Vendex Challenge

AUTHOR
Team A11

Project: Boosting sales and achieving operational excellence at Vendex

Vendex is a leading vending and coffee services company in Europe, with a turnover of 984 million euros on 2017 and employing approximately 5,500 people. Founded in 1967 and headquartered in London, Vendex. has grown its geographic market presence to 15 countries across Europe. Vendex serves more than 8 million consumers every day at its 165,000 point of sales addressing the growing need for out of home food and beverage services at the workplace and on the go.

Vendex roots stand for service excellence, a high-quality product offering and innovative concepts for out of home food and beverage services.

Currently, Vendex is putting a strong focus on boosting sales and profitability by improving pricing and assortment strategy and achieving operational excellence through Advanced Analytics.

Pre-Work

Load the data and do some initial data exploration

```
#import relevant libraries
library(readr)
library(dplyr)
library(lubridate)
library(data.table)
library(tidyverse)
library(tidymodels)
library(GGally)
library(corrplot)
library(ggplot2)
library(vip)         # Feature importance
library(probably)    # Thresholds evaluation
library(scales)
library(janitor)     # Data cleaning
```

```
library(skimr)      # Data summarization
library(themis)     # Extra recipes
library(pROC)

# load the data (update with your specific path)
#file_names <- list.files(path = './DATA/', pattern = '.csv', full.na
# check file_names for correct path
machines <- read.csv("../machine_data.csv")
products <- read.csv("../product_data.csv")
transactions <- read.csv("../transactional_data.csv")

# initial exploration beofre assignment tasks (commented out for read
#summary(machines)
#summary(transactions)
#summary(products)

#head(machines)
#head(transactions)
#head(products)
```

Part 1: General overview and Data Treament

Question 1 : Exploratory Analysis

a. How many machines are there?

There are 2495 machines in the data set.

```
# count number of rows
nrow(unique(machines))
```

```
[1] 2495
```

b. What percentage of them are small?

38.44% of all machines are small

```
# number of machines
machines %>%
  group_by(small_machine) %>%
```

```
  summarise(n=n(),.groups = 'drop') %>%
  mutate(freq = n / sum(n))
```

```
# A tibble: 2 × 3
  small_machine     n  freq
          <int> <int> <dbl>
1             0  1536 0.616
2             1   959 0.384
```

```
# alternative way (commented out for readability)
# prop.table(table(machines$small_machine))
```

c) How do they distribute in terms of location type i.e. transport, petrol station?

There are 58.52% of machines at transport stations, 13.75% at petrol stations, 27.7% at other locations as well as one Nan value.

```
machines %>%
  group_by(location_type) %>%
  summarise(n=n(),.groups = 'drop') %>%
  mutate(freq = n / sum(n))
```

```
# A tibble: 4 × 3
  location_type      n     freq
  <chr>          <int>    <dbl>
1 others           691 0.277
2 petrol station   343 0.137
3 transport       1460 0.585
4 <NA>               1 0.000401
```

d. How many products are there? Which category has the highest number of products?

There are 63 products. Carbonates and energy drinks has the highest number of products with 13 (20.63%).

```
#Calculate the number of products
print(nrow(unique(products)))
```

```
[1] 63
```

```r
# Calculate the categroy with the highest number of products
products %>%
  group_by(category) %>%
  summarise(n=n(),.groups = 'drop') %>%
  mutate(freq = n / sum(n)) %>%
  arrange(desc(freq))
```

```
# A tibble: 8 × 3
  category                          n   freq
  <chr>                         <int>  <dbl>
1 Carbonates and energy drinks     13 0.206
2 Chocolate based                  11 0.175
3 Juice, tea and smoothies          8 0.127
4 Salty                             8 0.127
5 Cookies, pastries and cereals     7 0.111
6 Sugar candy                       7 0.111
7 Water                             5 0.0794
8 Milk based                        4 0.0635
```

e. Which category has the highest and lowest average price? And within snacks or drinks?

Overall milk-based products (3.43 €) have the highest average price while sugar candy have the lowest average price (2.30 €).

```r
# category based price calculation
products %>%
  group_by(category) %>%
  summarise(avg_price=mean(price)) %>%
  arrange(desc(avg_price))
```

```
# A tibble: 8 × 2
  category                      avg_price
  <chr>                             <dbl>
1 Milk based                         3.42
2 Carbonates and energy drinks       3.26
3 Water                              3.26
4 Juice, tea and smoothies           2.86
5 Salty                              2.72
6 Cookies, pastries and cereals      2.39
7 Chocolate based                    2.31
8 Sugar candy                        2.3
```

Just between the groups snacks and drinks, drinks has a higher average price of 3,18 € and snacks a lower of 2.42 €.

```
# between snacks and drinks
products %>%
  group_by(type_drink_snack) %>%
  summarise(avg_price=mean(price)) %>%
  arrange(desc(avg_price))
```

```
# A tibble: 2 × 2
  type_drink_snack avg_price
  <chr>                <dbl>
1 drink                 3.18
2 snack                 2.42
```

Within the drinks segment, the highest price is from milk-based products (3.43 €) while the lowest is Juice, Tea and Smoothies (2.86€). Among the foods, the highest priced food group is 'salty' (2.73€), while the lowest price is from the Sugar Candy with 2.42 €.

```
# grouped by category and snack/drink
products %>%
  group_by(type_drink_snack, category) %>%
  summarise(avg_price=mean(price)) %>%
  arrange(desc(avg_price))
```

```
`summarise()` has grouped output by 'type_drink_snack'. You can
override using
the `.groups` argument.

# A tibble: 8 × 3
# Groups:   type_drink_snack [2]
  type_drink_snack category                     avg_price
  <chr>            <chr>                            <dbl>
1 drink            Milk based                        3.42
2 drink            Carbonates and energy drinks      3.26
3 drink            Water                             3.26
4 drink            Juice, tea and smoothies          2.86
5 snack            Salty                             2.72
6 snack            Cookies, pastries and cereals     2.39
7 snack            Chocolate based                   2.31
8 snack            Sugar candy                       2.3
```

f) Restricting the transactional data to March 2017, what's the average daily items among small and big machines?

Daily Average Items among small machines were 7.71, while for big machines they were 9.89. The boost in sales for bigger machines can be explained 2-fold: Firstly, bigger machines usually have more products and a greater selection, thus they should also have higher sales. Secondly, the bigger machines are probably set up at more favorable locations (e.g. more foot traffic).

```
# create the dataframe with restricted transactions data (March only)
# transactions_march  <- transactions %>%
#  filter(between(date, as.Date('2017-03-01'), as.Date('2017-03-31'))
transactions_march <- transactions[month(as.Date(transactions$date))
#select the necessary machine data columns and merge them with the da
machines_selec <- machines %>%
  select(machine, small_machine)

transactions_march_merged <- merge(transactions_march, machines_selec

# Calculate daily items per machine (incl. type of machine for later
march_average_data <- transactions_march_merged %>%
  group_by(machine, small_machine) %>%
  summarize(average_daily_items_per_machine = n() / n_distinct(as.Dat
```

```
`summarise()` has grouped output by 'machine'. You can override using
the
`.groups` argument.
```

```
# Calculate the average per group
march_average_data %>%
  group_by(small_machine) %>%
  summarise(average_daily_items_by_type = mean(average_daily_items_pe
```

```
# A tibble: 2 × 2
  small_machine average_daily_items_by_type
          <int>                       <dbl>
1             0                        9.90
2             1                        7.71
```

Question 2 : Trend Analysis

a) Is there a general trend in the number of snacks and drinks as the months progress from January to April? Is it the same for snacks and drinks? Why do you think that might be so?

