SPROCKET CENTRAL PTY LTD

# CUSTOMER ANALYIS

---

## An analysis of Sprocket Central Pty Ltd New Customers.

**We will create a model that clusters the customers into the 4 cluster categories. The model will be created by variables that are also available in the newcustomerlist data.**

**We will then use the model on the new customer list to try and segment the customers into 4 different segments.**

```
library(tidyverse)
library(lubridate)
library(scales) ## for scales
```

### 5 Modeling customer cluster

**select columns that would not be repeative per customer**

**Leave out columns used to calculate clusters**

```
lrfmp_customers <- read_csv("lrfmp_customers.csv")
lrfmp_customers_2 <- lrfmp_customers
```

```r
lrfmp_customers_2 <- lrfmp_customers_2 %>%
  select(3,18:26,28:40)
lrfmp_customers_3 <- lrfmp_customers_2 %>% select(21,1,2:14,22:23)
```

**Get distinct customers**

```r
lrfmp_customers_4 <- lrfmp_customers_3 %>%
  distinct(customer_id, .keep_all = TRUE)
dim(lrfmp_customers_4)
```

```
## [1] 3381   17
```

**We will create a model but leave out some columns and use their modifications for easier classification.**

**We will use age_group in place of age and dob, past_purchase_group in place of past_purchases and pvaluation_group in place of property_valuation. From the plots, in gender there wasn't a clear distinction.**

```r
customer_data <- lrfmp_customers_4 %>% select(-c(4,5,6,7,12,13,15))
dim(customer_data)
```

```
## [1] 3381   10
```

```r
head(customer_data %>% select(1:5))
```

```
## # A tibble: 6 x 5
##   cluster customer_id customer_name          age_group job_industry
##     <dbl>       <dbl> <chr>                  <chr>     <chr>
## 1       1           1 Laraine Medendorp      Older     Health
## 2       4           2 Eli Bockman            Middle    Financials
## 3       4           4 Talbot                 Older     IT
## 4       3           5 Sheila-kathryn Calton Middle    Not Available
## 5       3           6 Curr Duckhouse         Older     Retail
## 6       4           7 Fina Merali            Middle    Financials
```

```r
head(customer_data %>% select(6:10))
```

```
## # A tibble: 6 x 5
##   wealth_segment owns_car state past_purchase_group pvaluation_group
##   <chr>          <chr>    <chr> <chr>               <chr>
```

```
## 1 Mass           Yes      NSW   Excellent          Wealthy
## 2 Mass           Yes      NSW   Better             Wealthy
## 3 Mass           No       QLD   Good               Wealthy
## 4 Affluent       Yes      NSW   Good               Average
## 5 High Net       Yes      VIC   Good               Wealthy
## 6 Affluent       Yes      NSW   Bad                Wealthy
```

**We have 4 clusters with**

- Cluster 1:Most loyal
- Cluster 3:Regular
- Cluster 4:Hibernating
- Cluster 2:Seasonal

**We replace the digits in cluster with the words and recode as factor with 4 levels.**

```
customer_data$cluster[customer_data$cluster == 1] <- "most loyal"
customer_data$cluster[customer_data$cluster == 2] <- "seasonal"
customer_data$cluster[customer_data$cluster == 3] <- "regular"
customer_data$cluster[customer_data$cluster == 4] <- "hibernating"
```

```
class(customer_data$cluster)
```

```
## [1] "character"
```

```
customer_data %>% count(cluster, sort = T)
```

```
## # A tibble: 4 x 2
##   cluster         n
##   <chr>       <int>
## 1 most loyal   1168
## 2 regular      1100
## 3 hibernating   772
## 4 seasonal      341
```

**All columns are characters**

```
str(customer_data)
```

```
## tibble [3,381 x 10] (S3: tbl_df/tbl/data.frame)
##  $ cluster            : chr [1:3381] "most loyal" "hibernating" "hibernating" "regular" ...
##  $ customer_id        : num [1:3381] 1 2 4 5 6 7 8 9 11 12 ...
```

```
##  $ customer_name      : chr [1:3381] "Laraine Medendorp" "Eli Bockman" "Talbot" "Sheila-kat
##  $ age_group          : chr [1:3381] "Older" "Middle" "Older" "Middle" ...
##  $ job_industry       : chr [1:3381] "Health" "Financials" "IT" "Not Available" ...
##  $ wealth_segment     : chr [1:3381] "Mass" "Mass" "Mass" "Affluent" ...
##  $ owns_car           : chr [1:3381] "Yes" "Yes" "No" "Yes" ...
##  $ state              : chr [1:3381] "NSW" "NSW" "QLD" "NSW" ...
##  $ past_purchase_group: chr [1:3381] "Excellent" "Better" "Good" "Good" ...
##  $ pvaluation_group   : chr [1:3381] "Wealthy" "Wealthy" "Wealthy" "Average" ...
```

