Sales prediction model

Matej Salamunić

17/12/2020

1 Introduction

The aim of the project is to develop a system that predicts the success of sales by using existing company sales data. By analyzing the data and testing different machine learning algorithms we will try to find an effective sales prediction model to improve the decision-making process and optimize the sales process in the company.

The data set, which represents sales data, is a sample of IBM Watson Sales-Win-Loss obtained on the following GitHub *link*. These data set, WA_Fn-UseC_-Sales-Win-Loss.csv.zip, is also available through the GitHub repository of this project on following *link* and the data from this location were used in the project.

In the first step of the project the WA_Fn-UseC_-Sales-Win-Loss.csv.zip data was downloaded and loaded into sales_dataset, and an initial data review was performed based on which the name and format of the columns were adjusted. Then, data sets (train_sales_set, test_sales_set) were created for testing and analysis of algorithm results, and a data sets (sales_set, validation_set) for final validation of the selected model. And then data exploration and visualization were performed. The correlation between features and the features distribution was reviewed by plotting different graphs and the most important variables for predicting the outcome of sales opportunities have been identified.

The result of sales activities, Opportunity.Result, is recorded through two possible categories (Won or Loss) so its prediction can be made using a classification model. Accordingly, several algorithms were selected for the test (glm, naive bayes, knn, qda, rf, adaboost) and tested, as well as an ensemble model composed of these algorithms. Based on the test results, the final, best algorithm was selected. Given the data prevalence of Opportunity.Results data in the given data set, sensitivity and precision are used as the most important measures in the algorithms evaluation.

Finally, for the selected model with the best algorithm (rf), a final validation was performed with a data set that has not been used in the project so far (validation_set) in order to confirm the obtained measures from the previous test.

In the following chapters the main and explanatory elements of the program code are presented with the corresponding execution results while the complete code is attached in the appendix at the end of the report with other information on system implementation, session and processing time.

2 Method and Analysis

2.1 Download Data and Generate Data Sets

The data source for the project is located at the *link* from where the data was downloaded and saved to the project directory (dataset) and then all data is loaded into the data set sales_dataset.

```r
# Download data
# WA_Fn-UseC_-Sales-Win-Loss dataset
# project location on GitHub
# https://github.com/matej-s/Sales
# link used to download data
# "https://raw.githubusercontent.com/matej-s/Sales/
#main/dataset/WA_Fn-UseC_-Sales-Win-Loss.csv.zip"
zip_url <- "https://raw.githubusercontent.com/matej-s/Sales/
main/dataset/WA_Fn-UseC_-Sales-Win-Loss.csv.zip"
zip_file <- "WA_Fn-UseC_-Sales-Win-Loss.csv.zip"
zip_dest <- as.character(paste(getwd(), zip_file, sep = "/dataset/"))
download.file(zip_url, destfile = zip_dest)
#unzip in  working directory
unzip(zip_dest)

# move unzip data from working directory to /dataeet project directory
path_to_original_file <- file.path(getwd(), "WA_Fn-UseC_-Sales-Win-Loss.csv")
path_to_move_to <- file.path(getwd(), "dataset/WA_Fn-UseC_-Sales-Win-Loss.csv")
file.copy(from = path_to_original_file, to  = path_to_move_to)
file.remove(path_to_original_file)

# Dataset
# load sales_dataset with downloaded data
sales_dataset <- read_csv(path_to_move_to)
```

2.1.1 Initial Exploration

After the initial data check, removal of duplicate data and setting column names and format, the sales_data set, as shown below, contains 77829 records of sales opportunity with information about the client, product, and sales process status.

```r
# structure of data set
str(sales_dataset)
#tibble [77,829 x 18] (S3: tbl_df/tbl/data.frame)
# $ Opportunity.Result                    : Factor w/ 2 levels "Won","Loss": 1 2 1 2 2 2 1 2 2 2 ...
# $ Supplies.Subgroup                     : Factor w/ 11 levels "Batteries & Accessories",..: 3 3 6 9 .
# $ Supplies.Group                        : Factor w/ 4 levels "Car Accessories",..: 1 1 3 3 1 3 1 1 1
# $ Region                                : Factor w/ 7 levels "Mid-Atlantic",..: 4 5 5 2 5 5 5 5 4 5
# $ Route.To.Market                       : Factor w/ 5 levels "Fields Sales",..: 1 2 2 2 2 2 1 1 1 2
# $ Elapsed.Days.In.Sales.Stage           : num [1:77829] 76 63 24 16 69 89 111 82 68 18 ...
# $ Sales.Stage.Change.Count              : num [1:77829] 13 2 7 5 11 3 12 6 8 7 ...
# $ Total.Days.Identified.Through.Closing  : num [1:77829] 104 163 82 124 91 114 112 70 156 50 ...
# $ Total.Days.Identified.Through.Qualified: num [1:77829] 101 163 82 124 13 0 112 70 156 50 ...
# $ Opportunity.Amount.USD                : num [1:77829] 0 0 7750 0 69756 ...
# $ Client.Size.By.Revenue                : Factor w/ 5 levels "1","2","3","4",..: 5 3 1 1 1 5 4 1 1 1
# $ Client.Size.By.Employee.Count         : Factor w/ 5 levels "1","2","3","4",..: 5 5 1 1 1 1 5 1 5 1
# $ Revenue.From.Client.Past.Two.Years    : Factor w/ 5 levels "0","1","2","3",..: 1 1 1 1 1 1 1 1 1 1
# $ Competitor.Type                       : Factor w/ 3 levels "Known","None",..: 3 3 3 1 3 3 3 1 2 3
# $ Ratio.Days.Identified.To.Total.Days   : num [1:77829] 0.696 0 1 1 0 ...
# $ Ratio.Days.Validated.To.Total.Days    : num [1:77829] 0.114 1 0 0 0.141 ...
# $ Ratio.Days.Qualified.To.Total.Days    : num [1:77829] 0.154 0 0 0 0 ...
# $ Deal.Size.Category                    : Factor w/ 7 levels "1","2","3","4",..: 1 1 1 1 4 5 2 6 6 4
```

2.1.2 Project Data Sets

This project uses four data sets extracted from the downloaded data (sales_dataset). Data sets for testing algorithms and for validating the final model are created from the set of all data (sales_dataset) which is first divided into sales_set (90%) and validation_set (10%). After that, sales_set is divided into train_sales_set (80%) and test_sales_set (20%) which are used for training and testing algorithms. The validation_set will be used only at the end in the final validation of the final selected model.

```
# preparing data sets for train, test and final validation

# final validation set (validation_set)  will be 10% of the entire sales_dataset
set.seed(211120, sample.kind = "Rounding")
test_index <- createDataPartition(y = sales_dataset$Opportunity.Result,
                                  times = 1,
                                  p = 0.1,
                                  list = FALSE)
# final validation sets
sales_set <- sales_dataset[-test_index,]
validation_set  <- sales_dataset[test_index,]

# split sales_set (80/20) to train and test algorithms
set.seed(211120, sample.kind = "Rounding")
test2_index <- createDataPartition(y = sales_set$Opportunity.Result,
                                   times = 1,
                                   p = 0.2,
                                   list = FALSE)

train_sales_set <- sales_set[-test2_index,]
test_sales_set <- sales_set[test2_index,]
```

Table 1: Data sets

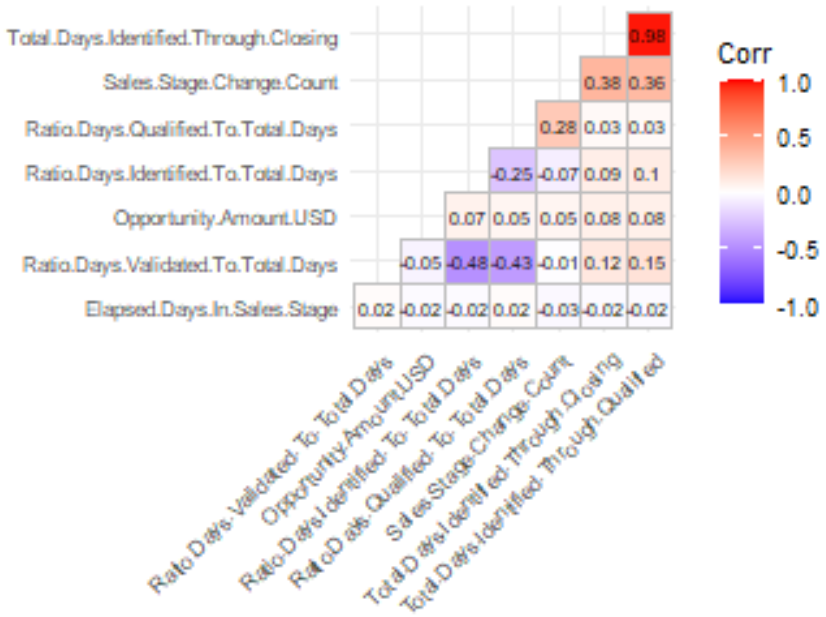
sales_dataset_split	rows	percentage	sales_set_split	rows	percentage
sales_set	70,045	90%	train_sales_set	56,035	80%
validation_set	7,784	10%	test_sales_set	14,010	20%

2.2 Exploration and Vizualization

In this section an exploratory data analysis is performed to gain better understanding of the data and to gather information that can be used in modeling.

2.2.1 Correlation

Using the heatmap correlation graph we will check the correlation between the numerical variables to see if there are highly correlated variables.



From the above heatmap we see that the correlation is moderate or lower for all variables except Total.Days.Identified.Through.Qualified and Total.Days.Identified.Through.Closing which are highly correlated (0.98). Highly correlated variables will have a negative impact on our models, and they represent a fairly equal source of information, so we will drop one of them (Total.Days.Identified.Through.Closing) from further analysis.

2.2.2 Near Zero Variance Predictors

With nearZeroVar function we can look for the predictors that have small number of unique values or those with a very large frequency difference between the most common and the second most common values. If there are predictors with these characteristics, they add little value to the prediction model and can be discarded. The results of nearZeroVar identification are shown below.

Table 2: nearZeroVar

	freqRatio	percentUnique	zeroVar	nzv
Supplies.Subgroup	1.0973	0.0157	FALSE	FALSE
Supplies.Group	1.8183	0.0057	FALSE	FALSE
Region	1.3845	0.0100	FALSE	FALSE
Route.To.Market	1.0709	0.0071	FALSE	FALSE
Elapsed.Days.In.Sales.Stage	2.1047	0.1942	FALSE	FALSE
Sales.Stage.Change.Count	1.7169	0.0300	FALSE	FALSE
Total.Days.Identified.Through.Qualified	2.7702	0.2170	FALSE	FALSE
Opportunity.Amount.USD	1.2596	12.9745	FALSE	FALSE
Client.Size.By.Revenue	11.3327	0.0071	FALSE	FALSE
Client.Size.By.Employee.Count	11.6892	0.0071	FALSE	FALSE
Revenue.From.Client.Past.Two.Years	24.3346	0.0071	FALSE	TRUE
Competitor.Type	4.7146	0.0043	FALSE	FALSE
Ratio.Days.Identified.To.Total.Days	4.7383	12.8289	FALSE	FALSE
Ratio.Days.Validated.To.Total.Days	1.0465	18.4924	FALSE	FALSE
Ratio.Days.Qualified.To.Total.Days	14.9790	13.3357	FALSE	FALSE
Deal.Size.Category	1.2013	0.0100	FALSE	FALSE

The result obtained indicates that one predictor (Revenue.From.Client.Past.Two.Years) has a near zero variance identification. Let's look at how unique values are represented in this predictor.

Revenue.From.Client.Past.Two.Years	n
0	62175
1	1585
2	1860
3	1870
4	2555

Although this predictor has one distinct value compared to the remaining four, it cannot be said that this predictor is entirely non-informative for our sales classification problem, therefore it will not be omitted from further analysis.

2.2.3 Statistics Summary

From the tables below, summary function results, we can get information that can indicate the variables that are normally distributed or we can check the categories distribution.

