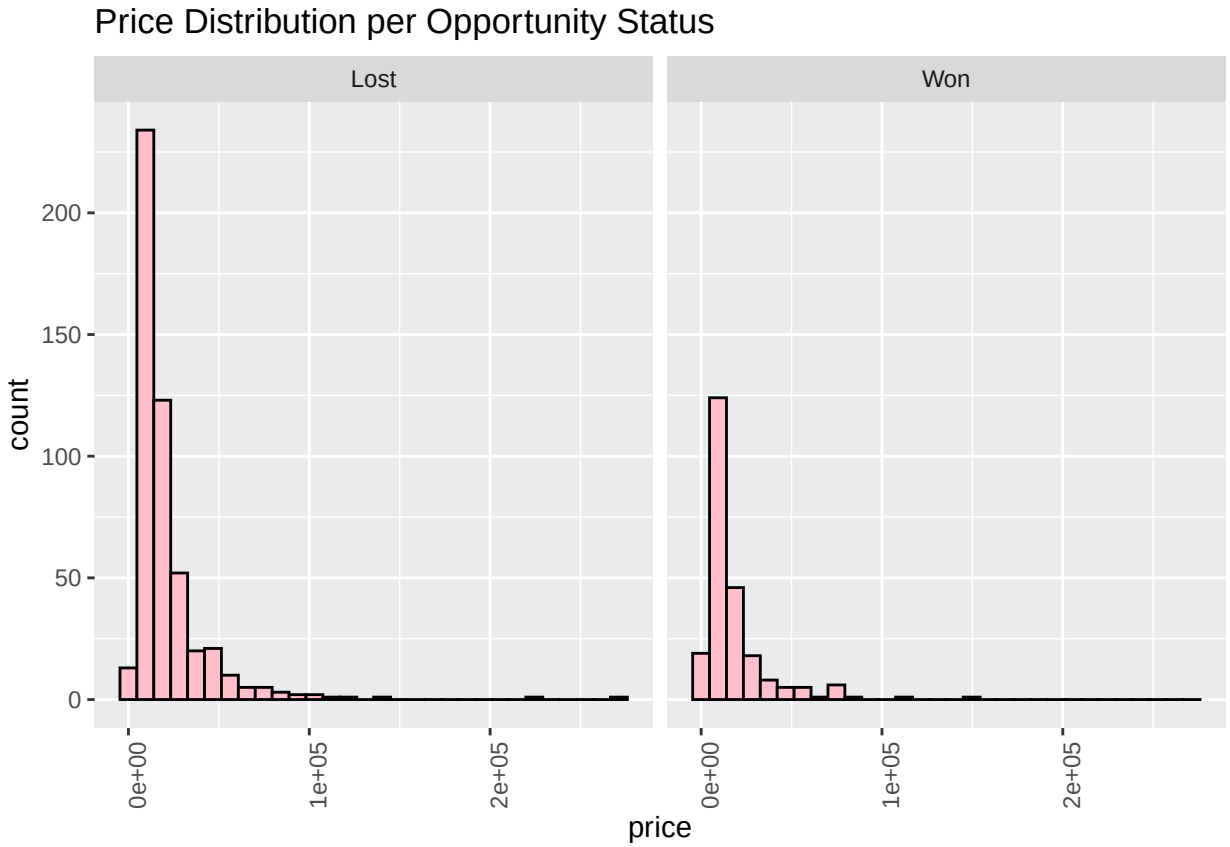
```
#Opportunities value per time

sales_report %>% mutate(time = as.numeric(sales_report$time)) %>%
  ggplot(aes(time, price, color = status)) +
  geom_point(aes(shape = market_class)) +
  scale_x_continuous(trans = "log10") +
  scale_y_continuous(trans = "log10")
```


```
sales_report %>% group_by(open_date, sale_class)  %>%
  ggplot(aes(open_date, price, color = sale_class)) +
  geom_point() +
  scale_y_continuous(trans = "log10") +
  facet_grid(~status)
```



The difference between the value of the minimum and the maximum opportunity is striking, which shows that the provision of services offered by the company serves both small companies and large corporations. In the following boxplot we can see the distribution in greater detail:

```
#Histogram Price Distribution All Opportunities per status
sales_report %>%
  ggplot(aes(price)) +
  geom_histogram(bins = 30, fill = "pink", col = "black") +
  ggtitle("Price Distribution per Opportunity Status") +
  facet_grid(~status) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



However, we can see that there is a greater volume of opportunities concentrated in lower bands. Below we see the average price per opportunity (all Opportunities) and the average of won opportunities and success rate:

```
#Average Price per status opportunity and Success Rate

dat <- sales_report %>% group_by(status) %>%
  summarize(Opps = n(), Ticket = mean(price))


dat %>% mutate(Rate = (dat[, 2:2]) / sum(dat[, 2])) %>% knitr::kable()
```

status	Opps	Ticket	Rate
Lost	495	20988.15	0.6780822
Won	235	17451.35	0.3219178

```
#Resume Prospects Opportunities

sales_resume <- sales_report %>%
  summarize(
    'Counts'=n_distinct(count_id),
    'Opportunities'=n_distinct(opportunity_ID),
    'Min_Price'=min(price),
    'Max_Price'=max(price),
```

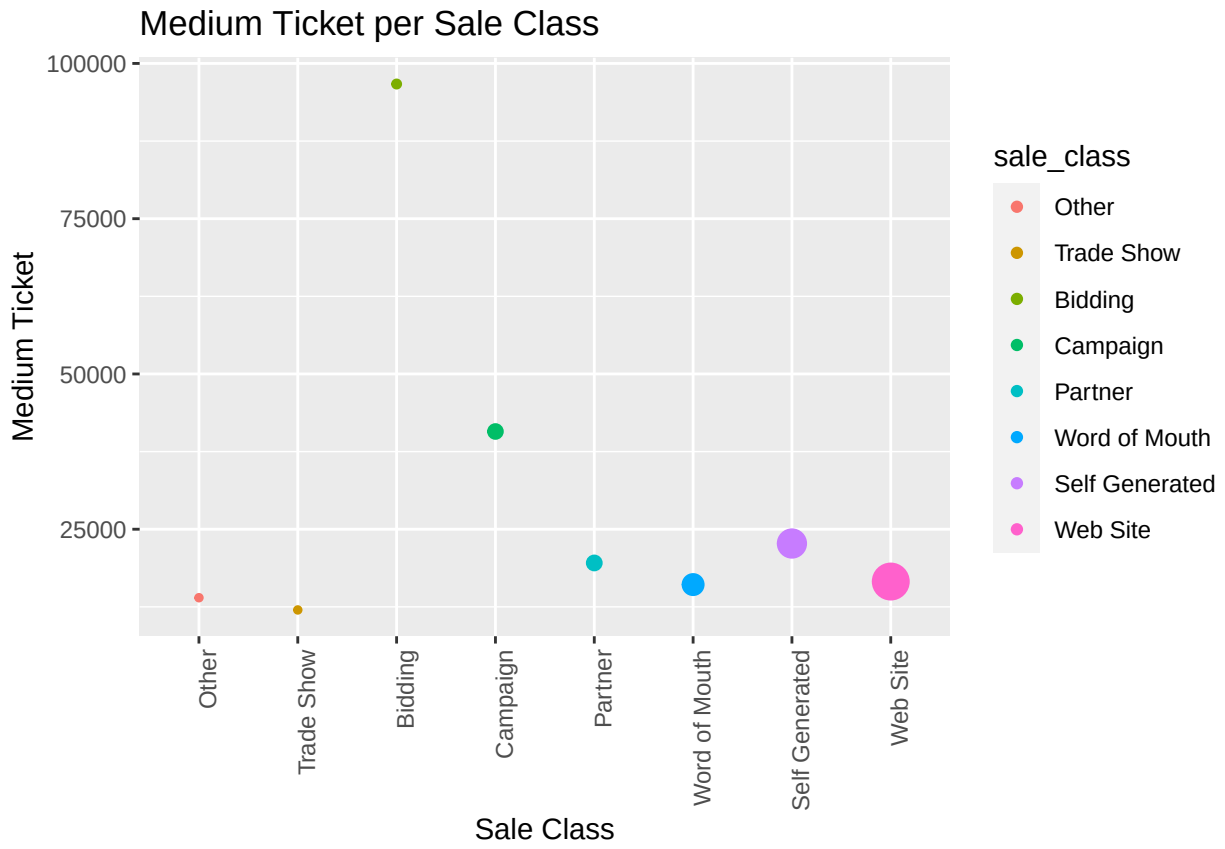
```
    'Average_Price'=mean(price),
    'Total_Values'=sum(price)
  )
sales_resume %>% knitr::kable()
```

Counts	Opportunities	Min_Price	Max_Price	Average_Price	Total_Values
730	730	793.8	272000	19849.59	14490200

