Bank Personal Loan Modelling

Assel_Kassenova

5/4/2023

Part 1.Data preparation& Exploration

```
# importing libraries
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.2     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.2     ✔ tibble    3.2.1
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(dplyr)
library(ggplot2)
library(rpart)
library(arules)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
##
## Attaching package: 'arules'
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
library(cluster)
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
# imorting dataset
df <- read_csv("Bank_Personal_loan_Modelling.csv")
```

```
## Rows: 5000 Columns: 14
## ── Column specification ────────────────────────────────────────────────────────────────────────────────────────────────────────────────
—
## Delimiter: ","
## dbl (14): ID_Customer, Age, Experience, Income, ZIP Code, Family, CCAvg, Edu...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Information about dataset: ID_Customer: Customer ID Age: Customer's age in completed years Experience: #years of professional experience Income: Annual income of the customer (in thousand dollars) ZIP Code: Home Address ZIP code. Family: the Family size of the customer CCAvg: Average spending on credit cards per month (in thousand dollars) Education:Education Level. 1: Undergrad; 2: Graduate;3: Advanced/Professional Mortgage: Value of house mortgage if any. (in thousand dollars) Personal Loan: Did this customer accept the personal loan offered in the last campaign?

Explorind Data

```
# NA values
sum(is.na(df))
```

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

```
#dublicates
sum(duplicated(df))
```

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

```
# summary statistics
summary(df)
```

```
##   ID_Customer       Age         Experience       Income         ZIP Code
##  Min.   :   1   Min.   :23.00   Min.   :-3.0   Min.   :  8.00   Min.   : 9307
##  1st Qu.:1251   1st Qu.:35.00   1st Qu.:10.0   1st Qu.: 39.00   1st Qu.:91911
##  Median :2500   Median :45.00   Median :20.0   Median : 64.00   Median :93437
##  Mean   :2500   Mean   :45.34   Mean   :20.1   Mean   : 73.77   Mean   :93152
##  3rd Qu.:3750   3rd Qu.:55.00   3rd Qu.:30.0   3rd Qu.: 98.00   3rd Qu.:94608
##  Max.   :5000   Max.   :67.00   Max.   :43.0   Max.   :224.00   Max.   :96651
##      Family          CCAvg          Education       Mortgage
##  Min.   :1.000   Min.   : 0.000   Min.   :1.000   Min.   :  0.0
##  1st Qu.:1.000   1st Qu.: 0.700   1st Qu.:1.000   1st Qu.:  0.0
##  Median :2.000   Median : 1.500   Median :2.000   Median :  0.0
##  Mean   :2.396   Mean   : 1.938   Mean   :1.881   Mean   : 56.5
##  3rd Qu.:3.000   3rd Qu.: 2.500   3rd Qu.:3.000   3rd Qu.:101.0
##  Max.   :4.000   Max.   :10.000   Max.   :3.000   Max.   :635.0
##  Personal Loan   Securities Account   CD Account       Online
##  Min.   :0.000   Min.   :0.0000     Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.000   1st Qu.:0.0000     1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.000   Median :0.0000     Median :0.0000   Median :1.0000
##  Mean   :0.096   Mean   :0.1044     Mean   :0.0604   Mean   :0.5968
##  3rd Qu.:0.000   3rd Qu.:0.0000     3rd Qu.:0.0000   3rd Qu.:1.0000
##  Max.   :1.000   Max.   :1.0000     Max.   :1.0000   Max.   :1.0000
##    CreditCard
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.294
##  3rd Qu.:1.000
##  Max.   :1.000
```

1. This dataset contain no NA values.
2. Summary of this datasets shows that variable Experience has value "-3". Incorrect value.
3. Zip Code is numeric value. we might have to delete it after descriptive analytics

Preparing Data

```
# replacing space in column names  by "_"
colnames(df) <- gsub(" ", "_", colnames(df))

# data cleaning
df <- df[df$Experience >= 0,] # removing all values less than 0 in Experience
```

Part 2.Descriptive Analytics

```
library(flexdashboard)
library(tidyverse)
library(highcharter)
```

```
## Highcharts (www.highcharts.com) is a Highsoft software product which is
```

```
## not free for commercial and Governmental use
```

```
library(gt)
library(htmltools)
```

Counts of Personal Loan, Mortgage, Securities Account, Online, and Credit Card

```
df %>%
  group_by(Personal_Loan, Mortgage, Securities_Account, Online, CreditCard) %>%
  summarise(count = n()) %>%
  ungroup() %>%
  gather(key = "variable", value = "value", -count) %>%
  mutate(value = ifelse(value == 1, "Accepted Loan", "Did Not Accept Loan")) %>%
  hchart(type = "column", hcaes(x = variable, y = count, group = factor(value), color = variable)) %>%
  hc_plotOptions(column = list(stacking = FALSE)) %>%
  hc_legend(enabled = TRUE) %>%
  hc_xAxis(title = list(text = "Variable")) %>%
  hc_yAxis(title = list(text = "Count")) %>%
  hc_title(text = "Counts of Personal Loan, Mortgage, Securities Account, Online, and Credit Card") %>%
  hc_colors(list('#f15c80', '#7cb5ec'))
```

```
## `summarise()` has grouped output by 'Personal_Loan', 'Mortgage',
## 'Securities_Account', 'Online'. You can override using the `.groups` argument.
```

Personal Loan Holders VS Securities Account and Credit Card Status

```
df %>%
  group_by(Personal_Loan, Securities_Account, CreditCard) %>%
  summarise(count = n()) %>%
  ungroup() %>%
  hchart(type = "column", hcaes(x = factor(ifelse(Securities_Account == 1 & CreditCard == 1, "Have Both", ifelse(Securities_Account == 1, "Have Securitie
s Account", ifelse(CreditCard == 1, "Have Credit Card", "Have None")))), y = count, group = factor(ifelse(Personal_Loan == 1, "Accepted Loan", "Did Not A
ccept Loan"), levels = c("Accepted Loan", "Did Not Accept Loan")))) %>%
  hc_plotOptions(column = list(stacking = FALSE)) %>%
  hc_legend(enabled = TRUE) %>%
  hc_xAxis(title = list(text = "Securities Account and Credit Card Holders")) %>%
  hc_yAxis(title = list(text = "Count")) %>%
  hc_title(text = "Counts of Personal Loan Holders based on Securities Account and Credit Card Status")
```

```
## `summarise()` has grouped output by 'Personal_Loan', 'Securities_Account'. You
## can override using the `.groups` argument.
```

Mortgage Holders VS Securities Account and Credit Card Status

```
df %>%
  group_by(Mortgage, Securities_Account, CreditCard) %>%
  summarise(count = n()) %>%
  ungroup() %>%
  hchart(type = "column", hcaes(x = factor(ifelse(Securities_Account == 1 & CreditCard == 1, "Have Both", ifelse(Securities_Account == 1, "Have Securitie
s Account", ifelse(CreditCard == 1, "Have Credit Card", "Have None")))), y = count, group = factor(ifelse(Mortgage == 0,"Don't have Mortgage", "Have Mor
tgage"), levels = c("Don't have Mortgage", "Have Mortgage")))) %>%
  hc_plotOptions(column = list(stacking = FALSE)) %>%
  hc_legend(enabled = TRUE) %>%
  hc_xAxis(title = list(text = "Securities Account and Credit Card Holders")) %>%
  hc_yAxis(title = list(text = "Count")) %>%
  hc_title(text = "Counts of Mortgage Holders based on Securities Account and Credit Card Status")
```