Opportunity.Result	Supplies.Subgroup	Supplies.Group	Region	Route.To.Market
Won :15793	Motorcycle Parts :13688	Car Accessories :44670	Mid-Atlantic: 6812	Fields Sales:33432
Loss:54252	Exterior Accessories :12474	Car Electronics : 249	Midwest :18835	Reseller :31218
NA	Garage & Car Care : 8757	Performance & Non-auto:24567	Northeast : 6628	Telecoverage: 561
NA	Shelters & RV : 8567	Tires & Wheels : 559	Northwest : 8582	Telesales : 2278
NA	Batteries & Accessories: 8240	NA	Pacific :13604	Other : 2556
NA	Replacement Parts : 6865	NA	Southeast : 8263	NA
NA	(Other) :11454	NA	Southwest : 7321	NA

Elapsed.Days.In.Sales.Stage	Sales.Stage.Change.Count	Total.Days.Identified.Through.Qualified	Opportunity.Amount.USD
Min. : 0.00	Min. : 1.000	Min. : 0.00	Min. : 0
1st Qu.: 19.00	1st Qu.: 2.000	1st Qu.: 4.00	1st Qu.: 15000
Median : 43.00	Median : 3.000	Median : 12.00	Median : 49000
Mean : 43.55	Mean : 2.954	Mean : 16.28	Mean : 91484
3rd Qu.: 65.00	3rd Qu.: 3.000	3rd Qu.: 24.00	3rd Qu.: 105000
Max. :210.00	Max. :23.000	Max. :208.00	Max. :1000000

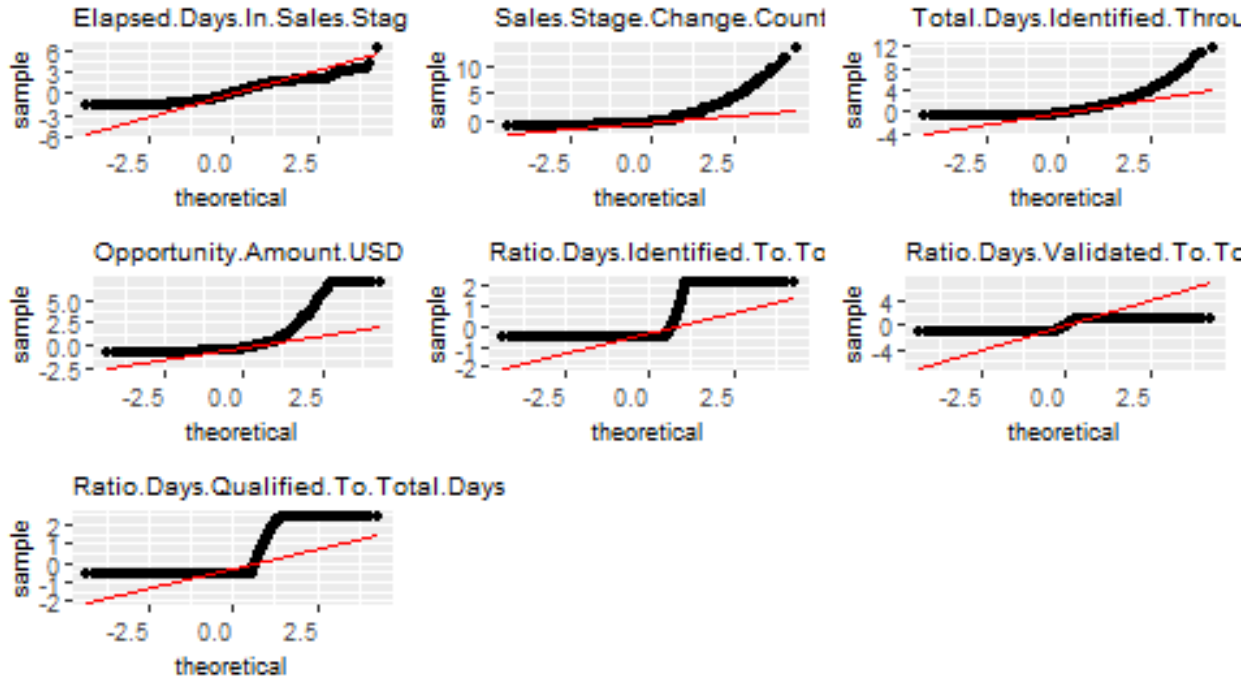
Client.Size.By.Revenue	Client.Size.By.Employee.Count	Revenue.From.Client.Past.Two.Years	Competitor.Type
1:53479	1:53326	0:62175	Known :10800
2: 3436	2: 4047	1: 1585	None : 8327
3: 4247	3: 4397	2: 1860	Unknown:50918
4: 4164	4: 3713	3: 1870	NA
5: 4719	5: 4562	4: 2555	NA

Ratio.Days.Identified.To.Total.Days	Ratio.Days.Validated.To.Total.Days	Ratio.Days.Qualified.To.Total.Days	Deal.Size.Category
Min. :0.0000	Min. :0.0000	Min. :0.0000	1:10850
1st Qu.:0.0000	1st Qu.:0.0000	1st Qu.:0.0000	2:13530
Median :0.0000	Median :0.4487	Median :0.0000	3:10783
Mean :0.2036	Mean :0.4886	Mean :0.1848	4:12243
3rd Qu.:0.1992	3rd Qu.:1.0000	3rd Qu.:0.1869	5:16254
Max. :1.0000	Max. :1.0000	Max. :1.0000	6: 4410
NA	NA	NA	7: 1975

From the tables above we can observe the prevalence in our response Opportunity.Result variable. Also, Elapsed.Days.In.Sales.Stage and Sales.Stage.Change.Count have a median value that is close to the mean which is indicator of the normal distribution.

2.2.4 QQ Plot

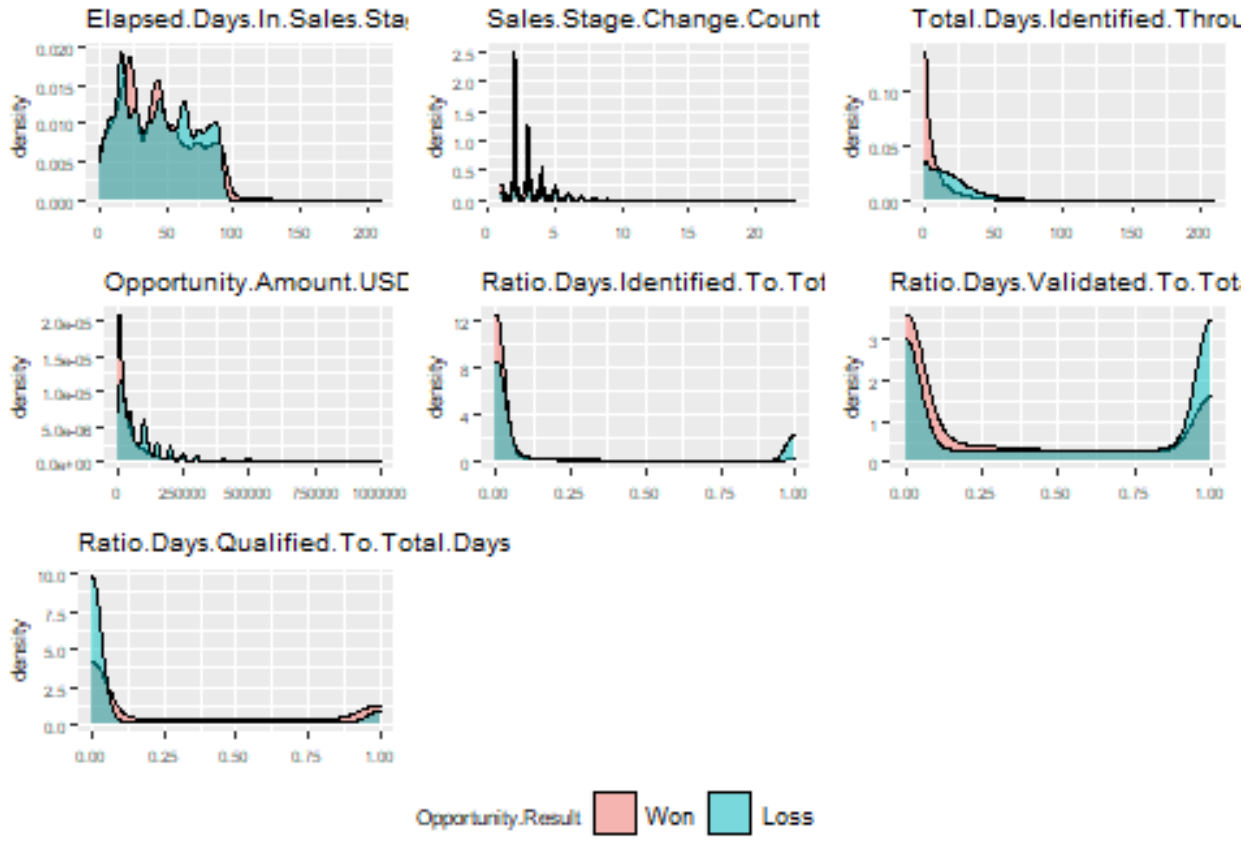
For additional insight into the variables we look at the qq plot which compare theirs distribution with the normal distribution.



For all these variables we have a deviation from the straight line, which indicates a deviation from the normal distribution. The Elapsed.Days.In.Sales.Stage distribution indicates that data has more extreme values than ones in normal distribution. For Sales.Stage.Change.Count, Total.Days.Identified.Through.Qualified and Opportunity.Amount.USD line is curved indicating distribution skewed to the right. The ratio columns, Ratio.Days.Identified.To.Total.Days, Ratio.Days.Validated.To.Total.Days and Ratio.Days.Qualified.To.Total.Days have data located at both extreme ends of the scale.

2.2.5 Density Plot

In the following density plots we will show the distribution of numerical variables grouped by Opportunity.Result to see how separate is the distribution of categories (Won, Loss).



Variables that have small overlaps in the distribution of categories are better predictors. From the graphs above we see that there is a overlap in categories distribution for all variables. For variables Elapsed.Days.In.Sales.Stage, Total.Days.Identified.Through.Qualified and Opportunity.Amount.USD there are small shifts in the distribution of categories and they have different peaks, so they are, looking at the density plot, slightly better predictors than other variables.

2.2.6 Prevalence

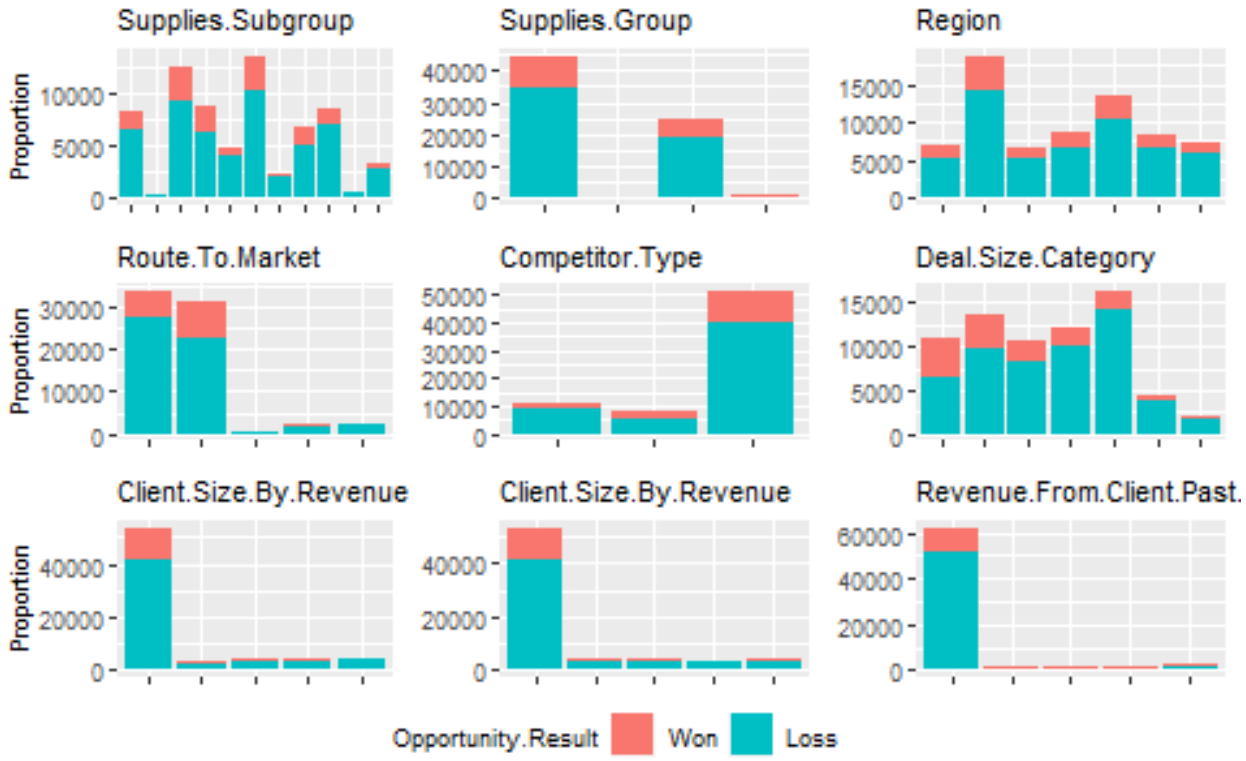
We are trying to predict a successful sales result "Won" so we will check what the Opportunity.Result categories distribution is to find out in what proportion "Won" is present in the data set.