```
sales_resume_won <- sales_report %>%  filter(status ==  "Won") %>%
  summarize(
    'Counts'=n_distinct(count_id),
    'Won'=n_distinct(opportunity_ID),
    'Min Price'=min(price),
    'Max Price'=max(price),
    'Average Price'=mean(price),
    'Total Values Won'=sum(price),
    'Success Rate'=(Won/sales_resume$Opportunities)
  )
sales_resume_won %>% knitr::kable()
```

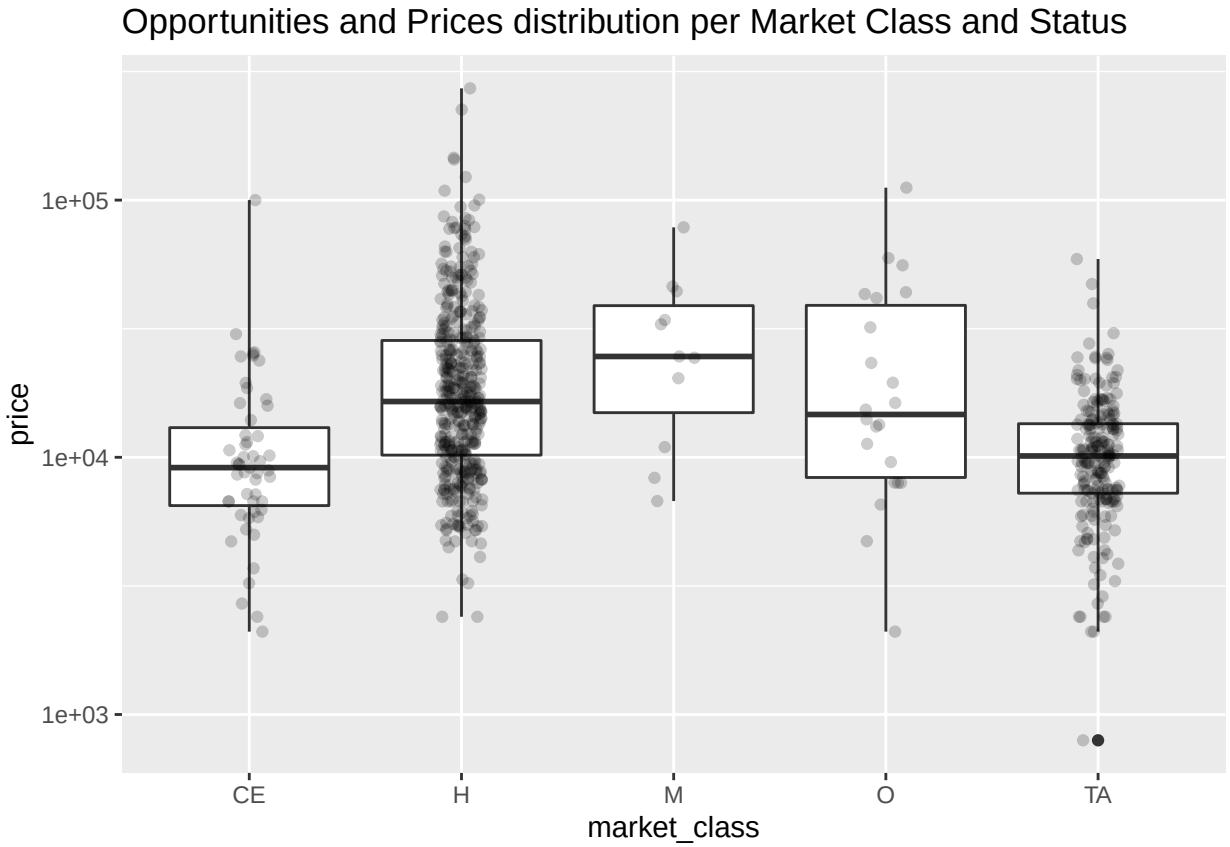
Counts	Won	Min Price	Max Price	Average Price	Total Values Won	Success Rate
235	235	793.8	146100	17451.35	4101067	0.3219178

```
sales <- sales_report %>% group_by(sale_class) %>%
  summarize(totalopp = n(), averageprice = mean(price)) %>%
  arrange(desc(averageprice))
sales %>% mutate(sale_class = reorder(sale_class, totalopp)) %>%
  ggplot(aes(x= sale_class, y = averageprice, color = sale_class)) +
  geom_point(aes(size=totalopp)) +
  ggtitle("Medium Ticket per Sale Class") +
  xlab("Sale Class")+
  ylab("Medium Ticket") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  guides(size=FALSE)
```

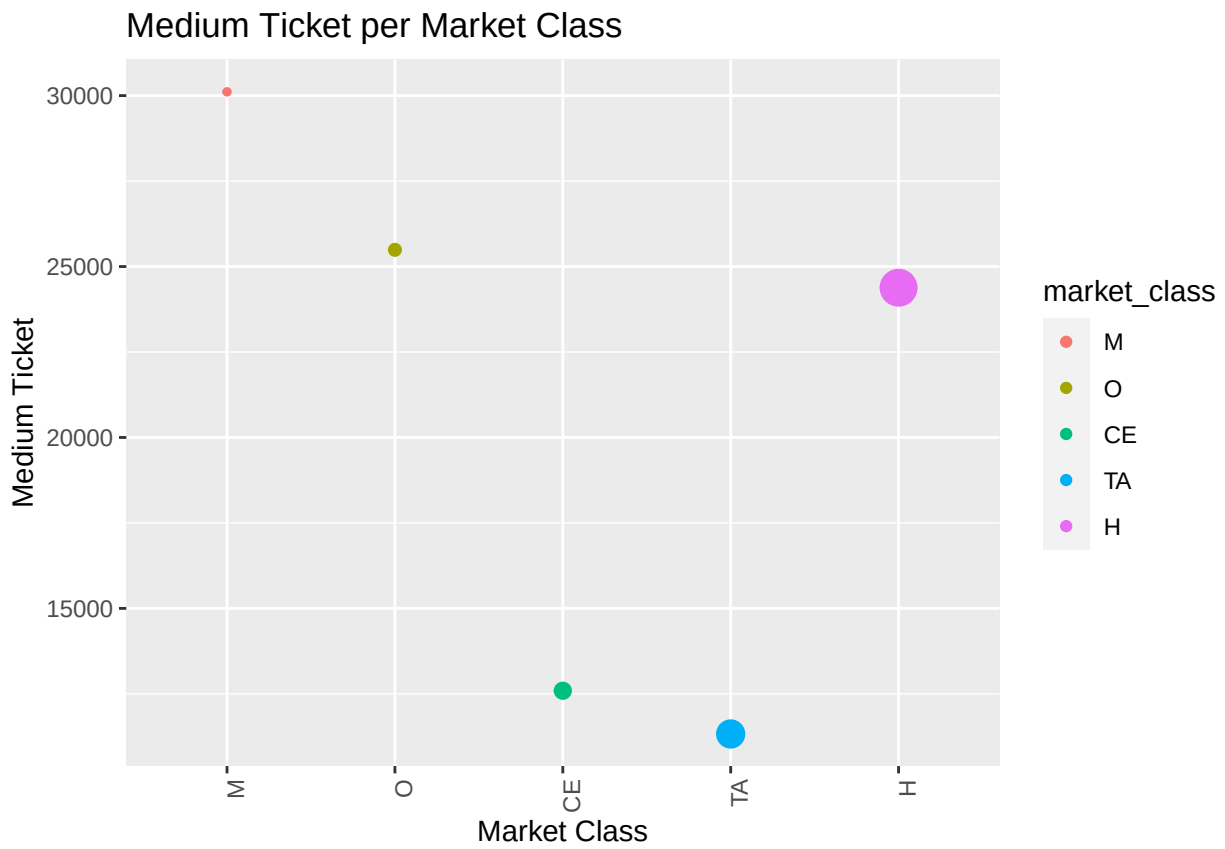


Continuing with the analysis of the dataset based on the price of the opportunities, the following boxplot shows us the distribution by value and type of market of potential clients, in which we can perceive that the largest volume is concentrated in hospitals and technical assistance:

```
#Boxplot Opportunities and prices distribution per Market Class
sales_report %>%
  group_by(market_class) %>% arrange(desc(price)) %>%
  ggplot(aes(market_class, price)) +
  scale_y_continuous(trans = "log10") +
  geom_boxplot(coef=3) +
  geom_jitter(width = 0.1, alpha = 0.2)  +
  ggtitle("Opportunities and Prices distribution per Market Class and Status")
```

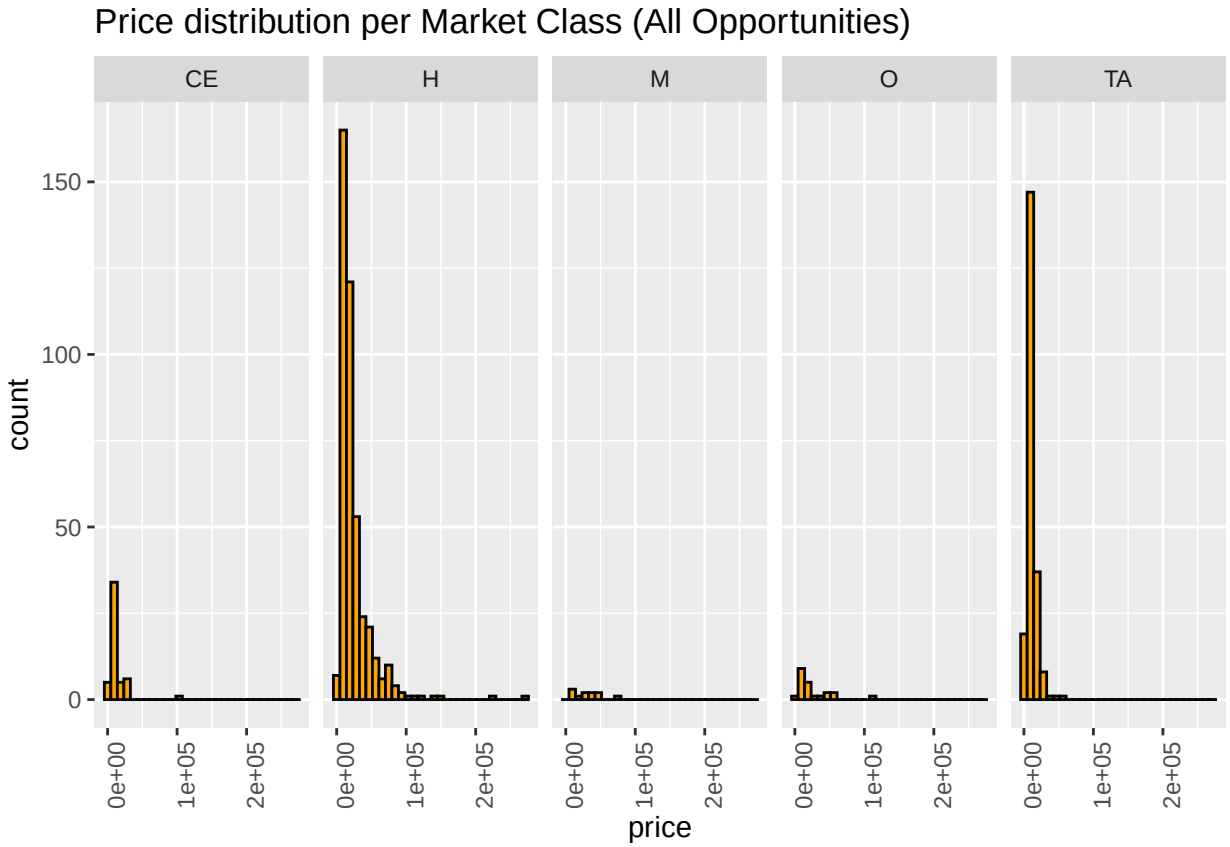