```
## `summarise()` has grouped output by 'Mortgage', 'Securities_Account'. You can
## override using the `.groups` argument.
```

Personal Loan Status based on Income

```
df %>%
  group_by(Personal_Loan, Income) %>%
  summarise(count = n()) %>%
  hchart(type = "column", hcaes(x = Income, y = count, group = factor(ifelse(Personal_Loan == 0, "Did Not Accepted Loan", "Accepted Loan"), levels = c("
Did Not Accepted Loan", "Accepted Loan" )))) %>%
  hc_plotOptions(column = list(stacking = FALSE)) %>%
  hc_legend(enabled = TRUE) %>%
  hc_xAxis(title = list(text = "Income Category")) %>%
  hc_yAxis(title = list(text = "Count")) %>%
  hc_title(text = "Personal Loan Status based on Income")
```

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

Education and Personal Loan

```
df %>%
  group_by(Education, Personal_Loan = factor(ifelse(Personal_Loan == 0, "Did Not Accepted Loan", "Accepted Loan"), levels = c( "Did Not Accepted Loan
", "Accepted Loan"))) %>%
  summarise(n = n()) %>%
  group_by(Education) %>%
  mutate(pct = n/sum(n)*100) %>%
  ungroup() %>%
  hchart(type = "column", hcaes(x = Education, y = pct, group = Personal_Loan), colorByPoint = TRUE) %>%
  hc_title(text = "Mortgage by Education") %>%
  hc_yAxis(title = "Percentage", labels = list(format = "{value}%")) %>%
  hc_tooltip(pointFormat = "<b>{point.y:.1f}%</b><br>") %>%
  hc_plotOptions(column = list(stacking = "normal")) %>%
  hc_legend(layout = "horizontal", align = "center", verticalAlign = "bottom")
```

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

Personal Loan by Family

```
df %>%
  group_by(Family, Personal_Loan = factor(ifelse(Personal_Loan == 0, "Did Not Accept Loan", "Accepted Loan"), levels = c("Did Not Accept Loan", "Accep
ted Loan"))) %>%
  summarise(n = n()) %>%
  group_by(Family) %>%
  mutate(pct = n/sum(n)*100) %>%
  ungroup() %>%
  hchart(type = "column", hcaes(x = Family, y = pct, group = Personal_Loan), colorByPoint = TRUE) %>%
  hc_title(text = "Personal Loan by Education") %>%
  hc_yAxis(title = "Percentage", labels = list(format = "{value}%")) %>%
  hc_tooltip(pointFormat = "<b>{point.y:.1f}%</b><br>") %>%
  hc_plotOptions(column = list(stacking = "normal")) %>%
  hc_legend(layout = "horizontal", align = "center", verticalAlign = "bottom")
```

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

Personal Loan by Age

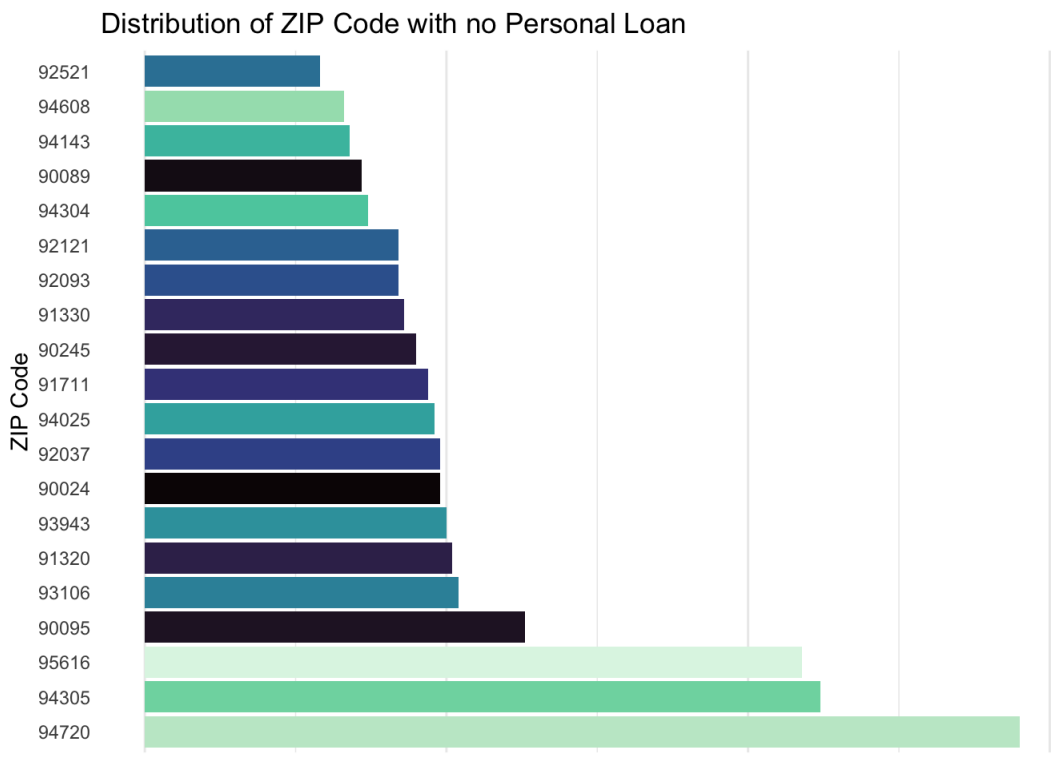
```
df %>%
  group_by(Personal_Loan, Age) %>%
  summarise(count = n()) %>%
  group_by(Age) %>%
  mutate(total = sum(count),
         percentage = paste0(round(100 * count/total, 2), "%")) %>%
  hchart(type = "column", hcaes(x = Age, y = count, group = factor(ifelse(Personal_Loan == 0, "Did Not Accept Loan", "Accepted Loan"), levels = c( "Did N
ot Accept Loan", "Accepted Loan")))) %>%
  hc_plotOptions(column = list(stacking = FALSE)) %>%
  hc_legend(enabled = TRUE) %>%
  hc_xAxis(title = list(text = "Income Category")) %>%
  hc_yAxis(title = list(text = "Percentage")) %>%
  hc_tooltip(pointFormat = "<b>{point.percentage}</b>") %>%
  hc_title(text = "Personal Loan Status based on Income")
```

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

Distribution of ZIP Code with no Personal Loan

```
df$ZIP_Code <- as.factor(df$ZIP_Code)

# subset data for Mortgage == 0
df_subset <- subset(df, Personal_Loan == 0)

# order ZIP Code by count of Mortgage == 0
zip_order <- as.character(names(sort(table(df_subset$ZIP_Code), decreasing = TRUE)))[1:20]