Opportunity.Result	rows	percentage
Won	15,793	23%
Loss	54,252	77%

Opportunity.Result has an imbalance where the prevalence of "Won", which is successful sales, in this data set is 23% and of "Loss" is 77%.

2.2.7 Class Distribution

Let's look at how classes are distributed within categorical variables and how distribution of Won and Loss classes of our target Opportunity.Result variable are within those classes.



Distribution by most categories is present in Supplies.Subgroup, Region and Deal.Size.Category while in other variables we have class imbalance distribution, with only one or two dominant class. The prevalence of the Opportunity.Result is reflected across all classes of the variables shown above.

2.2.8 Feature Selections

In the data set we have 16 variables (features) and each can be used as a predictor but not all have to be equally informative and useful in predicting successful sales. To estimate the importance of each variable we can use the function R filterVarImp which uses the area under the ROC curve as a measure of variable importance to predict outcomes (Won, Loss).

Table 3: variable importance

Feature	Won	Loss
Total.Days.Identified.Through.Qualified	0.739	0.739
Deal.Size.Category	0.636	0.636
Ratio.Days.Qualified.To.Total.Days	0.634	0.634
Revenue.From.Client.Past.Two.Years	0.631	0.631
Ratio.Days.Identified.To.Total.Days	0.623	0.623
Opportunity.Amount.USD	0.623	0.623
Ratio.Days.Validated.To.Total.Days	0.602	0.602
Sales.Stage.Change.Count	0.564	0.564
Route.To.Market	0.546	0.546
Supplies.Subgroup	0.524	0.524
Region	0.521	0.521
Elapsed.Days.In.Sales.Stage	0.515	0.515
Supplies.Group	0.515	0.515
Competitor.Type	0.511	0.511
Client.Size.By.Employee.Count	0.509	0.509
Client.Size.By.Revenue	0.508	0.508

From table above, the Total.Days.Identified.Through.Qualified is the most important for predicting, and of the remaining variables, half of them are of the order of 0.6 and the remaining half have an amount above 0.5.

A large number of variables can have a negative effect on performance of a model and such a model cannot be easily explained and interpreted. Taking into account information obtained so far on the data set the 6 most important variables that can be used to improve the execution time of certain models were selected (Total.Days.Identified.Through.Qualified, Deal.Size.Category, Ratio.Days.Qualified.To.Total.Days, Revenue.From.Client.Past.Two.Years, Ratio.Days.Identified.To.Total.Days, Opportunity.Amount.USD).

2.3 Modeling Approach

Form data exploration part we learned that prediction of successful sales may be challenging because of large distribution overlapping areas in variables. Also we have an imbalanced data set where the number of negatives (Loss sales) is much larger than the positives (Won sales).

Our higher concern is to detect won opportunity as correctly as possible, and it is more acceptable to wrongly predict a loss opportunity than an inaccurate prediction of a won opportunity. So from our model we hope to have sensitivity as high as possible. And to evaluate the models effectiveness we examine relations between precision and sensitivity (recall) using precision-recall plot, which will also help us to select our final model.

The goal is to successfully predict successful sale (Won) what we are trying to achieve by testing different algorithms and based on the calculated measures we will choose the most suitable algorithm for us. Algorithms we ar going to try are glm, naive bayes, knn, qda, rf, adaboost and we will also try the ensemble model, composed of all these algorithms. These algorithms were selected from a supervised learning algorithms domain that include our classification problem which is prediction of the outcome of sales opportunity.

For algorithm testing we use train_sales_set set and perform cross validation to train algorithm ten times. Then with best training result algorithms performance are evaluates according to test_sales_set and resulting measures are calculated using confusionMatrix function. As a basic measure, we use a baseline model result which is based on the guessing the outcome prediction.

3 Results

3.1 Baseline Model

Our baseline model is randomly guessing the outcome without using any predictors.

```
# baseline model
set.seed(211120, sample.kind = "Rounding")
# guess with equal probability of Opportunity.Result Won, Loss
baseline <- sample(c("Loss", "Won"), nrow(train_sales_set), replace = TRUE)
baseline <- factor(baseline, levels = c("Won", "Loss"))
```

Table 4: baseline model result

Model	Accuracy	Sensitivity	Specificity	Precision	F1_Score
baseline	0.5003658	0.4985753	0.5008871	0.2252781	0.3103338

These baseline model result will determine whether the models we test are better than chance.

3.2 Generalized Linear Model

In generalized linear model (glm) we use all predictors.

```
# glm model
set.seed(211120, sample.kind = "Rounding")
# train glm model
train_glm_all <- train(Opportunity.Result ~ .,
                       method = "glm",
                       data = train_sales_set,
                       trControl = trainControl(method = "cv", number = 10, p = 0.9))
# predict on test data set
glm_all_preds <- predict(train_glm_all, test_sales_set)
```

Table 5: glm model result

Model	Accuracy	Sensitivity	Specificity	Precision	F1_Score
glm	0.837045	0.4900285	0.9380702	0.6972973	0.5755717

The resulting measures of the generalized linear model are better than in the baseline model except for the sensitivity.

3.3 Naive Bayes Model

In naive bayes model (naive_bayes) we use all predictors.

```
# naive_bayes model
set.seed(211120, sample.kind = "Rounding")
# train naive_bayes model
train_nb_all <- train(Opportunity.Result ~ .  ,
                  data = train_sales_set,
```

```
                  method = "naive_bayes",
                  trControl = trainControl(method = "cv", number = 10, p = 0.9))
# predict on test data set
nb_preds_all <- predict(train_nb_all, test_sales_set)
```

Table 6: naive bayes model result

Model	Accuracy	Sensitivity	Specificity	Precision	F1_Score
naivebayes	0.8033547	0.5362456	0.8811169	0.5676944	0.5515221

In the naive bayes model the sensitivity is slightly above 0.5.

3.4 K-Nearest Neighbors

In k-mearest neighbor model (knn) we use previously selected most important predictors and categorical variables in our data sets are normalized (n_train_sales_set, n_test_sales_set) to make the algorithm work more efficiently.

```
# knn model
set.seed(211120, sample.kind = "Rounding")
# train knn model
train_knn_bp <- train(Opportunity.Result ~ Total.Days.Identified.Through.Qualified +
                      Deal.Size.Category + Ratio.Days.Qualified.To.Total.Days +
                      Revenue.From.Client.Past.Two.Years +
                      Ratio.Days.Identified.To.Total.Days + Opportunity.Amount.USD,
                      method = "knn",
                      data = n_train_sales_set,
                      trControl = trainControl(method = "cv", number = 10, p = 0.9))
# predict on test data set
knn_preds_bp <- predict(train_knn_bp, n_test_sales_set)
```

Table 7: knn model result

Model	Accuracy	Sensitivity	Specificity	Precision	F1_Score
knn	0.8451106	0.5485913	0.9314349	0.6996367	0.6149752

With the k-nearest neighbors model the sensitivity is a bit higher than in the previous one and the F-1 score is the best so far.

3.5 Quadratic Discriminant Analysis

In quadratic discriminant analysis model (qda) we use set of most important predictors.

```
# qda model
set.seed(211120, sample.kind = "Rounding")
# train qda model
train_qda_bp <- train(Opportunity.Result ~ Total.Days.Identified.Through.Qualified +
                      Deal.Size.Category + Ratio.Days.Qualified.To.Total.Days +
                      Revenue.From.Client.Past.Two.Years +
                      Ratio.Days.Identified.To.Total.Days + Opportunity.Amount.USD,
```

```
                      data = train_sales_set,
                      method = "qda",
                      trControl = trainControl(method = "cv", number = 10, p = 0.9))
# predict on test data set
qda_preds_bp <- predict(train_qda_bp, test_sales_set)
```

Table 8: qda model result

Model	Accuracy	Sensitivity	Specificity	Precision	F1_Score
qda	0.8129907	0.357075	0.9457193	0.6569598	0.4626743

Quadratic discriminant analysis model has the lowest sensitivity so far.

3.6 Random Forest

In random forest model (rf) we use all predictors, set number of trees (ntree argument) to 100 for number of decision trees and set to 10 number of predictors randomly sampled as candidates at each split (mtry argument). This algorithm processing can take some time (see appendix for details).

```
# !!!this  can take long execution time
# rf model
set.seed(211120, sample.kind = "Rounding")
# train rf model
train_rf_all <- train(Opportunity.Result ~ . ,
                  data = train_sales_set,
                  method = "rf",
                  ntree = 100,    #ntree = 100
                  tuneGrid = data.frame(mtry = seq(1:10)),
                  trControl = trainControl(method = "cv", number = 10, p = 0.9))
# predict on test data set
rf_preds_all <- predict(train_rf_all, test_sales_set)
```

Table 9: rf model result

Model	Accuracy	Sensitivity	Specificity	Precision	F1_Score
rf	0.8827266	0.6752137	0.9431389	0.7756364	0.7219496

Random forest model has the best results so far for all measures except specificity which is at the level of the best measures so far.

3.7 AdaBoost

In AdaBoost model (adaboost) we use set of most important predictors to enable processing in a shorter time frame. This algorithm is the most demanding in terms of processing in the project (see appendix for details).

```
# !!!this  can take very long execution time
# adaboost model
set.seed(211120, sample.kind = "Rounding")
# train adaboost model
```

```
train_adab_bp <- train(Opportunity.Result ~ Total.Days.Identified.Through.Qualified +
                       Deal.Size.Category + Ratio.Days.Qualified.To.Total.Days +
                       Revenue.From.Client.Past.Two.Years +
                       Ratio.Days.Identified.To.Total.Days + Opportunity.Amount.USD,
                       data = train_sales_set,
                       method = "adaboost",
                       trControl = trainControl(method = "cv", number = 10, p = 0.9))
# predict on test data set
adab_preds_bp <- predict(train_adab_bp, test_sales_set)
```

Table 10: adaboost model result

Model	Accuracy	Sensitivity	Specificity	Precision	F1_Score
adaboost	0.835546	0.5023742	0.9325408	0.6843467	0.5794085

AdaBoost model has sensitivity just above 0.5 and other measures are within the measures of previous algorithms.