```
type <- sales_report %>% group_by(market_class) %>%
  summarize(totalopp = n(), averageprice = mean(price)) %>%
  arrange(desc(averageprice))
type %>% mutate(market_class = reorder(market_class, totalopp)) %>%
  ggplot(aes(x= market_class, y = averageprice, color = market_class)) +
  geom_point(aes(size=totalopp)) +
  ggtitle("Medium Ticket per Market Class") +
  xlab("Market Class")+
  ylab("Medium Ticket") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  guides(size=FALSE)
```

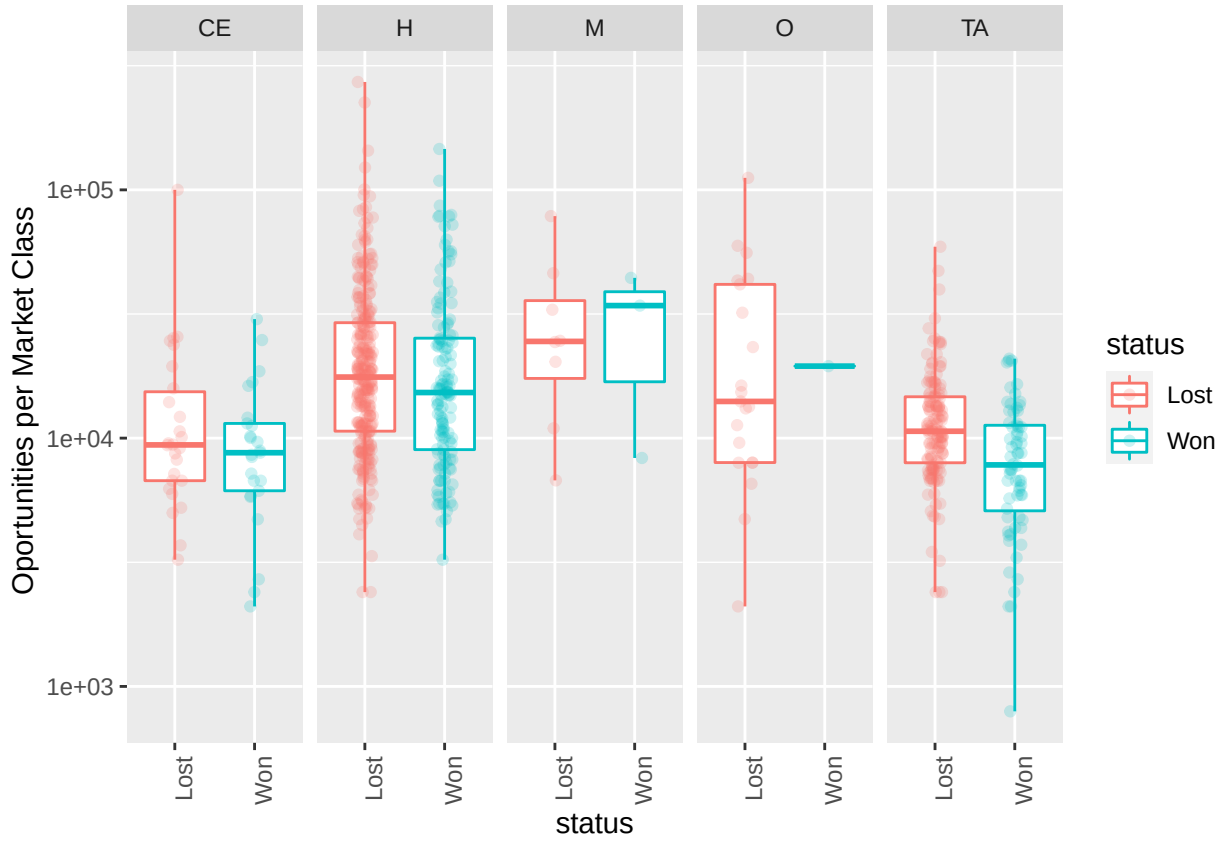


The following histograms will help us to better understand the irregular distribution of the value of opportunities taking into account several variables:

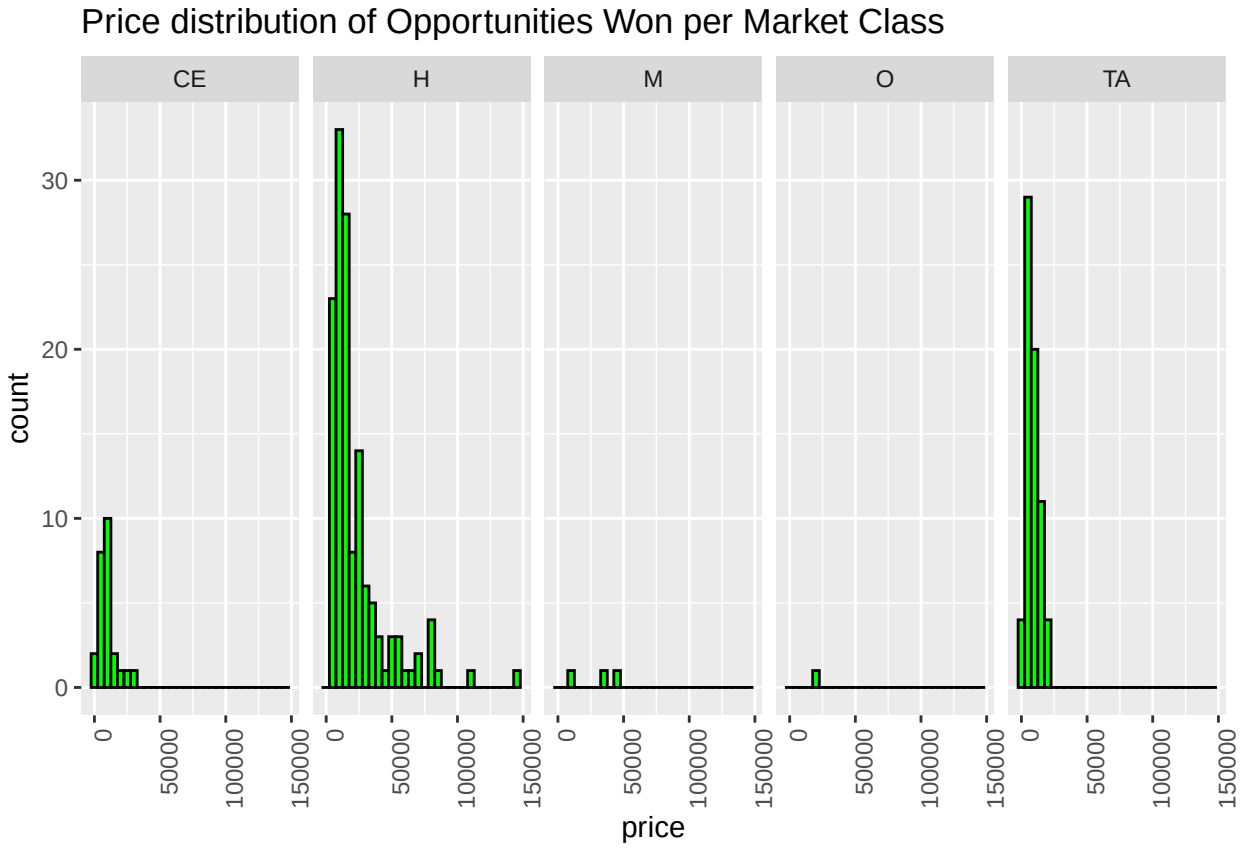
```
#Histogram Price Distribution per Market Class
sales_report %>%
  ggplot(aes(price)) +
  geom_histogram(bins = 30, fill = "orange", col = "black") +
  ggtitle("Price distribution per Market Class (All Opportunities)") +
  facet_grid(~market_class) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