- For snacks there is not real trend to be identified as the months progresses from January to April
- However, for drinks we identified that there is an overall upward trend over the same time period. This can be explained by the fact that with rising temperatures, people are more likely to buy drinks and thus, also buy them from our vending machines. Food consumption patterns are not influenced by the heat.

b) Is there shorter time period trend as well? Is it the same for snacks and drinks? What do you might be the cause?

- When looking at the short-term trend, they are very similar for both drinks and snacks: For both categories, there are around 4 spikes in equal distance.
- We believe that this can be attributed to the different consumption patterns of weekends and weekdays. We believe that the spiked represent weekend consumption patterns as more people use the transports and routes to travel and thus use vending machines more.

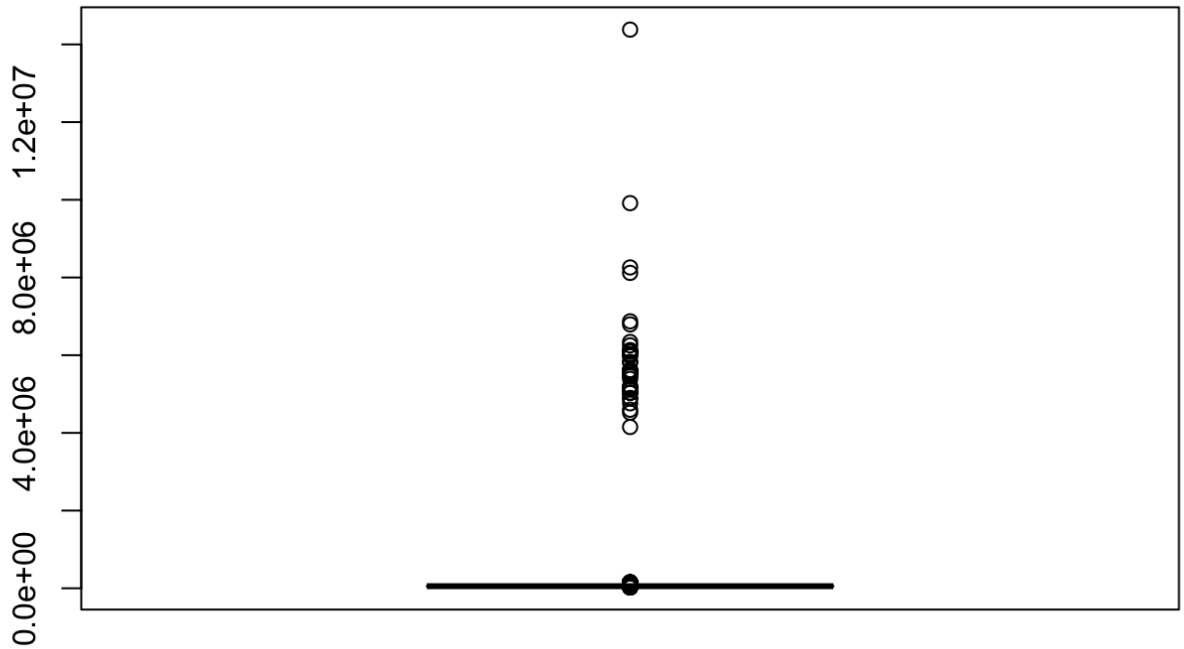
Question 3: Outlier & NA treatment

a) Are there outliers? How would you treat them?

When first analyzing the summary given in the assignment, we notice big differences between mean and median as well as the 3rd quartile and the maximum. This is a first indicator that we have outliers in our data. Thus, we took a closer look at the data behind the summary.

After plotting the data with a boxplot as well as computing the IQR, we can conclude that there are 184 outliers that we then removed. Since we plan to run a linear regression model later, those would otherwise skew our results heavily. We worked with the 1.5 IQR as an industry standard.

```r
# assign income_average column to a vector for easier work
avg_income <- machines$income_average

# 1st Check for outliers with Box plot
boxplot(avg_income)

# Identify any outliers
outliers1 <- boxplot(avg_income)$out
```




```r
# Check how many outliers
length(outliers1)
```

```
[1] 184
```

```r
# Calculate the outliers with the IQR range (double check the boxplot

# in order to effectively use the IQR we need to omit all NA values
avg_income_complete <- na.omit(avg_income)

#Calculate the Quartiles
```

```r
quartiles <- quantile(avg_income_complete, c(0.25, 0.75))

# Compute the IQR
iqr <- IQR(avg_income_complete)
iqr
```

```
[1] 11553
```

```r
# Compute the lower and upper limits of the IQR
lower_limit <- quartiles[1] - 1.5 * iqr
upper_limit <- quartiles[2] + 1.5 * iqr

# Check the lower and upper limits
print(lower_limit)
```

```
    25%
31265.5
```

```r
print(upper_limit)
```

```
    75%
77477.5
```

```r
# Find outliers
outliers <- avg_income_complete[avg_income_complete <= lower_limit |
length(outliers) # same as in boxplot
```