3.8 Ensemble Model

In the ensemble model we combine the results of previous predictions trying to improve the model.

```
# ensemble for alggorithms "glm", "naive_bayes", "knn", "qda", "rf", "adaboost"
# define algorithms for ensemble model
models <- c( "glm", "naive_bayes", "knn", "qda", "rf", "adaboost")
set.seed(211120, sample.kind = "Rounding")
# make predictions using already trained algorithms
preds <- sapply(models, function(model){
  if (model == "glm") {
  # glm use all predictors
    fit <- train_glm_all
  } else if (model == "naive_bayes") {
  # naive bayes uuse all predictors
    fit <- train_nb_all
  }else if (model == "knn") {
  # knn use set of most important predictors and normalization data
    fit <- train_knn_bp
  } else if (model == "qda") {
  # qda use set of most important predictors
    fit <- train_qda_bp
  } else if (model == "rf") {
  # Random forest use all predictors
    fit <- train_rf_all
  } else if (model == "adaboost") {
  # adaboost use set of most important predictors
    fit <- train_adab_bp
  }
# predictions and predictions for knn
  if (model == "knn") {
    # knn use data sets with normalization
    pred <- predict(object = fit, newdata = n_test_sales_set)
  }else {
```

```
    pred <- predict(object = fit, newdata = test_sales_set)
  }
# these predictions will be used for ensemble
  return(pred)
})
# Combine all models, use votes method to ensemble the predictions
votes <- rowMeans(preds == "Won")
ens_preds <- factor(ifelse(votes > 0.5, "Won", "Loss"))
ens_preds <- factor(ens_preds, levels=c("Won", "Loss"))
```

Table 11: ensemble model result

Model	Accuracy	Sensitivity	Specificity	Precision	F1_Score
ensemble	0.8479657	0.4529915	0.9629527	0.7806874	0.5733173

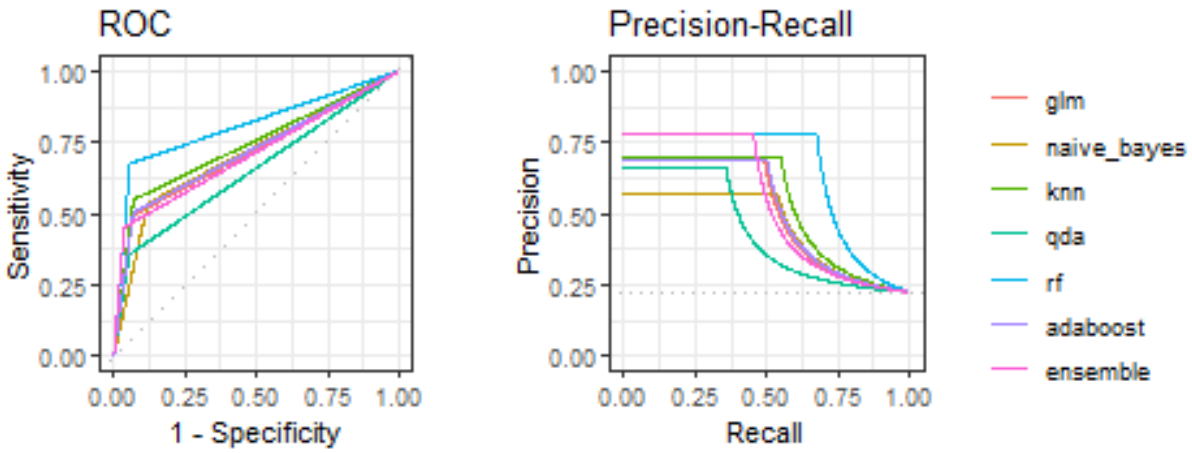
The ensemble model has slightly better measures of specificity and precision than other algorithms, accuracy is the second best, but sensitivity is below 0.5.

3.9 Best Model Selection

The results of all models are shown below in the table, and they are compared on the receiver operating characteristic (ROC) plot and precision–recall plot.

Table 12: Model results

Model	**Accuracy**	**Sensitivity**	**Specificity**	**Precision**	**F1_Score**
baseline	0.5004	0.4986	0.5009	0.2253	0.3103
glm	0.8370	0.4900	0.9381	0.6973	0.5756
naive_bayes	0.8034	0.5362	0.8811	0.5677	0.5515
knn	0.8451	0.5486	0.9314	0.6996	0.6150
qda	0.8130	0.3571	0.9457	0.6570	0.4627
rf	0.8827	0.6752	0.9431	0.7756	0.7219
adaboost	0.8355	0.5024	0.9325	0.6843	0.5794
ensemble	0.8480	0.4530	0.9630	0.7807	0.5733



Looking at the table and graphs above, we see that the ensemble model has the highest precision, only slightly higher than random forest. All other measures are best with the random forest model. In the random forest, compared to other models, we have the highest F-1 score and highest sensitivity (recall) and

the second best precision. Therefore, model with random forest algorithm is the selected as best model for our project.

3.10 Final Validation

Using our best model, random forest, we are perform final test with data never used so far in the project (validation_set). To train algorithm we are using sales_set and than making prediction against validation_set. As before, in the random forest model, processing can take some time (see appendix for details).

```
# !!!this can take long execution time
# final model rf, sales_set and validation set
set.seed(211120, sample.kind = "Rounding")
# train final rf model on sales_set
train_final_rf_all <- train(Opportunity.Result ~ . ,
                      data = sales_set,
                      method = "rf",
                      ntree = 100,    #ntree = 100
                      tuneGrid = data.frame(mtry = seq(1:10)),
                      trControl = trainControl(method = "cv", number = 10, p = 0.9))
# predict for final rf model on validation set
final_rf_preds_all <- predict(train_final_rf_all, validation_set)
```

Table 13: rf final model results

Model	Accuracy	Sensitivity	Specificity	Precision	F1_Score
final validation rf	0.881038	0.6683761	0.9429424	0.7732367	0.7169927

The measures obtained in the final validation correspond to the previously obtained values for random forest algorithm, so the results of final validation confirms the measures for random forest model.

4 Conclusion

This project was about developing sales prediction model for a set of sales data. At first download and import data in the program was performed, than after initial data exploration, the data preparation was made and data sets for the project were created.

In data exploration and visualization part first we looked for variables that could be omitted from further analysis. Then we looked at distributions looking for variables that are the best predictors.

In the results section, for the selected algorithms that were tested, the obtained measures are presented and the models were compared with each other. The random forest model was selected as the best model and its results were confirmed at the final validation (Accuracy 0.881, Sensitivity 0.668, F1 score 0.717).

To further improve the results we can use cost sensitive methods which treat our imbalanced data set and we can try to improve algorithms. For better balance of class distribution we can try undersampling and oversampling to transform train data set in preprocessing stage. And, to improve algorithms performance and results we can try with algorithms parameter tuning.

5 Appendix

Project related documentation (Sales.R, Sales_report.Rmd, Sales_report.pdf) and data set can be accessed on the GitHub link https://github.com/matej-s/Sales .

Sales.R - R code used to evaluate data set and build machine learning models.
Sales_report.Rmd - R Markdown script used to create the PDF report.
Sales_report.pdf - pdf report document, result of processing Rmd script.
WA_Fn-UseC_-Sales-Win-Loss.csv.zip - the project use these data set from the GitHub repository of this project.

The project used a Windows 10 computer with an i3 processor and 8 GB of RAM. The program code is written in R (version 4.0.2) and RStudio (version 1.3.1073) was used for development. For the described system, generating a pdf document (Sales_report.pdf) using the Sales_report.Rmd script takes about 7.5 hours, with the most time-consuming, about 90%, falling on the parts related to adaboost algorithm (takes about 4H for chapter 3.7 AdaBoost) and random forest algorithm (takes about 1H for chapter 3.6 Random Forest and 1H for chapter 3.10 Final Validation).

sessionInfo()
R version 4.0.2 (2020-06-22)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 18363)
Matrix products: default
locale:
[1] LC_COLLATE=English_United Kingdom.1252
[2] LC_CTYPE=English_United Kingdom.1252
[3] LC_MONETARY=English_United Kingdom.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United Kingdom.1252

attached base packages:
[1] stats graphics grDevices utils datasets methods base

loaded via a namespace (and not attached):
[1] Rcpp_1.0.5 digest_0.6.25 crayon_1.3.4 dplyr_1.0.2
[5] R6_2.4.1 lifecycle_0.2.0 magrittr_1.5 evaluate_0.14
[9] pillar_1.4.6 rlang_0.4.7 recosystem_0.4.3 rstudioapi_0.11
[13] vctrs_0.3.2 generics_0.0.2 ellipsis_0.3.1 rmarkdown_2.3
[17] tools_4.0.2 glue_1.4.1 purrr_0.3.4 yaml_2.2.1
[21] xfun_0.16 compiler_4.0.2 pkgconfig_2.0.3 htmltools_0.5.0
[25] knitr_1.29 tidyselect_1.1.0 tibble_3.0.3