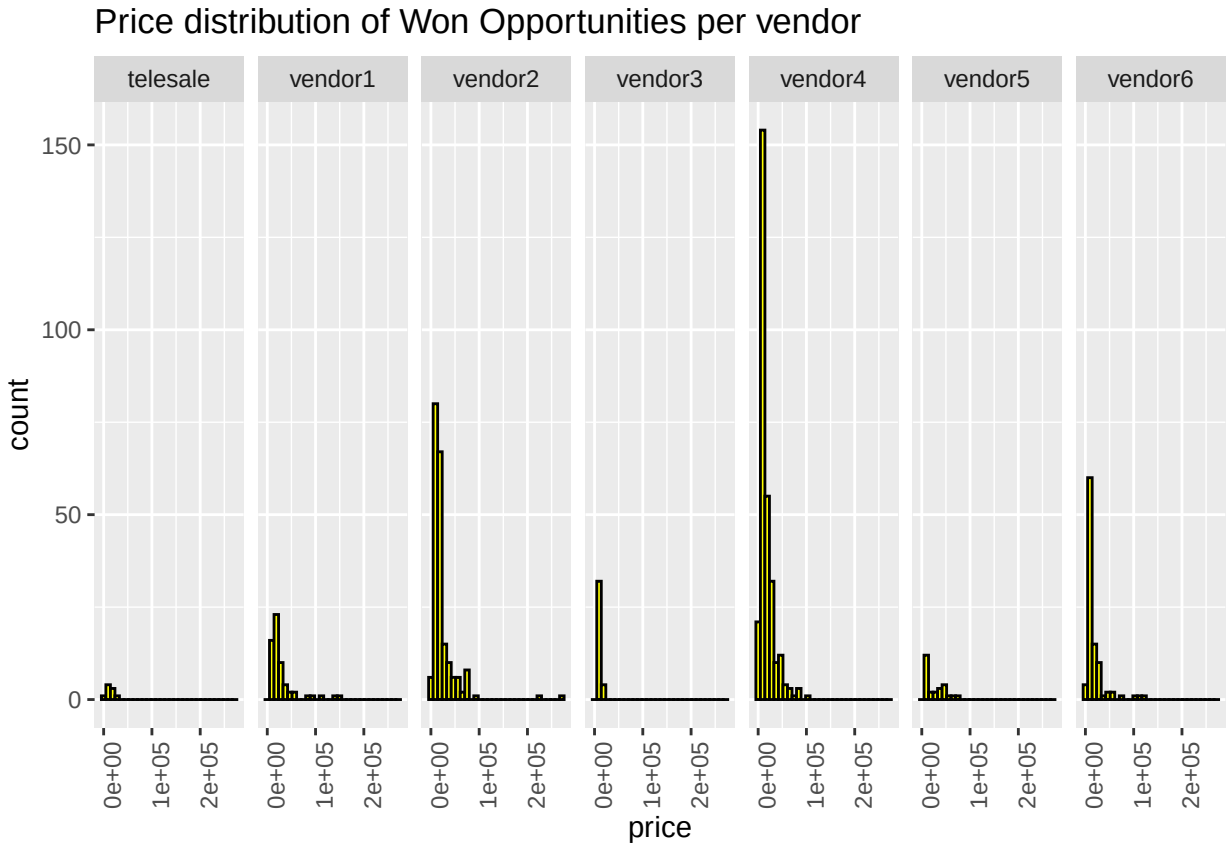

```
#Boxplot Opportunities per Market Class and Status
sales_report %>%
  group_by(status) %>% arrange(desc(price)) %>%
  ggplot(aes(status, price, color = status)) +
  scale_y_continuous(trans = "log10") +
  geom_boxplot(coef=3) +
  geom_jitter(width = 0.1, alpha = 0.2) +
  facet_grid(~market_class) +
  ylab("Oportunities per Market Class ") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



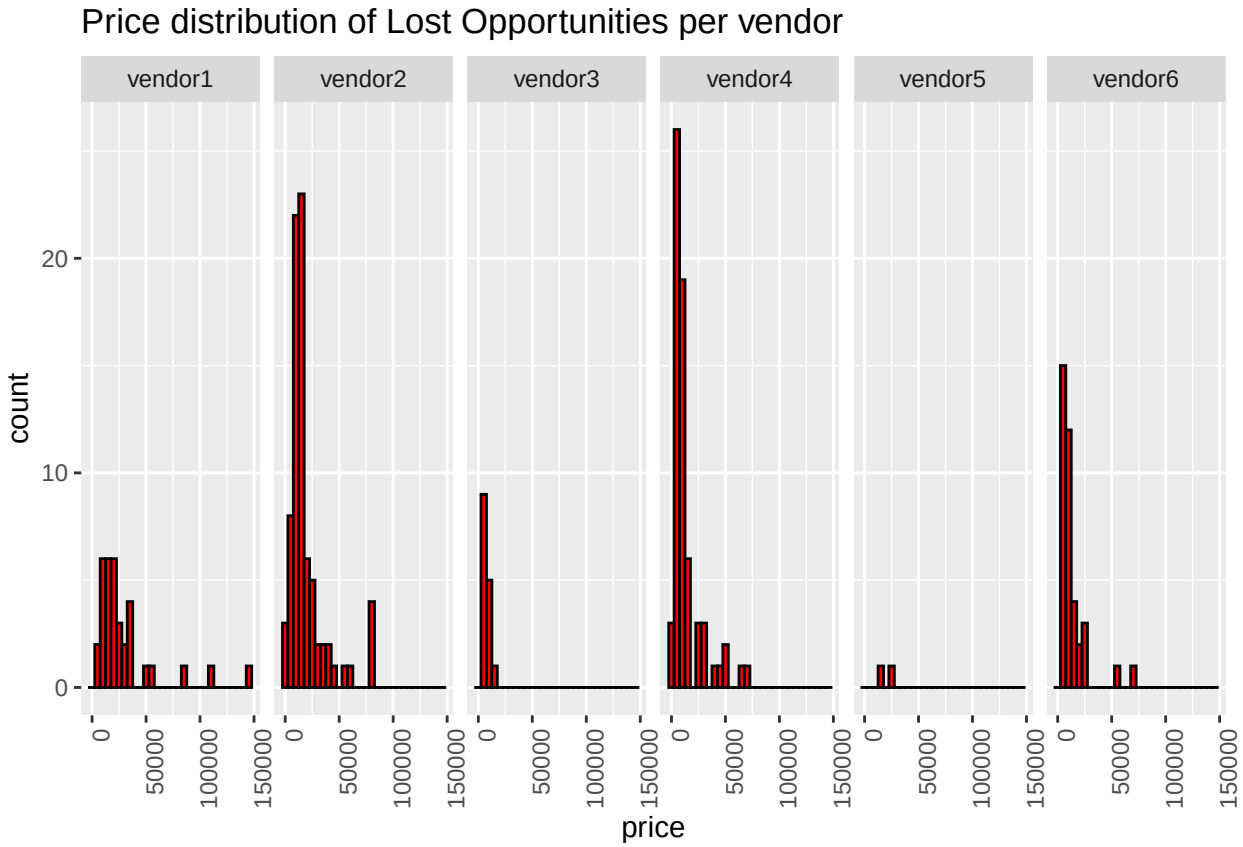

```
#Histogram Price Won Distribution per Market Class
sales_report %>% filter(status == "Won") %>%
  ggplot(aes(price)) +
  geom_histogram(bins = 30, fill = "green", col = "black") +
  ggtitle("Price distribution of Opportunities Won per Market Class") +
  facet_grid(~market_class) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```
#Histogram Won Price Distribution per vendor
sales_report %>% filter(status != "won") %>%
  ggplot(aes(price)) +
  geom_histogram(bins = 30, fill = "yellow", col = "black") +
  ggtitle("Price distribution of Won Opportunities per vendor") +
  facet_grid(~vendor) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```
#Histogram Lost Price Distribution per vendor
sales_report %>% filter(status != "Lost") %>%
  ggplot(aes(price)) +
  geom_histogram(bins = 30, fill = "red", col = "black") +
  ggtitle("Price distribution of Lost Opportunities per vendor") +
  facet_grid(~vendor) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



Finally, let's see the accuracy and standard deviation data before entering the predictions and results section:

```
#Overall accuracy

y_hat <- sample(c("Won", "Lost"), length(sales_report), replace = TRUE)

accuracy <- mean(y_hat == sales_report$status)

accuracy
```

```
## [1] 0.4739726
```

```
#Price Mean & SD

sales_report %>% group_by(status) %>% summarize(mean(price), sd(price))
```

```
## # A tibble: 2 x 3
##   status `mean(price)` `sd(price)`
##   <chr>          <dbl>       <dbl>
## 1 Lost          20988.      23400.
## 2 Won           17451.      18803.
```

3. PREDICTIONS AND RESULTS

Bearing in mind that the general approach to defining the best in machine learning is to define a _ loss function_, we will make our prediction testing seven generative models with RMSE (the most widely used loss function) as a measure to evaluate our algorithm performance until reaching a RMSE value equal to or less than 17451, since this is the average value of the opportunities won over the 6 years analyzed in the dataset.