# plot count of ZIP Code
ggplot(df_subset, aes(y = ZIP_Code, fill = ZIP_Code)) +
  geom_bar() +
  scale_fill_viridis_d(option = "mako") +
  scale_y_discrete(limits = zip_order) +
  theme_minimal() +
  theme(legend.position = "none",
        axis.text.x = element_blank(),
        axis.title.x = element_blank(),
        axis.line.y = element_blank(),
        axis.ticks.y = element_blank(),
        panel.grid.major.y = element_blank(),
        panel.grid.minor.y = element_blank()) +
  labs(title = "Distribution of ZIP Code with no Personal Loan", style = list(fontSize='25px',
                                                                              fontWeight= 'bold'),
       y = "ZIP Code")
```

```
## Warning: Removed 3352 rows containing non-finite values (`stat_count()`).
```



The rest of the Descriptive analytics can be on dasboard following link: “https://ad699.netlify.app/”

#Part 3.Building regression and classifier

```
df <- df %>%
  filter(Experience>=0)
df2 <- df %>%
  select(Age,Experience,Income,Family,CCAvg,Education,Mortgage,
         Securities_Account,CD_Account,Personal_Loan)
```

Online is removed because its an unspecified variable.

```
df$ZIP_Code <- as.numeric(df$ZIP_Code)
cor(df)
```

```
##                    ID_Customer          Age    Experience       Income
## ID_Customer         1.000000000 -0.0097943369 -0.0094055729 -0.017255228
## Age                -0.009794337  1.0000000000  0.9941014890 -0.058005788
## Experience         -0.009405573  0.9941014890  1.0000000000 -0.049244828
## Income             -0.017255228 -0.0580057884 -0.0492448284  1.000000000
## ZIP_Code            0.001980611 -0.0303859820 -0.0301130130 -0.026820677
## Family             -0.016382364 -0.0392786041 -0.0456099542 -0.155665764
## CCAvg              -0.025607184 -0.0508788729 -0.0489389617  0.646178154
## Education           0.021902227  0.0462218656  0.0182432552 -0.187992205
## Mortgage           -0.011171557 -0.0151839719 -0.0134592936  0.206920864
## Personal_Loan      -0.025123894 -0.0142035607 -0.0141208097  0.504227656
## Securities_Account -0.018960797  0.0004958185 -0.0004570698 -0.002327176
## CD_Account         -0.007084957  0.0032612003  0.0054504754  0.170170907
## Online             -0.001508087  0.0135424935  0.0135184842  0.014432567
## CreditCard          0.017293905  0.0074956879  0.0088760955 -0.004493256
##                     ZIP_Code       Family       CCAvg    Education
## ID_Customer         0.0019806113 -0.016382364 -0.025607184  0.021902227
## Age                -0.0303859820 -0.039278604 -0.050878873  0.046221866
## Experience         -0.0301130130 -0.045609954 -0.048938962  0.018243255
## Income             -0.0268206774 -0.155665764  0.646178154 -0.187992205
## ZIP_Code            1.0000000000  0.024633570 -0.010972918 -0.010510145
## Family              0.0246335703  1.000000000 -0.107230226  0.064032351
## CCAvg              -0.0109729177 -0.107230226  1.000000000 -0.133939348
## Education          -0.0105101451  0.064032351 -0.133939348  1.000000000
## Mortgage            0.0064930538 -0.020418837  0.109904537 -0.032558771
## Personal_Loan      -0.0007335735  0.063087978  0.369387930  0.138339036
## Securities_Account  0.0032386770  0.020155206  0.012476822 -0.007508034
## CD_Account          0.0215460184  0.015273754  0.137586572  0.014638931
## Online              0.0311028342  0.008465713 -0.003476440 -0.013932195
## CreditCard          0.0223712519  0.012905354 -0.007376833 -0.012604345
##                     Mortgage Personal_Loan Securities_Account   CD_Account
## ID_Customer        -0.011171557 -0.0251238943      -0.0189607971 -0.007084957
## Age                -0.015183972 -0.0142035607       0.0004958185  0.003261200
## Experience         -0.013459294 -0.0141208097      -0.0004570698  0.005450475
## Income              0.206920864  0.5042276560      -0.0023271758  0.170170907
## ZIP_Code            0.006493054 -0.0007335735       0.0032386770  0.021546018
## Family             -0.020418837  0.0630879776       0.0201552065  0.015273754
## CCAvg               0.109904537  0.3693879296       0.0124768215  0.137586572
## Education          -0.032558771  0.1383390364      -0.0075080343  0.014638931
## Mortgage            1.000000000  0.1423218419      -0.0037171431  0.089368528
## Personal_Loan       0.142321842  1.0000000000       0.0222158623  0.315769398
## Securities_Account -0.003717143  0.0222158623       1.0000000000  0.319055820
## CD_Account          0.089368528  0.3157693984       0.3190558203  1.000000000
## Online             -0.006754050  0.0061751844       0.0161014704  0.176767839
## CreditCard         -0.006909691  0.0027757292      -0.0170302693  0.280150625
##                     Online   CreditCard
## ID_Customer        -0.001508087  0.017293905
## Age                 0.013542493  0.007495688
## Experience          0.013518484  0.008876095
## Income              0.014432567 -0.004493256
## ZIP_Code            0.031102834  0.022371252
## Family              0.008465713  0.012905354
## CCAvg              -0.003476440 -0.007376833
## Education          -0.013932195 -0.012604345
## Mortgage           -0.006754050 -0.006909691
## Personal_Loan       0.006175184  0.002775729
## Securities_Account  0.016101470 -0.017030269
## CD_Account          0.176767839  0.280150625
## Online              1.000000000  0.008457430
## CreditCard          0.008457430  1.000000000
```

Correlation coefficients whose magnitude are between 0.7 and 1.0 indicate variables which can be considered highly correlated. Age and experience are also highly correlated because older people are normally have more working experience. So experience can be removed. Also Zipcode can be removed because its correlation with the personal loan is very low.