5.1 The entire code print

```
#' ## 1 Introduction
#'
#' The aim of the project is to develop a system that
#' predicts the success of sales by using existing company sales data.
#'
#' ## 2 Method and Analysis
#' ### 2.1 Download Data and Generate Data Sets
# to find total processing time
startTime_ALL_2 <- Sys.time()

# Install packages and call library for the project
inst_pack_load_lib <- function(need_pkg)
{ install_pkg <- need_pkg[!(need_pkg %in% installed.packages()[,"Package"])]
  if(length(install_pkg))
  { install.packages(install_pkg,  repos = "http://cran.us.r-project.org")}
```

```
  for(package_name in need_pkg)
  {    library(package_name,character.only=TRUE,quietly=TRUE)}
}

# required packages list
need_pkg <- c("tidyverse", "jjb", "caret" , "rpart",  "e1071",
              "kableExtra", "reshape2", "ggcorrplot", "knitr" , "gridExtra",
              "ggridges", "ggplot2",  "gtable", "grid", "egg" ,
              "lemon",  "ggpubr", "huxtable", "scales", "ggpubr",
              "naivebayes" , "fastAdaboost",  "ada", "precrec")

# install and load missing packages
inst_pack_load_lib(need_pkg);


# make directory dataset to save data set
mkdir("dataset")
# make directory figs to save figures
mkdir("figs")

# Download data
# WA_Fn-UseC_-Sales-Win-Loss dataset
# project location on GitHub
# https://github.com/matej-s/Sales
# link used to download data
# "https://raw.githubusercontent.com/matej-s/Sales/
#main/dataset/WA_Fn-UseC_-Sales-Win-Loss.csv.zip"
zip_url <- "https://raw.githubusercontent.com/matej-s/Sales/
main/dataset/WA_Fn-UseC_-Sales-Win-Loss.csv.zip"
zip_file <- "WA_Fn-UseC_-Sales-Win-Loss.csv.zip"
zip_dest <- as.character(paste(getwd(), zip_file, sep = "/dataset/"))
download.file(zip_url, destfile = zip_dest)
#unzip in  working directory
unzip(zip_dest)

# move unzip data from working directory to /dataeet project directory
path_to_original_file <- file.path(getwd(), "WA_Fn-UseC_-Sales-Win-Loss.csv")
path_to_move_to <- file.path(getwd(), "dataset/WA_Fn-UseC_-Sales-Win-Loss.csv")
file.copy(from = path_to_original_file, to  = path_to_move_to)
file.remove(path_to_original_file)

# Dataset
# load sales_dataset with downloaded data
sales_dataset <- read_csv(path_to_move_to)

#' ### 2.1.1 Initial Exploration
#Initial data exploration
str(sales_dataset)
dim(sales_dataset)
head(sales_dataset)
glimpse(sales_dataset)
# this data set  covers sales activities
```

```r
#is there empty data
sum(is.na(sales_dataset))
#[1] 0
#no empty data in data set

#is there duplicate Ids - Opportunity Number
duplicated(sales_dataset$`Opportunity Number`)
no_unique <- sales_dataset$`OpportunityNumber`[
                    duplicated(sales_dataset$`Opportunity Number`)]
length(no_unique)
#[1] 196
#there are 196 duplicates

#Remove duplicated rows on Opportunity Number
sales_dataset <- sales_dataset %>%
                 distinct(sales_dataset$`Opportunity Number`, .keep_all = TRUE)
nrow(sales_dataset)
#[1] 77829

# data preparation - column names, column format

# make column names without spaces
names(sales_dataset) <- make.names(names(sales_dataset))

# remove duplicate columns sales_dataset$`Opportunity Number`
sales_dataset = subset(sales_dataset, select =
                    -c(Opportunity.Number, sales_dataset..Opportunity.Number.))
str(sales_dataset)

# check column type and format columns
levels(as.factor(sales_dataset$Supplies.Subgroup))
# [1] "Batteries & Accessories" "Car Electronics"         "Exterior Accessories"
# [4] "Garage & Car Care"       "Interior Accessories"    "Motorcycle Parts"
# [7] "Performance Parts"       "Replacement Parts"       "Shelters & RV"
#[10] "Tires & Wheels"          "Towing & Hitches"

# make as factor
sales_dataset$Supplies.Subgroup=factor(sales_dataset$Supplies.Subgroup)

levels(as.factor(sales_dataset$Supplies.Group))
#[1] "Car Accessories"        "Car Electronics"        "Performance & Non-auto"
#[4] "Tires & Wheels"

# make as factor
sales_dataset$Supplies.Group=factor(sales_dataset$Supplies.Group)

levels(as.factor(sales_dataset$Region))
#[1] "Mid-Atlantic" "Midwest"      "Northeast"    "Northwest"    "Pacific"
#[6] "Southeast"    "Southwest"

# make as factor
sales_dataset$Region=factor(sales_dataset$Region)
```

```r
levels(as.factor(sales_dataset$Route.To.Market))
#[1] "Fields Sales" "Other"  "Reseller"  "Telecoverage"  "Telesales"
# make as factor
# set levels "Fields Sales" "Reseller"  "Telecoverage"  "Telesales"  "Other"
sales_dataset$Route.To.Market <- factor(sales_dataset$Route.To.Market, levels=
          c("Fields Sales", "Reseller", "Telecoverage", "Telesales", "Other"))

levels(as.factor(sales_dataset$Opportunity.Result))
#[1] "Loss" "Won"
# make as factor
# make base level for Opportunity.Result to Won
sales_dataset$Opportunity.Result <- factor(sales_dataset$Opportunity.Result,
                                           levels=c("Won", "Loss"))
# move Opportunity.Result column to the start
sales_dataset <- sales_dataset %>% select(Opportunity.Result, everything())

levels(as.factor(sales_dataset$Competitor.Type))
#[1] "Known"   "None"    "Unknown"
# make as factor
sales_dataset$Competitor.Type=factor(sales_dataset$Competitor.Type)

levels(as.factor(sales_dataset$Deal.Size.Category))
#[1] "1" "2" "3" "4" "5" "6" "7"
# make as factor
sales_dataset$Deal.Size.Category=factor(sales_dataset$Deal.Size.Category)

levels(as.factor(sales_dataset$Client.Size.By.Revenue))
#[1] "1" "2" "3" "4" "5"
# make as factor
sales_dataset$Client.Size.By.Revenue=factor(sales_dataset$Client.Size.By.Revenue)

levels(as.factor(sales_dataset$Client.Size.By.Employee.Count))
#[1] "1" "2" "3" "4" "5"
# make as factor
sales_dataset$Client.Size.By.Employee.Count=
    factor(sales_dataset$Client.Size.By.Employee.Count)

levels(as.factor(sales_dataset$Revenue.From.Client.Past.Two.Years))
#[1] "0" "1" "2" "3" "4"
# make as factor
sales_dataset$Revenue.From.Client.Past.Two.Years=
  factor(sales_dataset$Revenue.From.Client.Past.Two.Years)

# structure of data set
str(sales_dataset)
#tibble [77,829 x 18] (S3: tbl_df/tbl/data.frame)
# $ Opportunity.Result                    : Factor w/ 2 levels "Won","Loss": 1 2 1 2 2 2 1 2 2 2 ...
# $ Supplies.Subgroup                     : Factor w/ 11 levels "Batteries & Accessories",..: 3 3 6 9 .
# $ Supplies.Group                        : Factor w/ 4 levels "Car Accessories",..: 1 1 3 3 1 3 1 1 1
# $ Region                                : Factor w/ 7 levels "Mid-Atlantic",..: 4 5 5 2 5 5 5 5 4 5
# $ Route.To.Market                       : Factor w/ 5 levels "Fields Sales",..: 1 2 2 2 2 2 1 1 1 2
# $ Elapsed.Days.In.Sales.Stage           : num [1:77829] 76 63 24 16 69 89 111 82 68 18 ...
# $ Sales.Stage.Change.Count              : num [1:77829] 13 2 7 5 11 3 12 6 8 7 ...
```

```
# $ Total.Days.Identified.Through.Closing  : num [1:77829] 104 163 82 124 91 114 112 70 156 50 ...
# $ Total.Days.Identified.Through.Qualified: num [1:77829] 101 163 82 124 13 0 112 70 156 50 ...
# $ Opportunity.Amount.USD                 : num [1:77829] 0 0 7750 0 69756 ...
# $ Client.Size.By.Revenue                 : Factor w/ 5 levels "1","2","3","4",..: 5 3 1 1 1 5 4 1 1 1
# $ Client.Size.By.Employee.Count          : Factor w/ 5 levels "1","2","3","4",..: 5 5 1 1 1 1 5 1 5 1
# $ Revenue.From.Client.Past.Two.Years     : Factor w/ 5 levels "0","1","2","3",..: 1 1 1 1 1 1 1 1 1 1
# $ Competitor.Type                        : Factor w/ 3 levels "Known","None",..: 3 3 3 1 3 3 3 1 2 3
# $ Ratio.Days.Identified.To.Total.Days    : num [1:77829] 0.696 0 1 1 0 ...
# $ Ratio.Days.Validated.To.Total.Days     : num [1:77829] 0.114 1 0 0 0.141 ...
# $ Ratio.Days.Qualified.To.Total.Days     : num [1:77829] 0.154 0 0 0 0 ...
# $ Deal.Size.Category                     : Factor w/ 7 levels "1","2","3","4",..: 1 1 1 1 4 5 2 6 6 4

#' ### 2.1.2 Project Data Sets

# preparing data sets for train, test and final validation

# final validation set (validation_set)  will be 10% of the entire sales_dataset
set.seed(211120, sample.kind = "Rounding")
test_index <- createDataPartition(y = sales_dataset$Opportunity.Result,
                                  times = 1,
                                  p = 0.1,
                                  list = FALSE)
# final validation sets
sales_set <- sales_dataset[-test_index,]
validation_set  <- sales_dataset[test_index,]

# split sales_set (80/20) to train and test algorithms
set.seed(211120, sample.kind = "Rounding")
test2_index <- createDataPartition(y = sales_set$Opportunity.Result,
                                   times = 1,
                                   p = 0.2,
                                   list = FALSE)

train_sales_set <- sales_set[-test2_index,]
test_sales_set <- sales_set[test2_index,]

# data set used in projects
part1 <- nrow(sales_set) / nrow(sales_dataset)
part2 <- nrow(validation_set) / nrow(sales_dataset)

table_stes <- tibble(sales_dataset_split="sales_set",
                     rows=format(nrow(sales_set), big.mark= ',' ),
                     percentage = percent(part1, accuracy=1)     )

table_stes <- bind_rows(table_stes, tibble(sales_dataset_split="validation_set",
                        rows=format(nrow(validation_set), big.mark= ',' ),
                        percentage = percent(part2, accuracy=1)) )


part3 <- nrow(train_sales_set) / nrow(sales_set)
part4 <- nrow(test_sales_set) / nrow(sales_set)
table_sets_b <- tibble(sales_set_split="train_sales_set",
                       rows=format(nrow(train_sales_set), big.mark= ',' ),
```

```
                        percentage = percent(part3, accuracy=1) )

table_sets_b <- bind_rows(table_sets_b, tibble(sales_set_split="test_sales_set",
                          rows=format(nrow(test_sales_set), big.mark= ',' ),
                          percentage = percent(part4, accuracy=1) ))

kable( list(table_stes, table_sets_b), caption = 'Data sets',
            booktabs = TRUE, valign = 't')  %>%
            kable_styling(latex_options = "HOLD_position", font_size = 8)

#' ### 2.2 Exploration and Vizualization

# all predictors, all columns except Opportunity.Result
pred_col <- c("Supplies.Subgroup",
              "Supplies.Group",
              "Region",
              "Route.To.Market",
              "Elapsed.Days.In.Sales.Stage",
              "Sales.Stage.Change.Count",
              "Total.Days.Identified.Through.Closing",
              "Total.Days.Identified.Through.Qualified",
              "Opportunity.Amount.USD",
              "Client.Size.By.Revenue" ,
              "Client.Size.By.Employee.Count",
              "Revenue.From.Client.Past.Two.Years",
              "Competitor.Type",
              "Ratio.Days.Identified.To.Total.Days",
              "Ratio.Days.Validated.To.Total.Days",
              "Ratio.Days.Qualified.To.Total.Days",
              "Deal.Size.Category")

# categorical variables
categ_col <- c("Opportunity.Result",
               "Supplies.Subgroup",
               "Supplies.Group",
               "Region",
               "Route.To.Market",
               "Competitor.Type",
               "Deal.Size.Category",
               "Client.Size.By.Revenue" ,
               "Client.Size.By.Employee.Count",
               "Revenue.From.Client.Past.Two.Years" )

# continuous variables
cont_col <- c( "Elapsed.Days.In.Sales.Stage",
               "Sales.Stage.Change.Count",
               "Total.Days.Identified.Through.Closing",
               "Total.Days.Identified.Through.Qualified",
               "Opportunity.Amount.USD",
               "Ratio.Days.Identified.To.Total.Days",
               "Ratio.Days.Validated.To.Total.Days",
               "Ratio.Days.Qualified.To.Total.Days")
```

```r
#' ### 2.2.1 Correlation

# heatmap
# set for the correlation
cor_sales_set = subset(sales_set,
                select = -c(Opportunity.Result, Supplies.Subgroup,
                            Supplies.Group,  Region, Route.To.Market,
