Detecting shill bidders with Statistics

Simone Gastaldon, Martino Zaratin

1 Introduction

1.1 Shill bidding

Whoever is familiar with the context of online auctions might have witnessed, sometimes in his life, the uncomfortable experience of *shill bidding*. With the term *shill bidding* we refer to:

"the illegal practice of sellers who, during the auctions of their own products, place bids on their goods in order to drive up the price."

Usually the dishonest seller performs these actions in alliance with some accomplice of him, who is not interested in buying any product; his only purpose is rather to artificially raise the price of the auction. As already stated, nowadays this practice is considered illegal and many online marketplaces adopt strict regulations in order to avoid this phenomenon.

1.2 Outline

In this project, we will take a closer look to the practice of *shill bidding* in a statistical framework. We will consider a list of records about auctions that took place in one of the biggest e-commerce company worldwide. This dataset provides general information about the auctions, as well as specific details about the bidders and their behaviors. Our analysis is composed of the following steps:

1. **Data preprocessing.**
   This section consists in data filtering and cleaning. We analyze some of the global properties of our dataset and try to address issues such as data representation and encoding.

2. **Exploratory data analysis.**
   We will take a closer look to the most relevant variables via exploratory data analysis. We will provide plots and graphs in order to visualize features and relations. We will discuss the relevance of each variable, providing simple solutions when data is little informative.

3. **Regression methods.**
   In this section, we will proceed with the implementation of statistical regression methods to explore internal relationships. Specifically, our focus will be on studying the variable `Bidding_Ratio` based on the available features, including assessments of linearity and polynomial dependencies. Through rigorous testing and performance evaluations, we aim to gain insights into the suitability of these models for the regression task.

4. **Classification task.**
   We will consider a classification task: given the features describing the auction set and the behavior of a certain bidder, the task is to predict whether he is a legit bidder or an impostor, i. e. a *shill bidder*. We will implement different classification models, highlighting the mathematical assumpitons made during the process.

5. **Conclusions.**
   Finally we will draw conclusions: we will compare models with each other, focusing on pros and cons od such methods.

We begin by loading all the libraries we will need:

```
library(corrplot)
library(glmnet)
library(pROC)
library(MASS)
library(e1071)
library(car)
```

2 Data Preprocessing

2.1 Load the data

The dataset is available online at the UC Irvine Machine Learning Repository (here is a link to it). The creator scraped a large number of eBay auctions of a popular product and grouped all the relevant information in this file. It is composed of 6321 observations of 13 different variables.

```
Shill_bidding <- read.csv('Shill Bidding Dataset.csv')
```

Variables showcase different nature. Some of them provide general information about the setting of the auction, some others are related to the behavior of the bidders. Here follows a brief description of all variables:

- **Record_ID** : an integer identifying the record in the dataset.
- **Auction_ID**: an integer identifying the auction.
- **Bidder_ID**: a string which is the nickname of the bidder.
- **Bidder_Tendency**: a real number representing the tendency of a bidder to participate to auctions of few sellers rather than a diversified lot; this is a collusive act involving the fraudulent seller and an accomplice. The higher the value, the stronger the tendency.
- **Bidding_Ratio**: ratio that measures how frequently the bidder offers a new price. A shill bidder participates more frequently to raise the auction price and attract higher bids from legitimate participants.
- **Successive_Outbidding**: a real number showcasing the tendency of a bidder to raise its own bid. A shill bidder successively outbids himself even though he is the current winner to increase the price gradually with small consecutive increments.
- **Last_Bidding**: a real number displaying the tendency to be the last bidder. A shill bidder becomes inactive at the last stage of the auction (more than 90% of the auction duration) to avoid winning the auction.
- **Auction_Bids**: total number bids, with respect to the average. Auctions with SB activities tend to have a much higher number of bids than the average of bids in concurrent auctions.
- **Starting_Price_Average**: tendency to offer a small starting price. A shill bidder usually offers a small starting price to attract legitimate bidders into the auction.
- **Early_Bidding**: a real number displaying the activity during the first phase of the auction. A shill bidder tends to bid pretty early in the auction (less than 25% of the auction duration) to get the attention of auction users.
- **Winning_Ratio**: ratio between the number of auctions *lost* and the number of auctions attended. A shill bidder competes in many auctions but hardly wins any auctions.
- **Auction_Duration**: integer displaying how long an auction lasted.
- **Class**: a binary feature identifying the legitimacy of the bidder: 0 stands for normal bidding behavior, 1 stands for shill bidder.

2.2 Clean and filter the data

We take a first look at the features:

```
str(Shill_bidding)
```

```
## 'data.frame':    6321 obs. of  13 variables:
##  $ Record_ID              : int  1 2 3 4 5 8 10 12 13 27 ...
##  $ Auction_ID             : int  732 732 732 732 900 900 900 900 2370 600 ...
##  $ Bidder_ID              : chr  "_***i" "g***r" "t***p" "7***n" ...
##  $ Bidder_Tendency        : num  0.2 0.0244 0.1429 0.1 0.0513 ...
##  $ Bidding_Ratio          : num  0.4 0.2 0.2 0.2 0.222 ...
##  $ Successive_Outbidding  : num  0 0 0 0 0 0 0 1 1 0.5 ...
##  $ Last_Bidding           : num  2.78e-05 1.31e-02 3.04e-03 9.75e-02 1.32e-03 ...
##  $ Auction_Bids           : num  0 0 0 0 0 ...
##  $ Starting_Price_Average: num  0.994 0.994 0.994 0.994 0 ...
##  $ Early_Bidding          : num  2.78e-05 1.31e-02 3.04e-03 9.75e-02 1.24e-03 ...
##  $ Winning_Ratio          : num  0.667 0.944 1 1 0.5 ...
##  $ Auction_Duration       : int  5 5 5 5 7 7 7 7 7 7 ...
##  $ Class                  : int  0 0 0 0 0 0 0 1 1 1 ...
```

A few remarks come immediately to mind:

1. The variable `Record_ID` is useless and hence we can remove it.
2. There are no missing values:

```
sum(is.na(Shill_bidding))
```

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

3. All numerical variables have already been scaled to the interval $[0, 1]$. This is not relevant for predictive purposes since any regression model is invariant to scaling; however it is useful for data visualization and comparisons.

2.2.1 High cardinality categorical features

In the feature list there are two variables that require special attention: `Auction_ID` and `Bidder_ID`.

- **`Auction_ID`**: even though this is an integer-valued variable, this is not a quantitative variable. Integers shown do not refer to a numerical scale, since those numbers are simply used as ID's to identify auctions. Hence this should be considered as a categorical variable. However, one can check that this variable takes 807 different values across all observations, making it a *high cardinality categorical feature*. A one-hot encoding of all the possible values is simply unfeasible, since it would increase dramatically the number of features in the dataset.
- **`Bidder_ID`**: this is another high cardinality categorical feature. One can see that there are 1054 unique bidder ID's across the whole dataset. Once again, a one-hot encoding strategy would be a failure: this would require us to introduce 1053 variables to the list, making both computation and storage unfeasible..

A possible solution to this problem is by using a simple **aggregation function**: given a high cardinality categorical variable,

1. we calculate the relative frequency of each level;
2. we order levels by frequency;
3. we decide some threshold and group different levels into the same bin. The bins, which have a low cardinality, represent the new categorical features.