**assign digits to the different columns and convert to factors-the levels are not ordered thus the numbers are taken as per alphabetical order from A receiving the least to Z receiving the highest digit**

```
customer_data_3 <- customer_data
customer_data_3$age_group <- as.integer(factor(customer_data_3$age_group))
customer_data_3$job_industry <- as.integer(factor(customer_data_3$job_industry))
customer_data_3$wealth_segment <- as.integer(factor(customer_data_3$wealth_segment))
customer_data_3$owns_car <- as.integer(factor(customer_data_3$owns_car))
customer_data_3$state <- as.integer(factor(customer_data_3$state))
customer_data_3$past_purchase_group <- as.integer(factor(customer_data_3$past_purchase_group))
customer_data_3$pvaluation_group <- as.integer(factor(customer_data_3$pvaluation_group))
str(customer_data_3)
```

```
## tibble [3,381 x 10] (S3: tbl_df/tbl/data.frame)
##  $ cluster            : chr [1:3381] "most loyal" "hibernating" "hibernating" "regular" ...
##  $ customer_id        : num [1:3381] 1 2 4 5 6 7 8 9 11 12 ...
##  $ customer_name      : chr [1:3381] "Laraine Medendorp" "Eli Bockman" "Talbot" "Sheila-kat
##  $ age_group          : int [1:3381] 2 1 2 1 2 1 2 1 2 4 ...
##  $ job_industry       : int [1:3381] 4 3 5 7 9 3 7 1 8 6 ...
##  $ wealth_segment     : int [1:3381] 3 3 3 1 2 1 3 1 3 3 ...
##  $ owns_car           : int [1:3381] 2 2 1 2 2 2 1 2 1 1 ...
##  $ state              : int [1:3381] 1 1 2 1 3 1 1 1 3 2 ...
##  $ past_purchase_group: int [1:3381] 3 2 4 4 4 1 4 3 3 4 ...
##  $ pvaluation_group   : int [1:3381] 3 3 3 1 3 3 1 3 3 1 ...
```

**Convert columns to factor**

```
customer_data_1 <- customer_data
customer_data_1 <- customer_data %>%
  mutate_if(is.character, as.factor)
str(customer_data_1)
```

```
## tibble [3,381 x 10] (S3: tbl_df/tbl/data.frame)
##  $ cluster            : Factor w/ 4 levels "hibernating",..: 2 1 1 3 3 1 2 3 3 3 ...
```

```
##  $ customer_id        : num [1:3381] 1 2 4 5 6 7 8 9 11 12 ...
##  $ customer_name      : Factor w/ 3379 levels "Aarika Magog",..: 1906 981 3017 2861 743 115
##  $ age_group          : Factor w/ 4 levels "Middle","Older",..: 2 1 2 1 2 1 2 1 2 4 ...
##  $ job_industry       : Factor w/ 10 levels "Argiculture",..: 4 3 5 7 9 3 7 1 8 6 ...
##  $ wealth_segment     : Factor w/ 3 levels "Affluent","High Net",..: 3 3 3 1 2 1 3 1 3 3 ..
##  $ owns_car           : Factor w/ 2 levels "No","Yes": 2 2 1 2 2 2 1 2 1 1 ...
##  $ state              : Factor w/ 3 levels "NSW","QLD","VIC": 1 1 2 1 3 1 1 1 3 2 ...
##  $ past_purchase_group: Factor w/ 4 levels "Bad","Better",..: 3 2 4 4 4 1 4 3 3 4 ...
##  $ pvaluation_group   : Factor w/ 3 levels "Average","Minimum",..: 3 3 3 1 3 3 1 3 3 1 ...
```

```
customer_data_1$customer_id <- as.character(as.factor(customer_data_1$customer_id))
customer_data_1$customer_name <- as.character(as.factor(customer_data_1$customer_name))
table(customer_data_1$cluster)
```

```
##
## hibernating  most loyal     regular    seasonal
##         772        1168        1100         341
```

```
class(customer_data_1$cluster)
```

```
## [1] "factor"
```

```
customer_data_3$cluster <- factor(customer_data_3$cluster,
                                  levels = c("most loyal", "regular",
                                             "seasonal", "hibernating"))
customer_data_3$cluster <- factor(customer_data_3$cluster, ordered = TRUE)
customer_data_3$cluster <- fct_infreq(customer_data_3$cluster)
customer_data_3$cluster[1:5]
```

```
## [1] most loyal  hibernating hibernating regular     regular
## Levels: most loyal < regular < hibernating < seasonal
```

```
customer_data_3 <- customer_data_3 %>%
  mutate_if(is.numeric, as.factor)
```

```
str(customer_data_3)
```

```
## tibble [3,381 x 10] (S3: tbl_df/tbl/data.frame)
##  $ cluster            : Ord.factor w/ 4 levels "most loyal"<"regular"<..: 1 3 3 2 2 3 1 2 2
##  $ customer_id        : Factor w/ 3381 levels "1","2","4","5",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ customer_name      : chr [1:3381] "Laraine Medendorp" "Eli Bockman" "Talbot" "Sheila-kat
##  $ age_group          : Factor w/ 4 levels "1","2","3","4": 2 1 2 1 2 1 2 1 2 4 ...
##  $ job_industry       : Factor w/ 10 levels "1","2","3","4",..: 4 3 5 7 9 3 7 1 8 6 ...
##  $ wealth_segment     : Factor w/ 3 levels "1","2","3": 3 3 3 1 2 1 3 1 3 3 ...
##  $ owns_car           : Factor w/ 2 levels "1","2": 2 2 1 2 2 2 1 2 1 1 ...
##  $ state              : Factor w/ 3 levels "1","2","3": 1 1 2 1 3 1 1 1 3 2 ...
##  $ past_purchase_group: Factor w/ 4 levels "1","2","3","4": 3 2 4 4 4 1 4 3 3 4 ...
##  $ pvaluation_group   : Factor w/ 3 levels "1","2","3": 3 3 3 1 3 3 1 3 3 1 ...
```

