GROUP ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT127-3-2-PFDA

PROGRAMMING FOR DATA ANALYSIS

APU2F2409CS(CYB)

DATE ASSIGNED: 30 SEPTEMBER 2024

DATE COMPLETED: 9 NOVEMBER 2024

Group 7

No.	Student Name	TP Number
1.	Chloe Tan Jia Xin	TP070759
2.	Goh Yuan Kee	TP070126
3.	Ng Zhe Shen	TP071625
4.	Valen Christino	TP072897

Table of Contents

1.0 Introduction

In this assignment, we are given a dataset with 6000 rows (excluding header) and 22 columns called “credit_risk”. This dataset contains credit-related information and characteristics of bank’s customers, along with their credit score of either “good” or “bad”.

As a group of data analysts, our goal is to create a prediction model that has high accuracy in predicting whether a customer has the tendency to have “good” credit score. To achieve this, we need to first identify the most significant factors in achieving a “good” credit. However, with so many raw data in the dataset, it is impractical to evaluate each instance of data one by one to find a common pattern. Instead, we will use RStudio as our data analysis tool to analyze the dataset.

So, to begin the analysis, we picked four columns from the dataset as factors in the prediction model to form our initial hypothesis. These columns are picked purely based on joint logical deduction and assumption of every group member. Our hypothesis states that customers with **credit amount that are lesser than RM4000, status of employment being longer than 7 years, no known savings, and critical or existing credit history will indicate a “good” credit score**.

Using a variety of techniques such as data exploration, manipulation, transformation, and visualization, we will observe and evaluate the outcome of each column. During the procedure, we expect that certain columns may not have that great of an impact in determining the category of customers’ credit score. When instances like this occur, we will investigate other columns to find a suitable substitution. This process will be repeated until our prediction model achieves at least 70% accuracy. The accuracy is calculated using the formula: (**Number of selected rows / 3000) x 100%**. With a high accuracy credit score prediction model, we will be able to provide insightful recommendations to the bank’s stakeholders.

2.0 Data Preparation

2.1 Data Import

The following screenshot shows the code to import the dataset into R Studio. The dataset is held by the variable called “dataset”.

```r
dataset = read.csv("C:\\Users\\zhesh\\OneDrive\\Desktop\\PFDA Assignment\\credit_risk.csv", header=TRUE)
dataset
```

This is a portion of the output when trying to show the content of the dataset variable in console.

```
> dataset
      X checking_status  duration credit_history    purpose credit_amount savings_status employment
1     1              <0 12.000000       all paid retraining      339.0000           <100        >=7
2     4              <0 12.000000       all paid retraining      339.0000           <100        >=7
3     9              <0 12.000000       all paid retraining      339.0000           <100        >=7
4    10            >=200 12.000000      all paid   radio/tv      409.0000          >=1000    1<=X<4
5    11            >=200 12.000000      all paid   radio/tv      409.0000          >=1000    1<=X<4
6    13            >=200 12.000000      all paid   radio/tv      409.0000          >=1000    1<=X<4
7    15            >=200 12.000000      all paid   radio/tv      409.0000          >=1000    1<=X<4
8    18            >=200 12.000000      all paid   radio/tv      409.0000          >=1000    1<=X<4
9    29            >=200 12.000000      all paid   radio/tv      409.0000          >=1000    1<=X<4
10   35        0<=X<200  7.461352       all paid  education      433.0000           <100         <1
```

2.2 Data Cleaning

2.2.1 Converting Duration (Year) column to hold only integer value

```
#Converting Duration(Year) column to hold only integer value
dataset$`Duration(Year)`=as.integer(dataset$`Duration(Year)`)
# convert to integer "Installment_Commitment_Count" (assignment_file[9])
dataset$Installment_Commitment_Count = as.integer(dataset$Installment_Commitment_Count)
# convert to integer "Residence_Since"
dataset$Residence_Since = as.integer(dataset$Residence_Since)
# convert to integer "Age"
dataset$Age = as.integer(dataset$Age)
#Existing_Credits = convert to integer
dataset$Existing_Credits = as.integer(dataset$Existing_Credits)
# Num_Dependents = convert to integer
dataset$Num_Dependents = as.integer(dataset$Num_Dependents)
```

Certain columns, like Duration(Year), Age, Existing_Credits, and so on were converted to integer type by using as. Integer (). This change allows us to perform mathematical operations on these values during analysis and make it look clear.

2.2.2 Rounding the Credit_Amount(RM) column to 2 decimal places

```
#Rounding the Credit_Amount(RM) column to 2 decimal places
dataset$`Credit_Amount(RM)`=round(dataset$`Credit_Amount(RM)`,2)
dataset$`Credit_Amount(RM)`
```

Credit_Amount(RM) column was rounded to two decimal places to maintain consistency and make the values easier to interpret by using round().

```
R 4.4.1 · ~/
[457]  3905  3386   343  4594  3620  1721  3017   754  1950  2924  1659  7238  2764  4679  3092   448   654  1238  1245  3114  2569  5152  1037  1478
[481]  3573  1201  3622   960  1163  1209  3077  3757  1418  3518  1934  8318  1237   368  2122  2996  9034  1585  1301  1323  3123  5493  1126  1216
[505]  1207  1309  2360  6850  1413  8588   759  4686  2687   585  2255   609  1361  7127  1203   700  5507  3190  7119  3488  1113  7966  1532  1503
[529]  2302   662  2273  2631  1503  1311  3105  2319  1374  3612  7763  3049  1534  2032  6350  2864  1255  1333  2022  1552   626  8858   996  1750
[553]  6999  1995  1199  1331  2278  5003  3552  1928  2964  1546   683 12389  4712  1553  1372  2578  3979  6758  3234  5954  5433   806  1082  2788
[577]  2930  1927  2820   937  1056  3124  1388  2384  2133  2039  2799  1289  1217  2246   385  1965  1572  2718  1358   931  1442  4241  2775  3863
[601]  2329   918  1837  3349  1275  2828  4526  2671  2051  1300   741  1240  3357  3632  1808 12204  9157  3676  3441   640  3652  1530  3914  1858
[625]  2600  1979  2116  1437  4042  3832  3660  1553  1444  1980  1355  1393  1376 15653  1493  4370   750  1308  4623  1851  1880  7980  4583  1386
[649]   947   684  7476  1922  2303  8086  2346  3973   888 10222  4221  6361  1297   900  2241  1050  1047  6314  3496  3609  4843  3017  4139  5742
[673] 10366  2080  2580  4530  5150  5595  2384  1453  1538  2279  1478  5103  9857  6527  1347  2862  2753  3651   975  2631  2896  4716  2284  1236
[697]  1103   926  1800  1905  1123  6331  1377  2503  2528  5324  6560  2969  1206  2118   629  1198  2476  1138 14027  7596  3077  1505  3148  6148
[721]  1337   433  1228   790  2570   250  1316  1882  6416  1275  6403  1987   760  2603  3380  3990 11560  4380  6761  4280  2325  1048  3160  2483
[745] 14179  1797  2511  1274  5248  3029   428   976   841  5771  1555  1285  1299  1271  1393   691  5045  2124  2214 12680  2463  1155  3108  2901
[769]  3617  1655  2812  8065  3275  2223  1480  1371  3535  3509  5711  3872  4933  1940  1410   836  6468  1941  2675  2751  6224  5998  1188  6313
[793]  1221  2892  3062  2301  7511  1258   717  1549  1597  1795  4272   976  7472  9271   590   930  9283  1778   907   484  9629  3051  3931  7432
[817]  1338  1554 15857  1345  1101  3016  2712   731  3780  1602  3966  4165  8335  6681  2375  1216 11816  5084  2327  1082   886   601  2957  2611
[841]  5179  2993  1943  1559  3422  3976  6761  1249  1364   709  2235  4042  1471  1442 10875  1474   894  3343  3959  3577  5804  2169  2439  4526
[865]  2210  2221  2389  3331  7409   652  7678  1343  1382   874  3590  1322  1940  3595  1422  6742  7814  9277  2181  1098  4057   795  2825 15672
[889]  6614  7824  2442  1829  2171  5800  1169  8947  2606  1592  2186  4153  2625  3485 10477  1386  1278  1107  3763  3711  3594  3195  4454  4736
[913]  2991  2142  3161 18424  2848 14896  2359  3345  1817 12749  1366  2002  6872   697  1049 10297  1867  1344  1747  1670  1224   522  1498  1919
[937]   745  2063  6288  6842  3527  1546   929  1455  1845  8358  3349  2859  1533  3621  3590  2145  4113 10974  1893  1231  3656  1154  4006  3069
[961]  1740  2353  3556  2397   454  1715  2520  3568  7166  3939  1514  7393  1193  7297  2831  1258   753  2427  2538  1264  8386  4844  2923  8229
[985]  2028  1433  6289  1409  6579  1743  3565  1569  1936  3959  2390  1736  3857   804  1845  4576
 [ reached getOption("max.print") -- omitted 5000 entries ]
```

This is the data frame that is shown after executing the rounding Credit_Amount(RM) column to 2 decimal places.

2.2.3 Fix the typo

```
# fix the typo in Credit_History column ("no credits/all paid" -> "no credits")
dataset[dataset=="no credits/all paid"] = "no credits"
# fix the typo in the dataset ( "500<=X<10000" -> "500<=X<1000") to make it make sense :D
dataset[dataset=="500<=X<10000"] = "500<=X<1000"
```

Some columns had inconsistent values due to typos. For example, "no credits/all paid" was changed to "no credits", and "500<=X<10000" was corrected to "500<=X<1000". These changes ensure consistency across the dataset.

2.2.4 Function to get mode & Replace both NA values and blank spaces with the mode

```
# Function to get the mode
get_mode <- function(v) {
  uniq_v <- unique(v[!(is.na(v) | v == "")])  # Remove NA and blank spaces
  uniq_v[which.max(tabulate(match(v, uniq_v)))]  # Find the most frequent value
}
# Replace both NA values and blank spaces with the mode
dataset$Other_Payment_Plans[is.na(dataset$Other_Payment_Plans) | dataset$Other_Payment_Plans == ""] = get_mode(dataset$Other_Payment_Plans)
```

A custom function get_mode() was created to find the most frequent value (mode) in a column. This mode was used to replace missing or blank values in the Other_Payment_Plans column. Replacing missing values with the mode helps retain data consistency without removing any entries.

2.3 Data Pre-processing

Data Pre-processing is crucial to any dataset as it ensures that the dataset that we are working with is accurate before being analysed. This ensures that the analysis done on the data is of the highest quality as well as preventing unnecessary errors.

Some of the functions that we used to examine and prepare our dataset are functions built into the RStudio such as: summary(), factor(), and str()

- summary()

  The summary() function is a very useful tool for data pre-processing as it is used to generate a concise summary of the data that is referenced.

- factor()

  This function is used to sorts the dataset allowing for a more thorough understanding of the data. However, this function only works on a specific column of the dataset and cannot be used on the entire dataset.

- str()

  When used, this function provides a short summary of the dataset including: column name, datatype of each column, and the first few values of aech column

```
> str(dataset)
'data.frame':	6000 obs. of  22 variables:
 $ No.                              : int  0 1 2 3 4 5 6 7 8 9 ...
 $ Checking_Status                  : chr  "<0" "0<=X<200" "no checking" "<0" ...
 $ Duration(Year)                   : int  6 48 12 42 24 36 24 36 12 30 ...
 $ Credit_History                   : chr  "critical/order existing credit" "existing paid" "critical/order exist
ing credit" "existing paid" ...
 $ Purpose                          : chr  "radio/tv" "radio/tv" "education" "furniture/equipment" ...
 $ Credit_Amount(RM)                : num  1169 5951 2096 7882 4870 ...
 $ The_Statuses_of_Savings          : chr  "no known savings" "<100" "<100" "<100" ...
 $ Status_of_Employment(year_range): chr  ">=7" "1<=X<4" "4<=X<7" "4<=X<7" ...
 $ Installment_Commitment_Count     : int  4 2 2 2 3 2 3 2 2 4 ...
 $ Confidential_Personal_Status     : chr  "male single" "female div/dep/mar" "male single" "male single" ...
 $ Other_Party(type)                : chr  "none" "none" "none" "guarantor" ...
 $ Residence_Since                  : int  4 2 3 4 4 4 4 2 4 2 ...
 $ Property_Magnitude               : chr  "real estate" "real estate" "real estate" "life insurance" ...
 $ Age                              : int  67 22 49 45 53 35 53 35 61 28 ...
 $ Other_Payment_Plans              : chr  "stores" "stores" "stores" "stores" ...
 $ Housing                          : chr  "own" "own" "own" "for free" ...
 $ Existing_Credits                 : int  2 1 1 1 2 1 1 1 1 2 ...
 $ Job                              : chr  "skilled" "skilled" "unskilled resident" "skilled" ...
 $ Num_Dependents                   : int  1 1 2 2 2 2 1 1 1 1 ...
 $ Own_Telephone                    : chr  "yes" "none" "none" "none" ...
 $ Foreign_Worker                   : chr  "yes" "yes" "yes" "yes" ...
 $ Class                            : chr  "good" "bad" "good" "good" ...
```

Structure of data after data cleaning

```
factor(dataset$Credit_History)

72:31  (Untitled) ÷

Console  Terminal ×  Background Jobs ×

R 4.4.1 · ~/
[987] no credits                     existing paid
[989] existing paid                  critical/order existing credit
[991] critical/order existing credit all paid
[993] existing paid                  existing paid
[995] existing paid                  existing paid
[997] existing paid                  existing paid
[999] existing paid                  critical/order existing credit
 [ reached getOption("max.print") -- omitted 5000 entries ]
5 Levels: all paid critical/order existing credit ... no credits
```

Factor of the “Credit_History” column of the data called “dataset” after data cleaning

```
> summary(dataset)
      No.         Checking_Status     Duration(Year) Credit_History    
 Min.   :   0   Length:6000        Min.   : 4.0   Length:6000       
 1st Qu.:1500   Class :character   1st Qu.:12.0   Class :character  
 Median :3000   Mode  :character   Median :19.0   Mode  :character  
 Mean   :3000                      Mean   :21.9                     
 3rd Qu.:4499                      3rd Qu.:27.0                     
 Max.   :5999                      Max.   :72.0                     
   Purpose          Credit_Amount(RM) The_Statuses_of_Savings
 Length:6000        Min.   :  250     Length:6000            
 Class :character   1st Qu.: 1332     Class :character       
 Mode  :character   Median : 2290     Mode  :character       
                    Mean   : 3344                            
                    3rd Qu.: 4164                            
                    Max.   :18424                            
 Status_of_Employment(year_range) Installment_Commitment_Count
 Length:6000                      Min.   :1.000               
 Class :character                 1st Qu.:2.000               
 Mode  :character                 Median :3.000               
                                  Mean   :2.975               
                                  3rd Qu.:4.000               
                                  Max.   :4.000               
```

Summary of dataset after data cleaning

2.4 Data Exploration

Data Exploration is the explores the basic information of the dataset. This gives us a more thorough understanding of our dataset. Data Exploration can be done with many functions such as head(), tail(), names(), ncol(), nrow(), etc.

head() - prints the first few rows of the dataset

tail() - prints the last few rows of the dataset

ncol() - shows us the number of columns in the dataset

nrow() - shows us the number of rows in the dataset

names() - provides us with a complete list of column names in the dataset

3.0 Data Analysis

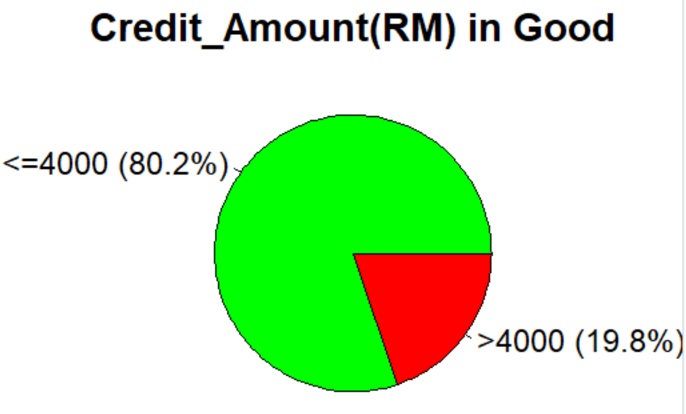
3.1 The Impact of Saving Status on Good Credit Score

Name: Ng Zhe Shen (TP071625)

3.1.1 How many Credit Amount <= 4000 is under "good" Class

```
#Filter data for Class == "good"
good = dataset[dataset$Class == "good", ]

#Calculate the number of rows for Credit_Amount(RM) <=4000 and >4000
count = c(sum(good$`Credit_Amount(RM)`<=4000), sum(good$`Credit_Amount(RM)`>4000))

#Use tibble to visualize the Credit_Amount(RM) column in a table form
install.packages("tibble")
library(tibble)
tibble(`Credit Amount(RM)` = c("<=4000",">4000"),
       Count = count)

#Calculating the percentage
percentage = round(count/sum(count)*100, 1)
library(ggplot2)
#Pie chart for Credit_Amount(RM)
pie(count, label=paste(c("<=4000",">4000")," (",percentage,"%",")",sep=""),
    radius=1,main="Credit_Amount(RM) in Good",col=c("green","red"))
```

```
> #Filter data for Class == "good"
> good = dataset[dataset$Class == "good", ]
> #Calculate the number of rows for Credit_Amount(RM) <=4000 and >4000
> count = c(sum(good$`Credit_Amount(RM)`<=4000), sum(good$`Credit_Amount(RM)`>4000))
> tibble(`Credit Amount(RM)` = c("<=4000",">4000"),
+        Count = count)
# A tibble: 2 × 2
  `Credit Amount(RM)` Count
  <chr>               <int>
1 <=4000               2406
2 >4000                 594
> 
```

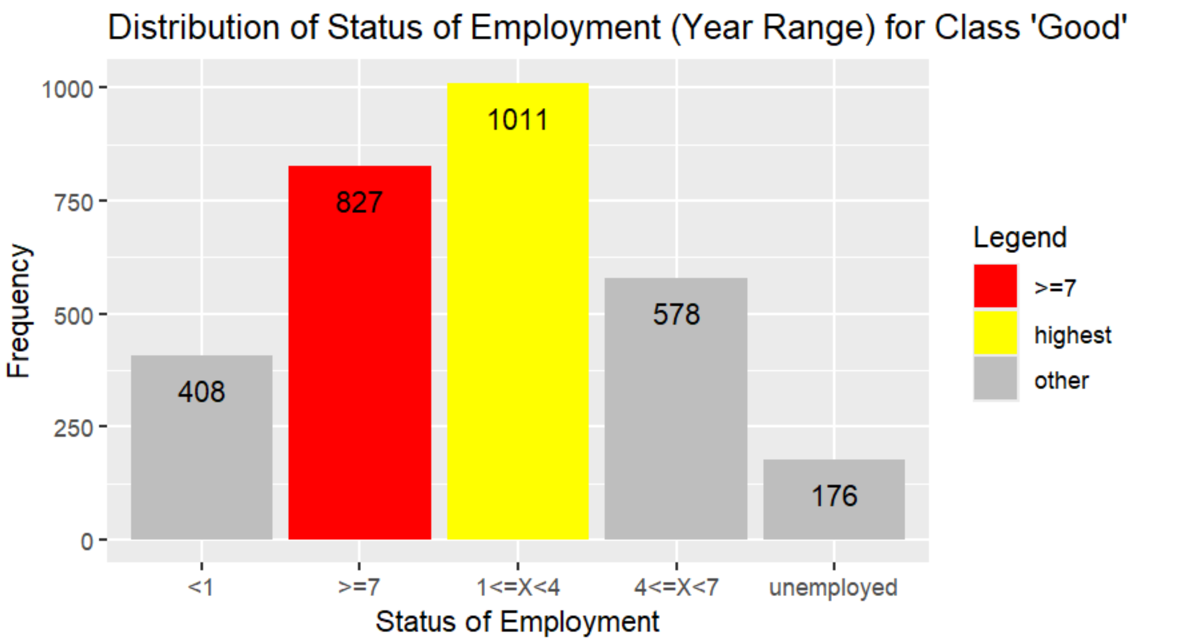


Based on the output, when Class == "good", there are 2406 people (80.2%) with credit amount of at most RM4000. This shows that credit amount has significant association with Class. Thus, this column aligns with the initial hypothesis.