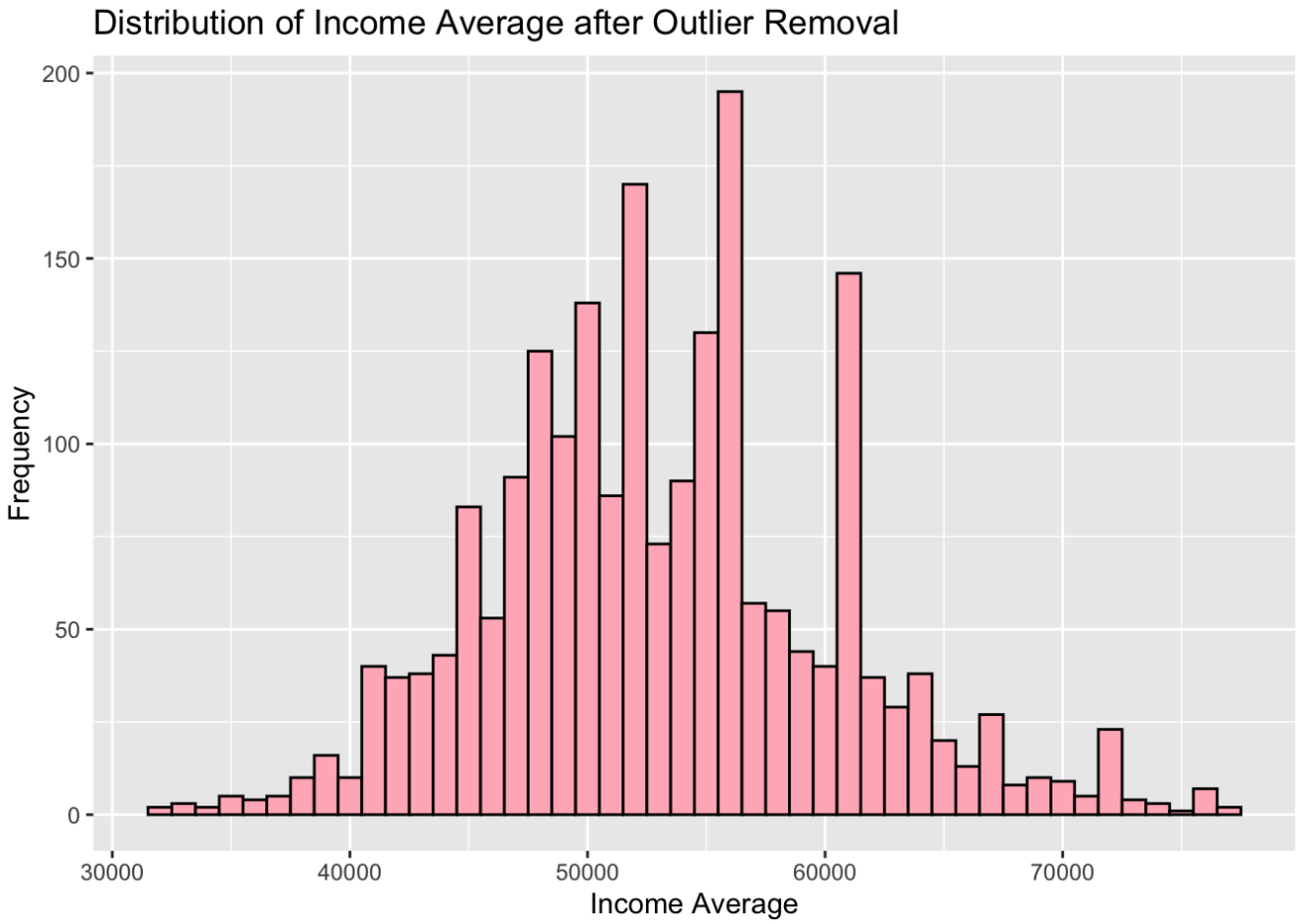
```
[1] 184
```

```r
# apply to the dataframe
machines <- machines[!(machines$income_average %in% outliers), ]
nrow(machines)
```

```
[1] 2311
```

```r
#plot distribution of average_income after removal of outliers
ggplot(data = machines, aes(x = income_average)) +
    geom_histogram(binwidth = 1000, color = "black", fill = "lightpin
    labs(title = "Distribution of Income Average after Outlier Remova
```

```
Warning: Removed 182 rows containing non-finite values
(`stat_bin()`).
```



b) Deal with NAs

There are several ways to deal with NA values.

1. **Drop the rows**: Remove data points that contain missing values. While it is the least “intrusive” method it can lead to a loss of information and explanatory power. We will only pursue the option when there are few rows missing. Another way could be to drop the feature (column) completely. This would only apply if that column has no great explanatory value (not the case here).

2. **Replace with mean or median**: Replace missing values with the mean/median of the non-missing values in the same column. While you are still changing/manipulating the distribution, this method helps to only do so minimally and maintain the overall distribution and structure of the data.

3. **Replace with 0**: Replace missing values with the 0, when the feature does not apply to all datapoints, eg. if the vending machine is not at a transport location it cannot have Working Passengers .

The individual choice of method depends on the business case. In this case, we saw that we have missing values in

- **location_type**: Will be dropped
- **train_AvgDailyPassengers**: Will be replaced by 0
- **train_AvgWorkingDayPassengers**: Will be replaced by 0
- **total_number_of_routes_600**: Will be replaced by 0
- **income_average**: Will be replaced by mean

Further explanations in the individual processing

```
# create overview of missing values
lapply(X = machines, FUN = function(x) sum(is.na(x)))
```

```
$machine
[1] 0

$location_type
[1] 1

$num_vendex_nearby_300
[1] 0

$train_AvgDailyPassengers
[1] 1196

$train_AvgWorkingDayPassengers
[1] 1196

$n_density_5km
[1] 0

$income_average
[1] 182

$total_number_of_routes_600
[1] 100

$num_hotels
[1] 0

$num_hotels_45
[1] 0
```

```
$small_machine
[1] 0
```

```r
# Method 1: Drop rows with missing values. As location_type has only
machines <- machines %>%
  filter(!is.na(location_type))

# Method 2: Replace by average to maintain the distribution for avera
# We omit NAs to calculate the true mean
mean_avg_income <- round(mean(na.omit(machines$income_average)))
machines$income_average <- machines$income_average %>%
  replace_na(mean_avg_income)

# Method 3: Replace with 0s
# For both daily variables as well as the routes feature we assume th

# Average Daily Passenger
machines$train_AvgDailyPassengers <- machines$train_AvgDailyPassenger
  replace_na(0)

# Average Daily Working Passenger
machines$train_AvgWorkingDayPassengers <- machines$train_AvgWorkingDa
  replace_na(0)

# Routes
machines$total_number_of_routes_600 <- machines$total_number_of_route
  replace_na(0)

# Check if all NA were dealt with
lapply(X = machines, FUN = function(x) sum(is.na(x)))
```

```
$machine
[1] 0

$location_type
[1] 0

$num_vendex_nearby_300
[1] 0

$train_AvgDailyPassengers
[1] 0
```

```
$train_AvgWorkingDayPassengers
[1] 0

$n_density_5km
[1] 0

$income_average
[1] 0

$total_number_of_routes_600
[1] 0

$num_hotels
[1] 0

$num_hotels_45
[1] 0

$small_machine
[1] 0
```

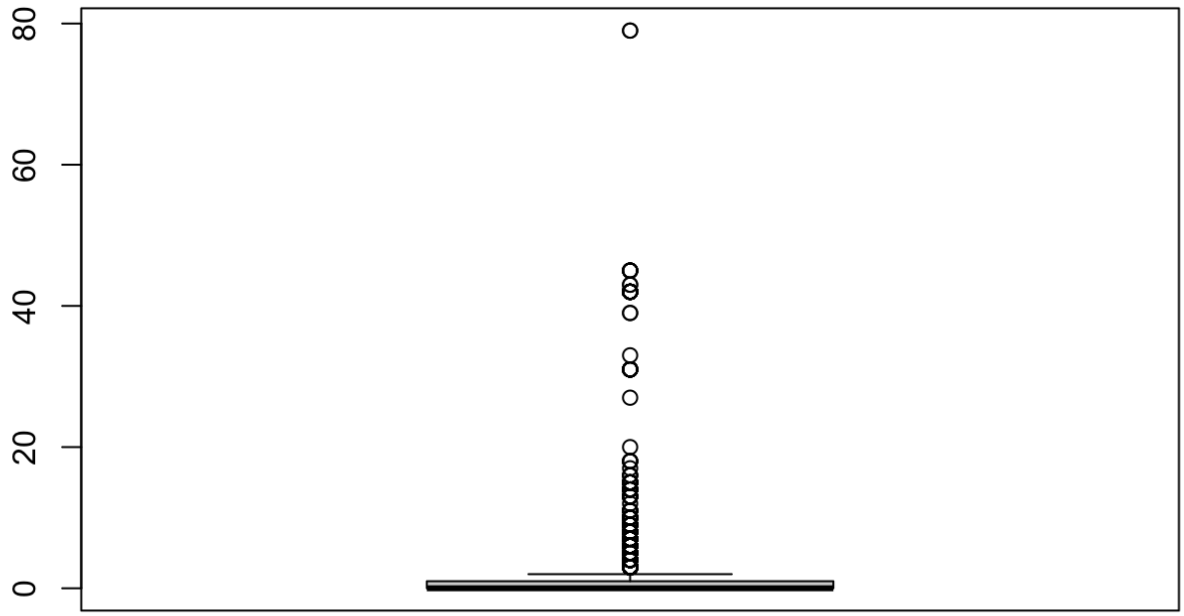
```r
# Check that only 184 values removed
nrow(machines)
```

```
[1] 2310
```