```
df2 <- df2 %>%
  select(Age,Income,Family,CCAvg,Education,Mortgage,
         Securities_Account,CD_Account,Personal_Loan)
```

```
set.seed(699)
df2 <- slice_sample(df2, prop =1)
train <- slice_head(df2, prop = 0.6)
valid <- slice_tail(df2, prop = 0.4)
```

Muitiple Linear Regression Model

```
mlr <- lm(data=train, formula=Personal_Loan ~.)
summary(mlr)
```

```
##
## Call:
## lm(formula = Personal_Loan ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.68244 -0.14079 -0.03393  0.07058  1.03874
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -4.085e-01  2.523e-02 -16.192  < 2e-16 ***
## Age                3.078e-04  3.896e-04   0.790  0.42955
## Income             3.119e-03  1.301e-04  23.971  < 2e-16 ***
## Family             3.238e-02  3.870e-03   8.365  < 2e-16 ***
## CCAvg              1.061e-02  3.269e-03   3.244  0.00119 **
## Education          8.125e-02  5.264e-03  15.435  < 2e-16 ***
## Mortgage           4.913e-05  4.339e-05   1.132  0.25762
## Securities_Account -5.028e-02  1.534e-02  -3.277  0.00106 **
## CD_Account         2.871e-01  1.948e-02  14.737  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2382 on 2959 degrees of freedom
## Multiple R-squared:  0.3772, Adjusted R-squared:  0.3755
## F-statistic:   224 on 8 and 2959 DF,  p-value: < 2.2e-16
```

```
mlr_backward <- step(mlr, direction = "backward")
```

```
## Start:  AIC=-8505.98
## Personal_Loan ~ Age + Income + Family + CCAvg + Education + Mortgage +
##     Securities_Account + CD_Account
##
##                    Df Sum of Sq    RSS     AIC
## - Age               1     0.035 167.99 -8507.4
## - Mortgage          1     0.073 168.03 -8506.7
## <none>                          167.95 -8506.0
## - CCAvg             1     0.597 168.55 -8497.4
## - Securities_Account  1     0.610 168.56 -8497.2
## - Family            1     3.972 171.92 -8438.6
## - CD_Account        1    12.327 180.28 -8297.8
## - Education         1    13.522 181.47 -8278.2
## - Income            1    32.615 200.57 -7981.3
##
## Step:  AIC=-8507.36
## Personal_Loan ~ Income + Family + CCAvg + Education + Mortgage +
##     Securities_Account + CD_Account
##
##                    Df Sum of Sq    RSS     AIC
## - Mortgage          1     0.074 168.06 -8508.1
## <none>                          167.99 -8507.4
## - CCAvg             1     0.596 168.58 -8498.8
## - Securities_Account  1     0.615 168.60 -8498.5
## - Family            1     3.940 171.93 -8440.6
## - CD_Account        1    12.325 180.31 -8299.2
## - Education         1    13.576 181.56 -8278.7
## - Income            1    32.582 200.57 -7983.2
##
## Step:  AIC=-8508.05
## Personal_Loan ~ Income + Family + CCAvg + Education + Securities_Account +
##     CD_Account
##
##                    Df Sum of Sq    RSS     AIC
## <none>                          168.06 -8508.1
## - CCAvg             1     0.585 168.65 -8499.7
## - Securities_Account  1     0.621 168.68 -8499.1
## - Family            1     3.964 172.03 -8440.9
## - CD_Account        1    12.441 180.50 -8298.1
## - Education         1    13.579 181.64 -8279.4
## - Income            1    34.129 202.19 -7961.3
```

```
summary(mlr_backward)
```

```
##
## Call:
## lm(formula = Personal_Loan ~ Income + Family + CCAvg + Education +
##     Securities_Account + CD_Account, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.69216 -0.13948 -0.03475  0.07231  1.04383
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -0.393040   0.017251 -22.784  < 2e-16 ***
## Income             0.003140   0.000128  24.521  < 2e-16 ***
## Family             0.032269   0.003861   8.358  < 2e-16 ***
## CCAvg              0.010494   0.003268   3.211  0.00134 **
## Education          0.081379   0.005261  15.467  < 2e-16 ***
## Securities_Account -0.050743   0.015339  -3.308  0.00095 ***
## CD_Account         0.288119   0.019461  14.805  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2382 on 2961 degrees of freedom
## Multiple R-squared:  0.3768, Adjusted R-squared:  0.3755
## F-statistic: 298.4 on 6 and 2961 DF,  p-value: < 2.2e-16
```

```
pred.train <- predict(mlr_backward, train)
accuracy(pred.train, train$Personal_Loan)
```

```
##                      ME      RMSE       MAE MPE MAPE
## Test set -3.605954e-15 0.2379593 0.1642916 NaN  Inf
```

```
pred.valid <- predict(mlr_backward, valid)
accuracy(pred.valid, valid$Personal_Loan)
```

```
##                   ME      RMSE       MAE MPE MAPE
## Test set -0.01100458 0.2250647 0.1552652 NaN  Inf
```

Errors (Smaller the better) ME: Mean error, error = acual - prediction MEA: mean absolute error MPE: mean percentage error MAPE: mean absolute precentage error RMSE: root mean square error sqrt(mean(error^2))

```
sd(train$Personal_Loan)
```

```
## [1] 0.301483
```

K-Nearest Neighbors

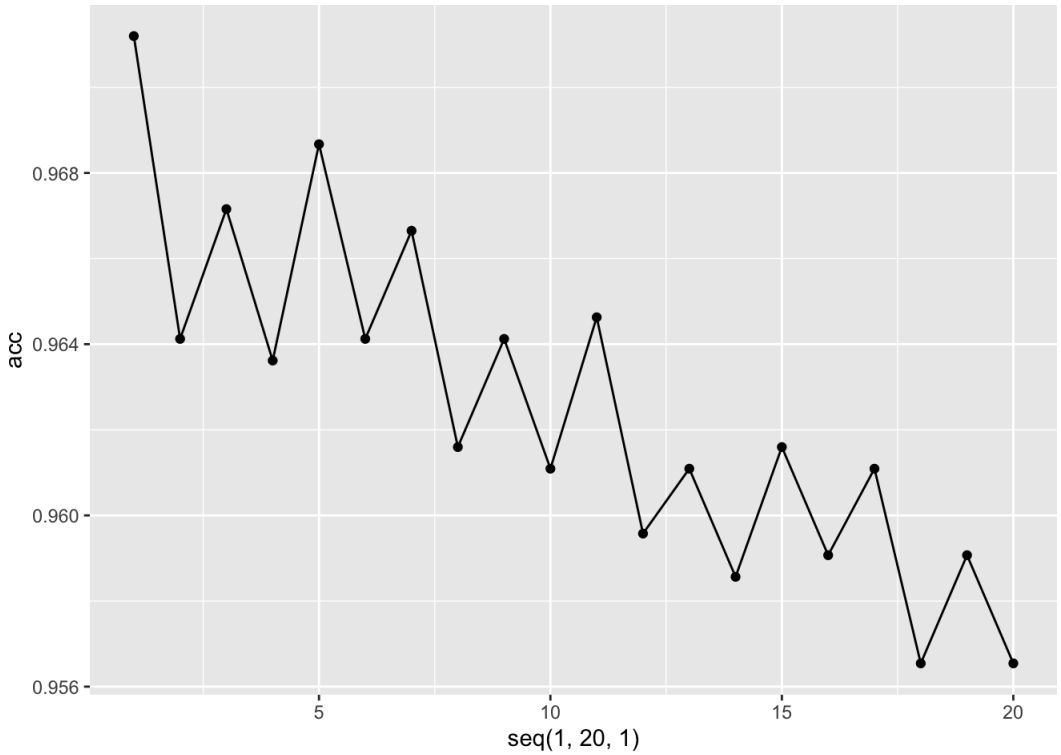
normalize the data

```
train_norm <- scale(train[,1:8])
train_norm <- data.frame(train_norm)
train_norm <- cbind(train_norm,train$Personal_Loan)
colnames(train_norm)[ncol(train_norm)] <- "Personal_Loan"
```

```
valid_norm <- scale(valid[,1:8])
valid_norm <- data.frame(valid_norm)
valid_norm <- cbind(valid_norm, valid$Personal_Loan)
colnames(valid_norm)[ncol(valid_norm)] <- "Personal_Loan"
```

find the best value of K

```
library(FNN)
acc <- c()
for(k in seq(1, 20, 1)){
  m <- knn(train=train_norm[,1:8], test=valid_norm[,1:8], cl=train_norm[,9],
           k=k)
  acc <- c(acc, mean(valid_norm[,9]==m))
}

ggplot() + geom_point(aes(x=seq(1, 20, 1), y=acc)) +
  geom_line(aes(x=seq(1, 20, 1), y=acc))
```



k=1 is the best. Predict by 1 neighbors However, we will choose k=5 which also has very high accuracy (>96%) because using 5 neighbors as indicates of the prediction is more reliable than using only one neighbor.