For example, taking in consideration the feature `Bidder_ID`, the frequency of each level is given by `table(Bidder_ID)`:

```
head(table(Shill_bidding$Bidder_ID))
```

```
##
## -***- -***_ -***9 -***a -***d -***e
##     5     1     1    11     1    10
```

We decided to group observations into three bins ('Low', 'Mid' and 'High' frequency). For thresholds, we chose two separation points lying half a standard deviation away from the mean:

```
sprintf('thresholds: % s, % s',
  round(mean(table(Shill_bidding$Bidder_ID)) - 0.5*sd(table(Shill_bidding$Bidder_ID))),
  round(mean(table(Shill_bidding$Bidder_ID)) + 0.5*sd(table(Shill_bidding$Bidder_ID))))
```

```
## [1] "thresholds: 2, 10"
```

We introduce two new variables, `Auction_size` and `Bidder_popularity`, which measure frequency of `Auction_ID` and `Bidder_ID`, respectively.

```
# get frequency columns:
Auction_ID <- Shill_bidding$Auction_ID
Auction_size <- rep(0, 6321)
sorted <- sort(unique(Auction_ID))
tb <- table(Auction_ID)
for (i in 1:length(sorted)) {
  Auction_size[Auction_ID == sorted[i]] <- tb[i]
}

Bidder_ID <- Shill_bidding$Bidder_ID
Bidder_popularity <- rep(0, 6321)
sorted <- sort(unique(Bidder_ID))
tb <- table(Bidder_ID)
for (i in 1:length(sorted)) {
  Bidder_popularity[Bidder_ID == sorted[i]] <- tb[i]
}
```

Now we transform these variables into actual categorical variables, using frequency thresholds:

```
head(Bidder_popularity)
```

```
## [1] Med  High Med  Med  High High
## Levels: Low Med High
```

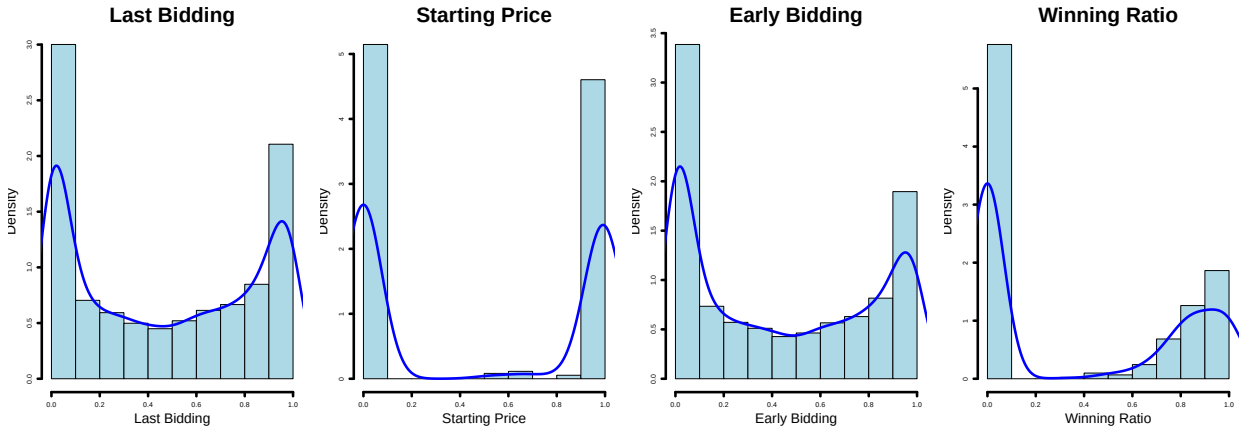
What we have done was to decrease the cardinality of the original categorical variables via frequency binning. This streamlines the data representation without compromising its readability, while maintaining the main information.
Finally we introduce the new variables and remove the old ones:

```
Shill_bidding$Auction_size <- Auction_size
Shill_bidding$Bidder_popularity <- Bidder_popularity
Shill_bidding <- Shill_bidding[4:15]
attach(Shill_bidding)
```

3 Exploratory Data Analysis

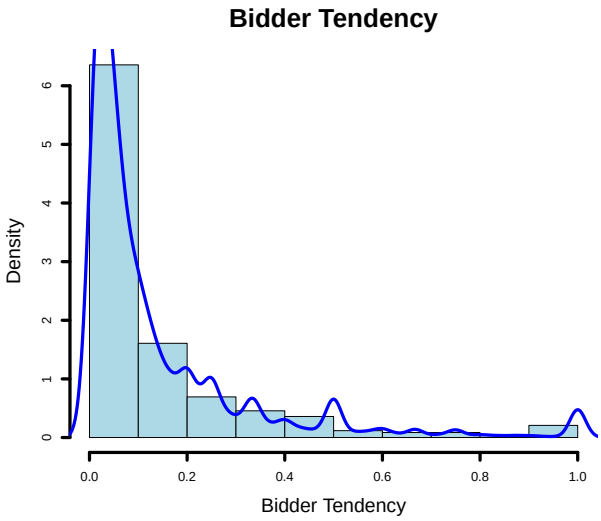
3.1 Continuous numerical variables

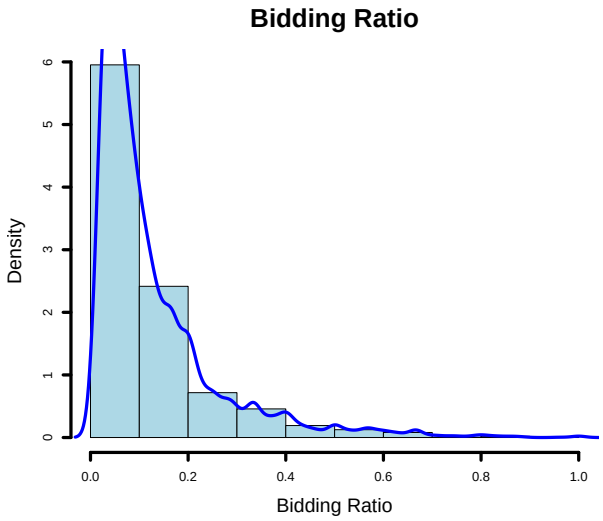
We begin by plotting histograms for the variables `Last Bidding`, `Starting Price`, `Early Bidding`, `Winning Ratio`.



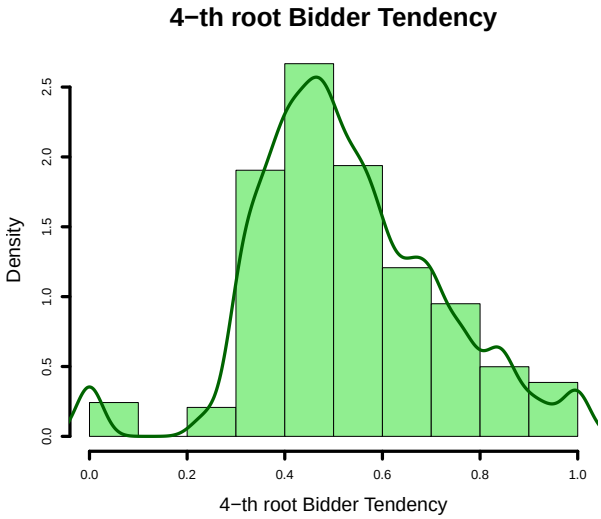
We notice that all these variables show a **bimodal distribution**. In particular, we notice high frequencies for values at the extremes of the range, i. e. 0 and 1.

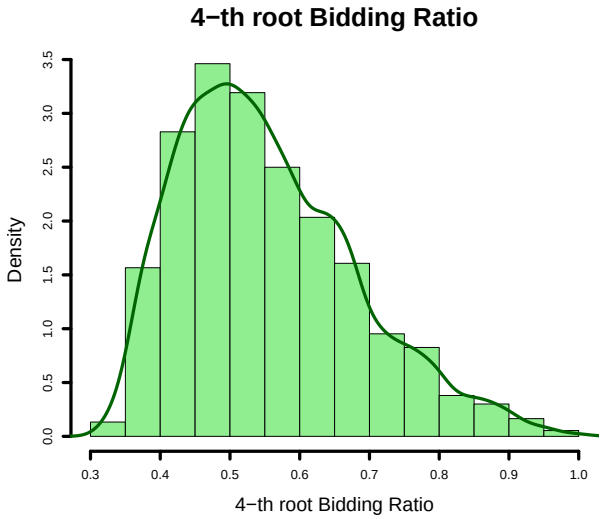
Let's plot now the variables `Bidder_Tendency` and `Bidding_Ratio`.