Question 4: Interpretation of Boxplot

According to the boxplot the median of hotels is 0. This is confirmed by our calculations.

```r
boxplot(machines$num_hotels)
```




```
median_hotels <- median(machines$num_hotels)
median_hotels
```

```
[1] 0
```

Question 5: Linear Model

Create a linear model with:

1. Machine size (big or small)
2. Income of the area
3. Number of routes in the area
4. Number of hotels with 4 and 5 stars in the area
5. 0-1 Indicator of whether the machine has train_AvgDailyPassengers informed
6. Number of other Vendex machines in the area

```
# create the Avg Daily Passenger Indicator (is_transport_loc)
machines <- machines %>% mutate(is_transport_loc = ifelse(train_AvgDa
```

```
# create the target variable
transactions_target <- transactions %>%
  group_by(machine) %>%
  summarise(average_daily_items_per_machine = n() / n_distinct(as.Dat

# merge datasets
df <- merge(transactions_target, machines, by = "machine")

# check that all merged well
colnames(df)
```

```
 [1] "machine"
"average_daily_items_per_machine"
 [3] "location_type"                    "num_vendex_nearby_300"
 [5] "train_AvgDailyPassengers"
"train_AvgWorkingDayPassengers"
 [7] "n_density_5km"                    "income_average"
 [9] "total_number_of_routes_600"       "num_hotels"
[11] "num_hotels_45"                    "small_machine"
[13] "is_transport_loc"
```

```
# set seed
set.seed(2022)

# create split
df_split <- initial_split(df, prop = .75, strata = average_daily_item
df_train <- training(df_split)
df_test <- testing(df_split)

# create score model
df_lm_model <- lm(formula = average_daily_items_per_machine ~ small_m
```

a) Do all variables show statistical significance? Which ones doesn't? How do you know?

In order to assess statistical significance we can look at the summary function of the model and look at the t- and p-values as well as the asterixes. If the t-value is above 2.5 or the Pr(>|t|) is lower than a certain threshold:

- 0.1/ * –> significance at 10% (probability to make a mistake in the amount of the estimate or higher)

- 0.5/** –> significance at 5%
- 0.01/*** –> significance at 1%

For our model this leads to the following interpretation:

- **Highly significant (*****): small_machine, is_transport_loc , num_hotels_45
- **Significant**(*): num_vendex_nearby_300
- **Not significant**: total_number_of_routes_600, income_average

```
summary(df_lm_model)
```

```
Call:
lm(formula = average_daily_items_per_machine ~ small_machine + 
    income_average + total_number_of_routes_600 + num_hotels_45 + 
    is_transport_loc + num_vendex_nearby_300, data = df_train)

Residuals:
    Min      1Q  Median      3Q     Max 
-10.853  -2.856  -1.054   1.754  36.599 

Coefficients:
                             Estimate Std. Error t value Pr(>|t|)    
(Intercept)                 8.091e+00  8.699e-01   9.301  < 2e-16 ***
small_machine              -1.700e+00  2.281e-01  -7.454 1.42e-13 ***
income_average             -1.406e-05  1.621e-05  -0.867   0.3858    
total_number_of_routes_600  2.450e-03  2.123e-03   1.154   0.2486    
num_hotels_45               4.491e-01  6.186e-02   7.260 5.83e-13 ***
is_transport_loc            3.396e+00  2.322e-01  14.627  < 2e-16 ***
num_vendex_nearby_300      -9.759e-02  4.068e-02  -2.399   0.0166 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.616 on 1723 degrees of freedom
Multiple R-squared:  0.1868,    Adjusted R-squared:  0.184 
F-statistic: 65.97 on 6 and 1723 DF,  p-value: < 2.2e-16
```

b) Build another linear model replacing “total_number_of_routes_600” with its log. Does this new variable show statistical significance?

The variable now is highly significant at 0.001 alpha-level (3 asterixes). Further, we observe that num_vendex_nearby_300 was bumped up in significance from 1 asterix to 3 asterixes. Thus, we will maintain the new variable.

```
# create new variable

df_log <-  df %>%
  mutate(log_transport = log10(total_number_of_routes_600 + 1)) %>% #
  select(-total_number_of_routes_600)

colnames(df_log)
```

```
 [1] "machine"
"average_daily_items_per_machine"
 [3] "location_type"                    "num_vendex_nearby_300"
 [5] "train_AvgDailyPassengers"
"train_AvgWorkingDayPassengers"
 [7] "n_density_5km"                    "income_average"
 [9] "num_hotels"                       "num_hotels_45"
[11] "small_machine"                    "is_transport_loc"
[13] "log_transport"
```

```
# set seed
set.seed(2022)
df_log_split <- initial_split(df_log, prop = .75, strata = average_da
df_log_train <- training(df_log_split)
df_log_test <- testing(df_log_split)

#create adapted model
df_lm_model_2 <- lm(formula = average_daily_items_per_machine ~ small

# check summary
summary(df_lm_model_2)
```

```
Call:
lm(formula = average_daily_items_per_machine ~ small_machine +
    income_average + log_transport + num_hotels_45 +
num_vendex_nearby_300 +
    is_transport_loc, data = df_log_train)

Residuals:
```

```
    Min      1Q  Median      3Q     Max 
-11.266  -2.894  -0.982   1.651  35.826 

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)    
(Intercept)             6.977e+00  8.879e-01   7.858 6.84e-15 ***
small_machine          -1.653e+00  2.264e-01  -7.302 4.30e-13 ***
income_average         -1.802e-05  1.610e-05  -1.120    0.263    
log_transport           1.321e+00  2.455e-01   5.381 8.40e-08 ***
num_hotels_45           4.088e-01  5.536e-02   7.384 2.37e-13 ***
num_vendex_nearby_300 -1.237e-01  2.859e-02  -4.326 1.61e-05 ***
is_transport_loc        3.188e+00  2.333e-01  13.664  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.58 on 1723 degrees of freedom
Multiple R-squared:  0.1996,    Adjusted R-squared:  0.1968 
F-statistic: 71.63 on 6 and 1723 DF,  p-value: < 2.2e-16
```

Final Model

For the final model, we removed income_average as it was not significant and kept the feature engineered variable log_transport.