```
model_knn <- knn(train=train_norm[,1:8], test=valid_norm[,1:8], cl=train_norm[,9],k=5)
```

#Accuracy of KNN model

```
sum(model_knn == valid_norm[,9]) / nrow(valid_norm)
```

```
## [1] 0.968671
```

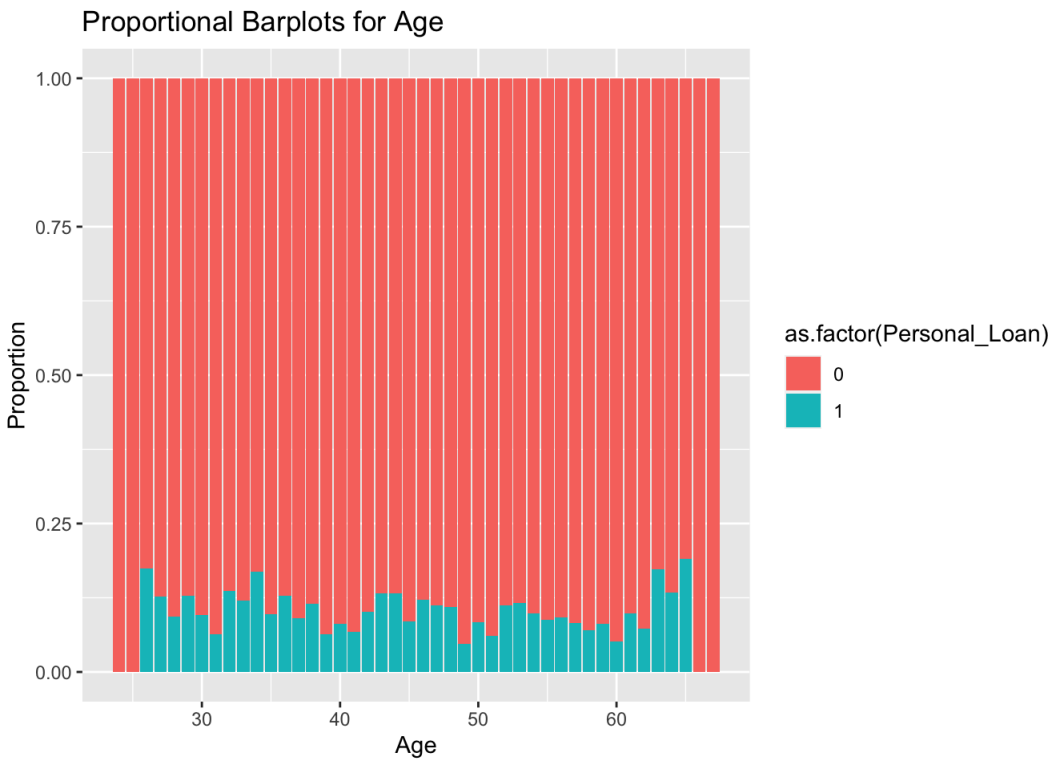
96.8% accuracy

Naive Bayes

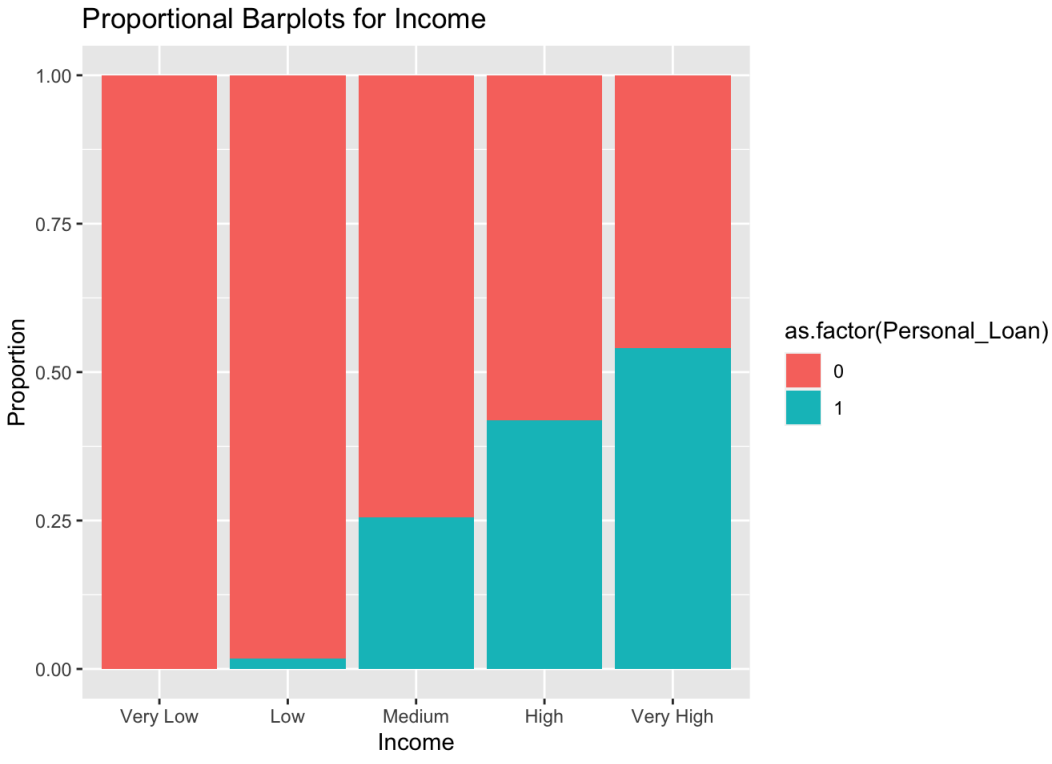
```
colnames(train)
```