3.1.2 How many Status of Employment >= 7 is under “good” Class

```
#Determine the highest frequency bar
highest_status <- names(table(
  good$`Status_of_Employment(year_range)`))[which.max(table(good$`Status_of_Employment(year_range)`))]
#Plot chart for Status_of_Employment(year_range)
ggplot(good, aes(x = `Status_of_Employment(year_range)`)) +
  geom_bar(aes(fill = ifelse(`Status_of_Employment(year_range)` == ">=7", ">=7",
                      ifelse(`Status_of_Employment(year_range)` == highest_status, "highest", "other")))) +
  scale_fill_manual(values = c(">=7" = "red", "highest" = "yellow", "other" = "grey")) +
  geom_text(stat = "count", aes(label = after_stat(count)), vjust = 2) +
  labs(title = "Distribution of Status of Employment (Year Range) for Class 'Good'",
       fill = "Legend",
       x = "Status of Employment",
       y = "Frequency")
```

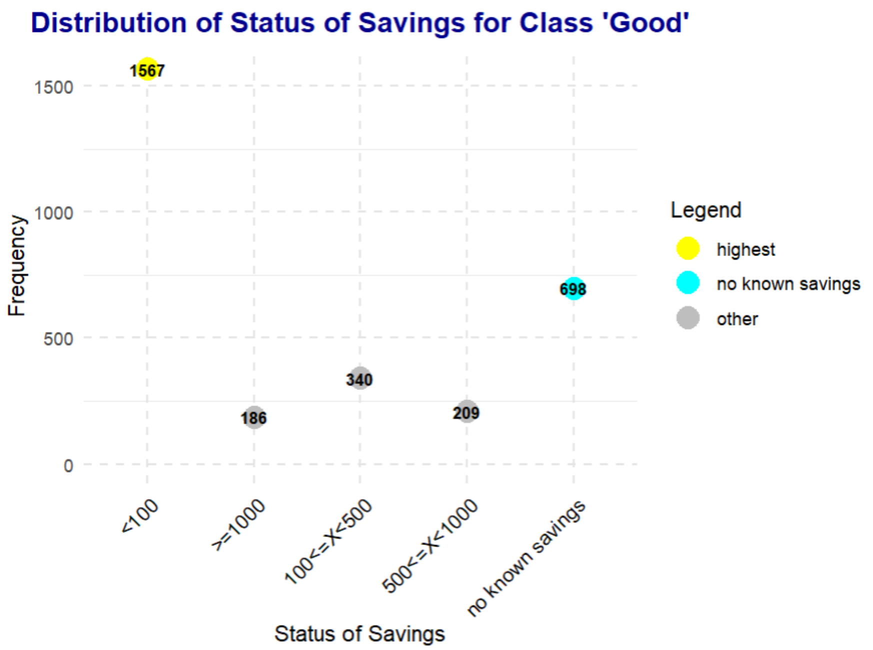
```
> #Determine the highest frequency bar
> highest_status <- names(table(
+   good$`Status_of_Employment(year_range)`))[which.max(table(good$`Status_of_Employment(year_range)`))]
> #Plot chart for Status_of_Employment(year_range)
> ggplot(good, aes(x = `Status_of_Employment(year_range)`)) +
+   geom_bar(aes(fill = ifelse(`Status_of_Employment(year_range)` == ">=7", ">=7",
+                       ifelse(`Status_of_Employment(year_range)` == highest_status, "highest", "other")))) +
+   scale_fill_manual(values = c(">=7" = "red", "highest" = "yellow", "other" = "grey")) +
+   geom_text(stat = "count", aes(label = after_stat(count)), vjust = 2) +
+   labs(title = "Distribution of Status of Employment (Year Range) for Class 'Good'",
+        fill = "Legend",
+        x = "Status of Employment",
+        y = "Frequency")
> 
```



Based on the output, although the chosen condition in the initial hypothesis, Status of Employment >= 7, has contributed a relatively high amount of data (827 out of 3000) towards the number of “good” Class, it is evidently not the condition with the highest amount of data, which is Status of Employment between 1 to 4 years (1011 out of 3000). So, Status of Employment 1<=X<=4 will be the better prediction factor to be considered.

3.1.3 How many Status of Savings == “no known savings” is under “good” Class

```
#Determine the highest value
highest_status = names(table(good$The_Statuses_of_Savings))[which.max(table(good$The_Statuses_of_Savings))]
# Create a dot plot
ggplot(good, aes(x = The_Statuses_of_Savings)) +
  stat_count(geom = "point", aes(y = after_stat(count),
  color = ifelse(The_Statuses_of_Savings == "no known savings", "no known savings",
          ifelse(The_Statuses_of_Savings == highest_status, "highest", "other"))),
  size = 5) +
  scale_color_manual(values = c("no known savings"="cyan", "highest"="yellow", "other"="grey")) +
  # Add count text labels above each dot
  stat_count(geom = "text", aes(label = after_stat(count), y = after_stat(count) + 0.5),
             color = "black", size = 3, fontface = "bold") +
  # Additional settings to expand the view
  expand_limits(y=0) +
  labs(title = "Distribution of Status of Savings for Class 'Good'",
       color = "Legend",
       x = "Status of Savings",
       y = "Frequency") +
  # Use theme_minimal and customize plot theme with theme()
  theme_minimal() +
  theme(
    panel.grid.major = element_line(color = "grey90", linetype = "dashed"),
    plot.title = element_text(face = "bold", size = 14, hjust = 0.5, color = "darkblue"),
    axis.text.x = element_text(angle = 45, hjust = 1, color = "black", size = 10)
  )
```

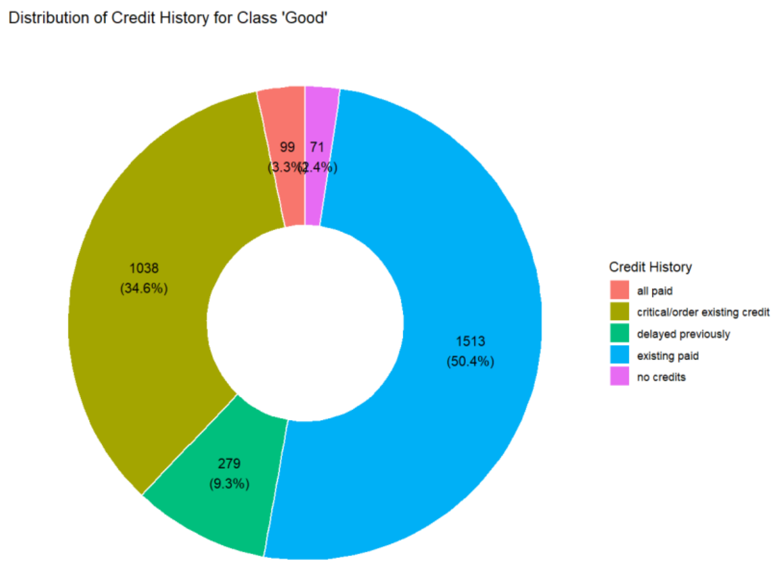


Looking at the graph, it is apparent that the initial hypothesis is flawed because it stated that Status of Savings is no known savings will result in high probability of “good” Class. In truth, it is Status of Savings < 100 that has the highest association with “good” Class (1567 compared to 698). Thus, the initial condition Status of Savings == “no known savings” will be replaced by Status of Savings < 100.

3.1.4 How many Credit History == “critical/order existing credit” in “good” Class

```r
# Create a frequency table
credit_history_counts = as.data.frame(table(good$Credit_History))
colnames(credit_history_counts) = c("Credit_History", "Count")
credit_history_counts$Percentage = round(credit_history_counts$Count /
                                          sum(credit_history_counts$Count)*100,1)
credit_history_counts
# Donut chart
ggplot(credit_history_counts, aes(x = 2, y = Count, fill = Credit_History)) +
  geom_col(width = 1, color = "white") +
  geom_text(aes(label=paste(Count,"\n(",Percentage,"%)",sep="")),position=position_stack(vjust=0.5)) +
  coord_polar(theta = "y") +
  xlim(0.8, 2.5) +
  labs(title = "Distribution of Credit History for Class 'Good'",
       fill = "Credit History") +
  theme_void() +
  theme(legend.position = "right")
```

```
> # Create a frequency table
> credit_history_counts = as.data.frame(table(good$Credit_History))
> colnames(credit_history_counts) = c("Credit_History", "Count")
> credit_history_counts$Percentage = round(credit_history_counts$Count /
+                                           sum(credit_history_counts$Count)*100,1)
> credit_history_counts
                 Credit_History Count Percentage
1                      all paid    99        3.3
2 critical/order existing credit  1038       34.6
3            delayed previously   279        9.3
4                 existing paid  1513       50.4
5                    no credits    71        2.4
> # Donut chart
> ggplot(credit_history_counts, aes(x = 2, y = Count, fill = Credit_History)) +
+   geom_col(width = 1, color = "white") +
+   geom_text(aes(label=paste(Count,"\n(",Percentage,"%)",sep="")),position=position_stack(vjust=0.5)) +
+   coord_polar(theta = "y") +
+   xlim(0.8, 2.5) +
+   labs(title = "Distribution of Credit History for Class 'Good'",
+        fill = "Credit History") +
+   theme_void() +
+   theme(legend.position = "right")
> |
```



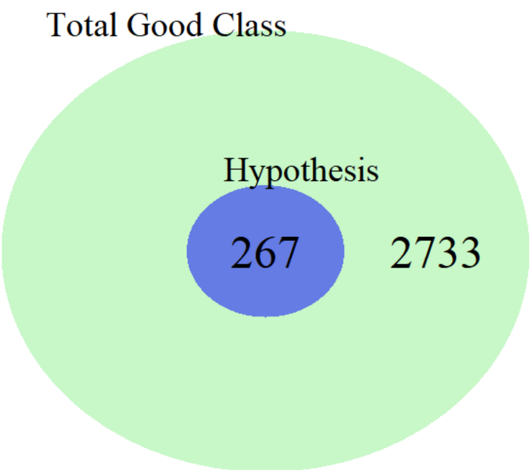
Based on the outcome, Credit History “critical/order existing credit” only represent 1038 instances of data out of the 3000 “good” Class. This category should not be considered into the prediction process as it only contributed 34.6% of the original data.

Though only 0.4% over half, Credit History is “existing paid” represents 1513 rows of data, making it the category with the highest amount of data. So, it will substitute the original factor in the hypothesis.

3.1.5 Outcome of Initial Hypothesis and Conclusion

```
#Visualization of Initial Hypothesis
install.packages("VennDiagram")
library(VennDiagram)
initial_hypothesis = nrow(dataset[(dataset$Class=="good") & (dataset$`Credit_Amount(RM)`<=4000) &
                                  (dataset$`Status_of_Employment(year_range)`=="1<=X<4") &
                                  (dataset$The_Statuses_of_Savings=="<100") &
                                  (dataset$Credit_History=="existing paid"), ])
#Venn Diagram - Initial Hypothesis vs Total Good Class
draw.pairwise.venn(
  area1 = 3000,         # Total "good" class
  area2 = initial_hypothesis,
  cross.area = initial_hypothesis,
  category = c("Total Good Class", "Hypothesis"),
  fill = c("lightgreen", "blue"),
  lty = "blank",
  cex = 2, cat.cex = 1.5, cat.pos = c(-20, 20), cat.dist = 0.02, scaled = TRUE
)
```

```
> library(VennDiagram)
Loading required package: grid
Loading required package: futile.logger
> initial_hypothesis = nrow(dataset[(dataset$Class=="good") & (dataset$`Credit_Amount(RM)`<=4000) &
+                                   (dataset$`Status_of_Employment(year_range)`=="1<=X<4") &
+                                   (dataset$The_Statuses_of_Savings=="<100") &
+                                   (dataset$Credit_History=="existing paid"), ])
> #Venn Diagram - Initial Hypothesis vs Total Good Class
> draw.pairwise.venn(
+   area1 = 3000,         # Total "good" class
+   area2 = initial_hypothesis,
+   cross.area = initial_hypothesis,
+   category = c("Total Good Class", "Hypothesis"),
+   fill = c("lightgreen", "blue"),
+   lty = "blank",
+   cex = 2, cat.cex = 1.5, cat.pos = c(-20, 20), cat.dist = 0.02, scaled = TRUE
+ )
(polygon[GRID.polygon.1], polygon[GRID.polygon.2], polygon[GRID.polygon.3], polygon[GRID.polygon.4], tex
t[GRID.text.5], text[GRID.text.6], text[GRID.text.7], text[GRID.text.8])
> 
```



After analyzing four columns that made up the initial hypothesis individually and making necessary adjustments to the chosen condition for each factor, it is time to visualize the outcome of the hypothesis when all four conditions are applied. Turns out, only 267 out of 3000 met all four conditions, which is an incredible low amount of data. The accuracy of this prediction model is merely 8.9%. So, improvements must be implemented, either by changing the condition of the columns or swapping out some of the columns with more suitable ones. More analysis will be carried out for other column to increase the prediction accuracy to at least 70%.

3.1.6 Additional features

1. **tibble()** – Modern version of tradition data frame. Used to present data in a table.
2. **paste()** – A function to concatenate multiple variables and Strings together. Used in pie chart to form the label.
3. **sep = “”** – Argument in **paste()** function to specify the separator between elements in paste() is nothing.
4. **table()** – A function to generate a table to store the frequency of each unique value appeared.
5. **which.max()** – This function returns the index of the maximum value in a vector. Combined with **table()**, it can return the value with the highest frequency of appearance.
6. **ifelse()** – Conditional function to categorize elements by whether a certain condition is met.
7. **labs()** – Modify title, legend title, x-axis and y-axis title.
8. **stat_count()** – Count the occurrence of each categorical value.
9. **geom = “point”** – Argument in **stat_count()**. Specify the graph is a point graph.
10. **geom = “text”** – Argument in **stat_count()**. Creates text label directly on top of each of the plot.
11. **after_stat()** – Display result calculated by stat_count().
12. **vjust** – Adjust the vertical alignment of the text
13. **font_face = “bold”** – Used in element_text() to make text bold.
14. **expand_limits(y=0)** – Ensure y-axis starts from 0.
15. **theme_minimal()** – Remove most grid lines for a cleaner look of the graph.
16. **theme** – Customize non-data elements e.g. grid lines, graph title, and axis text.
17. **panel.grid.major** – Refer to the grid lines.
18. **element_line()** – Modify lines in the graph.
19. **linetype = “dashed”** – Specify to replace solid lines with dashed lines.
20. **plot.title** – Refer to the plot title.
21. **element_text()** – Modify text elements in the graph.
22. **face = “bold”** – Argument in element_text to make the text bold.
23. **hjust** – Adjust the horizontal alignment of the text.
24. **axis.text.x** – Refer to the text on the x-axis.
25. **angle** – Rotate text to the specified angle.
26. **geom_col()** – Create bar chart using pre-determined value for height of the bars.

27. **coord_polar()** – Convert a chart from Cartesian coordinates to polar coordinates, which will turn a bar chart into a donut chart.
28. **theta = “y”** – Argument in coord_polar(), specify that bar heights are used to determine circular segment.
29. **xlim()** – Set the display limit of x-axis (width in donut chart).
30. **theme_void()** – Remove all background elements from the chart.
31. **legend.position = “right”** – Set the position of the legend to the right side of the chart.
32. **draw.pairwise.venn()** – Create Venn diagram with two sets.
33. **area1** – Argument in **draw.pairwise.venn()**. Define the first circle.
34. **area2** – Argument in **draw.pairwise.venn()**. Define the second circle.
35. **cross.area** – Argument in **draw.pairwise.venn()**. Define the area where the two circles overlap.
36. **category** – Argument in **draw.pairwise.venn()**. Label for each circle.
37. **lty** – Argument in **draw.pairwise.venn()**. Set the line type of the circles.
38. **cex** – Argument in **draw.pairwise.venn()**. Set font size for the value of the circles.
39. **cat.cex** – Argument in **draw.pairwise.venn()**. Set font size for the label of the circles.
40. **cat.pos** – Argument in **draw.pairwise.venn()**. Set position of the label for each circle.
41. **cat.dist** – Argument in **draw.pairwise.venn()**. Set the distance between the circle and label.
42. **scaled = TRUE** – Argument in **draw.pairwise.venn()**. When “TRUE”, the size of each circle is automatically set using their corresponding value.