### Rename cluster to loyalty

```
customer_data_3 <- customer_data_3 %>% rename(loyalty = cluster)
```

### Proportions

```
xtabs(~loyalty+age_group, data = customer_data_3)
```

```
##              age_group
## loyalty         1   2   3   4
##   most loyal  604 272  22 270
##   regular     562 245  28 265
##   hibernating 381 193  18 180
##   seasonal    168  85   7  81
```

```
library(tidymodels)
library(caret)
```

### Splitting data

```
set.seed(123)
parts <- initial_split(customer_data_3, prop = 0.7, strata = loyalty)
train_data_1 <- training(parts)
test_data <- testing(parts)
```

### Ordinal Logistic Regression using the polr from MASS package

```
library(MASS)
```

```
model_ordinal <- train(loyalty~age_group+job_industry+wealth_segment+owns_car+
                         state+past_purchase_group+pvaluation_group,
                       data = train_data_1,
                       method = "polr",
                       tuneGrid = expand.grid(method = "logistic"))
```

```
summary(model_ordinal)
```

```
##
## Coefficients:
```

```
##                          Value Std. Error t value
## age_group2             0.03713    0.09379  0.3959
## age_group3             0.23710    0.28503  0.8319
## age_group4             0.07470    0.09323  0.8012
## job_industry2          0.03461    0.30918  0.1119
## job_industry3          0.12171    0.25069  0.4855
## job_industry4          0.15385    0.25539  0.6024
## job_industry5         -0.08904    0.30391 -0.2930
## job_industry6          0.23259    0.25101  0.9266
## job_industry7          0.33776    0.25442  1.3276
## job_industry8          0.38800    0.27418  1.4151
## job_industry9          0.14098    0.26771  0.5266
## job_industry10         0.36158    0.35964  1.0054
## wealth_segment2        0.12999    0.10523  1.2352
## wealth_segment3       -0.14683    0.09265 -1.5848
## owns_car2              0.13255    0.07525  1.7615
## state2                 0.06098    0.10666  0.5717
## state3                 0.08580    0.09112  0.9416
## past_purchase_group2 -0.01508    0.10457 -0.1442
## past_purchase_group3 -0.05250    0.12613 -0.4162
## past_purchase_group4 -0.12452    0.09725 -1.2805
## pvaluation_group2      0.11097    0.12875  0.8618
## pvaluation_group3     -0.11836    0.09033 -1.3103
##
## Intercepts:
##                       Value   Std. Error t value
## most loyal|regular    -0.4706  0.2731    -1.7231
## regular|hibernating    0.8934  0.2738     3.2634
## hibernating|seasonal   2.3797  0.2789     8.5320
##
## Residual Deviance: 6122.189
## AIC: 6172.189
```

### Addind P-values

```r
prob <- pnorm(abs(summary(model_ordinal)$coefficients[, 3]),
              lower.tail = FALSE)*2
cbind(summary(model_ordinal)$coefficients, prob)
```

```
##                             Value Std. Error    t value         prob
## age_group2             0.03713276 0.09378991  0.3959142 6.921683e-01
## age_group3             0.23709967 0.28502680  0.8318505 4.054934e-01
## age_group4             0.07469712 0.09323098  0.8012050 4.230130e-01
## job_industry2          0.03460951 0.30918316  0.1119385 9.108721e-01
## job_industry3          0.12171009 0.25069273  0.4854951 6.273252e-01
## job_industry4          0.15385036 0.25538983  0.6024138 5.468987e-01
```

```
## job_industry5        -0.08903587 0.30391322 -0.2929648 7.695491e-01
## job_industry6         0.23259477 0.25100839  0.9266414 3.541128e-01
## job_industry7         0.33776247 0.25441975  1.3275796 1.843170e-01
## job_industry8         0.38800209 0.27417766  1.4151485 1.570250e-01
## job_industry9         0.14097519 0.26770951  0.5265976 5.984731e-01
## job_industry10        0.36157882 0.35963712  1.0053991 3.147047e-01
## wealth_segment2       0.12999039 0.10523411  1.2352496 2.167376e-01
## wealth_segment3      -0.14683231 0.09265257 -1.5847624 1.130203e-01
## owns_car2             0.13255407 0.07525048  1.7615045 7.815305e-02
## state2                0.06097743 0.10665679  0.5717163 5.675142e-01
## state3                0.08580353 0.09112182  0.9416354 3.463793e-01
## past_purchase_group2 -0.01508358 0.10456723 -0.1442477 8.853049e-01
## past_purchase_group3 -0.05249847 0.12612627 -0.4162374 6.772363e-01
## past_purchase_group4 -0.12451865 0.09724592 -1.2804511 2.003865e-01
## pvaluation_group2     0.11096511 0.12875229  0.8618495 3.887703e-01
## pvaluation_group3    -0.11835716 0.09032603 -1.3103327 1.900833e-01
## most loyal|regular   -0.47061692 0.27311961 -1.7231166 8.486745e-02
## regular|hibernating   0.89344858 0.27378207  3.2633568 1.101008e-03
## hibernating|seasonal  2.37972519 0.27891750  8.5320038 1.438344e-17
```

**Taking a p-value of 0.05 we see that none of the predictors are significant.**

**Accuracy rate of the for the training data**

```r
predict(model_ordinal, train_data_1) %>%
  bind_cols(train_data_1) %>%
  rename(pred ="...1", truth = loyalty) %>%
  accuracy(pred, truth)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.367
```