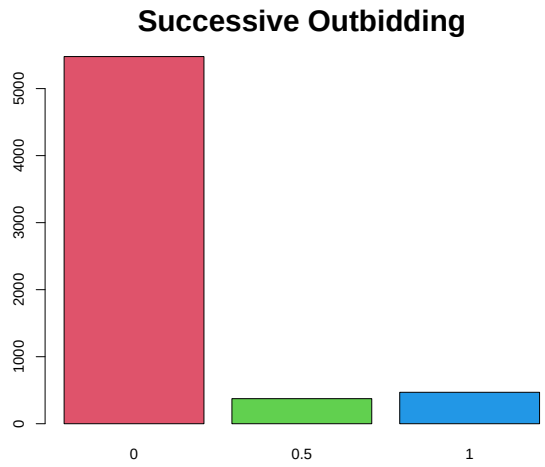
These variables instead showcase a **unimodal** distribution, with a very high density for values at the left extreme of the range. As a consequence, all these variables are heavily positive skewed.
A way to address this problem is by applying a transformation to the data points. We emphasize the fact that normality of the data is not a necessary condition for the models we are dealing with, like regression models. However, having data that is normally distributed can contribute to get better results.
As transformation, we decided to apply the *fourth root*: $x \mapsto \sqrt[4]{x}$. This way, the range of our variables, $[0, 1]$ will not be scaled or shifted.

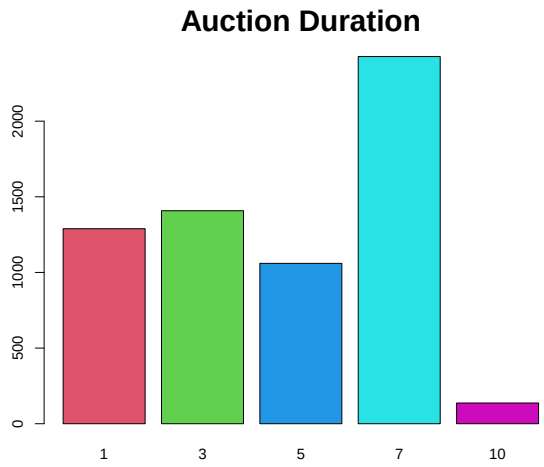




3.2 Discrete numerical variables

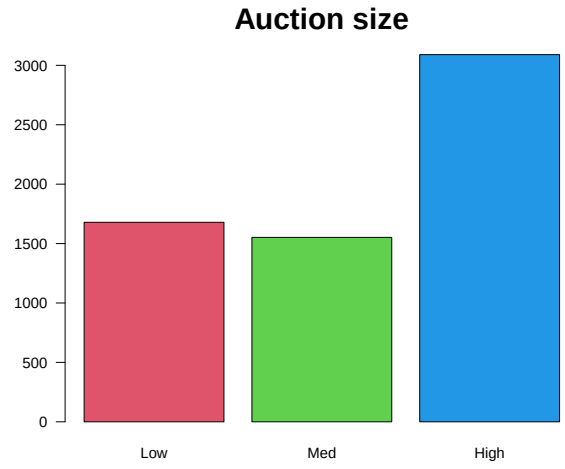
Among our numerical variables, two of them take discrete values: `Successive_Outbidding` and `Auction_duration`. These values and their frequencies are shown here:

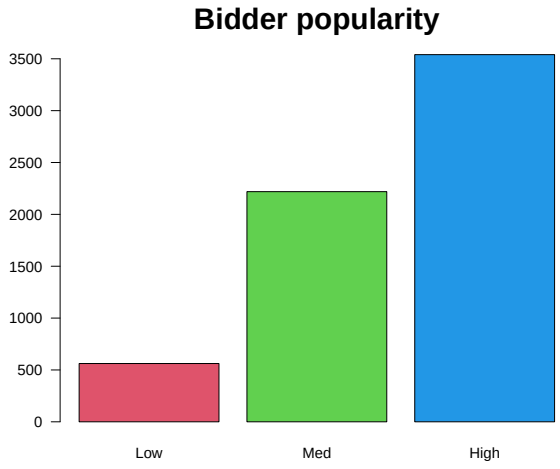




3.3 Categorical variables

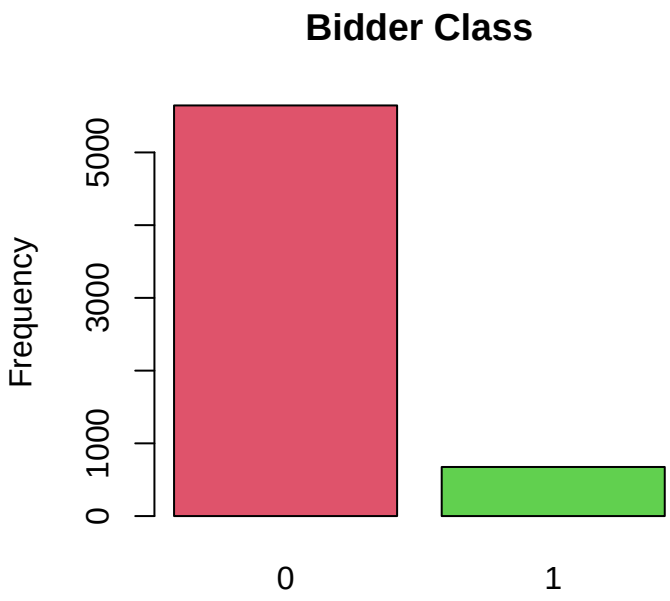
The two categorical variables in our dataset are the ones we created before. The frequency of each factor is related to the way we imposed our thresholds in the category grouping strategy.





3.4 Binary variables

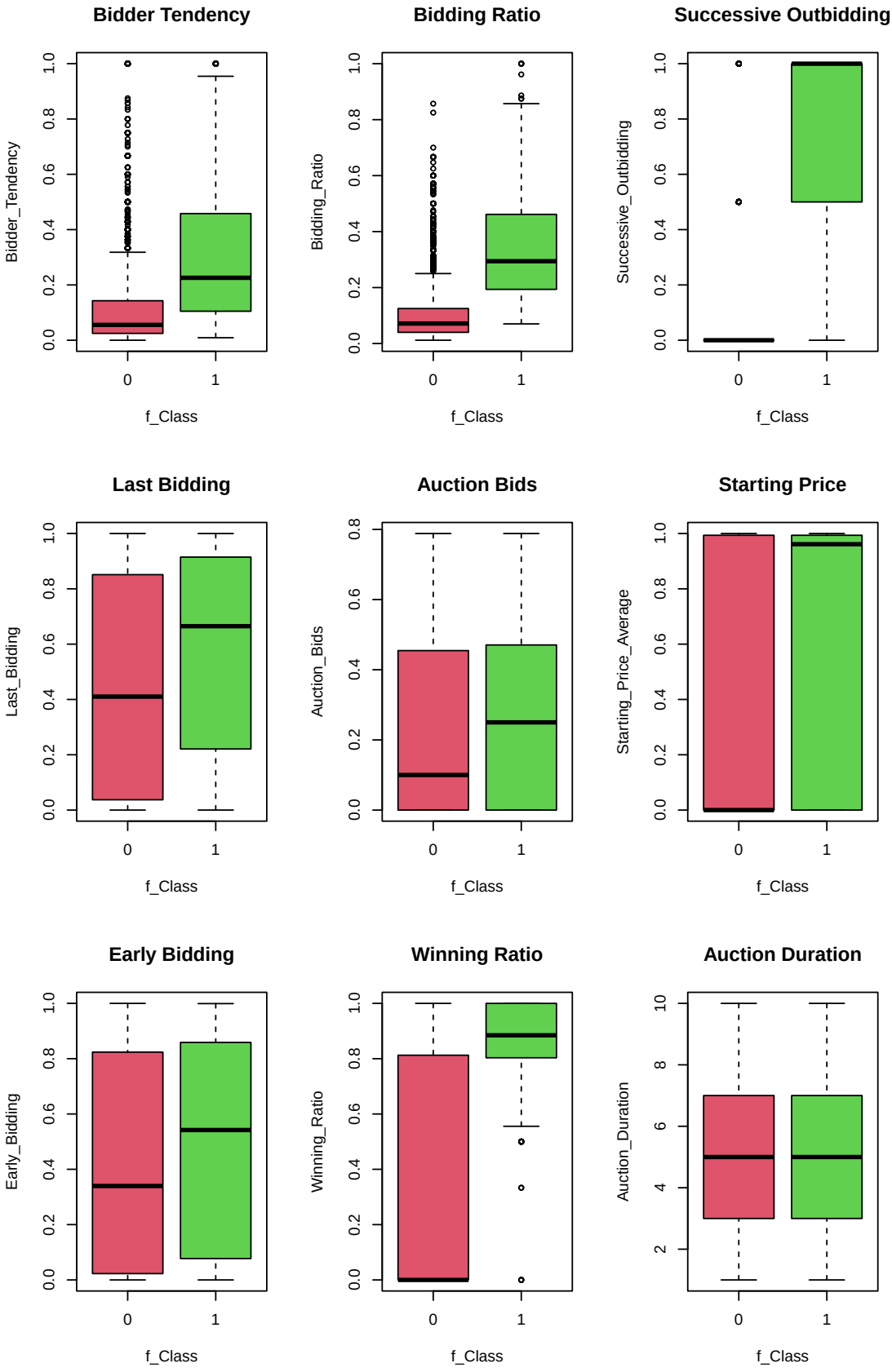
The variable `Class`, which is associated to the classification of bidders, is binary.