3.2 The Impact of Employment on Good Credit Score

Name: Valen Christino (TP072897)

3.2.1 Does Credit Amount Have an Impact on Good Credit Score

```r
# graph 1 violin graph of credit amount on good credit score
summary_credit = dataset %>%
  group_by(Class) %>%
  summarise(
    min_age = min(`Credit_Amount(RM)`, na.rm = TRUE),
    q1_age = quantile(`Credit_Amount(RM)`, 0.25, na.rm = TRUE),
    median_age = median(`Credit_Amount(RM)`, na.rm = TRUE),
    q3_age = quantile(`Credit_Amount(RM)`, 0.75, na.rm = TRUE),
    max_age = max(`Credit_Amount(RM)`, na.rm = TRUE),
   .groups = 'drop'    )

ggplot(dataset, aes(x = Class, y = `Credit_Amount(RM)`)) +
  geom_violin(trim = TRUE, fill = "yellow") +
  geom_boxplot(width = 0.1, aes(fill = Class)) +
  geom_text(data = summary_credit, aes(x = Class, y = min_age, label = round(min_age, 1)),
            vjust = -0.5, color = "blue") +
  geom_text(data = summary_credit, aes(x = Class, y = q1_age, label = round(q1_age, 1)),
            vjust = -0.5, color = "blue") +
  geom_text(data = summary_credit, aes(x = Class, y = median_age, label = round(median_age, 1)),
            vjust = -0.5, color = "blue") +
  geom_text(data = summary_credit, aes(x = Class, y = q3_age, label = round(q3_age, 1)),
            vjust = -0.5, color = "blue") +
  geom_text(data = summary_credit, aes(x = Class, y = max_age, label = round(max_age, 1)),
            vjust = -0.5, color = "blue") +
  ggtitle("Credit Amount Violin graph")
```

```
> ggplot(dataset, aes(x = Class, y = `Credit_Amount(RM)`)) +
+   geom_violin(trim = TRUE, fill = "yellow") +
+   geom_boxplot(width = 0.1, aes(fill = Class)) +
+   geom_text(data = summary_credit, aes(x = Class, y = min_age, label = round(min_age, 1)),
+             vjust = -0.5, color = "blue") +
+   geom_text(data = summary_credit, aes(x = Class, y = q1_age, label = round(q1_age, 1)),
+             vjust = -0.5, color = "blue") +
+   geom_text(data = summary_credit, aes(x = Class, y = median_age, label = round(median_age, 1)),
+             vjust = -0.5, color = "blue") +
+   geom_text(data = summary_credit, aes(x = Class, y = q3_age, label = round(q3_age, 1)),
+             vjust = -0.5, color = "blue") +
+   geom_text(data = summary_credit, aes(x = Class, y = max_age, label = round(max_age, 1)),
+             vjust = -0.5, color = "blue") +
+   ggtitle("Credit Amount Violin graph")
```

Figure 3.2.1(1 & 2): code for violin graph & console output

From the diagram, it appears that Credit Amount doesn't have a significant impact on credit score although we do see that most of the good credit score tend to have a lower Credit Amount compared to the bad credit score, averaging RM200 lower as well as having the majority being less than 4000, so we take the majority of good credit score to be <= RM4000 and use it in the next data.

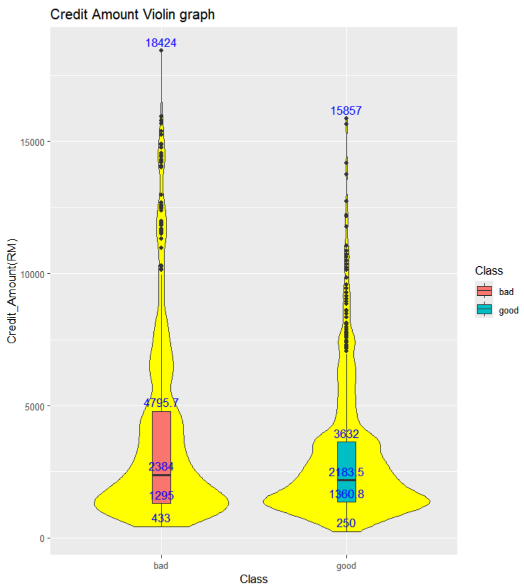


Figure 3.2.1(3): violin graph of Credit Amount

3.2.2 Does Credit Amount and Employment have an Impact to Credit Score

```r
# employment
# Filter the dataset for Class "good"
good_data = filter(dataset, Class == "good")

# Create a new variable 'Objective1' based on the objective
mutated_data = mutate(good_data, Credit_Category = ifelse(`Credit_Amount(RM)` <= 4000,
                                                          "not more than 4000",
                                                          "More than 4000"))

# summarize data
credit_employment = summarise(
  group_by(mutated_data, Credit_Category, `Status_of_Employment(year_range)`),
  Count = n(), .groups = 'drop')


ggplot(credit_employment, aes(x = `Status_of_Employment(year_range)`, y = Count, fill = Credit_Category)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("not more than 4000" = "green", "More than 4000" = "brown")) +
  labs(title = "Stacked Bar Chart of Class by Employment",
       x = "Credit History",
       y = "Count") +
  geom_text(data = credit_employment,
            aes(label = round(Count, 1)),
            position = position_stack(vjust = 0.5),
            size = 3.5,
            color = "red",
            fontface = "bold")
```

```
> ggplot(credit_employment, aes(x = `Status_of_Employment(year_range)`, y = Count, fill = Credit_Category)) +
+   geom_bar(stat = "identity") +
+   scale_fill_manual(values = c("not more than 4000" = "green", "More than 4000" = "brown")) +
+   labs(title = "Stacked Bar Chart of Class by Employment",
+        x = "Credit History",
+        y = "Count") +
+   geom_text(data = credit_employment,
+             aes(label = round(Count, 1)),
+             position = position_stack(vjust = 0.5),
+             size = 3.5,
+             color = "red",
+             fontface = "bold")
```

Figure 3.2.2(1 & 2): code for stacked bar chart & console output

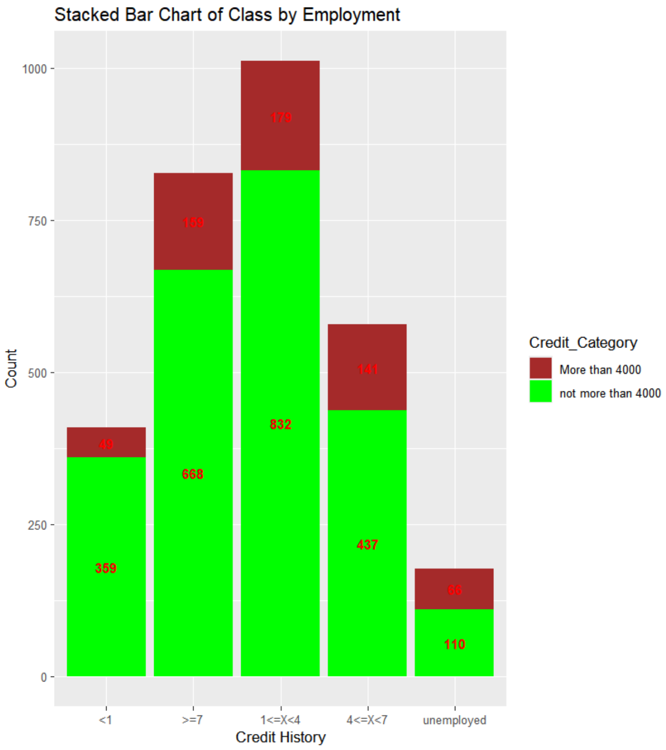


Figure 3.2.2(3): stacked bar chart of employment category

From the Stacked Bar Chart, we can see that employment has a significant majority in “>= 7 years” and “1<=x<=4 years” with “1<=x<=4 years” having the most share of the data and unemployed having the smallest share. This suggests that people who have good credit score tend to have an employment status of either “>= 7 years” or “1<=x<=4 years”, with only a minority having the status of “unemployed”.

3.2.3 Does Credit Amount, Employment, and Savings have an Impact to Credit Score

```
# savings

good_data_after_credit_amount = filter(dataset, Class == "good", `Credit_Amount(RM)` <= 4000)
good_data_after_employment_and_credit_amount =
  filter(good_data_after_credit_amount, `Status_of_Employment(year_range)` == ">=7" |
           `Status_of_Employment(year_range)` == "1<=X<4")


# summarize the newly filtered data
credit_employment_savings <- good_data_after_employment_and_credit_amount %>%
  group_by(The_Statuses_of_Savings, `Status_of_Employment(year_range)`) %>%
  summarise(count = n(), .groups = 'drop')


ggplot(credit_employment_savings, aes(x = The_Statuses_of_Savings, y = count, fill = `Status_of_Employment(year_range)`)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Stacked Bar Chart of Savings",
       x = "Savings Status",
       y = "Count") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))+
  geom_text(data = summary_new,
            aes(label = round(count, 1)),
            position = position_dodge(w = 0.9),
            vjust = -0.5,
            size = 3.5,
            color = "red",
            fontface = "bold")
```

```
> ggplot(credit_employment_savings, aes(x = The_Statuses_of_Savings, y = count, fill = `Status_of_Employment(year_range)`)) +
+   geom_bar(stat = "identity", position = "dodge") +
+   labs(title = "Stacked Bar Chart of Savings",
+        x = "Savings Status",
+        y = "Count") +
+   theme(axis.text.x = element_text(angle = 45, hjust = 1))+
+   geom_text(data = summary_new,
+             aes(label = round(count, 1)),
+             position = position_dodge(w = 0.9),
+             vjust = -0.5,
+             size = 3.5,
+             color = "red",
+             fontface = "bold")
```

Figure 3.2.3(1 &28): code for clustered bar chart & console output

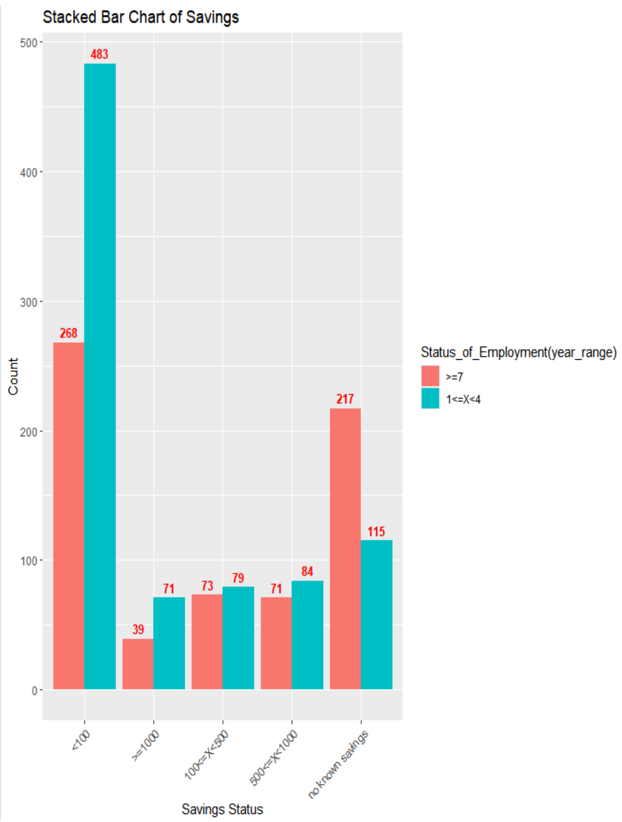


Figure 3.2.3(3): clustered bar chart of Savings Status

Based on the graph, we can conclude that savings status does have a significant impact on credit score. It shows that having a lower savings status ultimately leads to better Credit Score as we can see from the trend of having most of the data on either "no known savings" or "<100".

With that in mind, around half of the total data is concentrated on "<100" making it stand out the most. The employment status "1<=X<4" can also be concluded to have a better impact on Credit Score as in almost all saving statuses it can be seen as having larger numbers apart from the "no known savings category".

3.2.4 Does Credit Amount, Employment, Savings, and Credit History have an Impact to Credit Score

```
# credit history

good_data_after_employment_credit_amount_and_savings =
  filter(good_data_after_credit_amount,
         `Status_of_Employment(year_range)` == "1<=X<4",
         The_Statuses_of_Savings == "<100"
         )


credit_employment_savings_history <- good_data_after_employment_credit_amount_and_savings %>%
  group_by(Credit_History) %>%
  summarise(count = n(), .groups = 'drop')

ggplot(credit_employment_savings_history, aes(x = "", y = count, fill = Credit_History)) +
  geom_bar(stat = "identity", width = 1) +
  geom_label_repel(aes(label = paste(Credit_History, ":", count)),
                   position = position_stack(vjust = 0.5),
                   show.legend = FALSE)+
  coord_polar("y") +  # Convert bar chart to pie chart
  labs(title = "Credit History Distribution") +
  theme_void()
```

```
> ggplot(credit_employment_savings_history, aes(x = "", y = count, fill = Credit_History)) +
+   geom_bar(stat = "identity", width = 1) +
+   geom_label_repel(aes(label = paste(Credit_History, ":", count)),
+                    position = position_stack(vjust = 0.5),
+                    show.legend = FALSE)+
+   coord_polar("y") +  # Convert bar chart to pie chart
+   labs(title = "Credit History Distribution") +
+   theme_void()
```

Figure 3.2.4(1 & 2): code for pie chart & console output

*Note: the "good_data_after_credit_amount" variable is a reference to the previous variable in 3.2.3

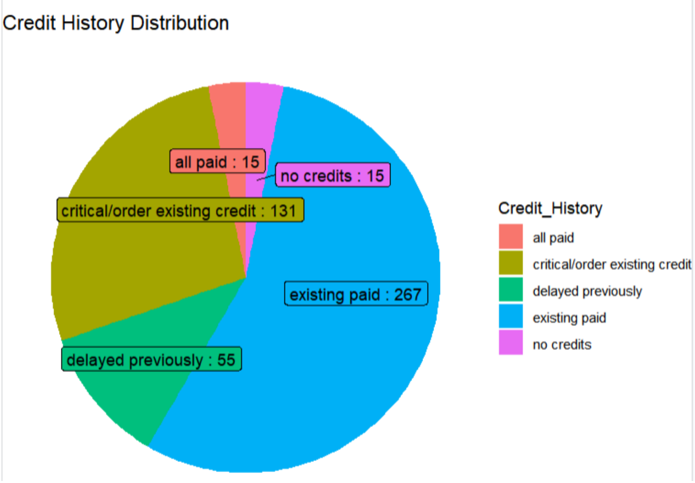


Figure 3.2.4(3): pie chart of Credit History

As we can see from the graph, the credit history of existing paid is the majority with more than 50% of the remaining data. This suggests that most people with good credit score have existing credits and successfully repaid it on time while people with "no credits" or "all paid" are likely to not have a good credit score.

3.2.5 Conclusion

these are the selected categories from our hypothesis:

- employment “1<=X<4”
- credit amount <= 4000
- saving status “< 100”
- credit history = "existing paid"

The result of this analysis is that our original hypothesis is already wrong, we predicted that saving status of “no known savings” and employment of “>=7” would have the highest probability of having a good credit score. But instead, when we did the analysis, we found that the saving status of “<100” and employment of “1<=X<4”.

With a result of 267, we have an 8.9% accuracy rate on our prediction. Suggesting that we need to revise some of the categories that we chose as it didn’t have a large enough impact to get the desired results of a high percentage in the number of good credit score data.

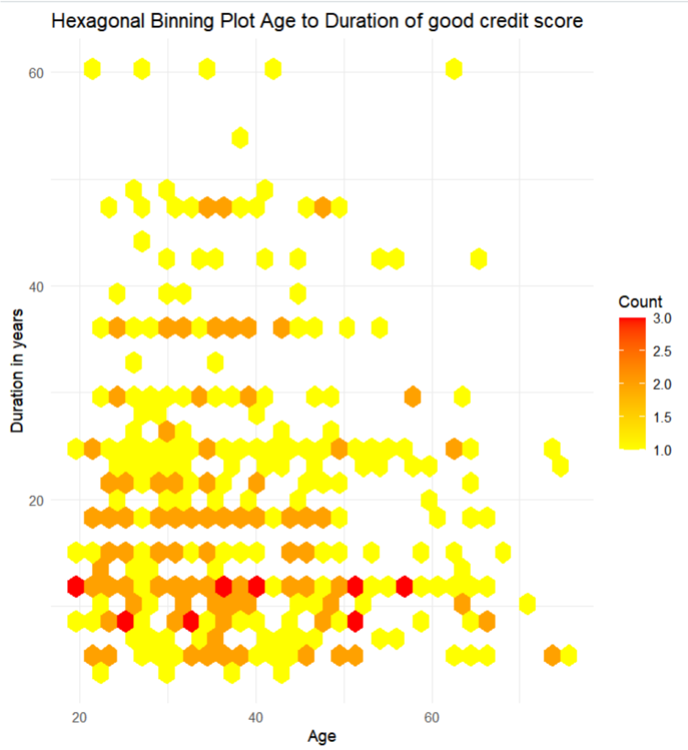
3.2.5 Additional Analysis

1) Hex Bin Plot of Age and Duration to Class

```
# age to duration of employment using geom_hex
summary_age_duration = summarise(
  group_by(good_data, Age, `Duration(Year)`),
  Count = n(), .groups = 'drop'
)

ggplot(summary_age_duration, aes(x = Age, y = `Duration(Year)`)) +
  geom_hex(bins = 30) +  # You can adjust the number of bins
  scale_fill_gradient(low = "yellow", high = "red") +
  labs(title = "Hexagonal Binning Plot Age to Duration of good credit score",
       x = "Age",
       y = "Duration in years",
       fill = "Count") +  # Legend title
  theme_minimal()
```

```
> ggplot(summary_age_duration, aes(x = Age, y = `Duration(Year)`)) +
+   geom_hex(bins = 30) +  # You can adjust the number of bins
+   scale_fill_gradient(low = "yellow", high = "red") +
+   labs(title = "Hexagonal Binning Plot Age to Duration of good credit score",
+        x = "Age",
+        y = "Duration in years",
+        fill = "Count") +  # Legend title
+   theme_minimal()
```

Figure 3.2.4(1 & 2): code for hex bin plot & console output



Out of curiosity, I tried to make a hex bin diagram to see what it would look like to compare two numerical columns this diagram shows that most people on the good credit score list are those whose age are from 20-50 and have a duration of less than 40. The “good_data” that is used here is referring to the good_data declared in the previous code in 3.2.2.