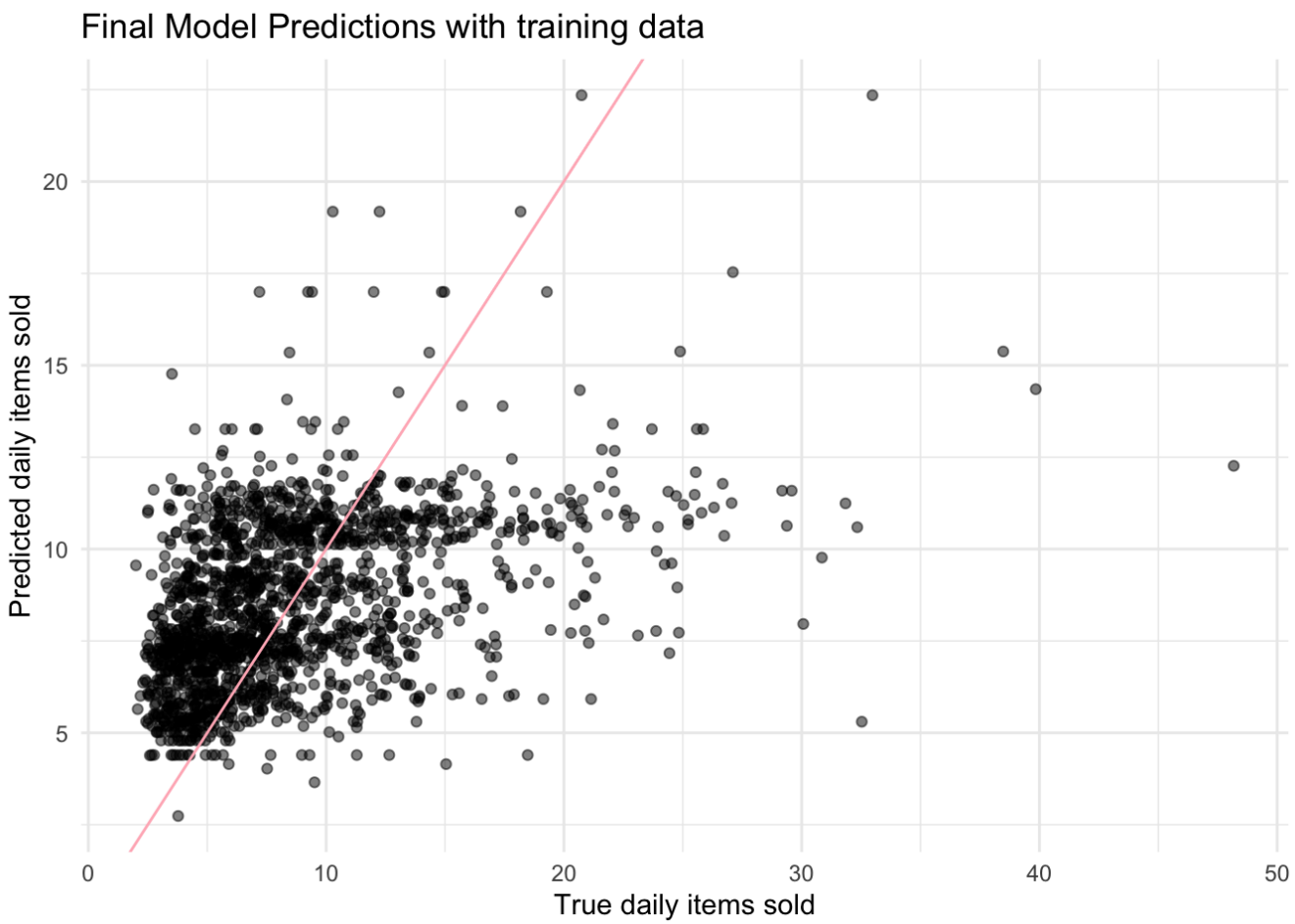
```
df_final <-  df_log %>%
  select(-income_average)

# set seed
set.seed(2022)
df_final_split <- initial_split(df_final, prop = .75, strata = averag
df_final_train <- training(df_final_split)
df_final_test <- testing(df_final_split)

#create adapted model
final_model<- lm(formula = average_daily_items_per_machine ~ small_ma
```

This model outputs and RMSE of 4.57, indicating that on average the predicted values from the model are about 4.57 units away from the true values of the Average Daily Items per machine.

The R-squared (R^2) value of 0.199 indicates that about 19.9% of the variability in the target variable can be explained by the predictor variables included in the model. This indicates that the model is not able to explain a large proportion of the

variability in the data, and there may be other factors or variables that influence the target variable but are not captured by the model.

```
# Define the function to compare predictions vs actual
predictions_vs_actual_plot <- function(df, x, y, x_title, y_title, ti
  x <- enquo(x)
  y <- enquo(y)
  ggplot(df, aes(!!x, !!y)) +
    geom_point(alpha = .5) +
    labs(x = x_title, y = y_title) +
    geom_abline(intercept = 0, slope = 1, color = 'lightpink') +
    ggtitle(title) +
    theme_minimal()
}

# Bind the predicted values to the training data
base_train_final_predictions <- df_final_train %>%
  bind_cols(.preds = predict(final_model, newdata = df_final_train))

# plot actual vs predicted
predictions_vs_actual_plot(df = base_train_final_predictions, x = ave
```



```
names(base_train_final_predictions)
```

```
 [1] "machine"
"average_daily_items_per_machine"
 [3] "location_type"                    "num_vendex_nearby_300"
 [5] "train_AvgDailyPassengers"
"train_AvgWorkingDayPassengers"
 [7] "n_density_5km"                    "num_hotels"
 [9] "num_hotels_45"                    "small_machine"
[11] "is_transport_loc"                 "log_transport"
[13] ".preds"
```

```
# Assess performance
rmse(base_train_final_predictions, truth = average_daily_items_per_ma
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard        4.57
```

```
rsq(base_train_final_predictions, truth = average_daily_items_per_mac
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rsq     standard       0.199
```

c) How many daily items less do small machines sell all other factors remaining equal?

In order to see the impact of having a small machine, holding all other variables constant, we will look at the coefficient. Based on this we can conclude that having a small machine leads o a decrease in machine sales of around 1.65 units on average. This makes intuitive sense and confirms our calculations in Task 1 which showed that smaller machines sell less units on average due to size, choice and location constraints

```
# check summary from coefficient table
summary(final_model)
```

```
Call:
lm(formula = average_daily_items_per_machine ~ small_machine +
    log_transport + num_hotels_45 + num_vendex_nearby_300 +
is_transport_loc,
    data = df_final_train)

Residuals:
    Min      1Q  Median      3Q     Max
-11.251  -2.896  -0.976   1.681  35.908

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)            6.04325    0.30480  19.827  < 2e-16 ***
small_machine         -1.64811    0.22639  -7.280 5.05e-13 ***
log_transport          1.30814    0.24523   5.334 1.09e-07 ***
num_hotels_45          0.40165    0.05499   7.303 4.26e-13 ***
num_vendex_nearby_300 -0.12359    0.02859  -4.323 1.63e-05 ***
is_transport_loc       3.17453    0.23301  13.624  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.58 on 1724 degrees of freedom
Multiple R-squared:  0.1991,    Adjusted R-squared:  0.1967
F-statistic: 85.69 on 5 and 1724 DF,  p-value: < 2.2e-16
```

d) What's effect on machine sales does having other nearby machines all other factors remaining equal?