**With logistic regression model we get 37% accuracy for the training set.**

**Testing with the tesing set**

```r
predict(model_ordinal, test_data) %>%
  bind_cols(test_data) %>%
  rename(pred ="...1", truth = loyalty) %>%
  accuracy(pred, truth)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.335
```

**The accuuracy for the testing set is 34% which is smaller than the training set an indication of overfitting problem.**

**Trying with the probit**

```r
set.seed(123)
model_probit <- train(loyalty~age_group+job_industry+wealth_segment+owns_car+
                         state+past_purchase_group+pvaluation_group,
                       data = train_data_1,
                       method = "polr",
                       tuneGrid = expand.grid(method = "probit"))
```

**Accuracy on training set**

```r
predict(model_probit, train_data_1) %>%
  bind_cols(train_data_1) %>%
  rename(pred ="...1", truth = loyalty) %>%
  accuracy(pred, truth)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.367
```

**Accuracy on the training set is 37% same as with the logistic regression**

**Accuracy with the tesing set**

```r
predict(model_probit, test_data) %>%
  bind_cols(test_data) %>%
  rename(pred ="...1", truth = loyalty) %>%
  accuracy(pred, truth)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.340
```

**Same accuracy on the testing set as with the logistic regression**

**CART model**

```r
library(rpartScore)
```

```r
set.seed(123)
model_cart <- train(loyalty~age_group+job_industry+wealth_segment+owns_car+
                         state+past_purchase_group+pvaluation_group,
                       data = train_data_1,
                       method = "rpartScore")
saveRDS(model_cart, "model_cart.rds")
```

```r
model_cart <- readRDS("model_cart.rds")
model_cart
```

```
## CART or Ordinal Responses
##
## 2365 samples
##    7 predictor
##    4 classes: 'most loyal', 'regular', 'hibernating', 'seasonal'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 2365, 2365, 2365, 2365, 2365, 2365, ...
## Resampling results across tuning parameters:
##
##   cp           split  prune  Accuracy   Kappa
##   0.005813953  abs    mr     0.3376332  0.009581549
##   0.005813953  abs    mc     0.3242579  0.000000000
##   0.005813953  quad   mr     0.3405798  0.012659833
##   0.005813953  quad   mc     0.3242579  0.000000000
##   0.008397933  abs    mr     0.3388598  0.009636844
##   0.008397933  abs    mc     0.3242579  0.000000000
##   0.008397933  quad   mr     0.3398629  0.011307406
##   0.008397933  quad   mc     0.3242579  0.000000000
##   0.011412575  abs    mr     0.3364004  0.007911394
##   0.011412575  abs    mc     0.3242579  0.000000000
##   0.011412575  quad   mr     0.3371890  0.008884228
##   0.011412575  quad   mc     0.3242579  0.000000000
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were cp = 0.005813953, split = quad
##  and prune = mr.
```

**The largest accuracy rate is about 34%**

**Accuracy**

```r
predict(model_cart, train_data_1) %>%
  bind_cols(train_data_1) %>%
  rename(pred ="...1", truth = loyalty) %>%
  accuracy(pred, truth)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.364
```

**Accuracy on the training set is about 36%**

**On the testing set**

```r
predict(model_cart, test_data) %>%
  bind_cols(test_data) %>%
  rename(pred ="...1", truth = loyalty) %>%
  accuracy(pred, truth)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.330
```

**Accuracy on the test set is lower at 33%**

**Ordinal Random Forest Model**

```r
library(randomForest)
library(ranger)
library(e1071)
```

```r
set.seed(123)
model_forest <- train(loyalty~age_group+job_industry+wealth_segment+owns_car+
                         state+past_purchase_group+pvaluation_group,
                       data = train_data_1,
                       method = "ordinalRF")
saveRDS(model_forest, "model_forest.rds")
model_forest <- readRDS("model_forest.rds")
```

#### Continuation Ratio Model

```r
library(VGAM)
set.seed(123)
model_vgam <- train(loyalty~age_group+job_industry+wealth_segment+owns_car+
                         state+past_purchase_group+pvaluation_group,
                       data = train_data_1,
                       method = "vglmContRatio", trace = FALSE)
model_vgam
```

```
## Continuation Ratio Model for Ordinal Data
##
## 2365 samples
##    7 predictor
##    4 classes: 'most loyal', 'regular', 'hibernating', 'seasonal'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 2365, 2365, 2365, 2365, 2365, 2365, ...
## Resampling results across tuning parameters:
##
##   parallel  link     Accuracy   Kappa
##   FALSE     logit    0.3356295  0.010314721
##   FALSE     probit   0.3360491  0.010586101
##   FALSE     cloglog  0.3378501  0.013541300
##   FALSE     cauchit  0.3359406  0.013202983
##   FALSE     logc     0.3410726  0.014747115
##    TRUE     logit    0.3356586  0.007243148
##    TRUE     probit   0.3369166  0.008668402
##    TRUE     cloglog  0.3381938  0.010256776
##    TRUE     cauchit  0.3348406  0.009362833
##    TRUE     logc     0.3381576  0.008564491
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were parallel = FALSE and link = logc.
```

```r
model_vgam_1 <- vglm(loyalty~age_group+job_industry+wealth_segment+owns_car+
                         state+past_purchase_group+pvaluation_group,
                     family = cumulative(parallel = FALSE, reverse = FALSE),
                     data = train_data_1)
model_vgam
```

```
## Continuation Ratio Model for Ordinal Data
##
## 2365 samples
```

```
##     7 predictor
##     4 classes: 'most loyal', 'regular', 'hibernating', 'seasonal'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 2365, 2365, 2365, 2365, 2365, 2365, ...
## Resampling results across tuning parameters:
##
##   parallel  link     Accuracy   Kappa
##   FALSE     logit    0.3356295  0.010314721
##   FALSE     probit   0.3360491  0.010586101
##   FALSE     cloglog  0.3378501  0.013541300
##   FALSE     cauchit  0.3359406  0.013202983
##   FALSE     logc     0.3410726  0.014747115
##    TRUE     logit    0.3356586  0.007243148
##    TRUE     probit   0.3369166  0.008668402
##    TRUE     cloglog  0.3381938  0.010256776
##    TRUE     cauchit  0.3348406  0.009362833
##    TRUE     logc     0.3381576  0.008564491
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were parallel = FALSE and link = logc.
```