Figure 3.2.4(3): hex bin plot comparing age and duration to good credit score

2) Pie Chart of purpose column that are of class "good"

```
slices_g = table(dataset$Purpose[dataset$Class == "good"])
pie (slices_g, labels = paste(dataset$Purpose, slices_g),
     main = "Purpose pie chart",
     clockwise = FALSE)
```

```
> pie (slices_g, labels = paste(dataset$Purpose, slices_g),
+      main = "Purpose pie chart",
+      clockwise = FALSE)
```

Figure 3.2.4(4 & 5): code for pie chart & console output

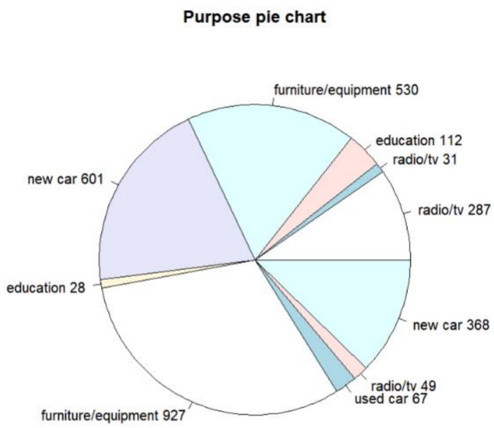


Figure 3.2.4(6): pie chart of purpose column

For this additional analysis, I looked at the purpose column to see if there was anything that could help in the hypothesis.

3) Clustered Bar Chart of "Checking Status" to class

```
summary_checking <- dataset %>%
  group_by(Checking_Status, Class) %>%
  summarise(count = n(), .groups = 'drop')

ggplot(summary_checking, aes(x = Checking_Status, y = count, fill = Class)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_fill_manual(values = c("bad" = "black", "good" = "yellow")) +
  geom_text(data = summary_checking,
            aes(x = Checking_Status, y = count, label = round(count, 1)),
            position = position_dodge(w = 0.9),
            vjust = -0.5,
            size = 3.5,
            color = "blue",
            fontface = "bold")
```

```
> ggplot(summary_checking, aes(x = Checking_Status, y = count, fill = Class)) +
+   geom_bar(stat = "identity", position = "dodge") +
+   scale_fill_manual(values = c("bad" = "black", "good" = "yellow")) +
+   geom_text(data = summary_checking,
+             aes(x = Checking_Status, y = count, label = round(count, 1)),
+             position = position_dodge(w = 0.9),
+             vjust = -0.5,
+             size = 3.5,
+             color = "blue",
+             fontface = "bold")
```

Figure 3.2.4(7 & 8): code for pie chart & console output

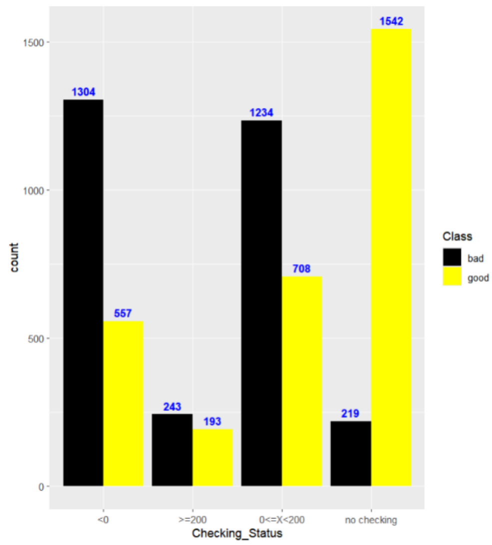


Figure 3.2.4(6): clustered bar chart of checking status

This is a graph showcasing the "Checking_Status" column. It shows that most of the good credit score is in "no checking" status and most of the bad credit score is in either the "<0" or "0<=X<200" status while ">=200" status has very little occurrences in both.

3.2.6 Additional Features

1) hexbin package - a package in RStudio that lets you analyze data using hexagonal binning(aggregating data into hexagonal bins)

2) dplyr package - a package in RStudio that is widely used for data manipulation and transformation

3) ggrepel package - an extension of the ggplot2 package which lets you improve placements of text labels and annotations

4) %>% (pipe operator) - an operator that takes the output of one function and uses it as input for the next function

5) group_by() - a main feature of the dplyr package, allows you to specify multiple columns to group, returns a data structure that is compatible with most other dplyr functions

6) summarise() - allows you to summarize the data into small aggregations
7) geom_violin() - allows you to display the density of data at different values
8) geom_label_repel() - part of the ggrepel package, it helps to avoid overlapping texts

9) label = - specifies the text that is supposed to be displayed
10) vjust = - is an aesthetic parameter that allows the adjusting of text labels in the graphs

11) ggtitle() - is used to add a title to the plot that is created
12) filter() - is used to quickly filter data based on your requirement, is also very readable compared to the alternative ways

13) ifelse() - a function that allows you to evaluate a condition and return different values according to the condition

14) mutate() - part of the dplyr package, allows operation on columns and create new columns based on calculations or transformations

15) n() - used to count the number of rows in a group when performing dplyr functions such as summarize()/group_by()/mutate()

16) .groups = - is used to control the grouping structure of the output from dplyr functions

17) scale_fill_manual() - allows you to manually set the fill colors for different groups or categories in a plot

18) labs() - is used to set or modify labels in a graph
19) position = “dodge” - an argument to make the geom_bar() function return a clustered bar graph

20) fontface = - is used to control the fonts of text in the graph
21) theme() - is used to customize the appearance of elements of the graph
22) axis.text.x - a component of the theme() function, is used to control the x axis rotation of text elements

23) element_text() - allows the custom text creation in a plot
24) angle = - is used in element_text() function as an argument to specify the rotational angle of text

25) hjust = - is used to control the horizontal position of text elements
26) position_dodge() - is used to adjust positions of overlapping elements in a graph
27) position_stack() - is used to create adjust bar charts in the stacked bar chart form

28) paste() - is used to format strings for display
29) show.legend = - an argument to configure if you can see the legend of a graph
30) coord_polar() - allows you to create polar coordinates that can help create pie charts

31) theme_void() - an additional funcion you can use to remove all background elements of a graph

32) geom_hex() - a function to create hexagonal binning plot
33) theme_minimal() - creates a minimalist theme for the graph

3.3 The Impact of Savings on Good Credit Score

Name: Goh Yuan Kee (TP070126)

3.3.1 The Impact of Good Credit Score

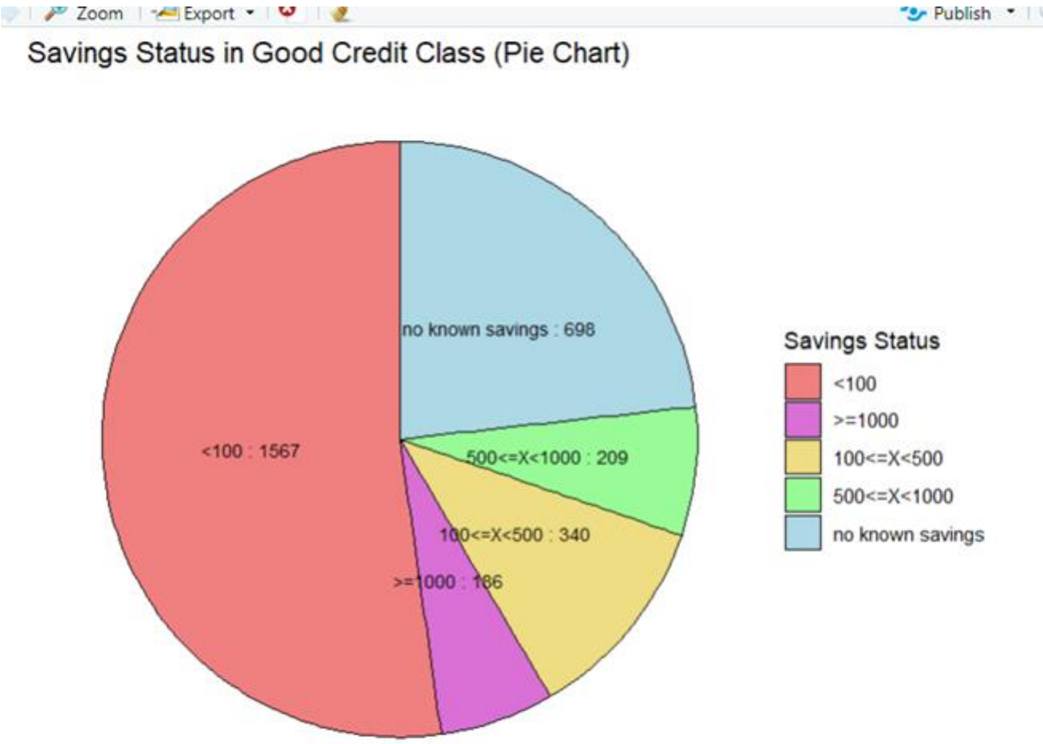
```
#column Q-V
#Customers with 1) low credit amount (0-4000), 2) long term employment(>=7), 3)no known savings,
#4) critical/order existing credit history tends to have a good credit score.
dataset[dataset$Class=="good", ]
nrow(dataset[dataset$Class=="good",] ) #3000
```

Analysis of Class == "good" Filter

First, the team decided to choose “good” filter for column Class. This is due to the reason that focusing on the “good” credit class allows the following analysis to be more meaningful by linking specific financial and demographic characteristics directly to credit outcomes. The filter for Class == “good” resulting in a total of 3000 entries.

3.3.2 The Impact of The_Statuses_of_Savings on Good Credit Score

```
# The_Statuses_of_Savings (no known saving) under good class
savings_data <- dataset %>%
  filter(Class == "good") %>%
  group_by(The_Statuses_of_Savings) %>%
  summarise(count = n())

# Custom colors for each savings status level
custom_colors <- c(
  "no known savings" = "lightblue",
  "<100" = "lightcoral",
  "500<=X<1000" = "palegreen",
  "100<=X<500" = "lightgoldenrod",
  ">=1000" = "orchid"
)

# Pie chart for The_Statuses_of_Savings
savings_pie <- ggplot(savings_data, aes(x = "", y = count, fill = The_Statuses_of_Savings)) +
  geom_bar(width = 1, stat = "identity", color = "black") +
  coord_polar("y") +
  labs(title = "Savings Status in Good Credit Class (Pie Chart)", fill = "Savings Status") +
  geom_text(aes(label = paste(The_Statuses_of_Savings, ":", count)),
            position = position_stack(vjust = 0.5), size = 3, color = "black") +
  scale_fill_manual(values = custom_colors) +
  theme_void() +
  theme(legend.position = "right")
savings_pie
```



<100 #1567

Analysis of The_Statuses_of_Savings == "<100" Filter

Based on the pie chart above, the “<100” is selected over “no known saving” that the teams decided for the hypothesis which is 1567 individuals are on “<100”. In contrast, “no known saving” which was chosen for the hypothesis, includes only 698 individuals, which is significantly lower.

3.3.3 The Impact of Credit_History on Good Credit Score

```
#credit history under class good
# Filter dataset to include only "good" credit class and count occurrences of each Credit History type
credit_history_data <- dataset %>%
  filter(Class == "good") %>%
  count(Credit_History)

# View the count data to ensure it only includes the "good" class
print(credit_history_data)

# Bar chart for Credit History in Good Credit Score Class
credit_history_bar <- ggplot(credit_history_data, aes(x = Credit_History, y = n, fill = Credit_History)) +
  geom_bar(stat = "identity", width = 0.7, color = "black") +
  labs(title = "Credit History Distribution in Good Credit Score Class", x = "Credit History", y = "Count") +
  geom_text(aes(label = n), vjust = -0.3, color = "black") +  # Adds count labels above each bar
  scale_fill_manual(values = c(
    "critical/order existing credit" = "lightcoral",
    "existing paid" = "palegreen",
    "no credits" = "skyblue",
    "delayed" = "lightgoldenrod",
    "fully paid" = "orchid"
  )) +
  theme_minimal() +
  theme(legend.position = "none", axis.text.x = element_text(angle = 45, hjust = 1))
credit_history_bar
```

Existing paid

Analysis of Credit_History == "existing paid" Filter

Based on the bar chart above, the teams chose “critical/order existing credit” as the hypothesis for Credit_History. The selection of “Existing paid” was favored over “critical/order existing credit” in which “Existing paid” is 1513 and “critical/order existing credit” is 1038 which is smaller than “Existing paid”.

3.3.4 The Impact of Credit_Amount on Good Credit Score

```r
# Filter dataset to include only "good" credit class
good_credit_data <- dataset %>%
  filter(Class == "good") %>%
  mutate(Credit_Category = ifelse(`Credit_Amount(RM)` <= 4000, "<=4000", ">4000"))

# View the filtered data to ensure only "good" class data is included with categorized Credit Amounts
print(good_credit_data)

# Box plot for Credit Amount Distribution (<=4000 and >4000) in Good Credit Score Class
credit_amount_box <- ggplot(good_credit_data, aes(x = Credit_Category, y = `Credit_Amount(RM)`, fill = Credit_Category)) +
  geom_boxplot(color = "black", width = 0.5, outlier.colour = "red", outlier.shape = 16) +
  scale_fill_manual(values = c("<=4000" = "lightgreen", ">4000" = "lightblue")) +
  labs(title = "Credit Amount Distribution in Good Credit Score Class",
       x = "Credit Amount Category",
       y = "Credit Amount (RM)") +
  theme_minimal() +
  theme(legend.position = "none")
credit_amount_box
```

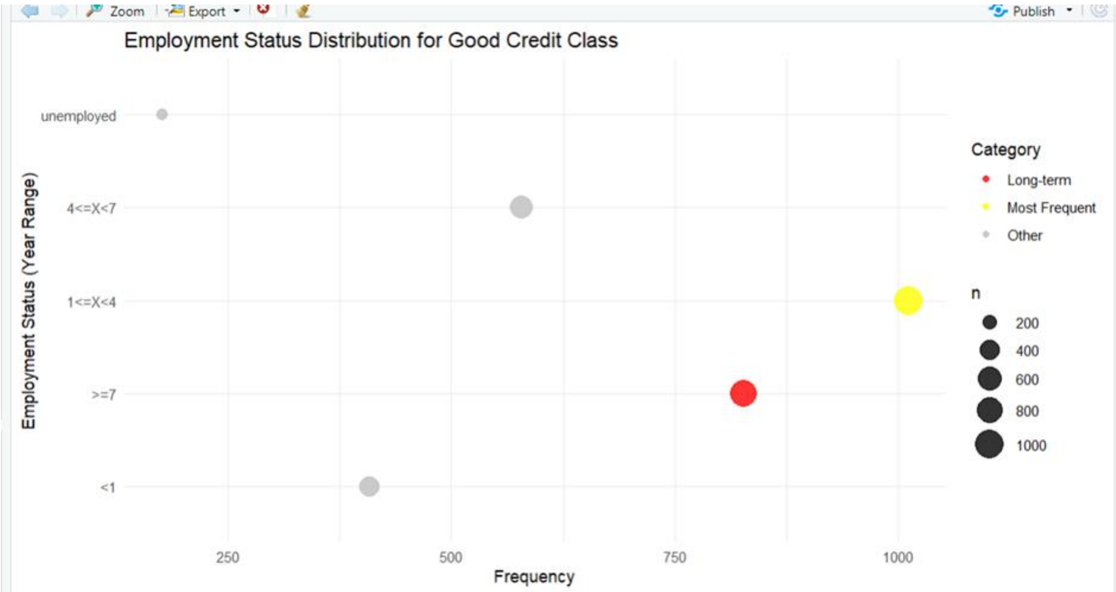
<=4000

Analysis of Credit_Amount(RM) <= 4000 Filter

Based on the box plot shown above, “<=4000” is more than “>=4000”.This is because the whiskers extend to the minimum and maximum values of the data, excluding outliers. The upper whisker for the >4000 category is much longer, indicating that there are higher credit amounts in this category. Additionally, the red dots represent outliers, which are significantly higher than the rest of the data points in this category.

3.3.4 The Impact of Status_of_Emplyment on Good Credit Score

```r
# Filter dataset to include only "good" credit class
good_data <- dataset %>%
  filter(Class == "good")

# Calculate the count of each employment status
employment_counts <- good_data %>%
  count(`Status_of_Employment(year_range)`)

# Identify the employment status with the highest count
highest_status <- employment_counts %>%
  filter(n == max(n)) %>%
  pull(`Status_of_Employment(year_range)`)

# Create a dot plot for Employment Status with special styling for ">=7" and the highest count
employment_status_dot_plot <- ggplot(employment_counts, aes(x = n, y = `Status_of_Employment(year_range)`)) +
  geom_point(aes(size = n, color = ifelse(`Status_of_Employment(year_range)` == ">=7", "Long-term",
                                          ifelse(`Status_of_Employment(year_range)` == highest_status, "Most Frequent", "Other"))),
             alpha = 0.8) +
  scale_color_manual(values = c("Long-term" = "red", "Most Frequent" = "yellow", "Other" = "grey")) +
  scale_size_continuous(range = c(3, 8)) +
  labs(title = "Employment Status Distribution for Good Credit Class",
       x = "Frequency",
       y = "Employment Status (Year Range)",
       color = "Category") +
  theme_minimal() +
  theme(legend.position = "right")
employment_status_dot_plot
```



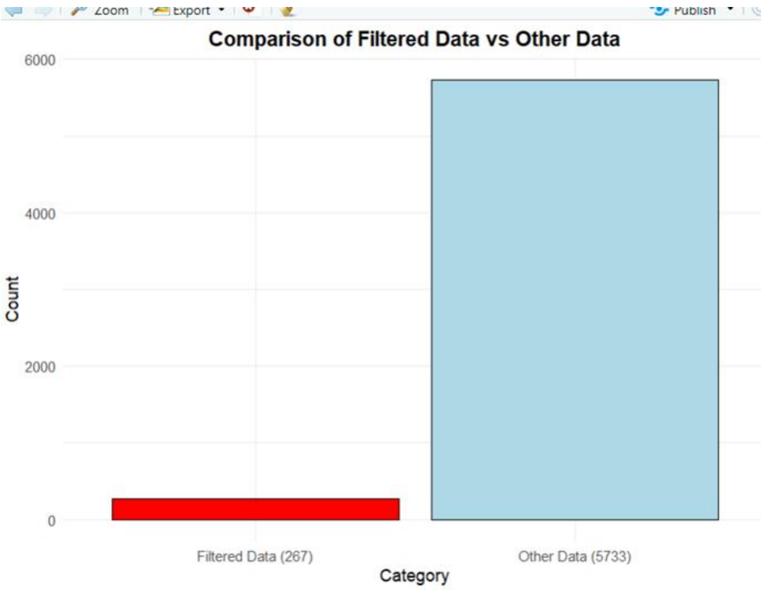
1<=x<4

Analysis of Status_of_Employment(year_range) == "1<=x<4" Filter

Based on the dot plot above, “1<=X<4” is more than “>=7” which was decided by the teams for the hypothesis. This is due to the fact that the larger the dot is, the higher the value is.