```
## [1] "Age"              "Income"           "Family"
## [4] "CCAvg"            "Education"        "Mortgage"
## [7] "Securities_Account" "CD_Account"       "Personal_Loan"
```

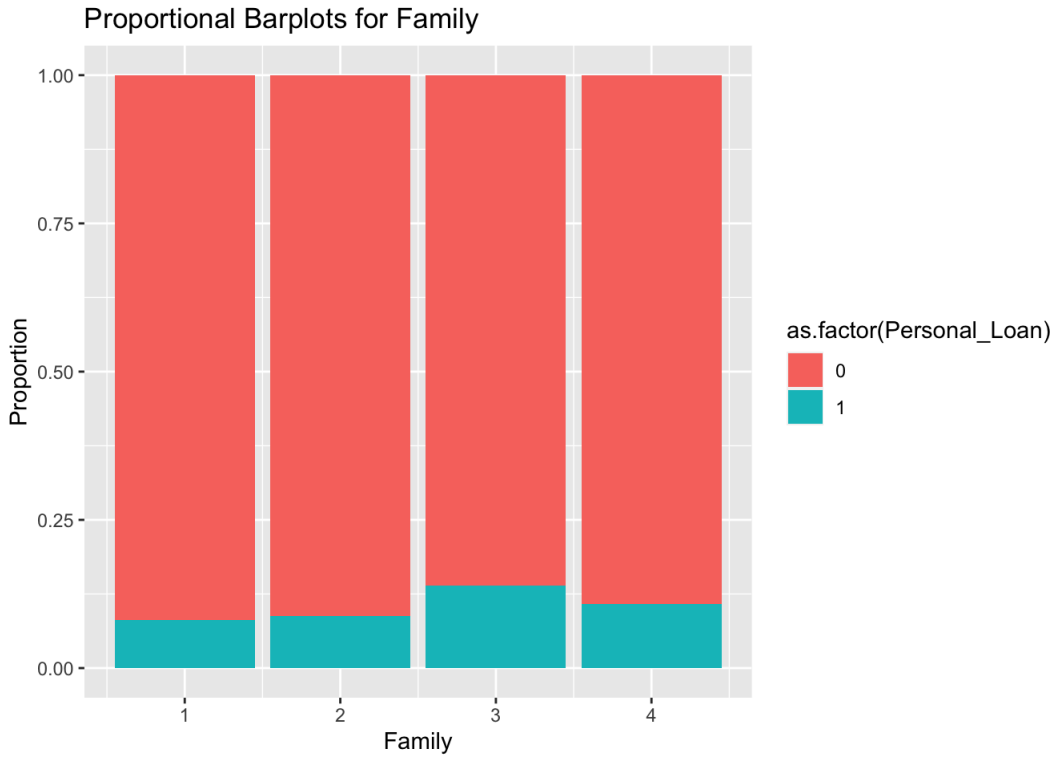
```
ggplot(train) +
  geom_bar(aes(x = Age, fill = as.factor(Personal_Loan)), position = "fill") +
  labs(title = "Proportional Barplots for Age",
       x = "Age",
       y = "Proportion")
```



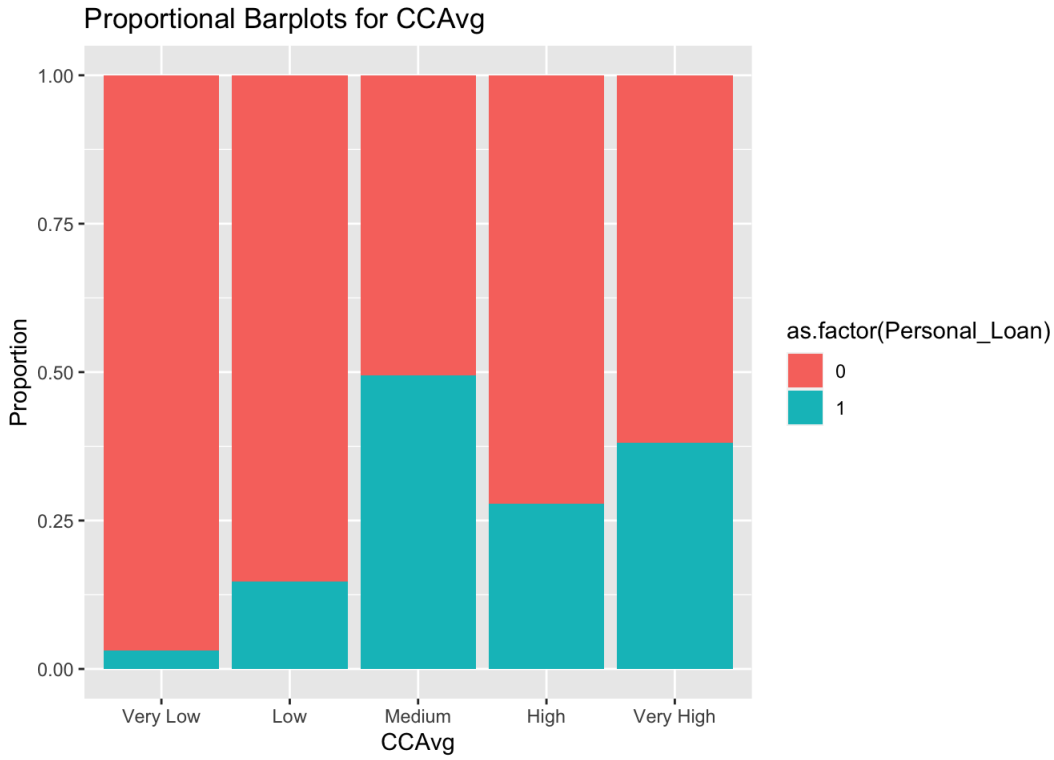

```
train1 <- train %>% mutate(Income = cut(Income, breaks = 5, labels = c("Very Low", "Low", "Medium", "High", "Very High"), include.lowest = TRUE))


ggplot(train1) +
  geom_bar(aes(x = Income, fill = as.factor(Personal_Loan)), position = "fill") +
  labs(title = "Proportional Barplots for Income",
       x = "Income",
       y = "Proportion")
```



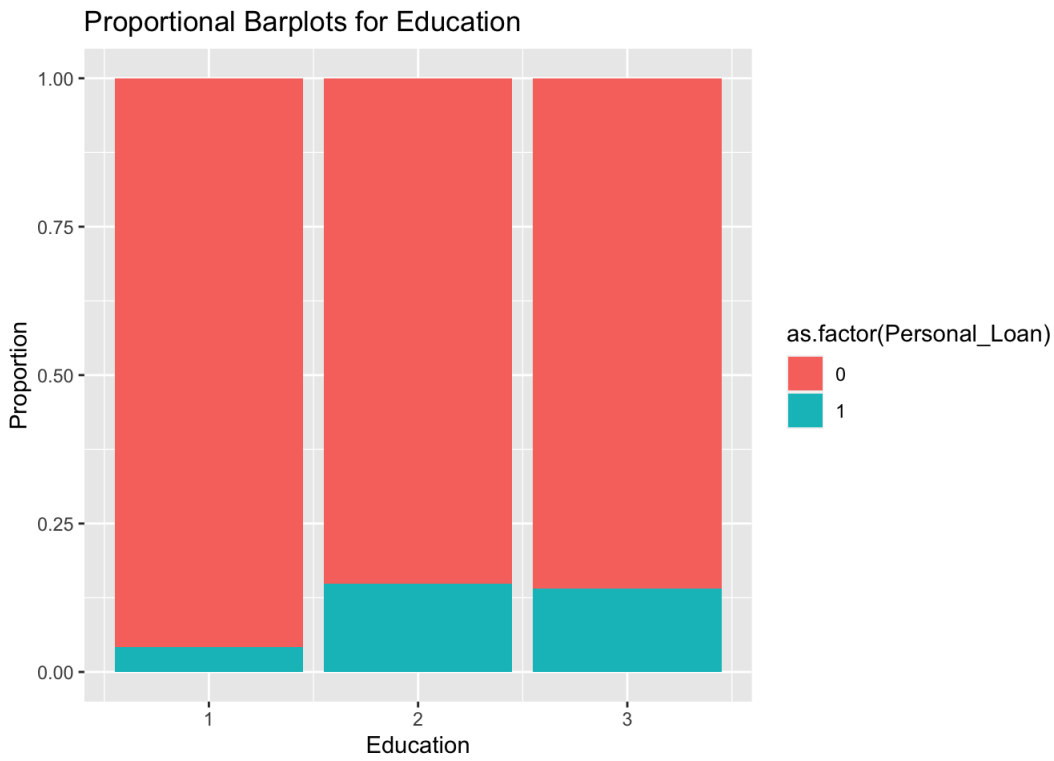