### Accuracy on training data

```
predict(model_vgam, train_data_1) %>%
  bind_cols(train_data_1) %>%
  rename(pred ="...1", truth = loyalty) %>%
  accuracy(pred, truth)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.372
```

### Accuracy on testing data

```
predict(model_vgam, test_data) %>%
  bind_cols(test_data) %>%
  rename(pred ="...1", truth = loyalty) %>%
  accuracy(pred, truth)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
```

```
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.328
```

**The different models almost have the same accuracy**

### Modeling with the non groups instead

```
data_customers <- lrfmp_customers %>%
  dplyr::select(38,3,18:20,22,24:26,30,31)
data_customers <- data_customers %>% rename(loyalty = cluster)
```

```
data_customers <- data_customers %>% distinct(customer_id, .keep_all = TRUE)
```

```
data_customers$loyalty[data_customers$loyalty == 1] <- "most loyal"
data_customers$loyalty[data_customers$loyalty == 2] <- "seasonal"
data_customers$loyalty[data_customers$loyalty == 3] <- "regular"
data_customers$loyalty[data_customers$loyalty == 4] <- "hibernating"
```

```
data_customers_3 <- data_customers
data_customers_3$gender[data_customers_3$gender == "Female"] <- 0
data_customers_3$gender[data_customers_3$gender == "Male"] <- 1
data_customers_3$gender[data_customers_3$gender == "Unidentified"] <- 2
data_customers_3$owns_car[data_customers_3$owns_car == "Yes"] <- 1
data_customers_3$owns_car[data_customers_3$owns_car == "No"] <- 0
data_customers_3$job_industry <- as.integer(factor(data_customers_3$job_industry))
data_customers_3$wealth_segment <- as.integer(factor(data_customers_3$wealth_segment))
data_customers_3$state <- as.integer(factor(data_customers_3$state))
str(data_customers_3)
```

```
## tibble [3,381 x 11] (S3: tbl_df/tbl/data.frame)
##  $ loyalty           : chr [1:3381] "most loyal" "hibernating" "hibernating" "regular" ...
##  $ customer_id       : num [1:3381] 1 2 4 5 6 7 8 9 11 12 ...
##  $ customer_name     : chr [1:3381] "Laraine Medendorp" "Eli Bockman" "Talbot" "Sheila-kath
##  $ gender            : chr [1:3381] "0" "1" "1" "0" ...
##  $ past_purchases    : num [1:3381] 93 81 33 56 35 6 31 97 99 58 ...
##  $ age               : num [1:3381] 69 42 61 46 57 47 61 50 69 29 ...
##  $ job_industry      : int [1:3381] 4 3 5 7 9 3 7 1 8 6 ...
##  $ wealth_segment    : int [1:3381] 3 3 3 1 2 1 3 1 3 3 ...
##  $ owns_car          : chr [1:3381] "1" "1" "0" "1" ...
##  $ state             : int [1:3381] 1 1 2 1 3 1 1 1 3 2 ...
##  $ property_valuation: num [1:3381] 10 10 9 4 9 9 4 12 8 4 ...
```

**Missing values**

```r
sum(is.na(data_customers_3))
```

```
## [1] 75
```

```r
sum(is.na(data_customers_3$age))
```

```
## [1] 75
```