3.3.5 Outcome of Initial Hypothesis and Conclusion

```r
# Create a data frame for plotting
data_for_bar_chart <- data.frame(
  Category = c("Filtered Data (267)", "Other Data (5733)"),
  Count = c(267, 5733)  # Counts for filtered and other data
)

# Create the bar chart using ggplot2
bar_chart <- ggplot(data_for_bar_chart, aes(x = Category, y = Count, fill = Category)) +
  geom_bar(stat = "identity", color = "black") +  # Create bars
  labs(title = "Comparison of Filtered Data vs Other Data",
       x = "Category",
       y = "Count") +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),  # Center the title
    legend.position = "none"  # Remove legend
  ) +
  scale_fill_manual(values = c("Filtered Data (267)" = "red", "Other Data (5733)" = "lightblue"))

# Display the bar chart
print(bar_chart)
```



After applying all filters — good credit class, savings less than RM100, "existing paid" credit history, credit amount ≤ RM4000, and employment between 1 to 4 years — the combined dataset yielded only 267 individuals, representing just 8.9% of the original good credit class entries. This outcome highlights that while each factor aligns with maintaining good credit, they do not collectively define the majority. The low percentage indicates that a broader range of conditions or alternative filters may need to be considered to effectively capture a more significant portion of good credit holders in the dataset. Further analysis with additional or adjusted criteria could increase the predictive accuracy of identifying good credit status.

Conclusion:

Based on the final combined conclusion, the filtered criteria yielded only 237 entries, which is significantly lower than expected and represents less than 60% of the total "good" credit class. This outcome highlights a gap between our initial hypothesis and the actual data trends. Given this low value, our team has decided to explore alternative columns and criteria in order to develop a more accurate and representative model. By adjusting our approach, we aim to increase the percentage to over 70%, thus improving the reliability of our findings and achieving a more comprehensive understanding of the factors associated with maintaining a good credit score.

3.3.6 Additional Analysis

For counting percentages

```r
#caculate the %
# Create a data frame for plotting
data_for_bar_chart <- data.frame(
  Category = c("Filtered Data (267)", "Other Data (5733)"),
  Count = c(267, 5733)  # Counts for filtered and other data
)

# Calculate the total count
total_count <- sum(data_for_bar_chart$Count)

# Calculate the percentage for each category
data_for_bar_chart$Percentage <- (data_for_bar_chart$Count / total_count) * 100

# Print the updated data frame with percentages
print(data_for_bar_chart)

# Check if the percentages sum to 100
percentage_sum <- sum(data_for_bar_chart$Percentage)
cat("Sum of Percentages: ", percentage_sum, "%\n")
```

```
R 4.4.1 · ~/
> # Print the updated data frame with percentages
> print(data_for_bar_chart)
             Category Count Percentage
1 Filtered Data (267)   267       4.45
2   Other Data (5733)  5733      95.55
>
> # Check if the percentages sum to 100
> percentage_sum <- sum(data_for_bar_chart$Percentage)
> cat("Sum of Percentages: ", percentage_sum, "%\n")
Sum of Percentages:  100 %
>
```

The analysis of the data reveals that the percentage of filtered data is 4.45%, while the percentage of other data is 95.55%. As observed, the percentage of filtered data is significantly below the threshold of 70%, which is considered low. To improve this situation and achieve a higher percentage of filtered data, the team has decided to modify the criteria used for filtering the data.

Furthermore, it is important to note that the sum of the percentages for both categories equals to100%, confirming the accuracy of the percentage calculations.

Trying to find another column

Trying for Job==skilled column under class==good

```
# Count the number of records that meet the criteria
nrow(dataset[(dataset$Class == "good")
             & (dataset$'Job' == "skilled"), ]) #1893
```

This is the output which is 1893 that seems good.

```
> # Count the number of records that meet the criteria
> nrow(dataset[(dataset$Class == "good")
+              & (dataset$'Job' == "skilled"), ])
[1] 1893
```

Trying for Housing ==own under class == good

```
# Count the number of records that meet the criteria
nrow(dataset[(dataset$Class == "good")
             & (dataset$'Housing' == "own"), ]) #2282
```

This is the output which is 2282.

```
# Count the number of records that meet the criteria
nrow(dataset[(dataset$Class == "good")
             & (dataset$'Housing' == "own"), ])
] 2282
```

Combined Job and Housing under class == good

```
# Count the number of records that meet the criteria
nrow(dataset[(dataset$Class == "good")
             & (dataset$'Job' == "skilled")
             & (dataset$'Housing' == "own"), ])#1477
```

This is the output that consider bad which is 1477. Due to this output, it is not suitable continue do for 4 of the hypotheses.

```
# Count the number of records that meet the criteria
nrow(dataset[(dataset$Class == "good")
             & (dataset$'Job' == "skilled")
             & (dataset$'Housing' == "own"), ])
1] 1477
```

3.3.7 Additional features

1. **group_by()** – Groups a data frame by one or more variables for subsequent operations.
2. **summarise()** – Reduces a data frame to summary statistics.
3. **ggplot()** – Initializes a ggplot object for creating visualizations.
4. **geom_bar()** – Creates a bar chart.
5. **geom_boxplot()** – Creates a box plot.
6. **coord_polar()** – Converts a Cartesian plot to polar coordinates, often used for pie charts.
7. **labs()** – Modifies titles and labels for the plot.
8. **scale_fill_manual()** – Manually sets the fill colors for different categories in a plot.
9. **theme_minimal()** – Applies a minimal theme to the plot, reducing clutter.
10. **theme()** – Customizes non-data elements of the plot, such as text and grid lines.
11. **element_text()** – Modifies text elements in the plot, such as font size and style.
12. **vjust** – Adjusts the vertical position of text labels.
13. **hjust** – Adjusts the horizontal position of text labels.
14. **ncol()** – Returns the number of columns in a data frame.
15. **summary()** – Provides a summary of the data frame, including statistics for each column.
16. **View()** – Opens a spreadsheet-style data viewer for the data frame.
17. **geom_point()** – Creates a scatter plot.
18. **geom_col()** – Creates a bar chart using pre-determined values for the height of the bars.
19. **theme_void()** – Removes all background elements from the plot.
20. **legend.position** – Specifies the position of the legend in the plot.
21. **alpha** – Adjusts the transparency of elements in the plot.
22. **geom_boxplot()** – Creates a box plot to visualize the distribution of a continuous variable.
23. **ifelse()** – Conditional function to categorize elements by whether a certain condition is met.
24. **tabulate()** – Creates a frequency table of the values in a vector.
25. **pull()** – Extracts a single column as a vector.
26. **cat()** – Concatenates and prints strings and values.
27. **paste()** – Concatenates strings with an optional separator.
28. **%>%** – The pipe operator from dplyr, chaining multiple functions together.

29. **ggplot2()**- R package for creating customizable data visualizations using a layered approach.

30. **dplyr()**-R package for efficient data manipulation and transformation.

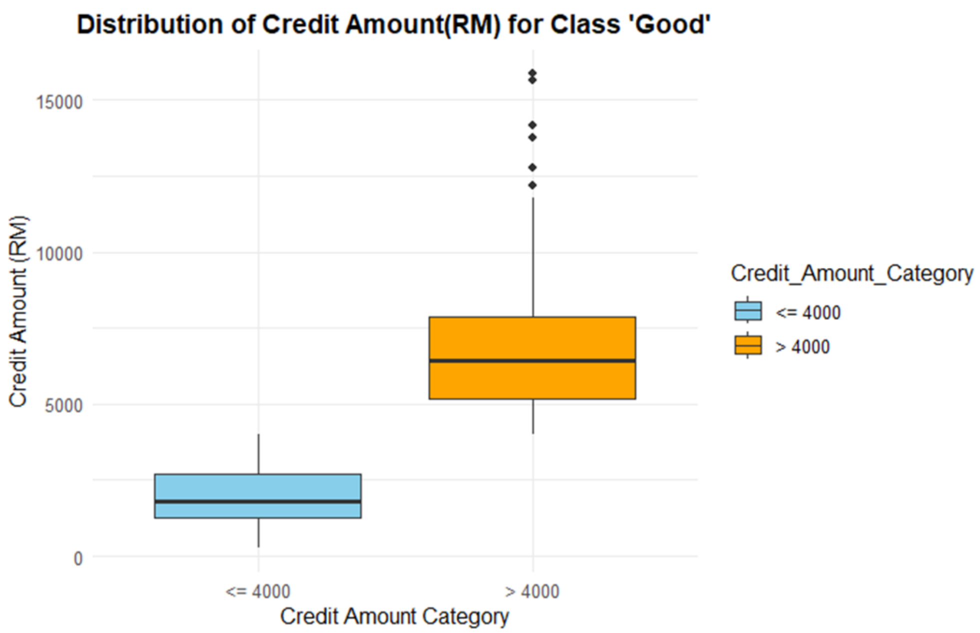
3.4 The Impact of Credit History on Good Credit Score

Name: Chloe Tan Jia Xin (TP070759)

3.4.1 The Impact of Credit_Amount on Good Credit Score

```
# Load required libraries
library(ggplot2)
library(dplyr)

# Calculate the counts for "good" class with Credit_Amount(RM) <= 4000 and > 4000
count_4000_or_less <- nrow(dataset %>% filter(Class == "good" & `Credit_Amount(RM)` <= 4000))#2406
count_above_4000 <- nrow(dataset %>% filter(Class == "good" & `Credit_Amount(RM)` > 4000))#594

# Create a new column for categorizing Credit_Amount(RM) into "<= 4000" and "> 4000"
dataset <- dataset %>%
  mutate(Credit_Amount_Category = ifelse(`Credit_Amount(RM)` <= 4000, "<= 4000", "> 4000"))

# Create the box plot using ggplot2
ggplot(dataset %>% filter(Class == "good"), aes(x = Credit_Amount_Category, y = `Credit_Amount(RM)`, fill = Credit_Amount_Category)) +
  geom_boxplot() +                          # Box plot
  labs(title = "Distribution of Credit Amount(RM) for Class 'Good'",
       x = "Credit Amount Category", y = "Credit Amount (RM)") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, face = "bold")) +
  scale_fill_manual(values = c("<= 4000" = "skyblue", "> 4000" = "orange"))
```

```
> # Load required libraries
> library(ggplot2)
> library(dplyr)
>
> # Calculate the counts for "good" class with Credit_Amount(RM) <= 4000 and > 4000
> count_4000_or_less <- nrow(dataset %>% filter(Class == "good" & `Credit_Amount(RM)` <= 4000))#2406
> count_above_4000 <- nrow(dataset %>% filter(Class == "good" & `Credit_Amount(RM)` > 4000))#594
>
> # Create a new column for categorizing Credit_Amount(RM) into "<= 4000" and "> 4000"
> dataset <- dataset %>%
+   mutate(Credit_Amount_Category = ifelse(`Credit_Amount(RM)` <= 4000, "<= 4000", "> 4000"))
>
> # Create the box plot using ggplot2
> ggplot(dataset %>% filter(Class == "good"), aes(x = Credit_Amount_Category, y = `Credit_Amount(RM)`, fill = Credit_Amount_Category)) +
+   geom_boxplot() +                          # Box plot
+   labs(title = "Distribution of Credit Amount(RM) for Class 'Good'",
+        x = "Credit Amount Category", y = "Credit Amount (RM)") +
+   theme_minimal() +
+   theme(plot.title = element_text(hjust = 0.5, face = "bold")) +
+   scale_fill_manual(values = c("<= 4000" = "skyblue", "> 4000" = "orange"))
```

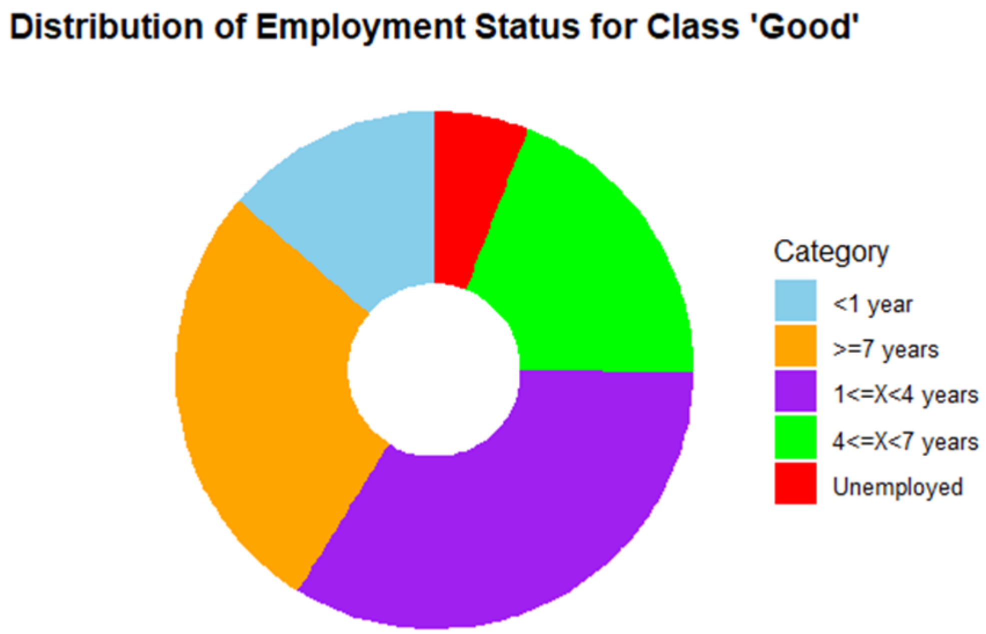


From the analysis above, there is approximately 2406 people have a credit amount of <=4000 while remaining 594 have a credit amount of exceeding RM4000. This indicates that most of the people in Class == "good" have a credit amount within the <=4000 range. Therefore, we will be using the majority of good credit score, which is <=4000 for the next analysis.

3.4.2 The Impact of Status_Of_Employment on Good Credit Score

```r
# Load required libraries
library(ggplot2)
library(dplyr)

# Calculate the counts for each employment status
count_less_than_1 <- nrow(dataset[(dataset$Class == "good") & (dataset$`Status_of_Employment(year_range)` == "<1"),]) # 408
count_7_or_more <- nrow(dataset[(dataset$Class == "good") & (dataset$`Status_of_Employment(year_range)` == ">=7"),]) # 827
count_1_to_4 <- nrow(dataset[(dataset$Class == "good") & (dataset$`Status_of_Employment(year_range)` == "1<=X<4"),]) # 1011
count_4_to_7 <- nrow(dataset[(dataset$Class == "good") & (dataset$`Status_of_Employment(year_range)` == "4<=X<7"),]) # 578
count_unemployed <- nrow(dataset[(dataset$Class == "good") & (dataset$`Status_of_Employment(year_range)` == "unemployed"),]) # 176

# Create a data frame with the counts for each employment status
employment_data <- data.frame(
  Category = c("<1 year", ">=7 years", "1<=X<4 years", "4<=X<7 years", "Unemployed"),
  Count = c(count_less_than_1, count_7_or_more, count_1_to_4, count_4_to_7, count_unemployed)
)

# Create the donut chart
ggplot(employment_data, aes(x = 2, y = Count, fill = Category)) +
  geom_bar(stat = "identity", width = 1) +           # Bar chart
  coord_polar("y", start = 0) +                      # Convert to polar coordinates (circular)
  labs(title = "Distribution of Employment Status for Class 'Good'") +
  theme_void() +                                     # Remove axis and grid
  theme(plot.title = element_text(hjust = 0.5, face = "bold")) +

  # Set x aesthetic to create a hole in the center (donut effect)
  scale_x_continuous(limits = c(1, 2.5)) +

  # Custom colors for each segment
  scale_fill_manual(values = c("<1 year" = "skyblue",
                               ">=7 years" = "orange",
                               "1<=X<4 years" = "purple",
                               "4<=X<7 years" = "green",
                               "Unemployed" = "red"))
```

```
> # Load required libraries
> library(ggplot2)
> library(dplyr)
>
> # Calculate the counts for each employment status
> count_less_than_1 <- nrow(dataset[(dataset$Class == "good") & (dataset$`Status_of_Employment(year_range)` == "<1"),]) # 408
> count_7_or_more <- nrow(dataset[(dataset$Class == "good") & (dataset$`Status_of_Employment(year_range)` == ">=7"),]) # 827
> count_1_to_4 <- nrow(dataset[(dataset$Class == "good") & (dataset$`Status_of_Employment(year_range)` == "1<=X<4"),]) # 1011
> count_4_to_7 <- nrow(dataset[(dataset$Class == "good") & (dataset$`Status_of_Employment(year_range)` == "4<=X<7"),]) # 578
> count_unemployed <- nrow(dataset[(dataset$Class == "good") & (dataset$`Status_of_Employment(year_range)` == "unemployed"),]) # 176
>
> # Create a data frame with the counts for each employment status
> employment_data <- data.frame(
+   Category = c("<1 year", ">=7 years", "1<=X<4 years", "4<=X<7 years", "Unemployed"),
+   Count = c(count_less_than_1, count_7_or_more, count_1_to_4, count_4_to_7, count_unemployed)
+ )
>
> # Create the donut chart
> ggplot(employment_data, aes(x = 2, y = Count, fill = Category)) +
+   geom_bar(stat = "identity", width = 1) +           # Bar chart
+   coord_polar("y", start = 0) +                      # Convert to polar coordinates (circular)
+   labs(title = "Distribution of Employment Status for Class 'Good'") +
+   theme_void() +                                     # Remove axis and grid
+   theme(plot.title = element_text(hjust = 0.5, face = "bold")) +
+
+   # Set x aesthetic to create a hole in the center (donut effect)
+   scale_x_continuous(limits = c(1, 2.5)) +
+
+   # Custom colors for each segment
+   scale_fill_manual(values = c("<1 year" = "skyblue",
+                                ">=7 years" = "orange",
+                                "1<=X<4 years" = "purple",
+                                "4<=X<7 years" = "green",
+                                "Unemployed" = "red"))
```