```
ggplot(train1) +
  geom_bar(aes(x = Family, fill = as.factor(Personal_Loan)), position = "fill") +
  labs(title = "Proportional Barplots for Family",
       x = "Family",
       y = "Proportion")
```



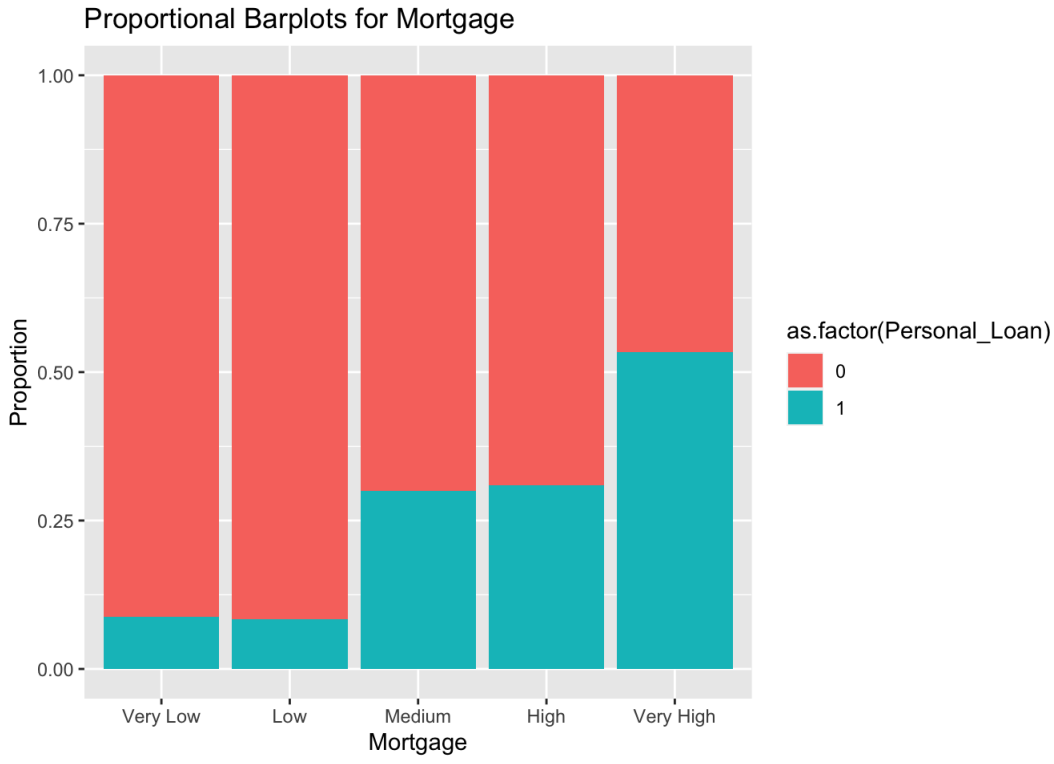

```
train1 <- train %>% mutate(CCAvg = cut(CCAvg, breaks = 5, labels = c("Very Low", "Low", "Medium", "High", "Very High"), include.lowest = TRUE))


ggplot(train1) +
  geom_bar(aes(x = CCAvg, fill = as.factor(Personal_Loan)), position = "fill") +
  labs(title = "Proportional Barplots for CCAvg",
       x = "CCAvg",
       y = "Proportion")
```



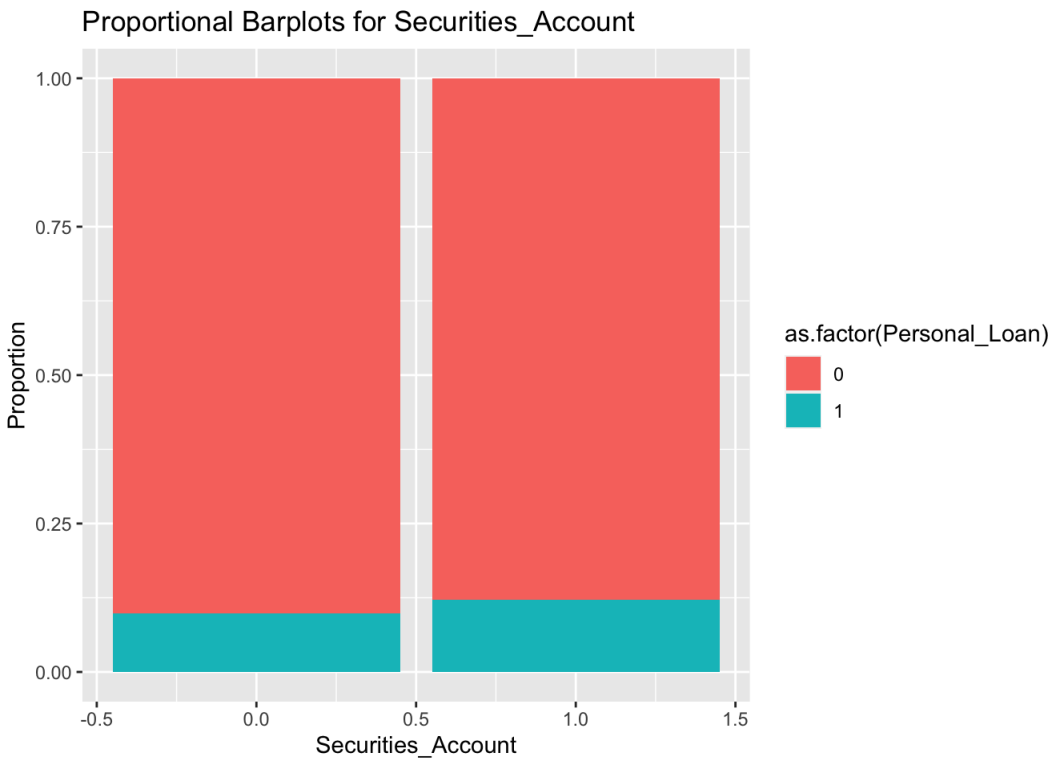