**There are 75 missing values and all are for the ages.**

**We can exclude the rows**

```r
data_customers_3 <- data_customers_3 %>% drop_na()
```

```r
data_customers_3$loyalty <- factor(data_customers_3$loyalty,
                                  levels = c("most loyal", "regular",
                                             "seasonal", "hibernating"))
data_customers_3$loyalty <- factor(data_customers_3$loyalty, ordered = TRUE)
data_customers_3$loyalty <- fct_infreq(data_customers_3$loyalty)
data_customers_3$loyalty[1:5]
```

```
## [1] most loyal  hibernating hibernating regular     regular
## Levels: most loyal < regular < hibernating < seasonal
```

```r
data_customers_3$gender <- as.factor(data_customers_3$gender)
data_customers_3$owns_car <- as.factor(data_customers_3$owns_car)
data_customers_3$job_industry <- as.factor(data_customers_3$job_industry)
data_customers_3$wealth_segment <- as.factor(data_customers_3$wealth_segment)
data_customers_3$state <- as.factor(data_customers_3$state)
str(data_customers_3)
```

```
## tibble [3,306 x 11] (S3: tbl_df/tbl/data.frame)
##  $ loyalty          : Ord.factor w/ 4 levels "most loyal"<"regular"<..: 1 3 3 2 2 3 1 2 2 2
##  $ customer_id      : num [1:3306] 1 2 4 5 6 7 8 9 11 12 ...
##  $ customer_name    : chr [1:3306] "Laraine Medendorp" "Eli Bockman" "Talbot" "Sheila-kath
##  $ gender           : Factor w/ 3 levels "0","1","2": 1 2 2 1 2 1 2 1 2 2 ...
##  $ past_purchases   : num [1:3306] 93 81 33 56 35 6 31 97 99 58 ...
##  $ age              : num [1:3306] 69 42 61 46 57 47 61 50 69 29 ...
##  $ job_industry     : Factor w/ 10 levels "1","2","3","4",..: 4 3 5 7 9 3 7 1 8 6 ...
##  $ wealth_segment   : Factor w/ 3 levels "1","2","3": 3 3 3 1 2 1 3 1 3 3 ...
##  $ owns_car         : Factor w/ 2 levels "0","1": 2 2 1 2 2 2 1 2 1 1 ...
##  $ state            : Factor w/ 3 levels "1","2","3": 1 1 2 1 3 1 1 1 3 2 ...
##  $ property_valuation: num [1:3306] 10 10 9 4 9 9 4 12 8 4 ...
```

### Proportions

```r
xtabs(~loyalty+wealth_segment, data = data_customers_3)
```

```
##                wealth_segment
## loyalty          1   2   3
##   most loyal   279 267 600
##   regular      249 286 537
##   hibernating  188 188 378
##   seasonal      90 104 140
```

```r
xtabs(~loyalty+gender, data = data_customers_3)
```

```
##                gender
## loyalty          0   1   2
##   most loyal   601 544   1
##   regular      562 510   0
##   hibernating  373 381   0
##   seasonal     171 163   0
```

```r
xtabs(~loyalty+job_industry, data = data_customers_3)
```

```
##                job_industry
## loyalty          1   2   3   4   5   6   7   8   9  10
##   most loyal    35  48 234 182  42 219 181  75 111  19
##   regular       27  37 215 164  43 223 183  71  88  21
##   hibernating   23  27 149 115  30 158 117  56  68  11
##   seasonal       9   7  63  51   5  75  60  26  27  11
```

```r
xtabs(~loyalty+owns_car, data = data_customers_3)
```

```
##                owns_car
## loyalty          0   1
##   most loyal   581 565
##   regular      529 543
##   hibernating  359 395
##   seasonal     163 171
```

### Splitting data

```r
set.seed(123)
parts <- initial_split(data_customers_3, prop = 0.7, strata = loyalty)
train_data <- training(parts)
test_data <- testing(parts)
```

### Ordinal Logistic Regression using the polr from MASS package

```
model_ordinal_2 <- train(loyalty~gender+age+past_purchases+job_industry+
                           wealth_segment+owns_car+state+property_valuation,
                       data = train_data,
                       method = "polr",
                       tuneGrid = expand.grid(method = "logistic"))
```

```
summary(model_ordinal_2)
```

```
##
## Coefficients:
##                          Value Std. Error    t value
## gender1               0.065961  7.601e-02  8.678e-01
## gender2             -11.809583  2.063e-07 -5.725e+07
## age                   0.003329  2.986e-03  1.115e+00
## past_purchases       -0.001710  1.321e-03 -1.294e+00
## job_industry2        -0.058561  3.218e-01 -1.820e-01
## job_industry3         0.039662  2.615e-01  1.517e-01
## job_industry4         0.089784  2.670e-01  3.362e-01
## job_industry5        -0.157291  3.260e-01 -4.825e-01
## job_industry6         0.184157  2.609e-01  7.057e-01
## job_industry7         0.176702  2.640e-01  6.693e-01
## job_industry8         0.110322  2.862e-01  3.854e-01
## job_industry9         0.056325  2.775e-01  2.030e-01
## job_industry10       -0.026630  3.763e-01 -7.077e-02
## wealth_segment2       0.020463  1.075e-01  1.904e-01
## wealth_segment3      -0.228362  9.446e-02 -2.417e+00
## owns_car1             0.041328  7.606e-02  5.433e-01
## state2               -0.086367  1.057e-01 -8.167e-01
## state3                0.023564  9.367e-02  2.516e-01
## property_valuation   -0.028196  1.478e-02 -1.908e+00
##
## Intercepts:
##                       Value         Std. Error    t value
## most loyal|regular   -7.578000e-01  3.307000e-01 -2.291300e+00
## regular|hibernating   5.988000e-01  3.306000e-01  1.811100e+00
## hibernating|seasonal  2.080900e+00  3.345000e-01  6.221600e+00
##
## Residual Deviance: 5989.424
## AIC: 6033.424
```