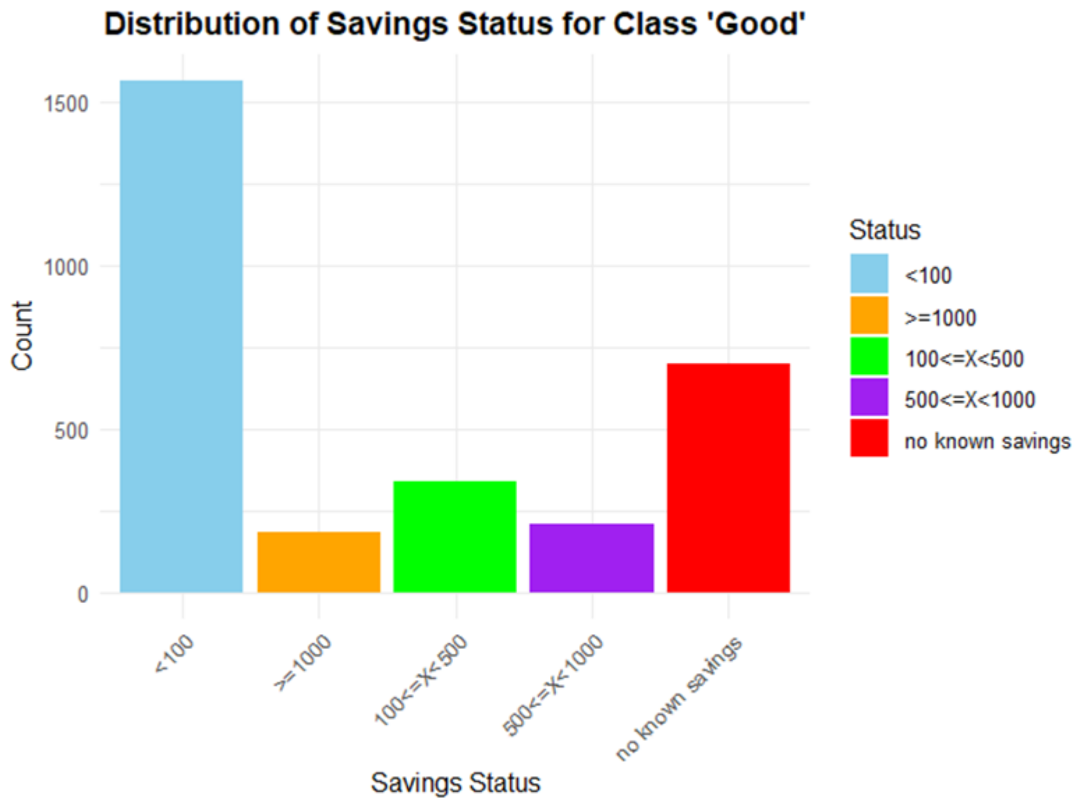


According to the analysis, the status of employment “1<=X<4 years” has the highest majority of data compared to the other categories. Although our initial hypothesis, which is status of employment “>=7”, also contributes a decent amount of data, the status of employment “1<=X<4” would be most ideal. As for the remaining statuses of employment, "<1”, “4<=X<7" and "unemployed" contribute lesser to the “good” class data.

3.4.3 The Impact of Status_Of_Savings on Good Credit Score

```r
# Load required libraries
library(ggplot2)
library(dplyr)

# Calculate the counts for each category of 'The_Statuses_of_Savings' for 'good' class
count_lessthan_100 <- nrow(dataset %>% filter(Class == "good" & `The_Statuses_of_Savings` == "<100"))
count_1000_or_more <- nrow(dataset %>% filter(Class == "good" & `The_Statuses_of_Savings` == ">=1000"))
count_100_to_500 <- nrow(dataset %>% filter(Class == "good" & `The_Statuses_of_Savings` == "100<=X<500"))
count_500_to_1000 <- nrow(dataset %>% filter(Class == "good" & `The_Statuses_of_Savings` == "500<=X<1000"))
count_no_savings <- nrow(dataset %>% filter(Class == "good" & `The_Statuses_of_Savings` == "no known savings"))


# Create a data frame with the counts for each savings status
savings_data <- data.frame(
  Status = c("<100", ">=1000", "100<=X<500", "500<=X<1000", "no known savings"),
  Count = c(1567, 186, 340, 209, 698)
)

# Create the bar chart
ggplot(savings_data, aes(x = Status, y = Count, fill = Status)) +
  geom_bar(stat = "identity") +  # Bar chart with heights proportional to count
  labs(title = "Distribution of Savings Status for Class 'Good'",
       x = "Savings Status", y = "Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_fill_manual(values = c("<100" = "skyblue", ">=1000" = "orange",
                               "100<=X<500" = "green", "500<=X<1000" = "purple",
                               "no known savings" = "red"))
```

```
> # Load required libraries
> library(ggplot2)
> library(dplyr)
>
> # Calculate the counts for each category of 'The_Statuses_of_Savings' for 'good' class
> count_lessthan_100 <- nrow(dataset %>% filter(Class == "good" & `The_Statuses_of_Savings` == "<100"))
> count_1000_or_more <- nrow(dataset %>% filter(Class == "good" & `The_Statuses_of_Savings` == ">=1000"))
> count_100_to_500 <- nrow(dataset %>% filter(Class == "good" & `The_Statuses_of_Savings` == "100<=X<500"))
> count_500_to_1000 <- nrow(dataset %>% filter(Class == "good" & `The_Statuses_of_Savings` == "500<=X<1000"))
> count_no_savings <- nrow(dataset %>% filter(Class == "good" & `The_Statuses_of_Savings` == "no known savings"))
>
>
> # Create a data frame with the counts for each savings status
> savings_data <- data.frame(
+   Status = c("<100", ">=1000", "100<=X<500", "500<=X<1000", "no known savings"),
+   Count = c(1567, 186, 340, 209, 698)
+ )
>
> # Create the bar chart
> ggplot(savings_data, aes(x = Status, y = Count, fill = Status)) +
+   geom_bar(stat = "identity") +  # Bar chart with heights proportional to count
+   labs(title = "Distribution of Savings Status for Class 'Good'",
+        x = "Savings Status", y = "Count") +
+   theme_minimal() +
+   theme(plot.title = element_text(hjust = 0.5, face = "bold")) +
+   theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
+   scale_fill_manual(values = c("<100" = "skyblue", ">=1000" = "orange",
+                                "100<=X<500" = "green", "500<=X<1000" = "purple",
+                                "no known savings" = "red"))
```

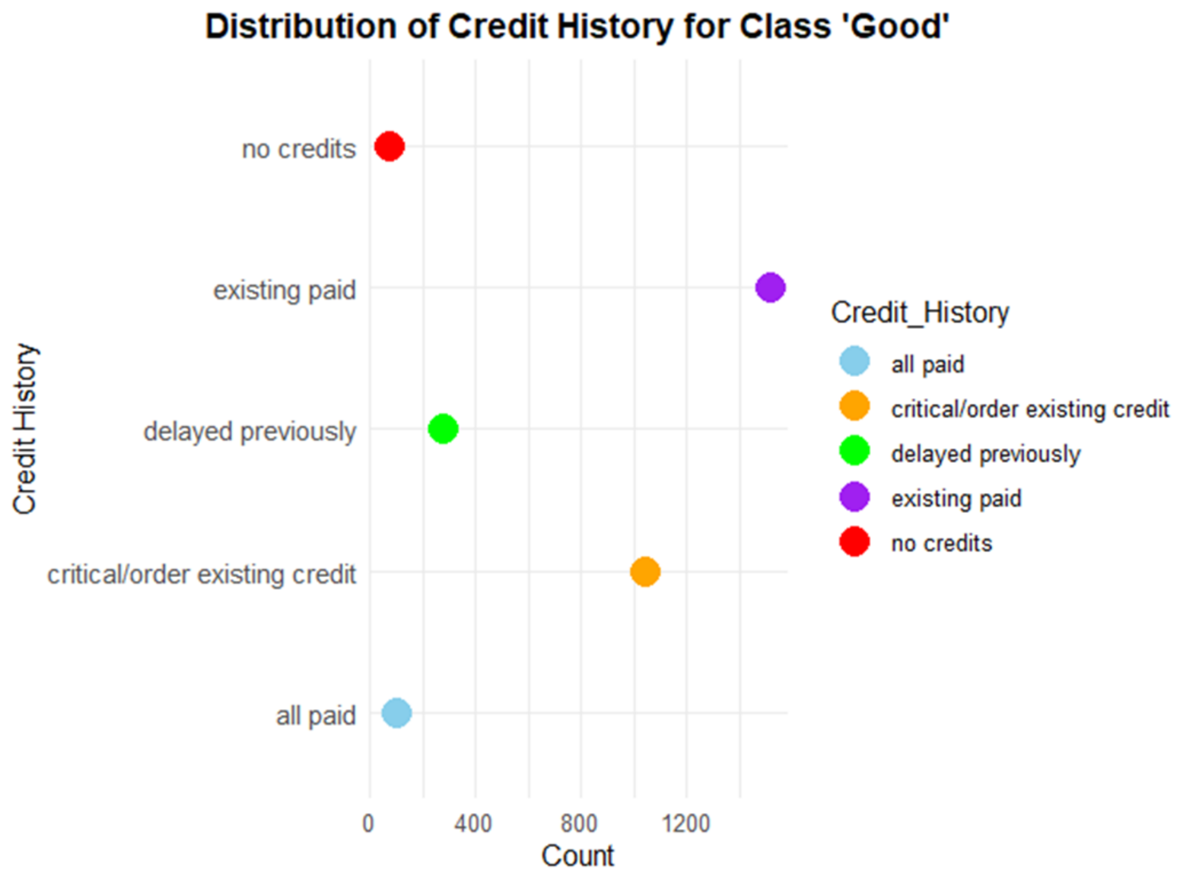


As we can see from the graph, the data that stands out most with the highest relevance to the "good" class is the Status of Savings "<100". This finding contradicts our initial hypothesis, which is the Status of Savings "no known savings" being the highest majority with the "good" class data. Thus, it would be preferrable to adjust the analysis and use the Status of Savings "<100" for a better prediction.

3.4.3 The Impact of Credit_History on Good Credit Score

```
#Credit History
# Load required libraries
library(ggplot2)

# Calculate the counts for each category of 'Credit_History' for 'good' class
count_all_paid <- nrow(dataset %>% filter(Class == "good" & `Credit_History` == "all paid"))
count_critical_order_existing_credit <- nrow(dataset %>% filter(Class == "good" & `Credit_History` == "critical/order existing credit"))
count_delayed_previously <- nrow(dataset %>% filter(Class == "good" & `Credit_History` == "delayed previously"))
count_existing_paid <- nrow(dataset %>% filter(Class == "good" & `Credit_History` == "existing paid"))
count_no_credits <- nrow(dataset %>% filter(Class == "good" & `Credit_History` == "no credits"))

# Create a data frame with the counts for each Credit_History category for Class "good"
credit_history_data <- data.frame(
  Credit_History = c("all paid", "critical/order existing credit", "delayed previously",
                     "existing paid", "no credits"),
  Count = c(99, 1038, 279, 1513, 71)
)

# Create the dot plot with discrete colors
ggplot(credit_history_data, aes(x = Count, y = Credit_History, color = Credit_History)) +
  geom_point(size = 5) +
  labs(title = "Distribution of Credit History for Class 'Good'",
       x = "Count", y = "Credit History") +
  scale_color_manual(values = c("all paid" = "skyblue",
                                "critical/order existing credit" = "orange",
                                "delayed previously" = "green",
                                "existing paid" = "purple",
                                "no credits" = "red")) +
  theme_minimal() +
  theme(axis.text.y = element_text(size = 10))
```

```
> #Credit History
> # Load required libraries
> library(ggplot2)
>
> # Calculate the counts for each category of 'Credit_History' for 'good' class
> count_all_paid <- nrow(dataset %>% filter(Class == "good" & `Credit_History` == "all paid"))
> count_critical_order_existing_credit <- nrow(dataset %>% filter(Class == "good" & `Credit_History` == "critical/order existing credit"))
> count_delayed_previously <- nrow(dataset %>% filter(Class == "good" & `Credit_History` == "delayed previously"))
> count_existing_paid <- nrow(dataset %>% filter(Class == "good" & `Credit_History` == "existing paid"))
> count_no_credits <- nrow(dataset %>% filter(Class == "good" & `Credit_History` == "no credits"))
>
> # Create a data frame with the counts for each Credit_History category for Class "good"
> credit_history_data <- data.frame(
+   Credit_History = c("all paid", "critical/order existing credit", "delayed previously",
+                      "existing paid", "no credits"),
+   Count = c(99, 1038, 279, 1513, 71)
+ )
>
> # Create the dot plot with discrete colors
> ggplot(credit_history_data, aes(x = Count, y = Credit_History, color = Credit_History)) +
+   geom_point(size = 5) +
+   labs(title = "Distribution of Credit History for Class 'Good'",
+        x = "Count", y = "Credit History") +
+   scale_color_manual(values = c("all paid" = "skyblue",
+                                 "critical/order existing credit" = "orange",
+                                 "delayed previously" = "green",
+                                 "existing paid" = "purple",
+                                 "no credits" = "red")) +
+   theme_minimal() +
+   theme(plot.title = element_text(hjust = 0.5, face = "bold")) +
+   theme(axis.text.y = element_text(size = 10))
```

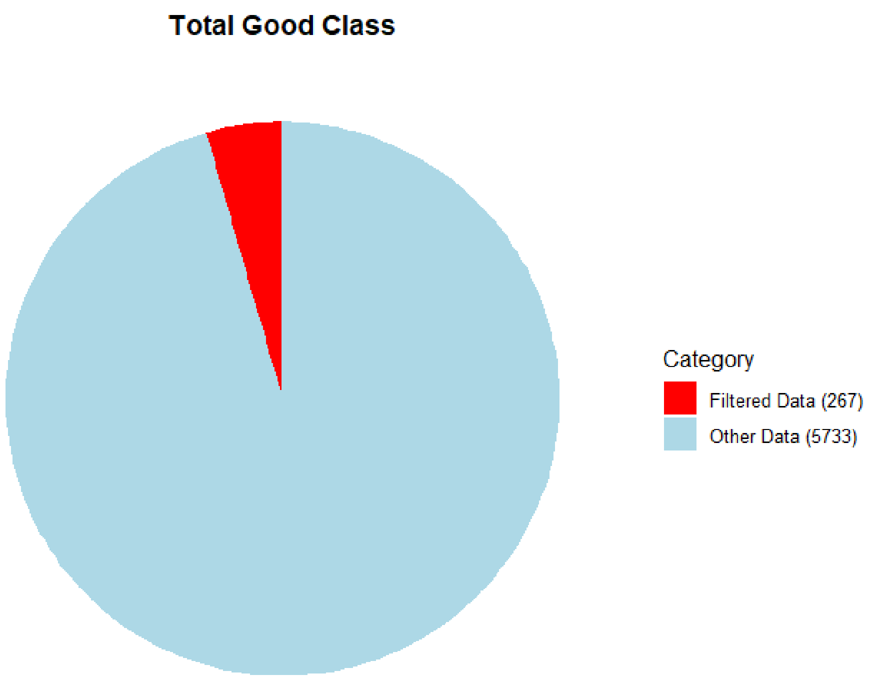


Based on the dot plot, Credit_History "existing paid" holds the highest share with 1513 entries, making it the leading category in the "good" class data. On the other hand, Credit_History "critical/other existing credit" totals up to 1038 entries, making it considerably lower. Because of its lesser share, using the Credit_History "existing paid" in our analysis would be optimal for a more influential prediction in the "good" class.

3.4.4 The Impact of Credit_Amount, Status_Of_Employment, Status_Of_Savings, and Credit_History on Good Credit Score

```r
# FINAL CONCLUSION FOR FIRST HYPOTHESIS
nrow(dataset[(dataset$Class=="good") & (dataset$The_Statuses_of_Savings=="<100") &
               (dataset$`Credit_Amount(RM)`<=4000) & (dataset$`Status_of_Employment(year_range)`=="1<=X<4") &
               (dataset$`Credit_History` == "existing paid"),])
# Load required library
library(ggplot2)

# Create a data frame for plotting
bar_data <- data.frame(
  Category = c("Filtered Data (267)", "Other Data (5733)"),
  Count = c(267, 6000 - 267)
)

# Create the pie chart using ggplot2
ggplot(bar_data, aes(x = "", y = Count, fill = Category)) +
  geom_bar(stat = "identity", width = 1) +        # Bar chart
  coord_polar("y", start = 0) +                   # Convert to pie chart
  labs(title = "Total Good Class") +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),  # Center the title
    axis.title.x = element_blank(),                        # Remove x-axis title
    axis.title.y = element_blank(),                        # Remove y-axis title
    axis.text = element_blank(),                           # Remove axis text
    axis.ticks = element_blank(),                          # Remove axis ticks
    panel.grid = element_blank()                           # Remove grid lines
  ) +
  scale_fill_manual(values = c("Filtered Data (267)" = "red", "Other Data (5733)" = "lightblue"))
```

```
> # Load required library
> library(ggplot2)
>
> # Create a data frame for plotting
> bar_data <- data.frame(
+   Category = c("Filtered Data (267)", "Other Data (5733)"),
+   Count = c(267, 6000 - 267)
+ )
>
> # Create the pie chart using ggplot2
> ggplot(bar_data, aes(x = "", y = Count, fill = Category)) +
+   geom_bar(stat = "identity", width = 1) +        # Bar chart
+   coord_polar("y", start = 0) +                   # Convert to pie chart
+   labs(title = "Total Good Class") +
+   theme_minimal() +
+   theme(
+     plot.title = element_text(hjust = 0.5, face = "bold"),  # Center the title
+     axis.title.x = element_blank(),                        # Remove x-axis title
+     axis.title.y = element_blank(),                        # Remove y-axis title
+     axis.text = element_blank(),                           # Remove axis text
+     axis.ticks = element_blank(),                          # Remove axis ticks
+     panel.grid = element_blank()                           # Remove grid lines
+   ) +
+   scale_fill_manual(values = c("Filtered Data (267)" = "red", "Other Data (5733)" = "lightblue"))
> |
```



Conclusion:

The graph above shows the final result of the Impact of Credit_Amount, Status_Of_Employment, Status_Of_Savings, and Credit_History on Good Credit Score. After analysing four columns with “good” class separately and applying all four conditions, it can be concluded that our original hypothesis is wrong. Having the result to be only at 267 out of 3000, we barely have an 8.9% of accuracy in our prediction. Because of the low impact on predicting a good credit score, this indicates that we need to reassess our initial conditions in our hypothesis. Either by replacing the columns with completely different ones or changing the conditions of the column, it will be necessary for further analysis to get our desired results in good credit score data.