We notice that the two classes are unbalanced. We keep this in mind since this might compromise the quality of our classification. After the classification task, we will discuss about the class imbalance.

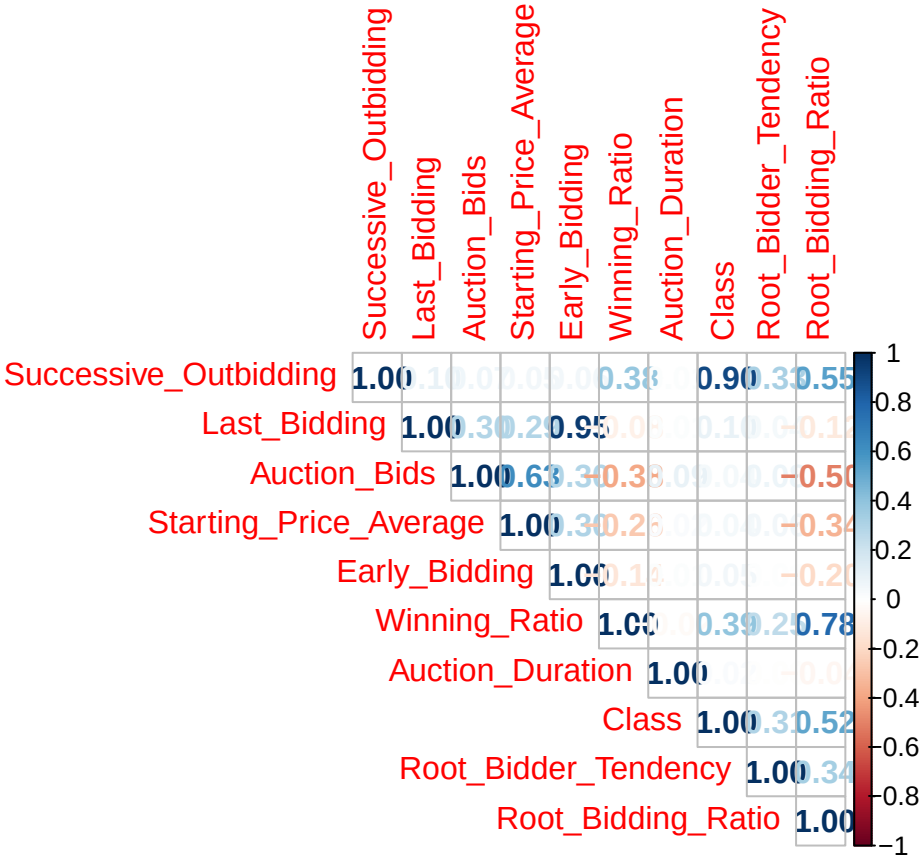
3.5 Relations with Class

Since our goal will be to distinguish between honest bidders and shill bidder, we visualize now the relations between each variable and the the classes of bidders.


Bidder Tendency
Bidder_Tendency
f_Class
Bidding Ratio
Bidding_Ratio
f_Class
Successive Outbidding
Successive_Outbidding
f_Class
Last Bidding
Last_Bidding
f_Class
Auction Bids
Auction_Bids
f_Class
Starting Price
Starting_Price_Average
f_Class
Early Bidding
Early_Bidding
f_Class
Winning Ratio
Winning_Ratio
f_Class
Auction Duration
Auction_Duration
f_Class

3.6 Correlation plot

We end up this section with the correlation plot of all the numerical variables:



4 Regression models

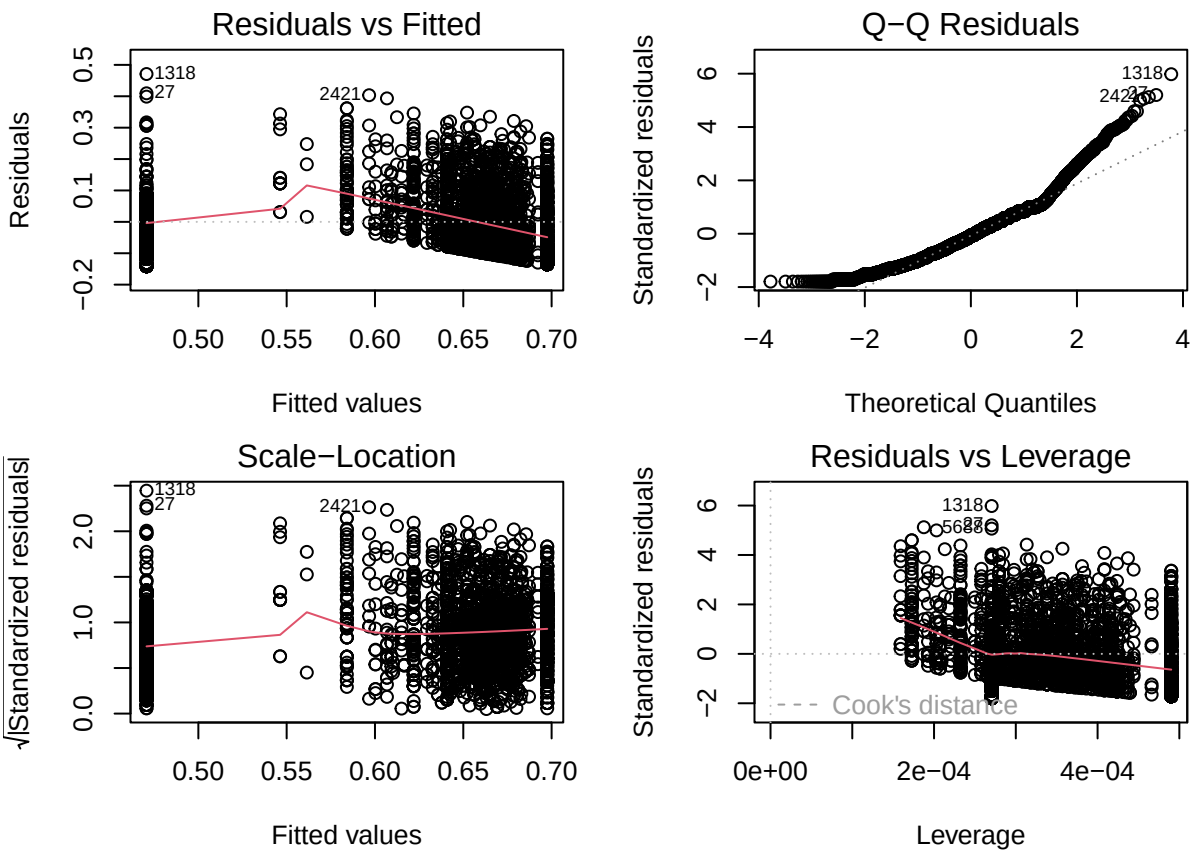
In this chapter, we aim to develop a robust model that helps us understand which features influence how frequently the bidder offers a new price. Our approach begins with a simple regression model which we will incrementally develop using various statistical methods. We will identify the optimal model based on the highest R-squared value and include the most significant variables for predicting `Bidding_Ratio`.