                            Competitor.Type, Deal.Size.Category,
                            Client.Size.By.Revenue,
                            Client.Size.By.Employee.Count,
                            Revenue.From.Client.Past.Two.Years  ))
# correlation matrix
cor_mat_sales <- round(cor(cor_sales_set),2)

# create the correlation heatmap
# library(reshape2)
melted_cor_mat_sales <- melt(cor_mat_sales)

# save qq plot in figures
png(file="figs/corr_1.png", width=480, height=240)
# show the correlation matrix
ggcorrplot(cor_mat_sales,
           hc.order = TRUE,
           type = "lower",
           lab = TRUE, lab_size = 2.5) +
   theme(axis.text.x = element_text(size = 8)) +
   theme(axis.text.y = element_text(size = 8))
dev.off()

# call heatmap graph in report
include_graphics("figs/corr_1.png",
                 auto_pdf = getOption("knitr.graphics.auto_pdf", FALSE), dpi=NA)

# drop Total.Days.Identified.Through.Closing from data sets
sales_set = subset(sales_set, select = -c(Total.Days.Identified.Through.Closing))
train_sales_set = subset(train_sales_set, select =
                          -c(Total.Days.Identified.Through.Closing))
test_sales_set = subset(test_sales_set, select =
                         -c(Total.Days.Identified.Through.Closing))
validation_set = subset(validation_set, select =
                         -c(Total.Days.Identified.Through.Closing))
pred_col <- pred_col[pred_col != "Total.Days.Identified.Through.Closing"]
cont_col <- cont_col[cont_col != "Total.Days.Identified.Through.Closing"]

#' ### 2.2.2 Near Zero Variance Predictors
# identification of near zero variance predictors using nearZeroVar
kable(nearZeroVar(sales_set[, pred_col],
    saveMetrics = TRUE), caption = 'nearZeroVar', digits = 4, booktabs=TRUE) %>%
    kable_styling(latex_options = "HOLD_position", font_size = 8)

# count values for Revenue.From.Client.Past.Two.Years
tbl1 <- sales_set %>% count(Revenue.From.Client.Past.Two.Years)
kable( tbl1, digits = 3, booktabs = TRUE, align = c("r", "r"))  %>%
```

```
      kable_styling(latex_options = "HOLD_position", font_size = 8)


#' ### 2.2.3 Statistics Summary
# data set variables summary statistics
#summary(sales_set)
temp <- sales_set[ , 1:5]
kable( summary(temp), booktabs = TRUE)  %>%
  kable_styling(latex_options = "scale_down", font_size = 8)
temp <- sales_set[ , 6:9]
kable( summary(temp), booktabs = TRUE)  %>%
  kable_styling(latex_options = "scale_down", font_size = 8)
temp <- sales_set[ , 10:13]
kable( summary(temp), booktabs = TRUE)  %>%
  kable_styling(latex_options = "scale_down", font_size = 8)
temp <- sales_set[ , 14:17]
kable( summary(temp), booktabs = TRUE)  %>%
  kable_styling(latex_options = "scale_down", font_size = 8)

#' ### 2.2.4 QQ Plot
# save qq plot in figures
png(file="figs/qq_1.png", width=480, height=270)
# QQ plots to look up if the feature is normally distributed
qq_grid <- lapply(cont_col, FUN=function(var) {
  sales_set %>%
    dplyr::select(all_of(var)) %>%
    ggplot(data = ., aes(sample = scale(.))) +
    stat_qq() +
    stat_qq_line(colour = "red") +
    theme(axis.text.x = element_text(hjust = 1)) +
    ggtitle(var)+
    theme(title =element_text(size=8),
          axis.text=element_text(size=9),
          axis.title=element_text(size=9))
})
do.call(grid.arrange, args=c(qq_grid, list(ncol=3)))
dev.off()

# call qq plots in report
include_graphics("figs/qq_1.png",
                 auto_pdf = getOption("knitr.graphics.auto_pdf", FALSE), dpi=NA)

#' ### 2.2.5 Density Plot
# save density plot in figures
png(file="figs/density_grid_1.png", width=480, height=330)
# grid of density plots
dens_grid <- lapply(cont_col, FUN=function(var) {
  # Build the plots
  ggplot(sales_set) +
    geom_density(aes_string(x = var, fill = "Opportunity.Result"), alpha = 0.5) +
    ggtitle(var)+
    theme(title =element_text(size=8),
          axis.text=element_text(size=6),
```

```
          axis.title=element_text(size=8), axis.title.x=element_blank())
})
do.call(grid_arrange_shared_legend, args=c(dens_grid, nrow = 3, ncol = 3))
dev.off()

# call density plots in report
include_graphics("figs/density_grid_1.png",
                 auto_pdf = getOption("knitr.graphics.auto_pdf", FALSE), dpi=NA)

#' ### 2.2.6 Prevalence
# distribution of Won Loss Opportunity.Result
sales_set_won <- sales_set[which(sales_set$Opportunity.Result == "Won"),]
sales_set_loss <- sales_set[which(sales_set$Opportunity.Result == "Loss"),]
part1 <- nrow(sales_set_won) / nrow(sales_set)
part2 <- nrow(sales_set_loss) / nrow(sales_set)

table_stes <- tibble(Opportunity.Result="Won",
                     rows=format(nrow(sales_set_won), big.mark= ',' ),
                     percentage = percent(part1, accuracy=1) )
table_stes <- bind_rows(table_stes, tibble(Opportunity.Result="Loss",
                    rows=format(nrow(sales_set_loss), big.mark= ',' ) ,
                    percentage =   percent(part2, accuracy = 1) ))

kable( table_stes, booktabs = TRUE, valign = 't')  %>%
  kable_styling(latex_options = "hold_position", font_size = 8)

#' ### 2.2.7 Class Distribution
# save categorical plots in figures
png(file="figs/class_1.png", width=480, height=300)
# categorical variables class distribution
p1 <- ggplot(data = sales_set) + geom_bar(aes(x=Supplies.Subgroup,
            fill=Opportunity.Result)) + labs(title = "Supplies.Subgroup") +
  theme(title =element_text(size=9), axis.text.x=element_blank(),
        axis.title.x = element_blank()) +
  theme(legend.position = "none")  + ylab("Proportion")

p2 <- ggplot(data = sales_set) + geom_bar(aes(x=Supplies.Group,
            fill=Opportunity.Result)) + labs(title = "Supplies.Group") +
  theme(title =element_text(size=9), axis.text.x=element_blank(),
        axis.title.x=element_blank(), axis.title.y = element_blank()) +
  theme(legend.position = "none")  + ylab("Proportion")

p3 <- ggplot(data = sales_set) + geom_bar(aes(x=Region,
             fill=Opportunity.Result)) + labs(title = "Region") +
  theme(title =element_text(size=9), axis.text.x=element_blank(),
        axis.title.x=element_blank(), axis.title.y = element_blank()) +
  theme(legend.position = "none")  + ylab("Proportion")

p4 <- ggplot(data = sales_set) + geom_bar(aes(x=Route.To.Market,
            fill=Opportunity.Result)) + labs(title = "Route.To.Market") +
  theme(title =element_text(size=9), axis.text.x=element_blank(),
        axis.title.x=element_blank()) +
  theme(legend.position = "none")  + ylab("Proportion")
```

```
p5 <- ggplot(data = sales_set) + geom_bar(aes(x=Competitor.Type,
            fill=Opportunity.Result)) + labs(title = "Competitor.Type") +
  theme(title =element_text(size=9), axis.text.x=element_blank(),
        axis.title.x=element_blank(), axis.title.y = element_blank()) +
  theme(legend.position = "none")  + ylab("Proportion")

p6 <- ggplot(data = sales_set) + geom_bar(aes(x=Deal.Size.Category,
            fill=Opportunity.Result)) + labs(title = "Deal.Size.Category") +
  theme(title =element_text(size=9), axis.text.x=element_blank(),
        axis.title.x=element_blank(), axis.title.y = element_blank()) +
  theme(legend.position = "none")  + ylab("Proportion")

p7 <- ggplot(data = sales_set) + geom_bar(aes(x=Client.Size.By.Revenue,
            fill=Opportunity.Result)) + labs(title = "Client.Size.By.Revenue") +
  theme(title =element_text(size=9), axis.text.x=element_blank(),
        axis.title.x=element_blank()) +
  theme(legend.position = "none")  + ylab("Proportion")

p8 <-ggplot(data = sales_set) + geom_bar(aes(x=Client.Size.By.Employee.Count,
            fill=Opportunity.Result)) + labs(title = "Client.Size.By.Revenue") +
  theme(title =element_text(size=9), axis.text.x=element_blank(),
        axis.title.x=element_blank(), axis.title.y = element_blank()) +
  theme(legend.position = "none")  + ylab("Proportion")

p9 <- ggplot(data = sales_set) + geom_bar(aes(x=Revenue.From.Client.Past.Two.Years,
     fill=Opportunity.Result))+labs(title = "Revenue.From.Client.Past.Two.Years") +
  theme(title =element_text(size=9), axis.text.x=element_blank(),
        axis.title.x=element_blank(), axis.title.y = element_blank()) +
  theme(legend.position = "none")  + ylab("Proportion")

ggarrange(p1, p2, p3, p4, p5, p6, p7, p8, p9, nrow=3, ncol=3,
          common.legend = TRUE, legend="bottom")
dev.off()

# call class plots in report
include_graphics("figs/class_1.png",
                 auto_pdf = getOption("knitr.graphics.auto_pdf", FALSE), dpi=NA)

#' ### 2.2.8 Feature Selections
# variable importance for Opportunity.Result
filterVarImp(x = sales_set[,pred_col],
             y = sales_set$Opportunity.Result)

# variable importance table
tbl_varimp <- tibble(Feature =
              rownames(filterVarImp(x = sales_set[,pred_col],
                                    y = sales_set$Opportunity.Result)),
               Won = filterVarImp(x = sales_set[,pred_col],
                                  sales_set$Opportunity.Result)$Won,
               Loss = filterVarImp(x = sales_set[,pred_col],
                                   sales_set$Opportunity.Result)$Loss) %>%
              arrange(desc(Won))
```

```r
kable( tbl_varimp, digits = 3, caption='variable importance',
       booktabs = TRUE, valign = 't' )  %>%
       kable_styling(latex_options = "HOLD_position", font_size = 8)


#' ### 2.3 Modeling Approach
#' The goal is to successfully predict successful sale (Won)
#' what we are trying to achieve by testing different algorithms.

#' For algorithm testing we use train_sales_set sets and perform cross validation to train algorithm te
#' Then with best training result algorithms performance are evaluates according to test_sales_set
#' and resulting measures are calculated using confusionMatrix function.
#' As a basic measure, we use a baseline model.



#' ## 3 Results

#' ### 3.1 Baseline Model
# baseline model
set.seed(211120, sample.kind = "Rounding")
# guess with equal probability of Opportunity.Result Won, Loss
baseline <- sample(c("Loss", "Won"), nrow(train_sales_set), replace = TRUE)
baseline <- factor(baseline, levels = c("Won", "Loss"))

# use confusionMatrix to view the results
cm_baseline_1 <- confusionMatrix(data = factor(baseline),
                reference = factor(train_sales_set$Opportunity.Result))
cm_baseline_2 <- confusionMatrix(data = factor(baseline),
                reference = factor(train_sales_set$Opportunity.Result),
                mode = "prec_recall", positive="Won")
# results table for baseline
baseline_results <- tibble(Model = "baseline",
                        Accuracy = cm_baseline_1$overall["Accuracy"],
                        Sensitivity = cm_baseline_1$byClass["Sensitivity"],
                        Specificity = cm_baseline_1$byClass["Specificity"],
                        Precision  = cm_baseline_2$byClass["Precision"],
                        F1_Score = cm_baseline_2$byClass["F1"])
#baseline_results
kable( baseline_results, caption = "baseline model result" )  %>%
       kable_styling(latex_options = "HOLD_position", font_size = 8)


#' ### 3.2 Generalized Linear Model
# glm model
set.seed(211120, sample.kind = "Rounding")
# train glm model
train_glm_all <- train(Opportunity.Result ~ .,
                       method = "glm",
                       data = train_sales_set,
                       trControl = trainControl(method = "cv", number = 10, p = 0.9))
# predict on test data set
glm_all_preds <- predict(train_glm_all, test_sales_set)
```

```r
## use confusionMatrix to view the results
cm_glm_1 <- confusionMatrix(data = factor(glm_all_preds),
                            reference = factor(test_sales_set$Opportunity.Result))
cm_glm_2 <-  confusionMatrix(data = factor(glm_all_preds),
                             reference = factor(test_sales_set$Opportunity.Result),
                             mode = "prec_recall", positive="Won")

# results table for glm model
glm_results <- tibble(Model = "glm",