3.4.5 Additional Features

1. **Dplyr package** – A package in R for data manipulation, providing functions like mutate(), filter(), select(), and more for data processing tasks.
2. **mutate()** – A function from the dplyr package that adds new variables or modifies existing variables in a data frame.
3. **ifelse()** – A function that tests a condition and returns one value if true and another if false, used for conditional transformations.
4. **filter()** – A dplyr function used to subset data based on specific conditions or criteria.
5. **%>%** – The pipe operator from the magrittr package (part of dplyr) that allows you to pass the output of one function as the input to the next function in a chain.
6. **geom_point** – A ggplot2 function used to create a scatter plot by adding points to the plot.
7. **labs()** – A function in ggplot2 to modify the labels of the plot, including title, x and y axes, and legend.
8. **theme_minimal()** – A function in ggplot2 that applies a minimalist theme to a plot by reducing most background elements.
9. **scale_fill_manual()** – A function in ggplot2 used to manually specify the colors for filling aesthetic mappings in a plot.
10. **coord_polar()** – A function in ggplot2 used to transform a Cartesian plot into polar coordinates (for example, in pie charts).
11. **theme_void()** – A function in ggplot2 that removes all background elements, grid lines, and axes, creating a clean, minimal plot.
12. **scale_x_continuous()** – A function in ggplot2 to control the x-axis scale, typically for continuous numeric variables.
13. **face = "bold"** – An argument used in element_text() to make the text bold.
14. **hjust** – An argument used in element_text() to control the horizontal alignment of text elements, where 0 is left, 0.5 is centered, and 1 is right-aligned.
15. **plot.title** – A reference to the title of the plot, which can be customized in theme() to adjust appearance.
16. **element_text()** – A function in ggplot2 used to modify text elements in a plot, such as font size, color, and style.
17. **axis.text.x** – A reference to the x-axis text labels, which can be customized using theme() to adjust the text appearance on the x-axis.

18. **angle** – An argument used to rotate text elements, such as axis labels, in element_text().
19. **geom_point** – A ggplot2 function used to add points to a plot, typically for scatter plots or other point-based visualizations.
20. **axis.text.y** – A reference to the y-axis text labels, which can be customized using theme() to adjust the appearance of the y-axis labels.
21. **theme()** – A function in ggplot2 used to customize non-data elements of a plot, such as axis labels, grid lines, and title.
22. **panel.grid** – A reference to the grid lines of the plot, which can be modified in theme() using panel.grid.major and panel.grid.minor.
23. **axis.ticks** – A reference to the ticks on the plot's axes, which can be customized (e.g., to remove ticks) using theme().
24. **category** – Refers to categorical data or a categorical aesthetic in a plot, often used to group data or define different visual elements based on category.
25. **geom_bar()** – creates bar charts to display the count of categorical data.

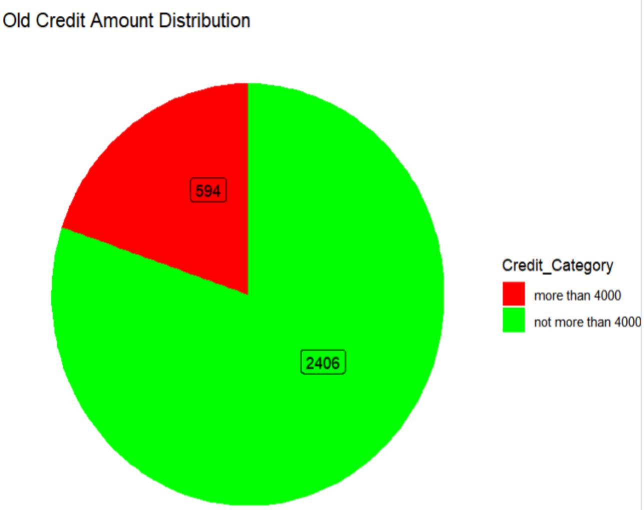
4.0 Additional Analysis to Improve Prediction Accuracy

4.1 Adjusting the objective "Credit_Amount(RM)<=4000" to "Credit_Amount(RM)<=6000"

For the original selection of credit amount <= 4000 it looked like the following:

```
Credit_Amount_data_old = mutate(good_data, Credit_Category =
                                  ifelse(`Credit_Amount(RM)` <= 4000,
                                         "not more than 4000","more than 4000"))

# summarize data
credit_old = summarise(
  group_by(Credit_Amount_data_old, Credit_Category),
  Count = n(), .groups = 'drop')

ggplot(credit_old, aes(x = "", y = Count, fill = Credit_Category)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y") +  # Convert bar chart to pie chart
  labs(title = "Old Credit Amount Distribution") +
  scale_fill_manual(values = c("not more than 4000" = "green", "more than 4000" = "red")) +
  geom_label_repel(aes(label = paste(Count)),
                   position = position_stack(vjust = 0.5),
                   show.legend = FALSE)+
  theme_void()
```

```
> ggplot(credit_old, aes(x = "", y = Count, fill = Credit_Category)) +
+   geom_bar(stat = "identity", width = 1) +
+   coord_polar("y") +  # Convert bar chart to pie chart
+   labs(title = "Old Credit Amount Distribution") +
+   scale_fill_manual(values = c("not more than 4000" = "green", "more than 4000" = "red")) +
+   geom_label_repel(aes(label = paste(Count)),
+                    position = position_stack(vjust = 0.5),
+                    show.legend = FALSE)+
+   theme_void()
```

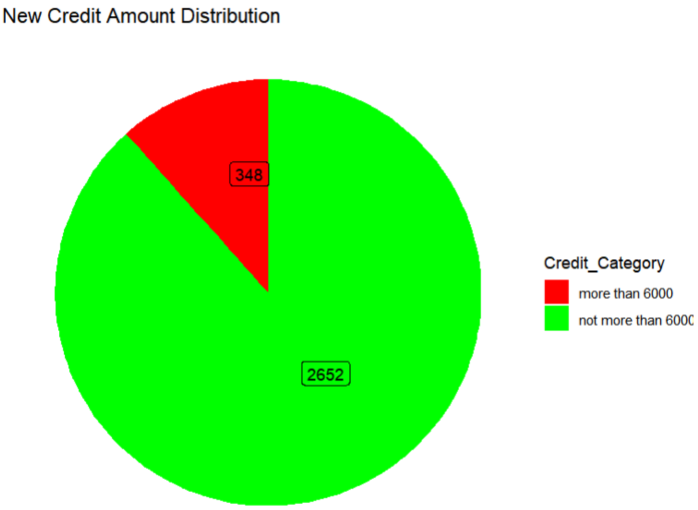


The original selection covered 2406 data which is approximately 80.2%. Since we felt that this was not enough, we decided to increase the scope of Credit_Amount(RM) to <= 6000. The impact can of this change can be seen below.

Following the decision, we adjusted our scope to have Credit_Amount(RM) be no more than 6000 instead of 4000 and ended up with the following:

```
Credit_Amount_data_new = mutate(good_data, Credit_Category =
                                  ifelse(`Credit_Amount(RM)` <= 6000,
                                         "not more than 6000","more than 6000"))
# summarize data
credit_new = summarise(
  group_by(Credit_Amount_data_new, Credit_Category),
  Count = n(), .groups = 'drop')

ggplot(credit_new, aes(x = "", y = Count, fill = Credit_Category)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y") +  # Convert bar chart to pie chart
  labs(title = "New Credit Amount Distribution") +
  scale_fill_manual(values = c("not more than 6000" = "green",
                               "more than 6000" = "red")) +
  geom_label_repel(aes(label = paste(Count)),
                   position = position_stack(vjust = 0.5),
                   show.legend = FALSE)+
  theme_void()
```

```
> ggplot(credit_new, aes(x = "", y = Count, fill = Credit_Category)) +
+   geom_bar(stat = "identity", width = 1) +
+   coord_polar("y") +  # Convert bar chart to pie chart
+   labs(title = "New Credit Amount Distribution") +
+   scale_fill_manual(values = c("not more than 6000" = "green",
+                                "more than 6000" = "red")) +
+   geom_label_repel(aes(label = paste(Count)),
+                    position = position_stack(vjust = 0.5),
+                    show.legend = FALSE)+
+   theme_void()
```



This is the result of the change. We can see that this change added 246 to the initial grouping of data which is the Credit Amount column.

Using the new change, this results in 2652 (~88.4%) data getting selected from 3000 instead of the original 2406 which is a good start.

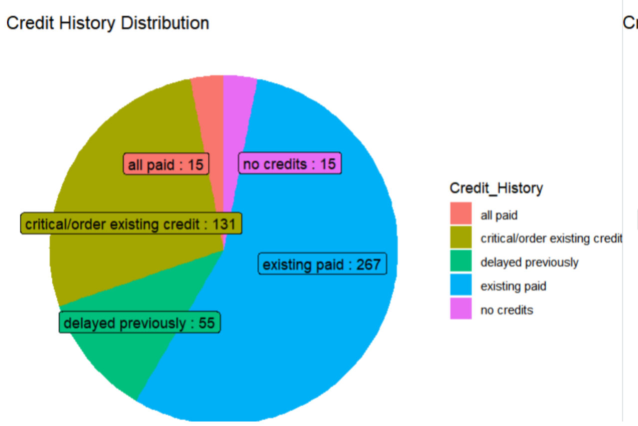
4.1.1 Impact on End Result

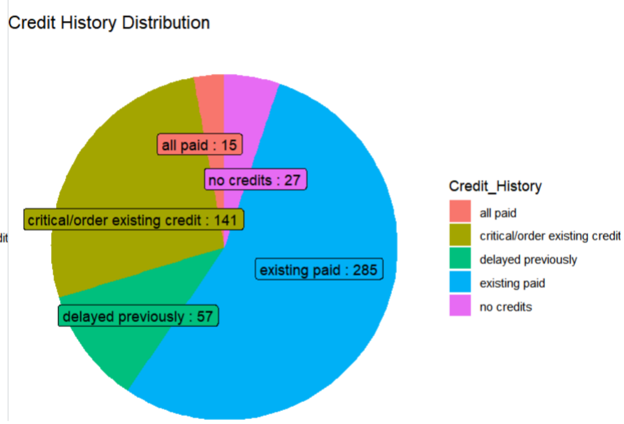
Following the first change, we can see the end result to be like the following:

```
good_data = filter(dataset, Class == "good")

good_data_after_credit_amount1 = filter(dataset, Class == "good", `Credit_Amount(RM)` <= 6000)

good_data_after_employment_credit_amount_and_savings1 =
  filter(good_data_after_credit_amount1,
         `Status_of_Employment(year_range)` == "1<=X<4",
         The_Statuses_of_Savings == "<100"
  )

credit_employment_savings_history1 <- good_data_after_employment_credit_amount_and_savings1 %>%
  group_by(Credit_History) %>%
  summarise(count = n(), .groups = 'drop')

ggplot(credit_employment_savings_history1, aes(x = "", y = count, fill = Credit_History)) +
  geom_bar(stat = "identity", width = 1) +
  geom_label_repel(aes(label = paste(Credit_History, ":", count)),
                   position = position_stack(vjust = 0.5),
                   show.legend = FALSE)+
  coord_polar("y") +  # Convert bar chart to pie chart
  labs(title = "Credit History Distribution") +
  theme_void()
```

```
> ggplot(credit_employment_savings_history1, aes(x = "", y = count, fill = Credit_History)) +
+   geom_bar(stat = "identity", width = 1) +
+   geom_label_repel(aes(label = paste(Credit_History, ":", count)),
+                    position = position_stack(vjust = 0.5),
+                    show.legend = FALSE)+
+   coord_polar("y") +  # Convert bar chart to pie chart
+   labs(title = "Credit History Distribution") +
+   theme_void()
```

Old **New**



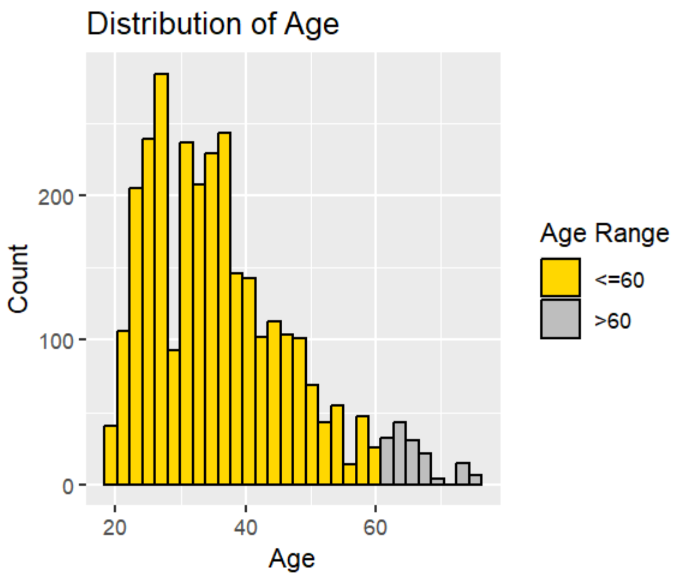


From this we can see that we managed to increase the result from 267 to 285 which is a good start to our adjustments, but the most impactful changes will be coming up in the next few changes.

4.2 Age <= 60 as Second Prediction Factor

```
#Histogram for Age
ggplot(good, aes(x = Age)) +
  geom_histogram(color = "black", aes(fill = ifelse(Age <= 60, "<=60", ">60"))) +
  scale_fill_manual(values = c("<=60" = "gold", ">60" = "grey")) +
  labs(fill = "Age Range") +
  labs(title = "Distribution of Age",
       x = "Age",
       y = "Count")

#Chi-square to see if the distribution is even
observed = table(good$Age)
chisq.test(observed, p = rep(1/length(observed), length(observed)))
```

```
> #Histogram for Age
> ggplot(good, aes(x = Age)) +
+   geom_histogram(color = "black", aes(fill = ifelse(Age <= 60, "<=60", ">60"))) +
+   scale_fill_manual(values = c("<=60" = "gold", ">60" = "grey")) +
+   labs(fill = "Age Range") +
+   labs(title = "Distribution of Age",
+        x = "Age",
+        y = "Count")
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
> #Chi-square to see if the distribution is even
> observed = table(good$Age)
> chisq.test(observed, p = rep(1/length(observed), length(observed)))

        Chi-squared test for given probabilities

data:  observed
X-squared = 1828, df = 52, p-value < 2.2e-16

> 
```

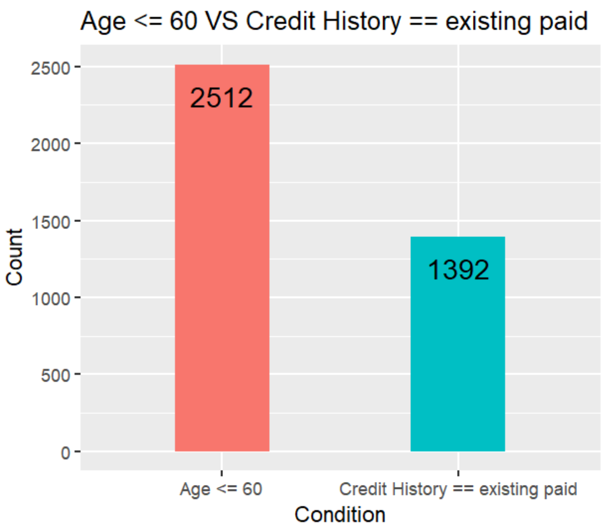


Looking at the Age column, which was never analyzed before, we realized that this column has great potential. Using the Chi-square test, the p value is $2.2e^{-16}$ (0.00000000000000022), which is far lower than 0.5. This outcome indicates that the distribution of data in this column is not evenly distributed. This is also proven by looking at the shape of the graph, as it spiked to the highest at the left, and slowly decline towards the right side. After thoughtful discussion, we decided to set the condition of this column to “Age <= 60” to maximize data prediction accuracy.