4.1 Simple Linear Regression

We start to initialise a simple regression model, which predicts `Bidding_Ratio` using only `Winning_Ratio` which is the highest correlated to our variable to predict. This is the reason that we start with this variable `Winning_Ratio`.

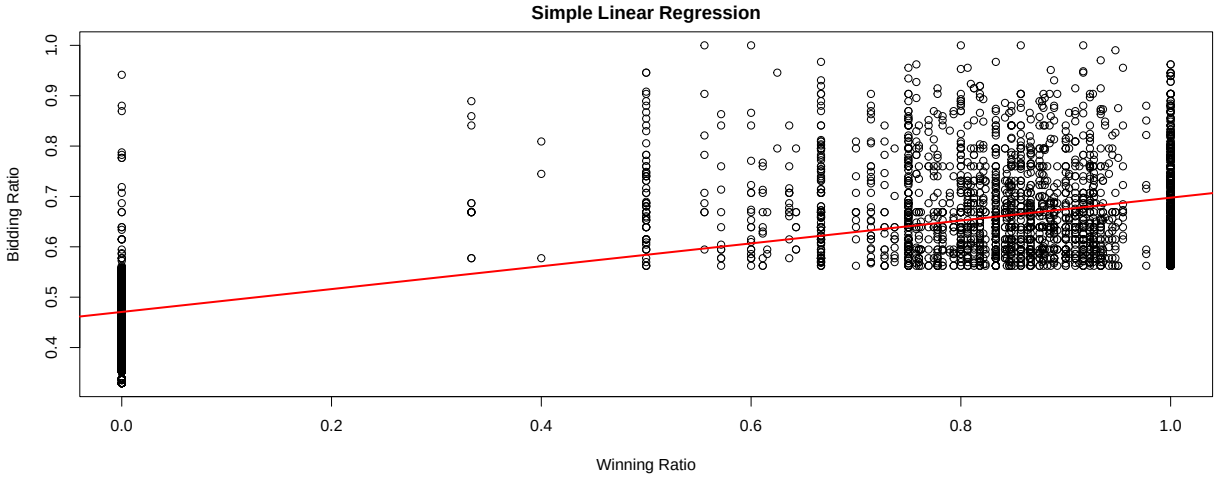
```
simple_model <- lm(Root_Bidding_Ratio~Winning_Ratio, data = Shill_bidding)
R2_simple <- summary(simple_model)$r.squared
summary(simple_model)
```

```
##
## Call:
## lm(formula = Root_Bidding_Ratio ~ Winning_Ratio, data = Shill_bidding)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.14131 -0.05742 -0.00664  0.04632  0.47084
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.470655   0.001295   363.4   <2e-16 ***
## Winning_Ratio 0.226865   0.002269   100.0   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07874 on 6319 degrees of freedom
## Multiple R-squared:  0.6128, Adjusted R-squared:  0.6127
## F-statistic:  9999 on 1 and 6319 DF,  p-value: < 2.2e-16
```



Since this is a regression model with only one variable, we plot the regression line on a scatter plot to see how well the model fits the data.

```
par(mfrow = c(1,1), mar=c(4,4,2,2))
plot(Winning_Ratio, Root_Bidding_Ratio,
     xlab = "Winning Ratio", ylab = "Bidding Ratio",
     main = "Simple Linear Regression")
abline(simple_model, col = "red", lwd = 2)
```



```
par(mfrow=c(1,1))
```

Instead of a not-good R-squared 0.61, this regression model doesn't fit the data well.

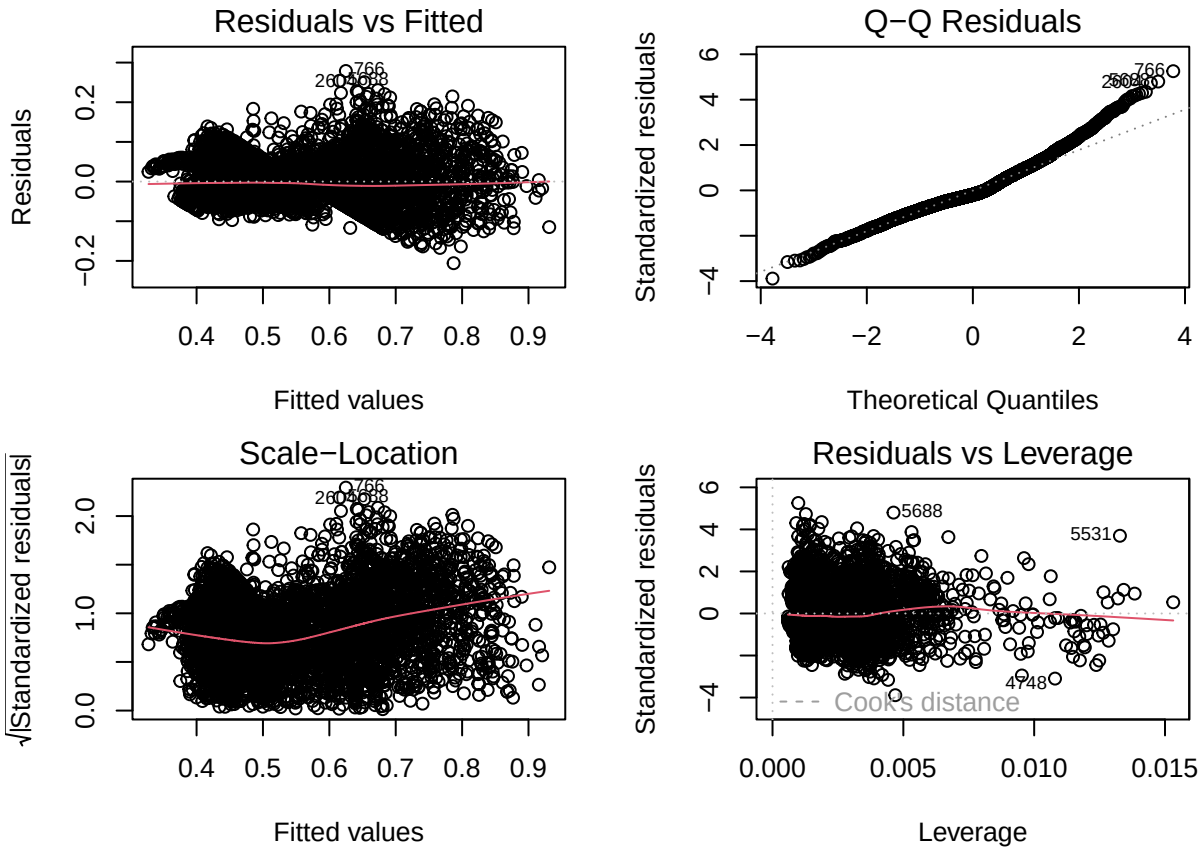
4.2 Multiple Linear Regression

Given the limitations of a single predictor, we decide to develop a multiple regression model utilizing all variables within the dataset. This approach yielded an R-squared score of 0.82.

```
multi_model <- lm(Root_Bidding_Ratio ~ ., data = Shill_bidding)
R2_mlr <- summary(multi_model)$r.squared
R2_mlr
```

```
## [1] 0.8244813
```

```
par(mfrow = c(2, 2), mar = c(4, 4, 2, 2))
plot(multi_model)
```



The multiple linear regression model has demonstrated substantial improvement compared to the simple linear regression model, based on our evaluation criteria. However, we have observed high correlation among certain variables in the current model. Therefore, our decision to remove variables exhibiting high multicollinearity (VIF > 5) is aimed at constructing an optimized model using the remaining variables.

```r
vif_values <- vif(multi_model)
vif_values > 5
columns_to_remove <- c("Last_Bidding", "Early_Bidding")
Shill_Bidding_reduced <- Shill_bidding[, !names(Shill_bidding) %in% columns_to_remove]
```

```r
multi_model1 <- lm(Root_Bidding_Ratio ~ ., data = Shill_Bidding_reduced)
R2_mlr1 <- summary(multi_model1)$r.squared
R2_mlr1
```

```
## [1] 0.8136446
```

Upon reviewing the R-squared score, it is evident that the multiple regression model did not exhibit any improvement over the previously best-performing model, as the R-squared decreased by 0.1. Given the potential implications of collinearity on the stability and interpretability of the model, a decision has been made to exclude the variables Department and `Early_Bidding` and `Last_Bidding` from further analysis.