### Addind P-values

```r
prob <- pnorm(abs(summary(model_ordinal_2)$coefficients[, 3]),
              lower.tail = FALSE)*2
cbind(summary(model_ordinal_2)$coefficients, prob)
```

```
##                                Value   Std. Error       t value          prob
## gender1                  0.065961177 7.601265e-02  8.677658e-01 3.855225e-01
## gender2                -11.809582796 2.062923e-07 -5.724684e+07 0.000000e+00
## age                      0.003328961 2.986422e-03  1.114699e+00 2.649796e-01
## past_purchases          -0.001709642 1.320882e-03 -1.294318e+00 1.955556e-01
## job_industry2           -0.058561499 3.217826e-01 -1.819909e-01 8.555899e-01
## job_industry3            0.039661958 2.615111e-01  1.516645e-01 8.794515e-01
## job_industry4            0.089783659 2.670339e-01  3.362257e-01 7.367007e-01
## job_industry5           -0.157290743 3.259955e-01 -4.824936e-01 6.294554e-01
## job_industry6            0.184157442 2.609496e-01  7.057203e-01 4.803621e-01
## job_industry7            0.176701605 2.640208e-01  6.692716e-01 5.033223e-01
## job_industry8            0.110321750 2.862292e-01  3.854315e-01 6.999178e-01
## job_industry9            0.056324533 2.775277e-01  2.029511e-01 8.391733e-01
## job_industry10          -0.026629939 3.762904e-01 -7.076965e-02 9.435811e-01
## wealth_segment2          0.020462949 1.074800e-01  1.903885e-01 8.490047e-01
## wealth_segment3         -0.228361673 9.446224e-02 -2.417492e+00 1.562789e-02
## owns_car1                0.041327929 7.606236e-02  5.433427e-01 5.868939e-01
## state2                  -0.086366690 1.057463e-01 -8.167346e-01 4.140801e-01
## state3                   0.023564169 9.366722e-02  2.515733e-01 8.013709e-01
## property_valuation      -0.028195610 1.478002e-02 -1.907684e+00 5.643204e-02
## most loyal|regular      -0.757760814 3.307174e-01 -2.291264e+00 2.194816e-02
## regular|hibernating      0.598780658 3.306166e-01  1.811103e+00 7.012493e-02
## hibernating|seasonal     2.080888190 3.344621e-01  6.221596e+00 4.921223e-10
```

**Taking p-value of 0.05 we get that only wealth segment and property valuation are significant.**

**Accuracy rate of the for the training data**

```r
predict(model_ordinal_2, train_data) %>%
  bind_cols(train_data) %>%
  rename(pred ="...1", truth = loyalty) %>%
  accuracy(pred, truth)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.360
```

**we get accuracy of 36%**

**Testing with the tesing set**

```
predict(model_ordinal_2, test_data) %>%
  bind_cols(test_data) %>%
  rename(pred ="...1", truth = loyalty) %>%
  accuracy(pred, truth)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.329
```

**We get accuracy of 33%, smaller than the testing set.**

**Trying with the probit**

```
set.seed(123)
model_probit_2 <- train(loyalty~gender+age+past_purchases+job_industry+
                           wealth_segment+owns_car+state+property_valuation,
                       data = train_data,
                       method = "polr",
                       tuneGrid = expand.grid(method = "probit"))
```

**Accuracy on training set**

```
predict(model_probit_2, train_data) %>%
  bind_cols(train_data) %>%
  rename(pred ="...1", truth = loyalty) %>%
  accuracy(pred, truth)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.359
```

**Accuracy of 36% on training set**

**Accuracy with the tesing set**

```r
predict(model_probit_2, test_data) %>%
  bind_cols(test_data) %>%
  rename(pred ="...1", truth = loyalty) %>%
  accuracy(pred, truth)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.329
```

**Accuracy of 33%**

**CART model**

```r
set.seed(123)
model_cart_2 <- train(loyalty~gender+age+past_purchases+job_industry+
                           wealth_segment+owns_car+state+property_valuation,
                      data = train_data,
                      method = "rpartScore")
saveRDS(model_cart_2, "model_cart_2.rds")
```

```r
model_cart_2 <- readRDS("model_cart_2.rds")
model_cart_2
```

```
## CART or Ordinal Responses
##
## 2312 samples
##    8 predictor
##    4 classes: 'most loyal', 'regular', 'hibernating', 'seasonal'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 2312, 2312, 2312, 2312, 2312, 2312, ...
## Resampling results across tuning parameters:
##
##   cp           split  prune  Accuracy   Kappa
##   0.004139073  abs    mr     0.3183866  -0.0016192012
##   0.004139073  abs    mc     0.3205697  -0.0020915569
##   0.004139073  quad   mr     0.3290718   0.0060158687
##   0.004139073  quad   mc     0.3219638  -0.0007812258
##   0.004635762  abs    mr     0.3206180  -0.0012868285
##   0.004635762  abs    mc     0.3224821  -0.0003111439
##   0.004635762  quad   mr     0.3302154   0.0052180984
##   0.004635762  quad   mc     0.3231367   0.0002599830
```

```
##    0.006092715  abs    mr      0.3245897   0.0003978784
##    0.006092715  abs    mc      0.3226198  -0.0003999741
##    0.006092715  quad   mr      0.3315642   0.0035061519
##    0.006092715  quad   mc      0.3227589  -0.0003037494
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were cp = 0.006092715, split = quad
##  and prune = mr.
```

**Accuracy**

```
predict(model_cart_2, train_data) %>%
  bind_cols(train_data) %>%
  rename(pred ="...1", truth = loyalty) %>%
  accuracy(pred, truth)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.347
```

**Accuracy on the training set is about 35%**

**On the testing set**

```
predict(model_cart_2, test_data) %>%
  bind_cols(test_data) %>%
  rename(pred ="...1", truth = loyalty) %>%
  accuracy(pred, truth)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.346
```