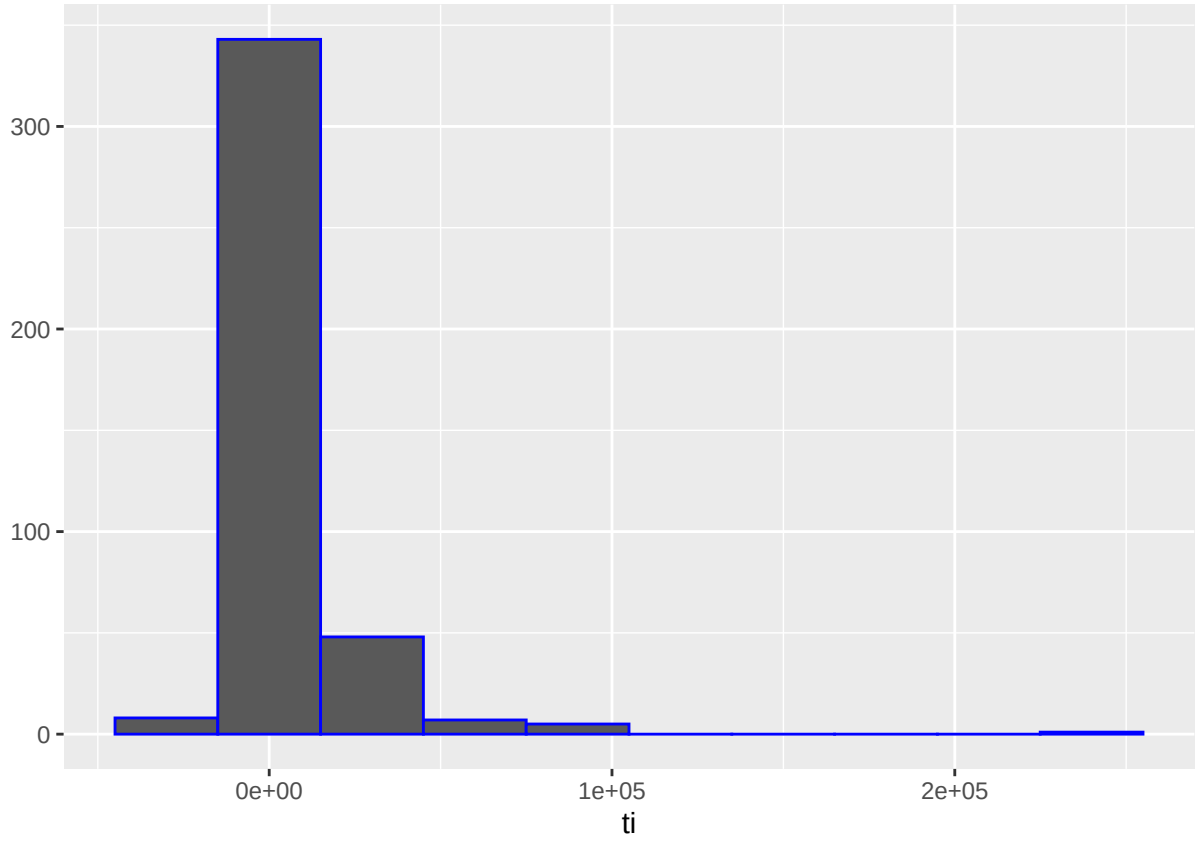
3.1 Naive By Mean Method

```
#Naive by Mean Method
mu <- mean(edx$price)
naive_RMSE <- RMSE(temp$price, mu)

RMSE_results = data.frame(Method = "Naive Model", RMSE = naive_RMSE)
RMSE_results
```

```
##        Method    RMSE
## 1 Naive Model 22632.1
```

3.2 Time Effect Model

```
#Time effect model#
time_avg <- edx %>% group_by(time) %>%
  summarise(ti = mean(price - mu))

qplot(ti, data = time_avg, bins = 10, color = I("blue"))
```




```
prediction1 <- mu + temp %>%
  left_join(time_avg, by="time") %>%
  pull(ti)

RMSE_time_effect_model <- RMSE(prediction1, temp$price)

RMSE_time_effect_model
```

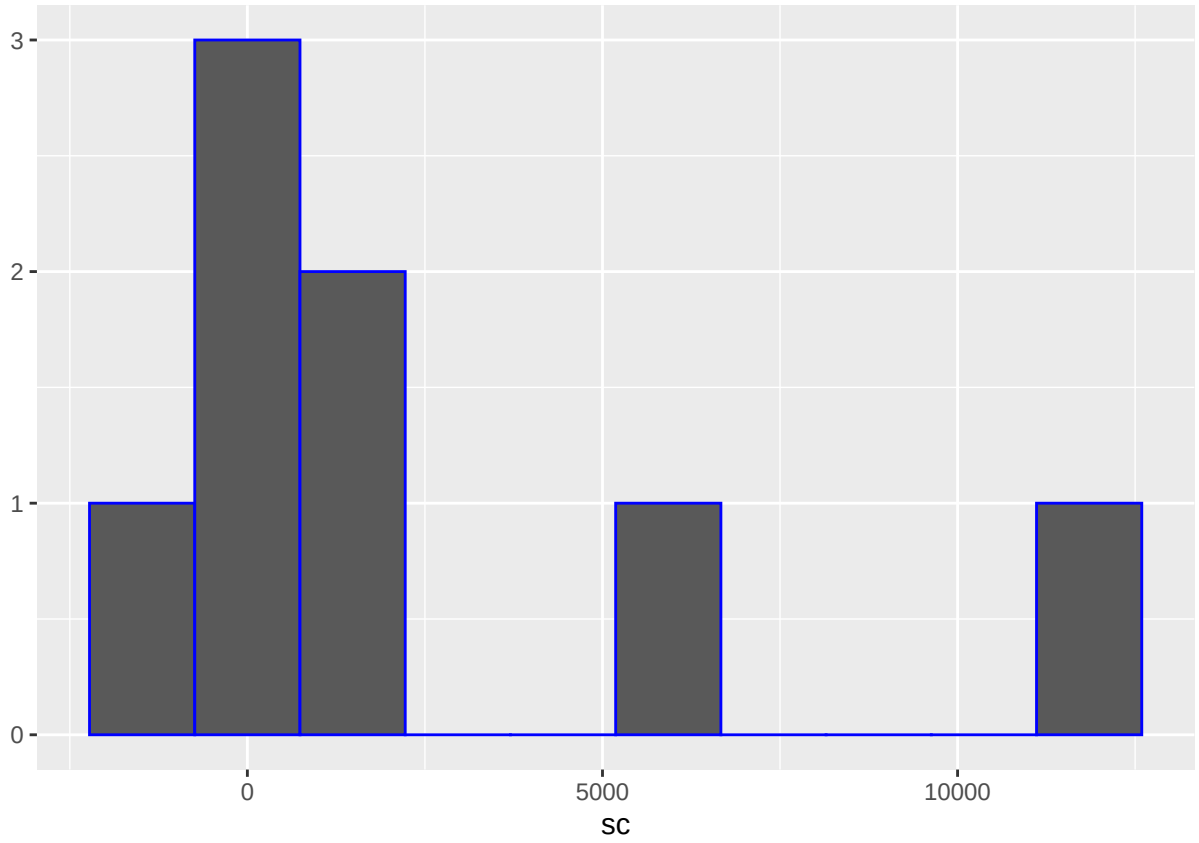
```
## [1] 14280.63
```

```
RMSE_results <-  bind_rows(RMSE_results,
                           data.frame(Method = "Time Effect Model",
                                      RMSE = RMSE_time_effect_model))
RMSE_results %>% knitr::kable()
```

Method	RMSE
Naive Model	22632.10
Time Effect Model	14280.63

3.3 Time and Sale Class Effect Model

```
#Time and Sale Class effect model

sale_avg <- edx %>%
  left_join(time_avg, by='time') %>%
  group_by(sale_class) %>%
  summarize(sc = mean(price - mu - ti))

qplot(sc, data = sale_avg, bins = 10, color = I("blue"))
```




```
prediction2 <- temp %>%
  left_join(time_avg, by='time') %>%
  left_join(sale_avg, by='sale_class') %>%
  mutate(pred = mu + ti + sc) %>%
  pull(pred)
RMSE(prediction2, temp$price)
```

```
## [1] 13755.38
```

```
RMSE_time_and_sale_class_effect_model <- RMSE(prediction2, temp$price)

RMSE_results <- bind_rows(RMSE_results,
```

```
                          data.frame(Method="Time & Sale Class Effect Model",
                                     RMSE = RMSE_time_and_sale_class_effect_model))
RMSE_results %>% knitr::kable()
```