```
ggplot(train1) +
  geom_bar(aes(x = Education, fill = as.factor(Personal_Loan)), position = "fill") +
  labs(title = "Proportional Barplots for Education",
       x = "Education",
       y = "Proportion")
```



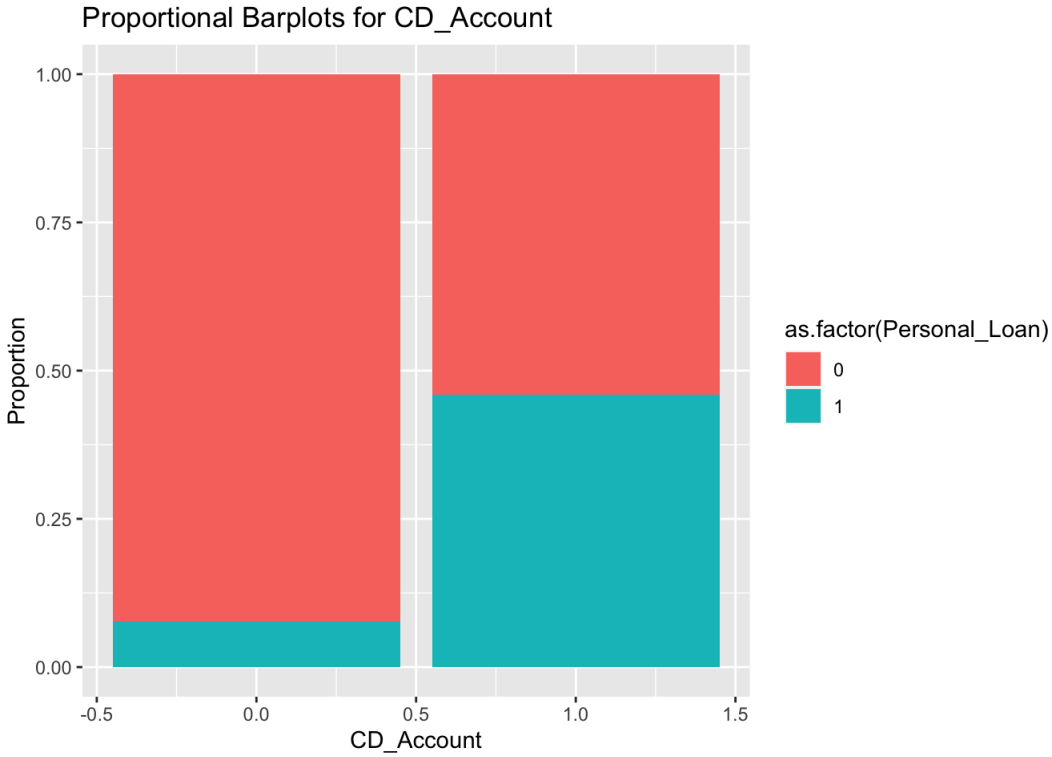

```
train1 <- train %>% mutate(Mortgage = cut(Mortgage, breaks = 5, labels = c("Very Low", "Low", "Medium", "High", "Very High"), include.lowest = TRUE))


ggplot(train1) +
  geom_bar(aes(x = Mortgage, fill = as.factor(Personal_Loan)), position = "fill") +
  labs(title = "Proportional Barplots for Mortgage",
       x = "Mortgage",
       y = "Proportion")
```




```
ggplot(train1) +
  geom_bar(aes(x = Securities_Account, fill = as.factor(Personal_Loan)), position = "fill") +
  labs(title = "Proportional Barplots for Securities_Account",
       x = "Securities_Account",
       y = "Proportion")
```




```
ggplot(train1) +
  geom_bar(aes(x = CD_Account, fill = as.factor(Personal_Loan)), position = "fill") +
  labs(title = "Proportional Barplots for CD_Account",
       x = "CD_Account",
       y = "Proportion")
```




```
library(e1071)
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
train_nb <- train %>% select(Income,CCAvg,Education,Mortgage,CD_Account,Personal_Loan)
valid_nb <- valid %>% select(Income,CCAvg,Education,Mortgage,CD_Account,Personal_Loan)
nb_model <- naiveBayes(data=train_nb, Personal_Loan~.)
nb_model
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         0         1
## 0.8989218 0.1010782
##
## Conditional probabilities:
##    Income
## Y        [,1]     [,2]
##   0  65.37256 40.81440
##   1 143.25667 32.19959
##
##    CCAvg
## Y       [,1]     [,2]
##   0 1.725746 1.602222
##   1 3.901733 2.072416
##
##    Education
## Y       [,1]     [,2]
##   0 1.841454 0.847134
##   1 2.246667 0.735499
##
##    Mortgage
## Y        [,1]      [,2]
##   0  53.62331  94.47066
##   1 100.46667 156.29415
##
##    CD_Account
## Y         [,1]      [,2]
##   0 0.03785607 0.1908839
##   1 0.28666667 0.4529600
```

```
pred_train <- predict(nb_model, newdata=train_nb)
table(pred_train, train_nb$Personal_Loan)
```

```
##
## pred_train    0    1
##          0 2427  127
##          1  241  173
```

```
mean(pred_train==train_nb$Personal_Loan)
```

```
## [1] 0.8760108
```

```
pred_valid <- predict(nb_model, newdata=valid_nb)
table(pred_valid, valid_nb$Personal_Loan)
```

```
##
## pred_valid    0    1
##          0 1665   70
##          1  134  110
```

```
mean(pred_valid==valid_nb$Personal_Loan)
```

```
## [1] 0.8969176
```

Accuracy 89.45%.

Part 4. Business insights

Cluster analysis has grouped customers into five distinct groups based on their characteristics such as income, age, credit card spending, mortgage balance, securities/CD accounts, and debt to income ratio. The model accuracy is 96%

The “confusion_matrix” table shows the number of true positives, true negatives, false positives, and false negatives generated by the model’s predictions. The “accuracy” variable calculates the proportion of correctly predicted values out of the total number of predictions made by the model, indicating that the model is 97.98% accurate in predicting personal loan acceptance.

Businesses can use the insights from cluster analysis and the accuracy of the predictive model to develop targeted marketing strategies and financial products to better serve different customer segments. By tailoring marketing efforts to specific customer groups, businesses can increase the efficiency of their marketing campaigns and improve customer engagement. For example, offering cost-saving measures to customers in the “Low-Income and Low-Spending” cluster or promoting high-end products and services to customers in the “High-Income and High-Spending” cluster can lead to increased customer satisfaction and loyalty. Additionally, developing targeted financial products, such as debt consolidation loans, for customers in the “High-Spending and High-Debt” cluster can help businesses meet their unique financial needs and improve their financial well-being. Overall, leveraging the insights from cluster analysis and predictive models can help businesses better understand and serve their customers, leading to increased revenue and growth opportunities.