Same as before we can look at the coefficient here. Having other machines nearby results in an average loss of -0.12 units. This is also intuitive. If there is another machine nearby, sales of the respective machines will go down due to increased competitive pressure

e) What are the real daily sales of the top & bottom20% machines according to your model? What's the top20%/bottom20% ratio?

The real daily sales of our top 20% machines are 12.15, and of our bottom 20% are 6.15. The ratio thus is 1.97.

```
#Creating the predictions
final_predictions <- df_final_train %>%
  bind_cols(predict(final_model, newdata = df_final_train)) %>%
  rename(prediction = ...13)
```

```
New names:
• `` -> `...13`
```

```
# Order the data by the predictions in descending order and extract t
top_20_percent_machines <- final_predictions %>%
  arrange(desc(prediction)) %>%
  head(n = nrow(final_predictions) * 0.2)

bottom_20_percent_machines <- final_predictions %>%
  arrange(desc(prediction)) %>%
  tail(n = nrow(final_predictions) * 0.2)

#Extract the real daily sales of the top 20% and bottom 20% machines:
top_20_percent_real_daily_sales <- top_20_percent_machines %>%
  select(average_daily_items_per_machine) %>%
  summarize(mean(average_daily_items_per_machine))

bottom_20_percent_real_daily_sales <- bottom_20_percent_machines %>%
  select(average_daily_items_per_machine) %>%
  summarize(mean(average_daily_items_per_machine))

#Calculate the top20%/bottom20% ratio:
ratio <- top_20_percent_real_daily_sales/bottom_20_percent_real_daily
ratio
```

```
  mean(average_daily_items_per_machine)
1                              1.974503
```

f) Given the following 2 locations for a big machine:

i. Supermarket entrance, 2 nearby hotels of 4 stars, 20 transport routes, no nearby machines ii. Transport station, no nearby hotels of 4 or 5 stars, 10 transport routes nearby, 3 nearby Vendex machines Which location would you choose and why?

Based on our current model, we would prefer the second location as it gives us on average higher daily unit sales by 1.09.

```
# define dataframes with relevant characteristics

location1 <- data.frame(
  is_transport_loc = 0,
  num_hotels_45 = 2,
  log_transport = log(20),
  num_vendex_nearby_300 = 0,
  small_machine = 0
)

location2 <- data.frame(
  is_transport_loc = 1,
  num_hotels_45 = 0,
  log_transport = log(10),
  num_vendex_nearby_300 = 3,
  small_machine = 0
)

sales_loc1 <- predict(final_model, newdata = location1)
sales_loc2 <- predict(final_model, newdata = location2)

diff <- sales_loc2 - sales_loc1
print(paste("Sales Location 1:", sales_loc1))
```

```
[1] "Sales Location 1: 10.7653815308787"
```

```
print(paste("Sales Location 2:", sales_loc2))
```

```
[1] "Sales Location 2: 11.8590989155943"
```

```
print(paste("Sales Location 2 has higher daily average unit sales by"
```

```
[1] "Sales Location 2 has higher daily average unit sales by
1.09371738471558"
```

Part 2: Introduction to Probability Models

Creating the relevant variable of the model

Question 1

Load the data and do some initial data exploration. Then merge the transactional dataset with the machine failures data set setting failure variable to 0 when no failure is recorded

```
# load the data
failures <- read.csv("../machine_failures.csv")
transactions <- read.csv("../transactional_data.csv")

# explore
summary(failures)
```

```
   machine           column       timestamp            failure
Min.   :   1.0   Min.   : 0.00   Length:12378       Min.   :1
1st Qu.: 858.2   1st Qu.:13.00   Class :character   1st Qu.:1
Median :1467.5   Median :24.00   Mode  :character   Median :1
Mean   :1459.4   Mean   :27.25                      Mean   :1
3rd Qu.:2072.0   3rd Qu.:39.00                      3rd Qu.:1
Max.   :2724.0   Max.   :69.00                      Max.   :1
```

```
summary(transactions)
```

```
   machine      timestamp             date               column
Min.   :   1   Length:1840477     Length:1840477     Min.   : 0.00
1st Qu.: 880   Class :character   Class :character   1st Qu.:13.00
Median :1472   Mode  :character   Mode  :character   Median :25.00
Mean   :1462                                        Mean   :27.03
3rd Qu.:2072                                        3rd Qu.:39.00
Max.   :2724                                        Max.   :69.00
product_name
Length:1840477
Class :character
Mode  :character
```

```r
head(failures)
```

```
  machine column           timestamp failure
1       1     50 2017-01-23 15:29:36       1
2       1     55 2017-01-28 11:14:06       1
3       1     33 2017-02-13 14:16:34       1
4       1     43 2017-02-27 09:14:18       1
5       1     47 2017-03-21 09:24:19       1
6       1     40 2017-03-26 12:21:14       1
```

```r
head(transactions)
```

```
  machine           timestamp       date column        product_name
1       1 2017-01-04 14:02:34 2017-01-04     55 water_(sparkling)_1
2       1 2017-01-02 15:00:20 2017-01-02     55 water_(sparkling)_1
3       1 2017-01-02 14:35:57 2017-01-02     56 water_(sparkling)_1
4       1 2017-01-13 14:01:38 2017-01-13     56 water_(sparkling)_1
5       1 2017-01-11 13:34:12 2017-01-11     56 water_(sparkling)_1
6       1 2017-01-29 11:34:31 2017-01-29     56 water_(sparkling)_1
```

```r
# Check rows and columnnames for merger (commented out for better rea
#colnames(transactions)
#colnames(failures)
#nrow(failures)
#nrow(transactions)

# merge failures with the transactions dataset
df <- merge(transactions,failures, by=c("column","machine", "timestam
df <- df %>% mutate(failure = replace_na(failure, 0))
```

Question 2

Create a variable called “last_vend” containing the timestamp of the previous sale of each machine

```r
# convert the 'timestamp' column to a POSIXct format
df$timestamp <- as.POSIXct(df$timestamp, format = "%Y-%m-%d %H:%M:%S"

# sort the data by machine and timestamp
df<- df %>%
  arrange(machine, timestamp)
```

```r
# create a new column 'last_vend' containing the timestamp of the pre
df <- df %>%
  group_by(machine) %>%
  mutate(last_vend = lag(timestamp)) %>%
  ungroup()
```

Question 3

Create a new column ‘deltahours’ containing the hours that passed since the last sale for each sale

```r
# we changd the unit format of deltahours to numeric values to avoid
df$deltahours <- as.numeric(difftime(df$timestamp, df$last_vend, unit
```

Question 4

Create an auxiliary data table called “machine_daily_average” with the average daily sales per machine. Use this auxiliary table to attach to every row of the transactional data table the average daily sales per machine.

```r
# for initial intuition: look for daily sales for machine
daily_sales <- df %>%
  group_by(machine, date) %>%
  summarise(total_sales_per_day = length(column))
```

```
`summarise()` has grouped output by 'machine'. You can override using
the
`.groups` argument.
```

```r
# group the daily sales by machine, and calculate the average daily s
machine_daily_average <- df %>%
  group_by(machine) %>%
  summarise(avg_daily_sales = n() / n_distinct(as.Date(date)))

# merge the machine_daily_average table with the transactions table t
df <- merge(df, machine_daily_average, by = "machine", all.x = TRUE)
```

Question 5

Create a new variable called “delta” in the transactional data table containing a normalized version of deltahours.

```r
# calculate the delta
df <- df %>%
  mutate(delta = as.numeric(deltahours / (24 / avg_daily_sales)))
```

Creating the model

Question 6

Select 30% of the machines in the transactional data for testing and 70% of the machines for training and train a linear logistic regression model called “m” to predict whether a machine has a failure as a function of variable delta.