Method	RMSE
Naive Model	22632.10
Time Effect Model	14280.63
Time & Sale Class Effect Model	13755.38

3.4 Time, Sale and Market Class Effect Model

```
#Time, Sale and Market Class Effect model

market_avg <- edx %>%
  left_join(time_avg, by='time') %>%
  left_join(sale_avg, by='sale_class') %>%
  group_by(market_class) %>%
  summarize(ma = mean(price - mu - ti - sc))


prediction3 <- temp %>%
  left_join(time_avg, by='time') %>%
  left_join(sale_avg, by='sale_class') %>%
  left_join(market_avg, by="market_class") %>%
  mutate(pred = mu + ti + sc + ma) %>%
  pull(pred)
RMSE(prediction3, temp$price)
```

```
## [1] 13849
```

```
RMSE_time_and_sale_and_market_class_effect_model <- RMSE(prediction3, temp$price)

RMSE_results <- bind_rows(RMSE_results,
                          data.frame(Method="Time, Sale & Market Class Effect Model",
                                     RMSE = RMSE_time_and_sale_and_market_class_effect_model))
RMSE_results %>% knitr::kable()
```

Method	RMSE
Naive Model	22632.10
Time Effect Model	14280.63
Time & Sale Class Effect Model	13755.38
Time, Sale & Market Class Effect Model	13849.00

Surprisingly, including the market class variable has not improved the RMSE of the previous algorithm. Let's see the effect of including the Status variable.

3.5 Time, Sale, Market Class and Status Effect Model

```
#Time, Sale & Market Class and Status Effect Model



status_avg <- edx %>%
  left_join(time_avg, by='time') %>%
  left_join(sale_avg, by='sale_class') %>%
  left_join(market_avg, by='market_class') %>%
  group_by(status) %>%
  summarize(st = mean(price - mu - ti - sc - ma))

prediction4 <- temp %>%
  left_join(time_avg, by='time') %>%
  left_join(sale_avg, by='sale_class') %>%
  left_join(market_avg, by="market_class") %>%
  left_join(status_avg, by="status") %>%
  mutate(pred = mu + ti + sc + ma + st) %>%
  pull(pred)
RMSE(prediction4, temp$price)
```

```
## [1] 13853.88
```

```
RMSE_time_sale_market_class_and_status_effect_model <- RMSE(prediction4, temp$price)

RMSE_results <- bind_rows(RMSE_results,
                          data.frame(Method="Time, Sale, Market Class & Status Effect Model",
                                     RMSE = RMSE_time_sale_market_class_and_status_effect_model))
RMSE_results %>% knitr::kable()
```

Method	RMSE
Naive Model	22632.10
Time Effect Model	14280.63
Time & Sale Class Effect Model	13755.38
Time, Sale & Market Class Effect Model	13849.00
Time, Sale, Market Class & Status Effect Model	13853.88

Including the Status variable also has worsened the result obtained not by the previous model, but the third one (the best so far). We will include below the last of the variables, Vendor.

3.6 Time, Sale, Market Class, Status and Vendor Effect Model

```
#Time, Vendor, Sale & Market Class and Status Effect Model

vendor_avg <- edx %>%
  left_join(time_avg, by='time') %>%
  left_join(sale_avg, by='sale_class') %>%
```

```
  left_join(market_avg, by='market_class') %>%
  left_join(status_avg, by="status") %>%
  group_by(vendor) %>%
  summarize(ven = mean(price - mu - ti - sc - ma - st))


prediction5 <- temp %>%
  left_join(time_avg, by='time') %>%
  left_join(sale_avg, by='sale_class') %>%
  left_join(market_avg, by="market_class") %>%
  left_join(status_avg, by="status") %>%
  left_join(vendor_avg, by="vendor") %>%
  mutate(pred = mu + ti + sc + ma + st + ven) %>%
  pull(pred)
RMSE(prediction5, temp$price)
```

```
## [1] 13380.32
```

```
RMSE_time_sale_market_class_vendor_and_status_effect_model <- RMSE(prediction5, temp$price)

RMSE_results <- bind_rows(RMSE_results,
                          data.frame(Method="Time, Sale, Market Class, Status & Vendor Effect Model",
                                     RMSE = RMSE_time_sale_market_class_vendor_and_status_effect_model))
RMSE_results %>% knitr::kable()
```

Method	RMSE
Naive Model	22632.10
Time Effect Model	14280.63
Time & Sale Class Effect Model	13755.38
Time, Sale & Market Class Effect Model	13849.00
Time, Sale, Market Class & Status Effect Model	13853.88
Time, Sale, Market Class, Status & Vendor Effect Model	13380.32

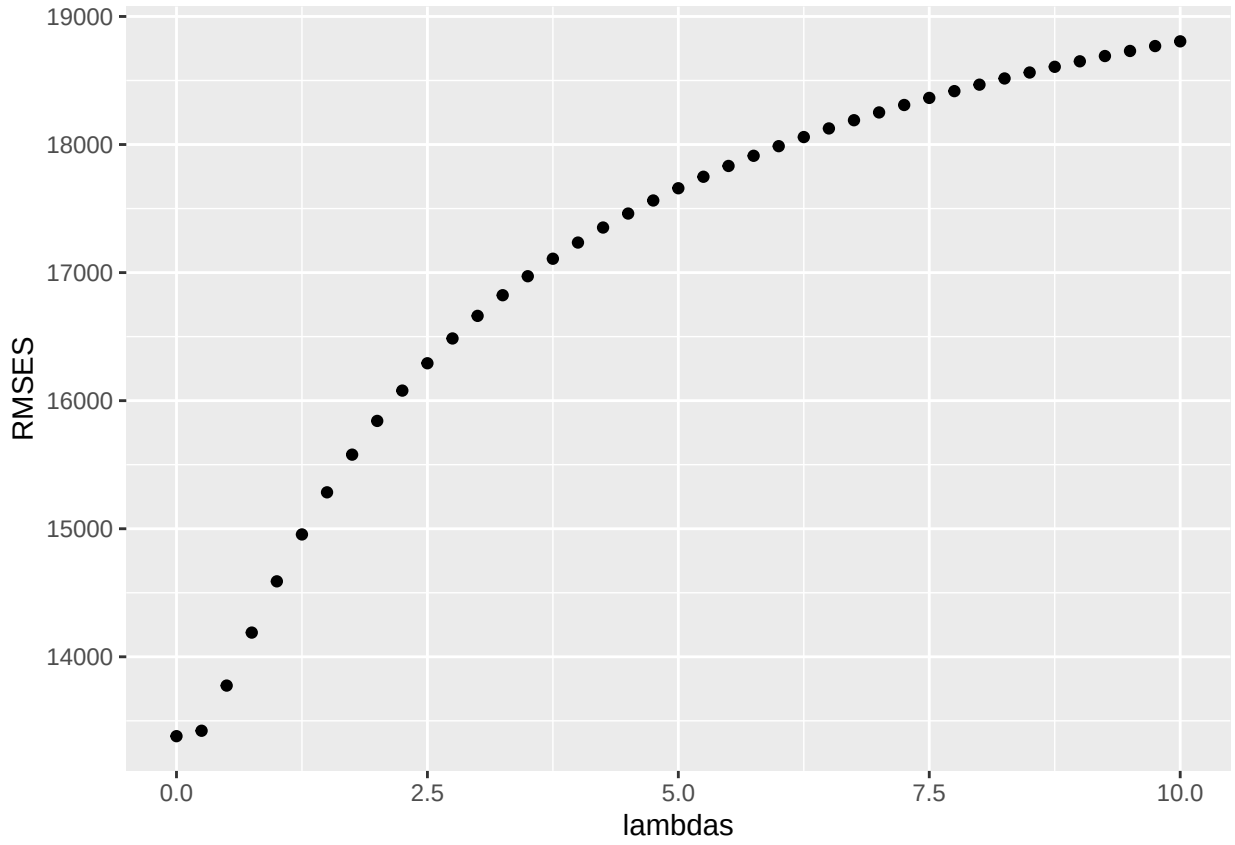
Including the Vendor variable along with the rest of the variables tested has resulted in obtaining the lowest value of RSME.