                      Accuracy = cm_glm_1$overall["Accuracy"],
                      Sensitivity = cm_glm_1$byClass["Sensitivity"],
                      Specificity = cm_glm_1$byClass["Specificity"],
                      Precision  = cm_glm_2$byClass["Precision"],
                      F1_Score = cm_glm_2$byClass["F1"])
kable( glm_results, caption = "glm model result" )  %>%
       kable_styling(latex_options = "HOLD_position", font_size = 8)


#' ### 3.3 Naive Bayes Model
# naive_bayes model
set.seed(211120, sample.kind = "Rounding")
# train naive_bayes model
train_nb_all <- train(Opportunity.Result ~ .  ,
                  data = train_sales_set,
                  method = "naive_bayes",
                  trControl = trainControl(method = "cv", number = 10, p = 0.9))
# predict on test data set
nb_preds_all <- predict(train_nb_all, test_sales_set)

## use confusionMatrix to view the results
cm_nb_1 <- confusionMatrix(data = factor(nb_preds_all), reference =
                             factor(test_sales_set$Opportunity.Result))
cm_nb_2 <- confusionMatrix(data = factor(nb_preds_all), reference =
                             factor(test_sales_set$Opportunity.Result),
            mode = "prec_recall", positive="Won")
# results table for naivebayes
naivebayes_results <- tibble(Model = "naivebayes",
                      Accuracy = cm_nb_1$overall["Accuracy"],
                      Sensitivity = cm_nb_1$byClass["Sensitivity"],
                      Specificity = cm_nb_1$byClass["Specificity"],
                      Precision  = cm_nb_2$byClass["Precision"],
                      F1_Score = cm_nb_2$byClass["F1"])
kable( naivebayes_results, caption = "naive bayes model result" )  %>%
       kable_styling(latex_options = "HOLD_position", font_size = 8)



#' ### 3.4 K-Nearest Neighbors

# train i test set normalization for knn
# function to normalize values
normalize <- function(x) {
  return ((x-min(x)) / (max(x) - min(x)))
```

```
}
#normalization train set n_
n_train_sales_set <- train_sales_set

#$ Opportunity.Result                     : Factor w/ 2 levels "Won","Loss": 1 2 2 2 2 2 2 2 2 2 ...
n_train_sales_set$Supplies.Subgroup <- as.numeric(
       factor(n_train_sales_set$Supplies.Subgroup))
n_train_sales_set$Supplies.Group <- as.numeric(
       factor(n_train_sales_set$Supplies.Group))
n_train_sales_set$Region <-  as.numeric(
       factor(n_train_sales_set$Region))
n_train_sales_set$Route.To.Market <-  as.numeric(
       factor(n_train_sales_set$Route.To.Market))
#$ Elapsed.Days.In.Sales.Stage            : num [1:56035] 76 69 89 82 68 18 35 16 81 83 ...
#Sales.Stage.Change.Count                : num [1:56035] 13 11 3 6 8 7 6 5 10 13 ...
#$ Total.Days.Identified.Through.Qualified: num [1:56035] 101 13 0 70 156 50 31 208 138 130 ...
#$ Opportunity.Amount.USD                 : num [1:56035] 0 69756 232522 450000 25
n_train_sales_set$Client.Size.By.Revenue <-  as.numeric(
       factor(n_train_sales_set$Client.Size.By.Revenue))
n_train_sales_set$Client.Size.By.Employee.Count <- as.numeric(
       factor(n_train_sales_set$Client.Size.By.Employee.Count))
n_train_sales_set$Revenue.From.Client.Past.Two.Years <-  as.numeric(
       factor(n_train_sales_set$Revenue.From.Client.Past.Two.Years))
n_train_sales_set$Competitor.Type <-  as.numeric(
       factor(n_train_sales_set$Competitor.Type))
#$ Ratio.Days.Identified.To.Total.Days    : num [1:56035] 0.696 0 0 0.264 0 ...
#$ Ratio.Days.Validated.To.Total.Days     : num [1:56035] 0.113985 0.141125 0.000877 0.73639 0.562821 .
#$ Ratio.Days.Qualified.To.Total.Days     : num [1:56035] 0.154 0 0 0 0.437 ...
n_train_sales_set$Deal.Size.Category <-  as.numeric(
       factor(n_train_sales_set$Deal.Size.Category))

n_train_sales_set <- data.frame(n_train_sales_set[1],
                                lapply(n_train_sales_set[2:17], normalize) )

#normalization test set n_ normalization
n_test_sales_set <- test_sales_set
#$ Opportunity.Result                     : Factor w/ 2 levels "Won","Loss": 1 2 2 2 2 2 2 2 2 2 ...
n_test_sales_set$Supplies.Subgroup <- as.numeric(
      factor(n_test_sales_set$Supplies.Subgroup))
n_test_sales_set$Supplies.Group <- as.numeric(
      factor(n_test_sales_set$Supplies.Group))
n_test_sales_set$Region <-  as.numeric(
      factor(n_test_sales_set$Region))
n_test_sales_set$Route.To.Market <-  as.numeric(
      factor(n_test_sales_set$Route.To.Market))
#$ Elapsed.Days.In.Sales.Stage            : num [1:56035] 76 69 89 82 68 18 35 16 81 83 ...
#Sales.Stage.Change.Count                : num [1:56035] 13 11 3 6 8 7 6 5 10 13 ...
#$ Total.Days.Identified.Through.Qualified: num [1:56035] 101 13 0 70 156 50 31 208 138 130 ...
#$ Opportunity.Amount.USD                 : num [1:56035] 0 69756 232522 450000 25
n_test_sales_set$Client.Size.By.Revenue <-  as.numeric(
      factor(n_test_sales_set$Client.Size.By.Revenue))
n_test_sales_set$Client.Size.By.Employee.Count <-  as.numeric(
      factor(n_test_sales_set$Client.Size.By.Employee.Count))
```

```r
n_test_sales_set$Revenue.From.Client.Past.Two.Years <-  as.numeric(
      factor(n_test_sales_set$Revenue.From.Client.Past.Two.Years))
n_test_sales_set$Competitor.Type <-  as.numeric(
      factor(n_test_sales_set$Competitor.Type))
#$ Ratio.Days.Identified.To.Total.Days    : num [1:56035] 0.696 0 0 0.264 0 ...
#$ Ratio.Days.Validated.To.Total.Days     : num [1:56035] 0.113985 0.141125 0.000877 0.73639 0.562821 .
#$ Ratio.Days.Qualified.To.Total.Days     : num [1:56035] 0.154 0 0 0 0.437 ...
n_test_sales_set$Deal.Size.Category <-  as.numeric(
      factor(n_test_sales_set$Deal.Size.Category))

n_test_sales_set <- data.frame(n_test_sales_set[1],
                               lapply(n_test_sales_set[2:17], normalize))


#normalize train i test set
# n_train_sales_set
# n_test_sales_set

# knn model
set.seed(211120, sample.kind = "Rounding")
# train knn model
train_knn_bp <- train(Opportunity.Result ~ Total.Days.Identified.Through.Qualified +
                      Deal.Size.Category + Ratio.Days.Qualified.To.Total.Days +
                      Revenue.From.Client.Past.Two.Years +
                      Ratio.Days.Identified.To.Total.Days + Opportunity.Amount.USD,
                      method = "knn",
                      data = n_train_sales_set,
                      trControl = trainControl(method = "cv", number = 10, p = 0.9))
# predict on test data set
knn_preds_bp <- predict(train_knn_bp, n_test_sales_set)

### use confusionMatrix to view the results
cm_knn_1  <- confusionMatrix(data = factor(knn_preds_bp), reference =
                              factor(n_test_sales_set$Opportunity.Result))
cm_knn_2 <- confusionMatrix(data = factor(knn_preds_bp), reference =
                             factor(n_test_sales_set$Opportunity.Result),
                             mode = "prec_recall", positive="Won")
# results table for knn
knn_results <- tibble(Model = "knn",
                        Accuracy = cm_knn_1$overall["Accuracy"],
                        Sensitivity = cm_knn_1$byClass["Sensitivity"],
                        Specificity = cm_knn_1$byClass["Specificity"],
                        Precision  = cm_knn_2$byClass["Precision"],
                        F1_Score = cm_knn_2$byClass["F1"])
kable( knn_results, caption = "knn model result" )  %>%
       kable_styling(latex_options = "HOLD_position", font_size = 8)



#' ### 3.5 Quadratic Discriminant Analysis
# qda model
set.seed(211120, sample.kind = "Rounding")
# train qda model
```

```r
train_qda_bp <- train(Opportunity.Result ~ Total.Days.Identified.Through.Qualified +
                      Deal.Size.Category + Ratio.Days.Qualified.To.Total.Days +
                      Revenue.From.Client.Past.Two.Years +
                      Ratio.Days.Identified.To.Total.Days + Opportunity.Amount.USD,
                      data = train_sales_set,
                      method = "qda",
                      trControl = trainControl(method = "cv", number = 10, p = 0.9))
# predict on test data set
qda_preds_bp <- predict(train_qda_bp, test_sales_set)

### use confusionMatrix to view the results
cm_qda_1 <- confusionMatrix(data = factor(qda_preds_bp), reference =
                            factor(test_sales_set$Opportunity.Result))
cm_qda_2 <- confusionMatrix(data = factor(qda_preds_bp), reference =
                            factor(test_sales_set$Opportunity.Result),
                            mode = "prec_recall", positive="Won")
# results table for qda
qda_results <- tibble(Model = "qda",
                      Accuracy = cm_qda_1$overall["Accuracy"],
                      Sensitivity = cm_qda_1$byClass["Sensitivity"],
                      Specificity = cm_qda_1$byClass["Specificity"],
                      Precision  = cm_qda_2$byClass["Precision"],
                      F1_Score = cm_qda_2$byClass["F1"])
kable( qda_results, caption = "qda model result")  %>%
       kable_styling(latex_options = "HOLD_position", font_size = 8)


#' ### 3.6 Random Forest
# !!!this  can take long execution time
# rf model
set.seed(211120, sample.kind = "Rounding")
# train rf model
train_rf_all <- train(Opportunity.Result ~ . ,
                  data = train_sales_set,
                  method = "rf",
                  ntree = 100,    #ntree = 100
                  tuneGrid = data.frame(mtry = seq(1:10)),
                  trControl = trainControl(method = "cv", number = 10, p = 0.9))
# predict on test data set
rf_preds_all <- predict(train_rf_all, test_sales_set)

# use confusionMatrix to view the results
cm_RF_1 <- confusionMatrix(data = factor(rf_preds_all), reference =
                           factor(test_sales_set$Opportunity.Result))
cm_RF_2 <- confusionMatrix(data = factor(rf_preds_all), reference =
                           factor(test_sales_set$Opportunity.Result),
                           mode = "prec_recall", positive="Won")
# results table for rf
rf_results <- tibble(Model = "rf",
                      Accuracy = cm_RF_1$overall["Accuracy"],
                      Sensitivity = cm_RF_1$byClass["Sensitivity"],
                      Specificity = cm_RF_1$byClass["Specificity"],
                      Precision  = cm_RF_2$byClass["Precision"],
```

```
                        F1_Score = cm_RF_2$byClass["F1"])
kable( rf_results, caption = "rf model result" )  %>%
       kable_styling(latex_options = "HOLD_position", font_size = 8)


#' ### 3.7 AdaBoost
# !!!this  can take very long execution time
# adaboost model
set.seed(211120, sample.kind = "Rounding")
# train adaboost model
train_adab_bp <- train(Opportunity.Result ~ Total.Days.Identified.Through.Qualified +
                       Deal.Size.Category + Ratio.Days.Qualified.To.Total.Days +
                       Revenue.From.Client.Past.Two.Years +
                       Ratio.Days.Identified.To.Total.Days + Opportunity.Amount.USD,
                       data = train_sales_set,
                       method = "adaboost",
                       trControl = trainControl(method = "cv", number = 10, p = 0.9))
# predict on test data set
adab_preds_bp <- predict(train_adab_bp, test_sales_set)

# use confusionMatrix to view the results
cm_adab_1 <- confusionMatrix(data = factor(adab_preds_bp), reference =
                               factor(test_sales_set$Opportunity.Result))
cm_adab_2 <- confusionMatrix(data = factor(adab_preds_bp), reference =
                               factor(test_sales_set$Opportunity.Result),
                               mode = "prec_recall", positive="Won")
# results table for adaboost
adaboost_results <- tibble(Model = "adaboost",
                         Accuracy = cm_adab_1$overall["Accuracy"],
                         Sensitivity = cm_adab_1$byClass["Sensitivity"],
                         Specificity = cm_adab_1$byClass["Specificity"],
                         Precision  = cm_adab_2$byClass["Precision"],
                         F1_Score = cm_adab_2$byClass["F1"])