4.3 Polynomial regression

In order to account for potential complex and non-linear interactions between variables, we plan to employ a polynomial regression model. This will involve incorporating the square of each of the presently selected features as predictors with the aim of enhancing the model's adherence to the data.

```r
model_pr <- lm(Root_Bidding_Ratio ~
               poly(Root_Bidder_Tendency, 2, raw = TRUE) +
               poly(Successive_Outbidding, 2, raw = TRUE) +
               poly(Auction_Bids, 2, raw = TRUE) +
               poly(Starting_Price_Average, 2, raw = TRUE) +
               poly(Winning_Ratio, 2, raw = TRUE) +
               poly(Auction_Duration, 2, raw = TRUE) +
               poly(Class, 2, raw = TRUE) +
               poly(Auction_size, 2, raw = TRUE) +
               poly(Bidder_popularity, 2, raw = TRUE),
               data = Shill_Bidding_reduced)
R2_pr <- summary(model_pr)$r.squared
R2_pr
```

```
## [1] 0.8348606
```

The polynomial regression model utilizing all variables yielded an improvement of 0.1 in the R-squared value compared to the previous model, demonstrating enhanced predictive capability.

We employ backward feature selection to enhance the polynomial regression model, systematically eliminating the variable that contributed the most to minimizing the AIC score.

```r
back_model <- step(model_pr, direction = "backward")
```

```r
R2_back <- summary(back_model)$r.squared
R2_back
```

```
## [1] 0.8348189
```

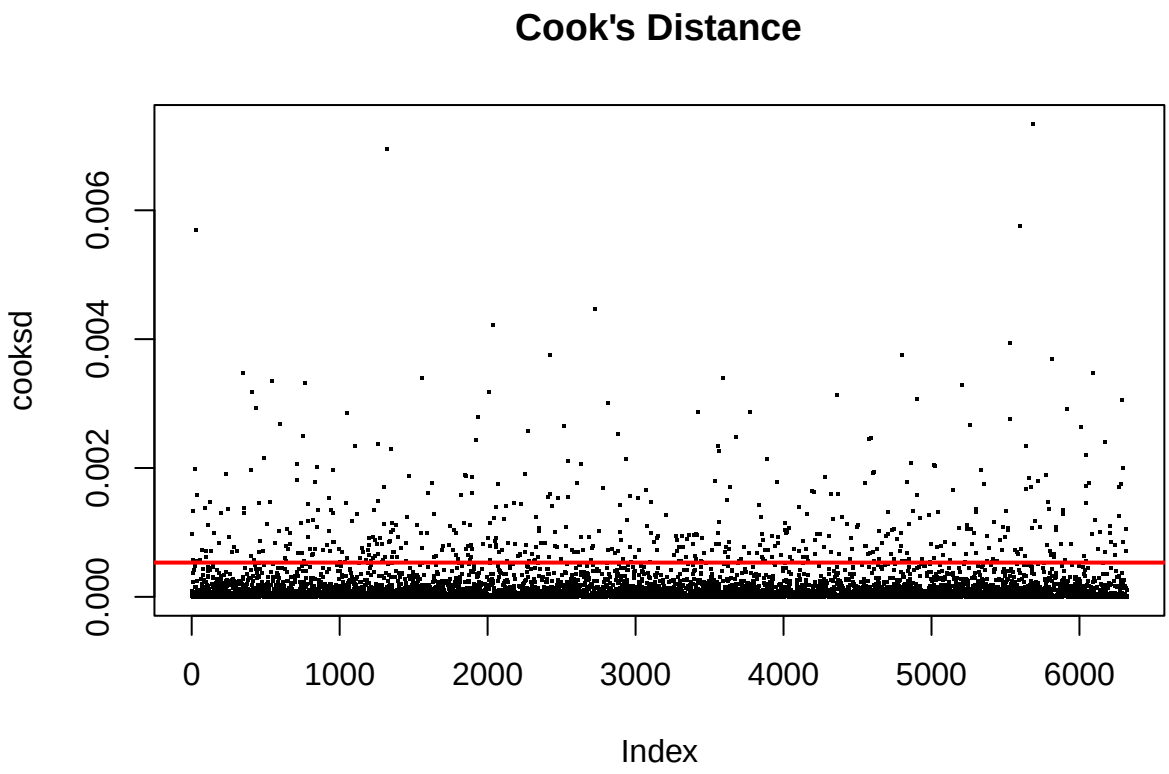
```r
model_pr1 <- lm(Root_Bidding_Ratio ~ poly(Root_Bidder_Tendency, 2, raw = TRUE) +
                  poly(Successive_Outbidding, 2, raw = TRUE) +
                  poly(Auction_Bids, 2, raw = TRUE) +
                  poly(Starting_Price_Average, 2, raw = TRUE) +
                  poly(Winning_Ratio, 2, raw = TRUE) +
                  poly(Auction_size, 2, raw = TRUE) +
                  poly(Bidder_popularity, 2, raw = TRUE), data=Shill_Bidding_reduced)
R2_pr1 <- summary(model_pr1)$r.squared
R2_pr1
```

```
## [1] 0.8348189
```

4.4 Final Model

As a final step, influential points with a Cook's distance exceeding three times the mean Cook's distance across all observations were identified and removed from the analysis.