The intercept of the function is -6.90 and the delta is 0.56. When reversing the log of the intercept we can see that when the delta is 0, the probability of failure is 0.1%.

```r
# select 70% of machines for training
set.seed(2022)
train_machines <- sample(unique(df$machine), round(0.7 * length(uniqu

# split the data into training and testing sets
train <- df[df$machine %in% train_machines, ]
test <- df[!df$machine %in% train_machines, ]

# train the logistic regression model
m <- glm(failure ~ delta, data = train, family = binomial)
```

```
Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
# print coefficients and intercept
delta <- m$coefficients["delta"]
intercept <- m$coefficients["(Intercept)"]

print(paste('Intercept:', intercept, 'Delta:', delta))
```

```
[1] "Intercept: -6.90465900782892 Delta: 0.561031107047334"
```

```r
# for interpretation we decided to reverse log on the intercept
logit2prob <- function(logit){
```

```
  odds <- exp(logit)
  prob <- odds / (1 + odds)
  return(prob)
}

# compute probability of failure when delta = 0
prob_delta_0 <- logit2prob(intercept)

# print updated message
cat('The probability of failure when delta equals 0 is', prob_delta_0
```

```
The probability of failure when delta equals 0 is 0.001002096
```

6a: What is the ROC AUC for the train and test set?

The ROC AUC for the Test set is 0.922 and for the Training set it is 0.919. These are very close values which indicates that there are no signs of overfitting.

```
# Obtain predicted probabilities for train set
train$pred_prob <- predict(m, train, type = "response")

# Calculate AUC on train set
train_auc <- roc(train$failure, train$pred_prob)
```

```
Setting levels: control = 0, case = 1

Setting direction: controls < cases
```

```
auc(train_auc)
```

```
Area under the curve: 0.9191
```

```
# Obtain predicted probabilities for test set
test$pred_prob <- predict(m, test, type = "response")

# Calculate AUC on test set
test_auc <- roc(test$failure, test$pred_prob)
```

```
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```

```
auc(test_auc)
```

```
Area under the curve: 0.9222
```

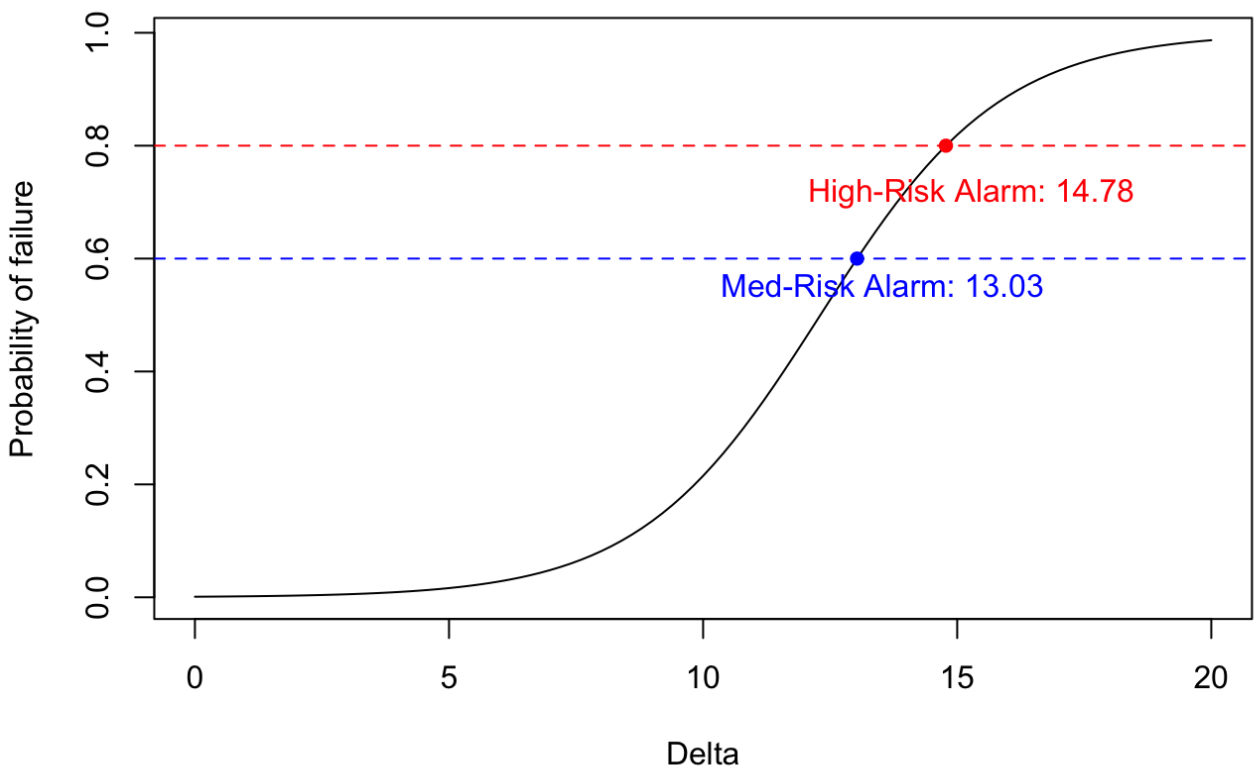
6b: Plot the function of probability of failure with respect to delta to gain intuition (includes 6c i.)

The threshold delta for Medium Risk alarms is 13.03 and the delta for High Risk Alarms is 14.78

```
# 6b: Base Graph
# Define a function to calculate the predicted probability of failure
prob_func <- function(delta_val) {
  prob_val <- 1 / (1 + exp(-(coef(m)[1] + coef(m)[2] * delta_val)))
  return(prob_val)
}

# Calculate the predicted probabilities for the sequence of delta val
delta_seq <- seq(0, 5, by = 0.1)
prob_seq <- sapply(delta_seq, prob_func)

# Plot the curve of predicted probabilities against delta values
curve(prob_func(x), from = 0, to = 20, xlab = "Delta", ylab = "Probab