4.2.1 Replacing Credit History == “existing paid” with Age <= 60

```
#Class = "good", Credit Amount <= 6000, Credit History == "existing paid"
good_ca_ch = nrow(good[(good$`Credit_Amount(RM)`<=6000) &
                        (good$Credit_History=="existing paid"), ])
good_ca_ch

#Class = "good", Credit Amount <= 6000, Age <= 60
good_ca_age = nrow(good[(good$`Credit_Amount(RM)`<=6000) &
                         (good$Age<=60), ])
good_ca_age

#Difference between the two
ch_vs_age = data.frame(Category = c("Credit History == existing paid", "Age <= 60"),
                       Count = c(good_ca_ch, good_ca_age))
ggplot(ch_vs_age, aes(x = Category, y = Count, fill = Category)) +
  geom_bar(stat = "identity", width = 0.4) +
  geom_text(aes(label = Count), vjust =2, color = "black", size = 5) +
  labs(title = "Age <= 60 VS Credit History == existing paid",
       x = "Condition",
       y = "Count") +
  theme(legend.position = "none")
```

```
> #Class = "good", Credit Amount <= 6000, Credit History == "existing paid"
> good_ca_ch = nrow(good[(good$`Credit_Amount(RM)`<=6000) &
+                         (good$Credit_History=="existing paid"), ])
> good_ca_ch
[1] 1392
> #Class = "good", Credit Amount <= 6000, Age <= 60
> good_ca_age = nrow(good[(good$`Credit_Amount(RM)`<=6000) &
+                          (good$Age<=60), ])
> good_ca_age
[1] 2512
> #Difference between the two
> ch_vs_age = data.frame(Category = c("Credit History == existing paid", "Age <= 60"),
+                        Count = c(good_ca_ch, good_ca_age))
> ggplot(ch_vs_age, aes(x = Category, y = Count, fill = Category)) +
+   geom_bar(stat = "identity", width = 0.4) +
+   geom_text(aes(label = Count), vjust =2, color = "black", size = 5) +
+   labs(title = "Age <= 60 VS Credit History == existing paid",
+        x = "Condition",
+        y = "Count") +
+   theme(legend.position = "none")
>
```

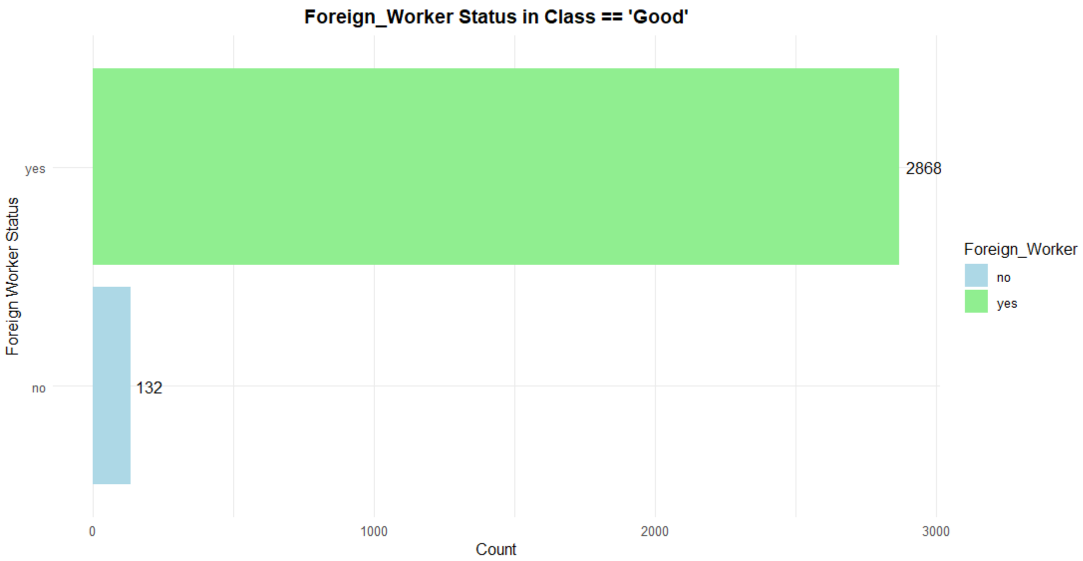


By replacing Credit History == “existing paid” in the original hypothesis with Age <= 60, the prediction accuracy increases by almost twofold – from 46.40% (1392/3000) to 83.73% (2512/3000). Hence, Age <= 60 will continue to serve as the second factor for our prediction.

4.3 Foreign_Worker == "yes" as Third Prediction Factor

```r
#Foreign_Worker
# Load necessary library
library(ggplot2)

# Count rows where Foreign_Worker == "yes" and Class == "good"
count_yes <- nrow(dataset[dataset$Foreign_Worker == "yes" & dataset$Class == "good", ]) #2868

# Count rows where Foreign_Worker == "no" and Class == "good"
count_no <- nrow(dataset[dataset$Foreign_Worker == "no" & dataset$Class == "good", ]) #132

# Create a data frame with the counts
counts_df <- data.frame(
  Foreign_Worker = c("yes", "no"),
  Count = c(count_yes, count_no)
)

# Display the counts
print(counts_df)

# Create a horizontal bar chart
ggplot(data = counts_df, aes(x = Count, y = Foreign_Worker, fill = Foreign_Worker)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = Count), hjust = -0.3) +
  labs(title = "Count of Foreign Worker Status in Good Credit Class", x = "Count", y = "Foreign Worker Status") +
  scale_fill_manual(values = c("lightblue", "lightgreen")) +
  theme_minimal()
```

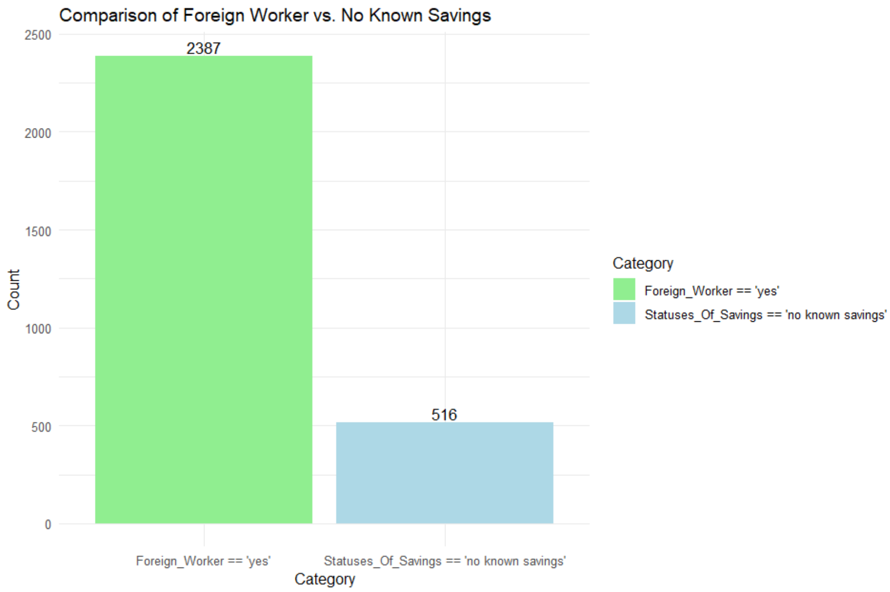
```
> #Foreign_Worker
> # Load necessary library
> library(ggplot2)
>
> # Count rows where Foreign_Worker == "yes" and Class == "good"
> count_yes <- nrow(dataset[dataset$Foreign_Worker == "yes" & dataset$Class == "good", ]) #2868
>
> # Count rows where Foreign_Worker == "no" and Class == "good"
> count_no <- nrow(dataset[dataset$Foreign_Worker == "no" & dataset$Class == "good", ]) #132
>
> # Create a data frame with the counts
> counts_df <- data.frame(
+   Foreign_Worker = c("yes", "no"),
+   Count = c(count_yes, count_no)
+ )
>
> # Display the counts
> print(counts_df)
  Foreign_Worker Count
1            yes  2868
2             no   132
>
> # Create a horizontal bar chart
> ggplot(data = counts_df, aes(x = Count, y = Foreign_Worker, fill = Foreign_Worker)) +
+   geom_bar(stat = "identity") +
+   geom_text(aes(label = Count), hjust = -0.3) +
+   labs(title = "Count of Foreign Worker Status in Good Credit Class", x = "Count", y = "Foreign Worker Status") +
+   scale_fill_manual(values = c("lightblue", "lightgreen")) +
+   theme_minimal()
> |
```



As we can see from the graph, the column "Foreign_Worker" has a notable impact on achieving good credit score. With a result of 2868 in the "good" class, it is clear that this column strongly contributes to increasing the result. Although this column has not been analyzed, the data could play an important role in our final results.

4.3.1 Replacing Statuses_Of_Savings == “no known savings” with Foreign_Worker == "yes"

```
# Load necessary library
library(ggplot2)

# Count for Foreign_Worker and Statuses_Of_Savings
count_foreign_worker <- nrow(dataset[dataset$Class == "good" & dataset$`Credit_Amount(RM)` <= 6000
                                     & dataset$Age <= 60 & dataset$Foreign_Worker == "yes", ])
count_savings_status <- nrow(dataset[dataset$Class == "good" & dataset$`Credit_Amount(RM)` <= 6000
                                     & dataset$Age <= 60 & dataset$The_Statuses_of_Savings == "no known savings", ])

# Create data frame with the counts
count_data <- data.frame(
  Category = c("Foreign_Worker == 'yes'", "Statuses_Of_Savings == 'no known savings'"),
  Count = c(count_foreign_worker, count_savings_status)
)

# Create the bar chart
ggplot(count_data, aes(x = Category, y = Count, fill = Category)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = Count), vjust = -0.2, size = 4) +
  labs(title = "Comparison of Foreign Worker vs. No Known Savings",
       x = "Category",
       y = "Count") +
  scale_fill_manual(values = c("Foreign_Worker == 'yes'" = "lightgreen",
                               "Statuses_Of_Savings == 'no known savings'" = "lightblue")) +
  theme_minimal()
> # Load necessary library
> library(ggplot2)
>
> # Count for Foreign_Worker and Statuses_Of_Savings
> count_foreign_worker <- nrow(dataset[dataset$Class == "good" & dataset$`Credit_Amount(RM)` <= 6000
+                                      & dataset$Age <= 60 & dataset$Foreign_Worker == "yes", ])
> count_savings_status <- nrow(dataset[dataset$Class == "good" & dataset$`Credit_Amount(RM)` <= 6000
+                                      & dataset$Age <= 60 & dataset$The_Statuses_of_Savings == "no known savings", ])
>
> # Create data frame with the counts
> count_data <- data.frame(
+   Category = c("Foreign_Worker == 'yes'", "Statuses_Of_Savings == 'no known savings'"),
+   Count = c(count_foreign_worker, count_savings_status)
+ )
>
> # Create the bar chart
> ggplot(count_data, aes(x = Category, y = Count, fill = Category)) +
+   geom_bar(stat = "identity") +
+   geom_text(aes(label = Count), vjust = -0.2, size = 4) +
+   labs(title = "Comparison of Foreign Worker vs. No Known Savings",
+        x = "Category",
+        y = "Count") +
+   scale_fill_manual(values = c("Foreign_Worker == 'yes'" = "lightgreen",
+                                "Statuses_Of_Savings == 'no known savings'" = "lightblue")) +
+   theme_minimal()
> |
```



From the graph, we can see that the count for Foreign_Worker == "yes" is much higher, with a result of 2387, compared to the Statuses_Of_Savings == "no known savings", only having 516 rows in the "good" credit score. Hence, it is most suitable to replace Statuses_Of_Savings with Foreign_Worker as it will give higher accuracy rate of our analysis conducted.

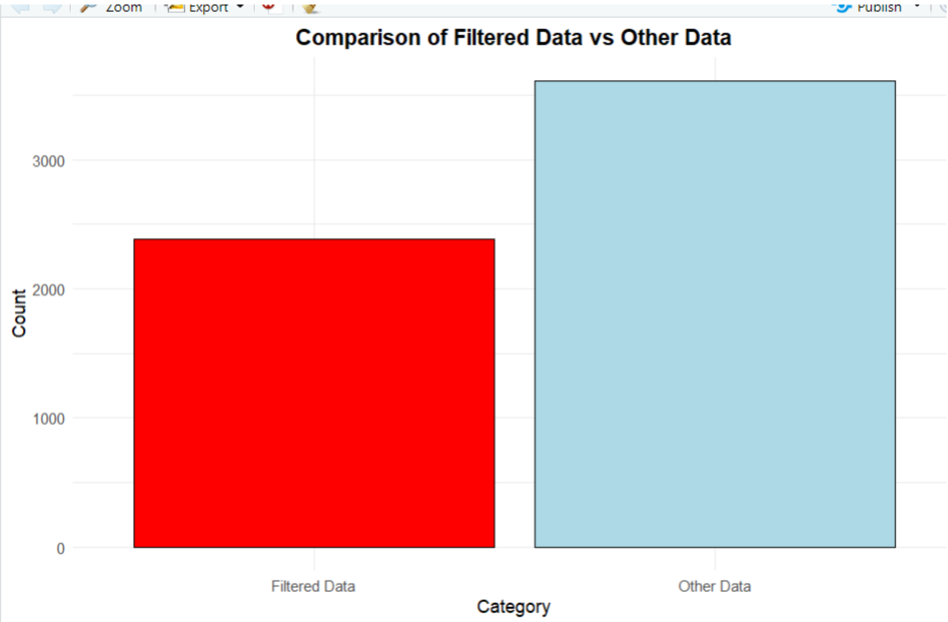
4.4 Status_of_Employment != "Unemployed"

```
# Count the number of records that meet the criteria
count <- nrow(dataset[(dataset$Class == "good")
                      & (dataset$`Credit_Amount(RM)` <= 6000)
                      & (dataset$`Age` <= 60)
                      & (dataset$Foreign_Worker == "yes")
                      & (dataset$`Status_of_Employment(year_range)` != "unemployed"),])

print(count)
```

After applying the new combination of columns (`Class == "good"`, `Credit Amount (RM) <= 6000`, `Foreign Worker == "yes"`, Age <=60 and `Status of Employment != "unemployed"`), we obtained a significantly better result than previous attempts. The new criteria yielded 2294 records that meet these conditions, indicating a much larger subset of "good" credit cases.

```
> # Count the number of records that meet the criteria
> count <- nrow(dataset[(dataset$Class == "good")
+                       & (dataset$`Credit_Amount(RM)` <= 6000)
+                       & (dataset$`Age` <= 60)
+                       & (dataset$Foreign_Worker == "yes")
+                       & (dataset$`Status_of_Employment(year_range)` != "unemployed"),])
>
> print(count)
[1] 2294
```

```
# Data frame for the bar chart
data_for_bar_chart <- data.frame(
  Category = c("Filtered Data", "Other Data"),
  Count = c(count, nrow(dataset) - count)
)

# Create the bar chart with ggplot2
library(ggplot2)
bar_chart <- ggplot(data_for_bar_chart, aes(x = Category, y = Count, fill = Category)) +
  geom_bar(stat = "identity", color = "black") +
  labs(title = "Comparison of Filtered Data vs Other Data",
       x = "Category",
       y = "Count") +
  scale_fill_manual(values = c("Filtered Data" = "red", "Other Data" = "lightblue")) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
    legend.position = "none"
  )

# Display the bar chart
print(bar_chart)
```



This outcome suggests that this combination of factors may be more effective in predicting a "good" credit score. By visualizing these results in a bar chart, we can clearly see the increased count of records that fit the new conditions compared to other data. This improvement supports our hypothesis and shows potential for a more accurate model for identifying customers with good credit scores.

The new conclusion will be:

Class == good

Credit amount <= 6000

Age <= 60

Foreign worker == yes

Status of Employment != unemployed

Using these adjusted conditions could ultimately lead to more reliable predictions, aligning closer to our goal of achieving a predictive model with greater than 70% accuracy.

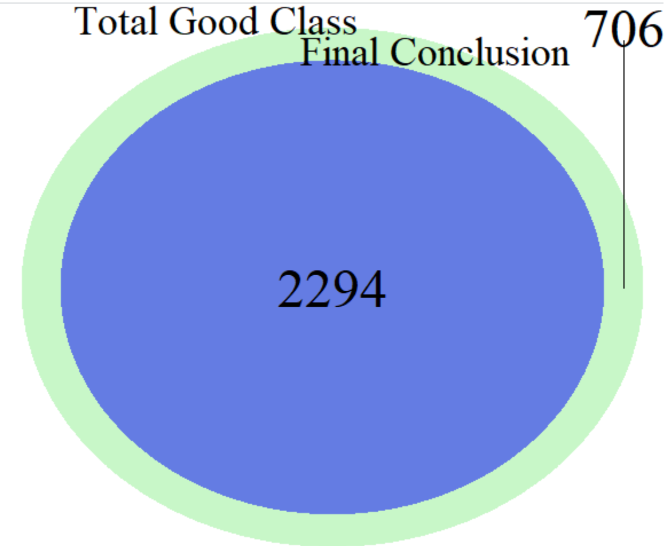
4.5 Additional Features

1. **chiq.test()** – Performs chi-squared contingency table tests and goodness-of-fit tests.

5.0 Final Conclusion

```
#Final Conclusion
final_conclusion = nrow(good[(good$`Credit_Amount(RM)`<=6000) &
                              (good$Age <= 60) &
                              (good$Foreign_Worker == "yes") &
                              (good$`Status_of_Employment(year_range)`!= "unemployed"), ])
final_conclusion

#Venn Diagram for final conclusion
draw.pairwise.venn(
  area1 = 3000,          # Total "good" class
  area2 = final_conclusion,
  cross.area = final_conclusion,
  category = c("Total Good Class", "Final Conclusion"),
  fill = c("lightgreen", "blue"),
  lty = "blank",
  cex = 2, cat.cex = 1.5, cat.pos = c(-20, 20), cat.dist = 0.02, scaled = TRUE
)
```

```
> #Final Conclusion
> final_conclusion = nrow(good[(good$`Credit_Amount(RM)`<=6000) &
+                              (good$Age <= 60) &
+                              (good$Foreign_Worker == "yes") &
+                              (good$`Status_of_Employment(year_range)`!= "unemployed"), ])
> final_conclusion
[1] 2294
> #Venn Diagram for final conclusion
> draw.pairwise.venn(
+   area1 = 3000,          # Total "good" class
+   area2 = final_conclusion,
+   cross.area = final_conclusion,
+   category = c("Total Good Class", "Final Conclusion"),
+   fill = c("lightgreen", "blue"),
+   lty = "blank",
+   cex = 2, cat.cex = 1.5, cat.pos = c(-20, 20), cat.dist = 0.02, scaled = TRUE
+ )
(polygon[GRID.polygon.154], polygon[GRID.polygon.155], polygon[GRID.polygon.156], polygon[GRID.polygon.1
57], text[GRID.text.158], text[GRID.text.159], lines[GRID.lines.160], text[GRID.text.161], text[GRID.tex
t.162])
>
```



After extensive analysis, we managed to identify columns with low impact as well as columns with significant impact to the amount of selected data as substitution. The final conclusion that we've made is customers with **Credit Amount less or equals to RM6000, at the Age of at most 60 years old, is a Foreign Worker, and not unemployed** will have **good** credit score. With this new conclusion, our credit score prediction model managed to achieve 76.47% accuracy.

6.0 References

Alboukadel. (n.d., November 13). *GGPlot Legend Title, Position and Labels*. Datanovia. https://www.datanovia.com/en/blog/ggplot-legend-title-position-and-labels/

Bobbitt, Z. (2022, August 11). *How to Use hjust & vjust to Move Elements in ggplot2*. Statology. https://www.statology.org/hjust-vjust-ggplot2/

chisq.test: Pearson's Chi-squared Test for Count Data. (n.d.). RDocumentation. https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/chisq.test

draw.pairwise.venn: Draw a Venn Diagram with Two Sets. (n.d.). RDocumentation. https://www.rdocumentation.org/packages/VennDiagram/versions/1.7.3/topics/draw.pairwise.venn

ggplot2. Create Elegant Data Visualisations Using the Grammar of Graphics. (n.d.). https://ggplot2.tidyverse.org/

ggplot2 axis titles, labels, ticks, limits and scales. (n.d.). R CHARTS. https://r-charts.com/ggplot2/axis/

Pie Charts in R. (n.d.). DataCamp. https://www.datacamp.com/doc/r/pie