kable( adaboost_results, caption = "adaboost model result" )  %>%
       kable_styling(latex_options = "HOLD_position", font_size = 8)


#' ### 3.8 Ensemble Model
# ensemble for alggorithms "glm", "naive_bayes", "knn", "qda", "rf", "adaboost"
# define algorithms for ensemble model
models <- c( "glm", "naive_bayes", "knn", "qda", "rf", "adaboost")
set.seed(211120, sample.kind = "Rounding")
# make predictions using already trained algorithms
preds <- sapply(models, function(model){
  if (model == "glm") {
  # glm use all predictors
    fit <- train_glm_all
  } else if (model == "naive_bayes") {
  # naive bayes uuse all predictors
    fit <- train_nb_all
  }else if (model == "knn") {
  # knn use set of most important predictors and normalization data
    fit <- train_knn_bp
```

```
  } else if (model == "qda") {
  # qda use set of most important predictors
    fit <- train_qda_bp
  } else if (model == "rf") {
  # Random forest use all predictors
    fit <- train_rf_all
  } else if (model == "adaboost") {
  # adaboost use set of most important predictors
    fit <- train_adab_bp
  }
# predictions and predictions for knn
  if (model == "knn") {
    # knn use data sets with normalization
    pred <- predict(object = fit, newdata = n_test_sales_set)
  }else {
    pred <- predict(object = fit, newdata = test_sales_set)
  }
# these predictions will be used for ensemble
  return(pred)
})
# Combine all models, use votes method to ensemble the predictions
votes <- rowMeans(preds == "Won")
ens_preds <- factor(ifelse(votes > 0.5, "Won", "Loss"))
ens_preds <- factor(ens_preds, levels=c("Won", "Loss"))

# use confusionMatrix to view the results
cm_ens_1 <- confusionMatrix(data = factor(ens_preds), reference =
                            factor(test_sales_set$Opportunity.Result))
cm_ens_2 <- confusionMatrix(data = factor(ens_preds), reference =
                            factor(test_sales_set$Opportunity.Result),
                            mode = "prec_recall", positive="Won")
# results table for Ensemble
ensemble_results <- tibble(Model = "ensemble",
                        Accuracy = cm_ens_1$overall["Accuracy"],
                        Sensitivity = cm_ens_1$byClass["Sensitivity"],
                        Specificity = cm_ens_1$byClass["Specificity"],
                        Precision  = cm_ens_2$byClass["Precision"],
                        F1_Score = cm_ens_2$byClass["F1"])
kable(ensemble_results, caption = "ensemble model result")  %>%
      kable_styling(latex_options = "HOLD_position", font_size = 8)



#' ### 3.9 Best Model Selection
# all model results in table model_results
# results baseline
model_results <- tibble(Model = "baseline",
                        Accuracy = cm_baseline_1$overall["Accuracy"],
                        Sensitivity = cm_baseline_1$byClass["Sensitivity"],
                        Specificity = cm_baseline_1$byClass["Specificity"],
                        Precision  = cm_baseline_2$byClass["Precision"],
                        F1_Score = cm_baseline_2$byClass["F1"])
# glm
```

```r
model_results <- model_results %>% add_row(Model = "glm",
                        Accuracy = cm_glm_1$overall["Accuracy"],
                        Sensitivity = cm_glm_1$byClass["Sensitivity"],
                        Specificity = cm_glm_1$byClass["Specificity"],
                        Precision = cm_glm_2$byClass["Precision"],
                        F1_Score = cm_glm_2$byClass["F1"])
# naive_bayes
model_results <- model_results %>% add_row(Model = "naive_bayes",
                        Accuracy = cm_nb_1$overall["Accuracy"],
                        Sensitivity = cm_nb_1$byClass["Sensitivity"],
                        Specificity = cm_nb_1$byClass["Specificity"],
                        Precision = cm_nb_2$byClass["Precision"],
                        F1_Score = cm_nb_2$byClass["F1"])
# knn
model_results <- model_results %>% add_row(Model = "knn",
                       Accuracy = cm_knn_1$overall["Accuracy"],
                       Sensitivity = cm_knn_1$byClass["Sensitivity"],
                       Specificity = cm_knn_1$byClass["Specificity"],
                       Precision = cm_knn_2$byClass["Precision"],
                       F1_Score = cm_knn_2$byClass["F1"])
# qda
model_results <- model_results %>% add_row(Model = "qda",
                       Accuracy = cm_qda_1$overall["Accuracy"],
                       Sensitivity = cm_qda_1$byClass["Sensitivity"],
                       Specificity = cm_qda_1$byClass["Specificity"],
                       Precision = cm_qda_2$byClass["Precision"],
                       F1_Score = cm_qda_2$byClass["F1"])
# rf
model_results <- model_results %>% add_row(Model = "rf",
                       Accuracy = cm_RF_1$overall["Accuracy"],
                       Sensitivity = cm_RF_1$byClass["Sensitivity"],
                       Specificity = cm_RF_1$byClass["Specificity"],
                       Precision = cm_RF_2$byClass["Precision"],
                       F1_Score = cm_RF_2$byClass["F1"])
# adaboost
model_results <- model_results %>% add_row(Model = "adaboost",
                       Accuracy = cm_adab_1$overall["Accuracy"],
                       Sensitivity = cm_adab_1$byClass["Sensitivity"],
                       Specificity = cm_adab_1$byClass["Specificity"],
                       Precision = cm_adab_2$byClass["Precision"],
                       F1_Score = cm_adab_2$byClass["F1"])
#ensemble
model_results <- model_results %>% add_row(Model = "ensemble",
                       Accuracy = cm_ens_1$overall["Accuracy"],
                       Sensitivity = cm_ens_1$byClass["Sensitivity"],
                       Specificity = cm_ens_1$byClass["Specificity"],
                       Precision  = cm_ens_2$byClass["Precision"],
                       F1_Score = cm_ens_2$byClass["F1"])
#model_results
kable( model_results, caption = "Model results", digits = 4 )  %>%
       kable_styling(latex_options = "HOLD_position", font_size = 8) %>%
       row_spec(0,bold=TRUE)
```

```
# Show ROC and Precision-Recall curves for all models, using precrec package
# glm  scores and labels for precrec package
scores_glm <-  str_replace_all(glm_all_preds, 'Won', '1')
scores_glm <-  str_replace_all(scores_glm, 'Loss', '0')
scores_glm <- as.numeric(as.character(scores_glm))

labels_glm <- str_replace_all(test_sales_set$Opportunity.Result, 'Won', '1')
labels_glm <- str_replace_all(labels_glm, 'Loss', '0')
labels_glm <- as.numeric(as.character(labels_glm))

# naive bayes scores and labels for precrec package
scores_nb <-  str_replace_all(nb_preds_all, 'Won', '1')
scores_nb <-  str_replace_all(scores_nb, 'Loss', '0')
scores_nb <- as.numeric(as.character(scores_nb))

labels_nb <- str_replace_all(test_sales_set$Opportunity.Result, 'Won', '1')
labels_nb <- str_replace_all(labels_nb, 'Loss', '0')
labels_nb <- as.numeric(as.character(labels_nb))

# knn scores and labels for precrec package
scores_knn <-  str_replace_all(knn_preds_bp, 'Won', '1')
scores_knn <-  str_replace_all(scores_knn, 'Loss', '0')
scores_knn <- as.numeric(as.character(scores_knn))

#knn use set with normalization
labels_knn <- str_replace_all(n_test_sales_set$Opportunity.Result, 'Won', '1')
labels_knn <- str_replace_all(labels_knn, 'Loss', '0')
labels_knn <- as.numeric(as.character(labels_knn))

# qda scores and labels for precrec package
scores_qda <-  str_replace_all(qda_preds_bp, 'Won', '1')
scores_qda <-  str_replace_all(scores_qda, 'Loss', '0')
scores_qda <- as.numeric(as.character(scores_qda))

labels_qda <- str_replace_all(test_sales_set$Opportunity.Result, 'Won', '1')
labels_qda <- str_replace_all(labels_qda, 'Loss', '0')
labels_qda <- as.numeric(as.character(labels_qda))

# rf scores and labels for precrec package
scores_rf <-  str_replace_all(rf_preds_all, 'Won', '1')
scores_rf <-  str_replace_all(scores_rf, 'Loss', '0')
scores_rf <- as.numeric(as.character(scores_rf))

labels_rf <- str_replace_all(test_sales_set$Opportunity.Result, 'Won', '1')
labels_rf <- str_replace_all(labels_rf, 'Loss', '0')
labels_rf <- as.numeric(as.character(labels_rf))

# adaboost scores and labels for precrec package
scores_adab <-  str_replace_all(adab_preds_bp, 'Won', '1')
scores_adab <-  str_replace_all(scores_adab, 'Loss', '0')
scores_adab <- as.numeric(as.character(scores_adab))

labels_adab <- str_replace_all(test_sales_set$Opportunity.Result, 'Won', '1')
```

```
labels_adab <- str_replace_all(labels_adab, 'Loss', '0')
labels_adab <- as.numeric(as.character(labels_adab))

# ensemble scores and labels for precrec package
scores_ens <-  str_replace_all(ens_preds, 'Won', '1')
scores_ens <-  str_replace_all(scores_ens, 'Loss', '0')
scores_ens <- as.numeric(as.character(scores_ens))

labels_ens <- str_replace_all(test_sales_set$Opportunity.Result, 'Won', '1')
labels_ens <- str_replace_all(labels_ens, 'Loss', '0')
labels_ens <- as.numeric(as.character(labels_ens))

# save results ROC, PRC plots in figures
png(file="figs/result_plot_1.png", width=480, height=180)

# one plot for "glm", "naive_bayes", "knn", "qda", "rf", "adaboost" and  ensemble
# join score vectors
scores1 <-  join_scores(scores_glm, scores_nb, scores_knn, scores_qda,
                        scores_rf, scores_adab, scores_ens)
# join label vectors
labels1 <- join_labels(labels_glm, labels_nb, labels_knn, labels_qda,
                       labels_rf, labels_adab, labels_ens)
# specify model names and test data set names
mmmdat1 <- mmdata(scores1, labels1, modnames=
          c("glm", "naive_bayes", "knn", "qda", "rf", "adaboost", "ensemble" ),
          dsids = c(1, 2, 3, 4, 5, 6, 7))
# calculate curves for multiple models and multiple test datasets
mmcurves <- evalmod(mmmdat1)
# show average ROC curves
p1 <-  autoplot(mmcurves, "ROC") + labs(title="ROC")
# show average Precision-Recall curves
p2 <- autoplot(mmcurves, "PRC") + labs(title="Precision-Recall")
ggarrange(p1, p2,  nrow=1, ncol=2, common.legend = TRUE, legend="right" )
dev.off()


# call results plots in report
include_graphics("figs/result_plot_1.png",
                 auto_pdf = getOption("knitr.graphics.auto_pdf", FALSE), dpi=NA)


#' ### 3.10 Final Validation
# !!!this can take long execution time
# final model rf, sales_set and validation set
set.seed(211120, sample.kind = "Rounding")
# train final rf model on sales_set
train_final_rf_all <- train(Opportunity.Result ~ . ,
                      data = sales_set,
                      method = "rf",
                      ntree = 100,    #ntree = 100
                      tuneGrid = data.frame(mtry = seq(1:10)),
                      trControl = trainControl(method = "cv", number = 10, p = 0.9))
# predict for final rf model on validation set
```

```r
final_rf_preds_all <- predict(train_final_rf_all, validation_set)


# use confusionMatrix to view the results
final_cm_RF_1 <- confusionMatrix(data = factor(final_rf_preds_all),
                          reference = factor(validation_set$Opportunity.Result))
final_cm_RF_2 <- confusionMatrix(data = factor(final_rf_preds_all),
                          reference = factor(validation_set$Opportunity.Result),
                          mode = "prec_recall", positive="Won")

# final validation results table for  rf model on validation set
final_val_results <- tibble(Model = "final validation rf",
                        Accuracy = final_cm_RF_1$overall["Accuracy"],
                        Sensitivity = final_cm_RF_1$byClass["Sensitivity"],
                        Specificity = final_cm_RF_1$byClass["Specificity"],
                        Precision  = final_cm_RF_2$byClass["Precision"],
                        F1_Score = final_cm_RF_2$byClass["F1"])
kable(final_val_results, caption = "rf final model results")  %>%
  kable_styling(latex_options = "HOLD_position", font_size = 8)



# end Time ALL
endTime_All_2 <-  Sys.time()
# total time ALL
endTime_All_2 - startTime_ALL_2


# end.
```