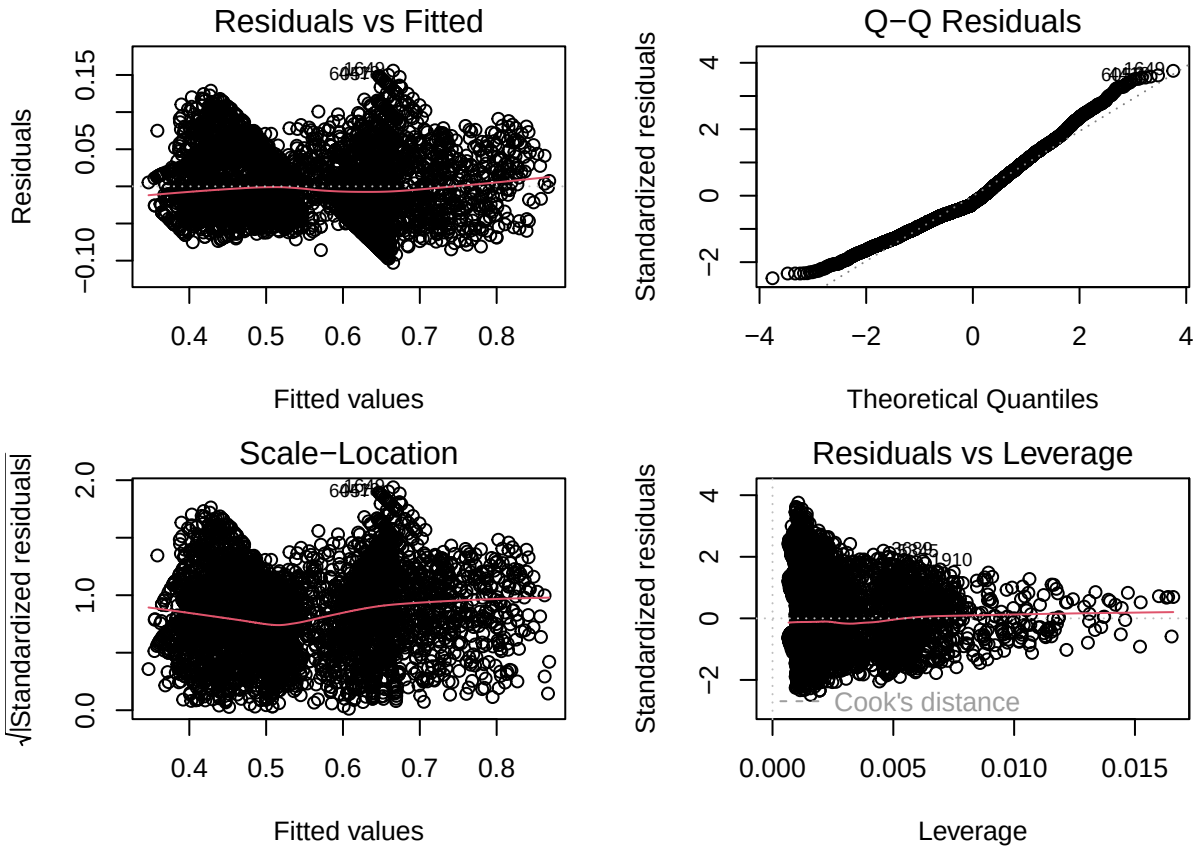
```r
cooksd <- cooks.distance(model_pr1)
par(mfrow = c(1, 1))
plot(cooksd, pch = ".", cex = 2, main = "Cook's Distance")
abline(h = 3 *mean(cooksd, na.rm=TRUE), col='red', lwd=2)
```



```
influential <- which(cooksd > 3*mean(cooksd, na.rm=TRUE))
final_Shill_Bidding <- Shill_Bidding_reduced[-influential, ]
```

```
final_model <- lm(Root_Bidding_Ratio ~ poly(Root_Bidder_Tendency, 2, raw = TRUE) +
                    poly(Successive_Outbidding, 2, raw = TRUE) +
                    poly(Auction_Bids, 2, raw = TRUE) +
                    poly(Starting_Price_Average, 2, raw = TRUE) +
                    poly(Winning_Ratio, 2, raw = TRUE) +
                    poly(Auction_size, 2, raw = TRUE) +
                    poly(Bidder_popularity, 2, raw = TRUE), data=final_Shill_Bidding)
final_R2 <- summary(final_model)$r.squared
final_R2
```

```
## [1] 0.8689815
```



We attain an R-squared value of 0.86, signifying the success of our model as the most effective one.

5 Classification models

In this section we face a binary classification task. The goal is to build a classifier which is able to predict whether a certain bidder is a honest one or a *shill bidder*. In order to do so, we adopt a supervised learning approach: we separate all the samples into two classes based on the values of the variable `Class`. Then we train a classifier built with a certain number of predictor making use of the information provided by the remaining variables in our dataset. The training procedure is aimed to find the values of predictors which minimize a certain loss function, which varies for each model.

In our analysis, we will go over four different classification models: Logistic Regression, Linear Discriminant Analysis, Quadratic Discriminant Analysis and Naive Bayes. We will evaluate the performance of each one of them with certain criteria and finally make comparisons between models.

5.1 Logistic regression

The first model we consider is a Logistic classifier with 15 predictors, one for each independent variable. In order to reduce the number of predictors, we tested a feature selection method: backward substitution with BIC criterion. However, this resulted in poor results, since almost all predictors resulted in being eliminated. Hence we chose to implement **Lasso Logistic regression** with the library `glmnet`. We chose Lasso to encourage sparsity in the solution. To invoke a Logistic Classifier, the option `family = "binomial"` must be selected in the function `glmnet`.Looking at the library description, we can see that the objective function for logistic regression is the penalized negative binomial log-likelihood:

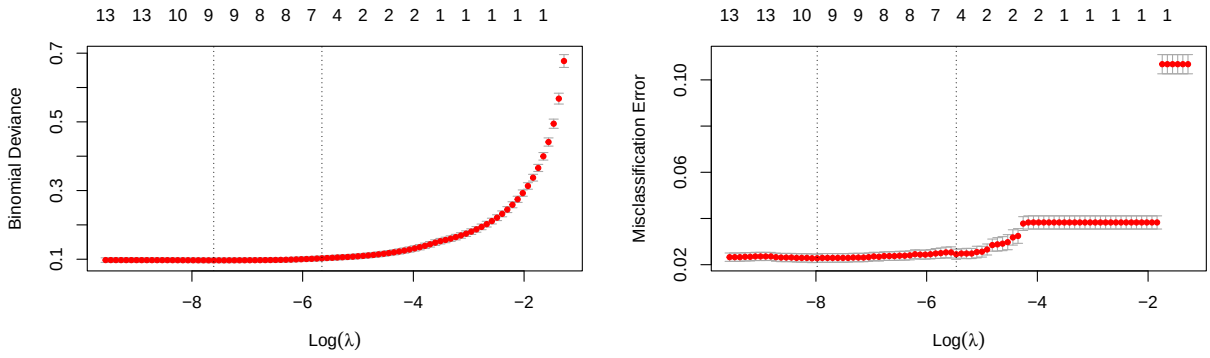
$$\min_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} -\Big[\frac{1}{N}\sum_{i=1}^{N} y_i(\beta_0 + x_i^T\beta) - \log(1+\exp(\beta_0 + x_i^T\beta))\Big] + \lambda[||\beta||_1]$$

The function `glmnet` automatically builds multiple versions of our model with different values for the hyper-parameter λ. To choose the one that guarantees the best result we adopt a **Cross Validation** method, which is already implemented via `cv.glmnet`. The default number of folds is set to 10.

```
X <- model.matrix(~ .-1, data = Shill_bidding[,-8])
y <- Shill_bidding[,8]

set.seed(123)
cvfit <- cv.glmnet(X, y, family = "binomial")
```

The default method to select the hyper-parameter is by minimizing the Binomial Deviance. WAnother possibility is to select the best hyper-parameter based on the misclassification error, i. e. the hyper-parameter that guarantees the best accuracy. This is set via the option `type.measure = "class"`.



The dotted lines in the graph above indicate two special values: `lambda.min` is the value of λ that gives minimum mean cross-validated error; `lambda.1se` is the value of λ that gives the most regularized model such that the cross-validated error is within one standard error of the minimum.

```
coef(cvfit, s = "lambda.min")
```

```
## 15 x 1 sparse Matrix of class "dgCMatrix"
##                                      s1
## (Intercept)            -12.45981376
## Successive_Outbidding    9.74113567
## Last_Bidding             0.20421399
## Auction_Bids             .
## Starting_Price_Average   .
## Early_Bidding            .
## Winning_Ratio            4.56074932
## Auction_Duration         0.08380566
## Auction_sizeLow         -0.75424927
## Auction_sizeMed          .
## Auction_sizeHigh         0.45167185
## Bidder_popularityMed    -0.06475875
## Bidder_popularityHigh    .
## Root_Bidder_Tendency     0.58930366
## Root_Bidding_Ratio       4.67890585
```

Looking at the coefficients we can see the sparsity properties of the solution.