Finally, let's see if we can improve the result by applying the Regularization process.

3.6 Regularization of Time, Sale, Market Class, Status and Vendor Effect Model

```
#Regularization


lambdas <- seq(-0, 10, 0.25)
RMSES <- sapply(lambdas, function(l){
  mu <- mean(edx$price)
  ti <- edx %>%
    group_by(time) %>%
    summarize(ti = sum(price - mu)/(n()+l))
  sc <- edx %>%
    left_join(ti, by="time") %>%
```

```
    group_by(sale_class) %>%
    summarize(sc = sum(price - ti - mu)/(n()+l))
  ma <- edx %>%
    left_join(ti, by="time") %>%
    left_join(sc, by="sale_class") %>%
    group_by(market_class) %>%
    summarize(ma = sum(price - ti - sc - mu)/(n()+l))
  st <- edx %>%
    left_join(ti, by="time") %>%
    left_join(sc, by="sale_class") %>%
    left_join(ma, by="market_class") %>%
    group_by(status) %>%
    summarize(st = sum(price - ti - sc - ma - mu)/(n()+l))
  ven <- edx %>%
    left_join(ti, by="time") %>%
    left_join(sc, by="sale_class") %>%
    left_join(ma, by="market_class") %>%
    left_join(st, by="status") %>%
    group_by(vendor) %>%
    summarize(ven = sum(price - ti - sc - ma - st - mu)/(n()+l))

  predicted_prices <- temp %>%
    left_join(ti, by="time") %>%
    left_join(sc, by="sale_class") %>%
    left_join(ma, by="market_class") %>%
    left_join(st, by="status") %>%
    left_join(ven, by="vendor") %>%
    mutate(pred = ti + sc + ma + st + ven + mu) %>%
    pull(pred)
  return(RMSE(predicted_prices, temp$price))
})

qplot(lambdas, RMSES)
```




```
lambda <- lambdas[which.min(RMSES)]
lambda
```

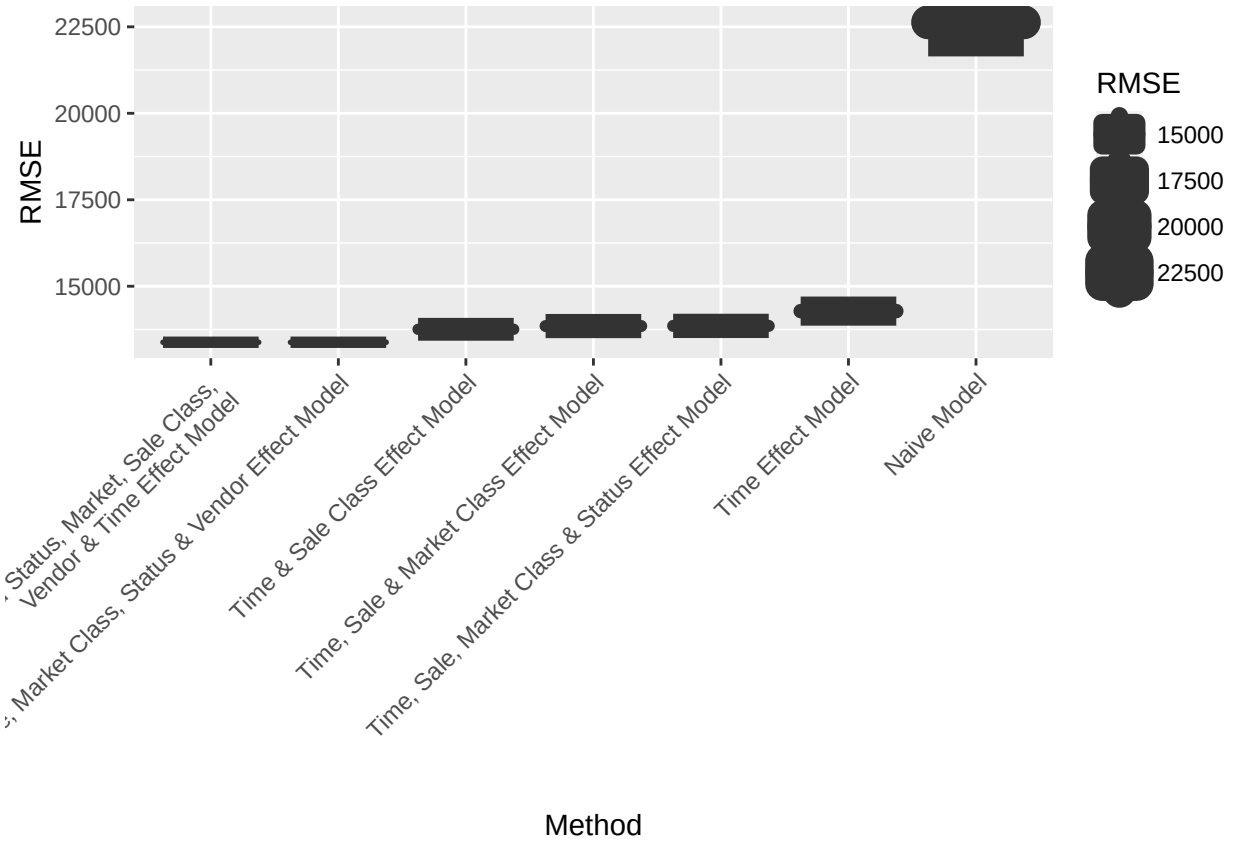
```
## [1] 0
```

```
RMSE_results <- bind_rows(RMSE_results,
                          data.frame(Method="Regularized Status, Market, Sale Class,
                                     Vendor & Time Effect Model", RMSE = min(RMSES)))
RMSE_results %>% knitr::kable()
```

Method	RMSE
Naive Model	22632.10
Time Effect Model	14280.63
Time & Sale Class Effect Model	13755.38
Time, Sale & Market Class Effect Model	13849.00
Time, Sale, Market Class & Status Effect Model	13853.88
Time, Sale, Market Class, Status & Vendor Effect Model	13380.32
Regularized Status, Market, Sale Class, Vendor & Time Effect Model	13380.32

```
#Boxplot comparative effect models

RMSE_results %>%  mutate(Method = reorder(Method, RMSE)) %>%
```

```
ggplot() +
geom_boxplot(aes(Method, RMSE, , size= RMSE)) + theme(axis.text.x = element_text(angle = 45, hjust =
```



4. CONCLUSION

We have achieved the lowest RSME result, $13,380.32, with the Time, Sale, Market Class, Status & Vendor Effect Model. The Regularization model has not improved the result of this one.

Therefore, we can conclude that the expected opportunity value is $13,380.32, for which we will have to take into account all the variables available in our dataset, namely, Time, Origin of the Client, Market & Sale class and Status.