##############
#6ci
# Define the probability functions for the med-risk and high-risk ala
med_risk_func <- function(delta_val) {
  return(prob_func(delta_val) - 0.6)
}

high_risk_func <- function(delta_val) {
  return(prob_func(delta_val) - 0.8)
}

# Find the roots of the probability functions to determine the delta
med_risk_delta <- uniroot(med_risk_func, c(0, 20))$root
high_risk_delta <- uniroot(high_risk_func, c(0, 20))$root
paste('Medium Risk Threshold:', med_risk_delta, 'High Risk Threshold:
```

```
[1] "Medium Risk Threshold: 13.0297998986025 High Risk Threshold:
14.7780637496776"
```

```r
## Add values to our plot
# Add horizontal lines at 60% and 80% probability levels
abline(h = 0.6, col = "blue", lty = 2)
abline(h = 0.8, col = "red", lty = 2)

# Add points for med-risk and high-risk thresholds
points(med_risk_delta, 0.6, col = "blue", pch = 16)
points(high_risk_delta, 0.8, col = "red", pch = 16)

# Add the delta value as text next to the points
text(med_risk_delta + 0.5, 0.5, paste0("Med-Risk Alarm: ", round(med_
text(high_risk_delta + 0.5, 0.5, paste0("High-Risk Alarm: ", round(hi
```



6cii: Alarms per day

When only accounting for the medium risk alarms, we will face a daily average of 41 alarms (Total of 3694 over the complete period). When only accounting for the high risk alarms, we will face a daily average of 28 alarms (Total of 2484 over the complete period).

```r
# Calculate the average number of med-risk alarms fired per day
med_risk_alarms <- df %>%
  filter(delta > med_risk_delta) %>%
  summarize(Total_med_risk_alarms = n(), med_risk_daily_alarm = Total

# Calculate the average number of high-risk alarms fired per day
high_risk_alarms <- df %>%
  filter(delta > high_risk_delta) %>%
  summarize(Total_high_risk_alarms = n(), high_risk_daily_alarm = Tot


# Combine the med-risk and high-risk alarms into a single data frame
alarm_summary <- data.frame(
  Alarm_Type = c("Med-Risk", "High-Risk"),
  Total_Alarms = c(med_risk_alarms$Total_med_risk_alarms, high_risk_a
  Avg_Alarms_Per_Day = c(med_risk_alarms$med_risk_daily_alarm, high_r
)
alarm_summary
```

```
  Alarm_Type Total_Alarms Avg_Alarms_Per_Day
1   Med-Risk         3694           41.04444
2  High-Risk         2484           27.60000
```

6ciii: Proportion of false alarms

For Medium-risk alarms we face a FPR of 20.14% and for High-Risk only alarms we face a False-Positive Rate of 8.85%.

```r
# proportion of false med_alarms
proportion_false_med_alarms <- df %>%
  filter(delta > med_risk_delta) %>%
  summarize(Total_med_false_alarms = sum(failure == 0), False_med_dai

# proportion of false high_alarms
proportion_false_high_alarms <- df %>%
  filter(delta > high_risk_delta) %>%
  summarize(Total_high_false_alarms = sum(failure == 0), False_high_d


# Combine the med-risk and high-risk alarms into a single data frame
alarm_summary_updated <- data.frame(
  Alarm_Type = c("Med-Risk", "High-Risk"),
```

```
  Total_Alarms = c(med_risk_alarms$Total_med_risk_alarms, high_risk_a
  Total_False_Alarms = c(proportion_false_med_alarms$Total_med_false_
  Avg_Alarms_Per_Day = c(med_risk_alarms$med_risk_daily_alarm, high_r
  Avg_False_Alarms_per_Day = c(proportion_false_med_alarms$False_med_
  prop_false_alarms = c(proportion_false_med_alarms$prop_false_med_al

)

alarm_summary_updated
```

```
  Alarm_Type Total_Alarms Total_False_Alarms Avg_Alarms_Per_Day
1   Med-Risk         3694                744           41.04444
2  High-Risk         2484                220           27.60000
  Avg_False_Alarms_per_Day prop_false_alarms
1                 8.266667        0.20140769
2                 2.444444        0.08856683
```

6d: Calculate the annual profit of both EWS systems

If we set use the new medium-risk-EWS the annual profit we will be increased by 2.43%. If we use the high-risk version, our profit will increase by 2.6%. Thus we recommend implementing the high-risk EWS.

1. Select transactions that would trigger the alarm

```
# Medium risk
df_med_alarm <- df %>%
  filter(delta > med_risk_delta)

# High risk
df_high_alarm <- df %>%
  filter(delta > high_risk_delta)
```

2. Calculate threshold hours according to the alarm threshold

```
# Medium risk
df_med_alarm <- df_med_alarm %>%
  group_by(machine, date) %>%
  mutate(daily_sales = length(column), threshold_hours = med_risk_del

# High risk
df_high_alarm <- df_high_alarm %>%
```

```
  group_by(machine, date) %>%
  mutate(daily_sales = length(column), threshold_hours = med_risk_del
```

3. Adjust threshold hours and delta for the repair time

```
# Medium risk
df_med_alarm <- df_med_alarm %>%
  mutate(threshold_hours_fixed = threshold_hours + 1.5, delta_fixed=

# High risk
df_high_alarm <- df_high_alarm %>%
  mutate(threshold_hours_fixed = threshold_hours + 1.5, delta_fixed=
```

4. Calculate won sales

```
# Medium risk
df_med_alarm <- df_med_alarm %>%
  mutate(won_sales = failure * (delta - delta_fixed))

# High risk
df_high_alarm <- df_high_alarm %>%
  mutate(won_sales = failure * (delta - delta_fixed))
```

5. Calculate additional revenues as a sum of won sales times the average margin

```
# Medium risk
additional_revenues_med_risk <- sum(df_med_alarm$won_sales, na.rm = T

# High risk
additional_revenues_high_risk <- sum(df_high_alarm$won_sales, na.rm =
```

6. Calculate cost of the system as the cost of sending an operator to check times the number of “false alarms”.

```
# find out time span of data to adjust the current FPR
earliest_date <- min(df$date)
latest_date <- max(df$date)

time_span <- as.numeric(difftime(latest_date, earliest_date, units =
time_factor <- 12/time_span
```

```r
# current costs
old_costs <- nrow(unique(df[, "machine", drop = FALSE])) * (2.2/time_

# Medium risk
cost_of_system_med_risk <- proportion_false_med_alarms$Total_med_fals
additional_costs_med_risk <- cost_of_system_med_risk - old_costs

#h High risk
cost_of_system_high_risk <- proportion_false_high_alarms$Total_high_f
additional_costs_high_risk <- cost_of_system_high_risk - old_costs
```

7. Calculate profit increase as a % of the total profit (profit increase / (margin_per_item * number_of_transactions))

```r
#current profit
old_profit <- 1.7 * nrow(df)

# Medium risk
additional_profit_med_risk <- additional_revenues_med_risk - addition
profit_increase_med_risk <- additional_profit_med_risk/ old_profit

# High risk
additional_profit_high_risk <- additional_revenues_high_risk - additi
profit_increase_high_risk <- additional_profit_high_risk/ old_profit

cbind(profit_increase_med_risk , profit_increase_high_risk )
```

```
     profit_increase_med_risk profit_increase_high_risk
[1,]               0.02428239                0.02600039
```