Time for predictions:

```
lasso.prob <- predict(cvfit, newx = X, s = "lambda.min", type = 'response')
n <- length(lasso.prob)

lasso.pred <- rep("No", n)
lasso.pred[lasso.prob>0.5] <- "Yes"
```

The confusion matrix looks like this:

```
##           Class
## lasso.pred    0    1
##        No  5560   53
##        Yes   86  622
```

5.2 Linear Discriminant Analysis

LDA makes more assumptions about the underlying data then the logistic regression: in particular, the most relevant one is the normality of independent variables, given each class.
Let's test its effectiveness. We decided to use the same predictors as before. This is because we would like to compare models between each other in the end of the project.

```
Selected_variables <- X[, c(1,2, 6, 7, 8, 10, 11, 13, 14)]
lda.fit <- lda(Class ~ Selected_variables)

lda.pred <- predict(lda.fit, Shill_bidding)
```

When it comes to prediction, we obtain the following confusion matrix:

```
##    Class
##        0    1
##   0 5480    2
##   1  166  673
```

The misclassified sample is pretty much the same as before. However, we notice that LDA commits a very little I-type error.
We observe that if we raise threshold for the decision boundary, we can further improve the accuracy (i. e. the misclassification error). This comes at a reasonable cost, since I-type error will increase too:

```
lda.class <- rep(0, 6321)
lda.class[lda.pred$posterior[,2]>= 0.7] <- 1

lda2.conf.matrix <- table(lda.class, Class)
lda2.conf.matrix
```

```
##          Class
## lda.class    0    1
##         0 5515    6
##         1  131  669
```

5.3 Quadratic Discriminant Analysis

With QDA we get rid of the assumption of homoscedasticity between classes. Let's verify if this improves the quality of the predictions:

```
qda.fit <- qda(Class ~ Selected_variables, data = Shill_bidding)

qda.class <- predict(qda.fit, Shill_bidding)$class

qda.conf.matrix <- table(qda.class, Class)
qda.conf.matrix
```

```
##          Class
## qda.class    0    1
##         0 5524    4
##         1  122  671
```

The results are pretty good and the best so far. In the conclusion we will try to tell reasons why this model works better than the previous ones.

5.4 Naive Bayes

The main assumption underlying Naive Bayes is the independence of predictors. Once again we report here the confusion matrix, while delaying the discussion of the remaining performance measures to the following section.

```
nb.fit <- naiveBayes(Class ~ Successive_Outbidding +
                             Last_Bidding +
                             Winning_Ratio +
                             Auction_Duration +
                             Auction_size_low +
                             Auction_size_high +
                             Bidder_popularity_med +
                             Root_Bidder_Tendency +
                             Root_Bidding_Ratio, data=Shill_bidding)

nb.class <- predict(nb.fit, Shill_bidding)
nb.conf.matrix <- table(nb.class, Class)
nb.conf.matrix
```

```
##          Class
## nb.class    0    1
##        0 5526    4
##        1  120  671
```

6 Conclusions

6.1 Regression

Given that we found the final regression model to be the most accurate, we decide to compare the Mean Squared Error (MSE) on the dataset between the initial and final model.

```
base.pred <- predict(simple_model, newdata = Shill_bidding)
base.mse <- mean((Shill_bidding$Root_Bidding_Ratio - base.pred)^2)
print(paste("MSE of the initial model on test set:", base.mse))
```

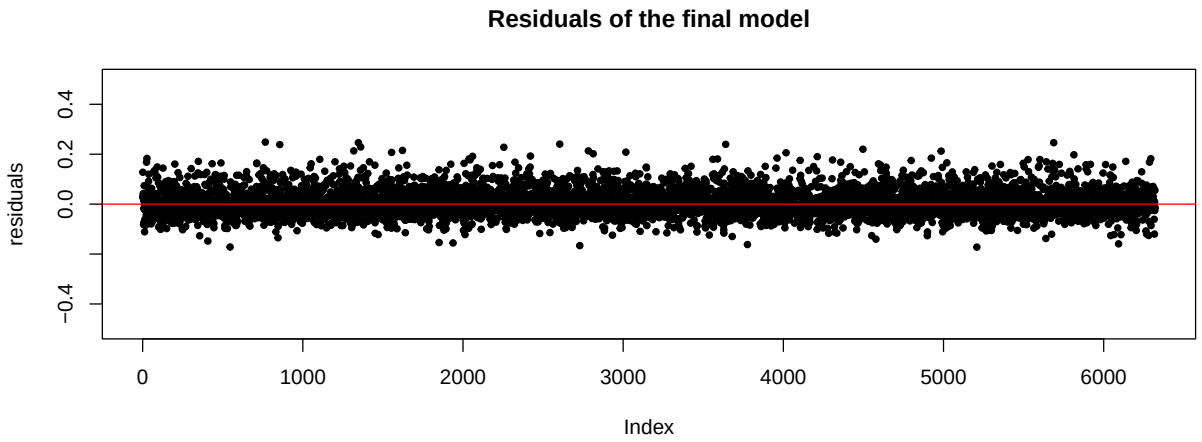
```
## [1] "MSE of the initial model on test set: 0.00619834316189899"
```

```
final.pred <- predict(final_model, newdata = Shill_bidding)
final.mse <- mean((Shill_bidding$Root_Bidding_Ratio - final.pred)^2)
print(paste("MSE of the final model on test set:", final.mse))
```

```
## [1] "MSE of the final model on test set: 0.00268825791879931"
```

Additionally, we showcase the Residuals of the final model.

```
residuals <- Shill_bidding$Root_Bidding_Ratio - final.pred
par(mfrow = c(1, 1))
plot(residuals, ylim=c(-0.5,0.5), pch = 20, main = "Residuals of the final model")
abline(h = 0, col = "red")
```



Model	MSE	R-squared
Simple Model	0.006	0.61
Final Model	0.002	0.86

The final model illustrates a reduction in Mean Squared Error (MSE) and an increase in the R-squared value compared to the initial model.

6.2 Classification

First of all let's compare some performance measures for all the classification models:

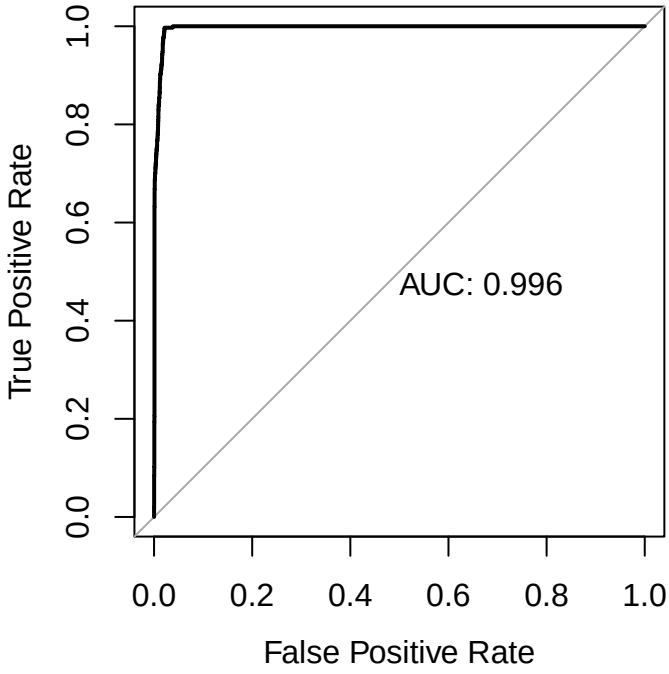
Table 2: Performance measures

Measure	Logistic Regression	LDA	LDA high thresh.	QDA	Naive Bayes
Error	0.0219	0.0265	0.0216	0.0199	0.0196
PPV (Precision)	0.8785	0.8021	0.8362	0.8461	0.8482
TPR (Recall/ Sensitivity)	0.9214	0.9970	0.9911	0.9940	0.9940
F1 score	0.8994	0.8890	0.9071	0.9141	0.9154
TNR (Specificity)	0.9847	0.9705	0.9767	0.9783	0.9787
FPR	0.0152	0.0294	0.0232	0.0216	0.0212

In general, we can see that the general error is low in every case (equivalently, the accuracy is very high). However we cannot trust accuracy to state that our models are good. Indeed, since classes are unbalanced, even a trivial classificator would gain a reasonably high accuracy on this dataset. |
In order to state the quality of our models, we need to refer to a different performance measure. We can either choose Sensitivity or Specificity, for example, in order to take into account I type error and II type error. In both cases we see that the number of False Positive and False negative is very little in comparison with the number of True Positive.

Among the models tested two were linear classifiers: Logistic Regression and LDA. The two obtained a similar accuracy, however with slight variations to the decision boundary LDA could perform even better than the first one.

Finally, we can display these results with one last plot, the ROC curve. Since ROC curves are almost identical for all models, we plot only the one for the Logistic Regression:


True Positive Rate
False Positive Rate
AUC: 0.996
0.0
0.2
0.4
0.6
0.8
1.0