**Accuracy on the testing set is also about 35%**

**Continuation Ratio Model**

```
set.seed(123)
model_vgam_2 <- train(loyalty~gender+age+past_purchases+job_industry+
                           wealth_segment+owns_car+state+property_valuation,
```

```
                        data = train_data,
                        method = "vglmContRatio", trace = FALSE)
model_vgam_2
```

```
## Continuation Ratio Model for Ordinal Data
##
## 2312 samples
##    8 predictor
##    4 classes: 'most loyal', 'regular', 'hibernating', 'seasonal'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 2312, 2312, 2312, 2312, 2312, 2312, ...
## Resampling results across tuning parameters:
##
##   parallel  link     Accuracy   Kappa
##   FALSE     logit    0.3334104  0.006421815
##   FALSE     probit   0.3337045  0.006758306
##   FALSE     cloglog  0.3347265  0.008261142
##   FALSE     cauchit  0.3336412  0.007436043
##   FALSE     logc           NaN          NaN
##    TRUE     logit    0.3414795  0.015454859
##    TRUE     probit   0.3412608  0.014815421
##    TRUE     cloglog  0.3380711  0.010233244
##    TRUE     cauchit  0.3373274  0.011225365
##    TRUE     logc     0.3383719  0.002206090
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were parallel = TRUE and link = logit.
```

#### Accuracy on training data

```
predict(model_vgam_2, train_data) %>%
  bind_cols(train_data) %>%
  rename(pred ="...1", truth = loyalty) %>%
  accuracy(pred, truth)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.346
```

**Accuracy on training set about 35%**

**Accuracy on testing data**

```
predict(model_vgam_2, test_data) %>%
  bind_cols(test_data) %>%
  rename(pred ="...1", truth = loyalty) %>%
  accuracy(pred, truth)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.341
```

**Accuracy about 34%**

**The grouped data produces models that are not so different with the non grouped data models.**

## Customer List

**We will predict using the model_ordinal model on the grouped data on the new list**

```
new_customers <- read_csv("newcustomerlist_2.csv")
```

**rename columns and create groups**

```
new_customers_1 <- new_customers %>%
  rename(past_purchases = past_3_years_bike_related_purchases,
         job_industry = job_industry_category,
         deceased = deceased_indicator,
         dob = DOB)
new_customers_1$first_name[is.na(new_customers_1$first_name)] <- ""
new_customers_1$last_name[is.na(new_customers_1$last_name)] <- ""
new_customers_1 <- unite(new_customers_1, customer_name,
                         first_name:last_name, sep = " ")
new_customers_1$customer_name <- str_squish(new_customers_1$customer_name)
```

### Wealth Segment

### Shorten some classifications

### The 3 classes are Mass Customer, High Net Worth and Affluent Customer

```
new_customers_1 <- new_customers_1 %>%
    mutate(wealth_segment = case_when(
        str_detect(wealth_segment, "Affluent") ~ "Affluent",
        str_detect(wealth_segment, "High") ~ "High Net",
        str_detect(wealth_segment, "Mass") ~ "Mass",
        TRUE ~ wealth_segment
    ))
```

### Shorten some job industry names

```
new_customers_1 <- new_customers_1 %>%
    mutate(job_industry = case_when(
        str_detect(job_industry, "Financial") ~ "Financials",
        str_detect(job_industry, "Telecomm") ~ "Telecomms",
        TRUE ~ job_industry
    ))
```

### Groups-age__group,property__valuation group

###age group

```
new_customers_2 <- new_customers_1 %>%
  mutate(age_group = case_when(
    age <= 35 ~ "Youth",
    age > 35 & age <= 55 ~ "Middle",
    age > 55 ~ "Older"
  ))
```

### past purchases

```
new_customers_2 <- new_customers_2 %>%
  mutate(past_purchase_group = case_when(
    past_purchases <= 24 ~ "Bad",
    past_purchases > 24 & past_purchases <= 59 ~ "Good",
    past_purchases > 59 & past_purchases <= 84 ~ "Better",
    past_purchases >= 85 ~ "Excellent"
  ))
```

**Property valuation**

```
new_customers_2 <- new_customers_2 %>%
  mutate(pvaluation_group = case_when(
    property_valuation <= 3 ~ "Minimum",
    property_valuation > 3 & property_valuation <= 7 ~ "Average",
    property_valuation > 7 ~ "Wealthy"
  ))
```

**Missing values**

```
sum(is.na(new_customers_2))
```

```
## [1] 157
```

**Columns with missing values**

```
names(which(colSums(is.na(new_customers_2)) > 0))
```

```
## [1] "dob"       "job_title" "age"       "age_group"
```

**Number of missing values in age and age group**

```
sum(is.na(new_customers_2$age))
```

```
## [1] 17
```

```
sum(is.na(new_customers_2$age_group))
```

```
## [1] 17
```

**The 17 missing values in age_group resulted from the 17 missing values from age.**

**Replace the NAs in age_group with unknown**

```
new_customers_2$age_group[is.na(new_customers_2$age_group)] <- "unknown"
```

**The other columns with missing values can be untouched**

**Checking the data**

```
new_customers_2 %>% count(gender, sort = T)
```

```
## # A tibble: 3 x 2
##   gender     n
##   <chr>  <int>
## 1 Female   513
## 2 Male     470
## 3 U         17
```

```
new_customers_2 %>% count(job_industry, sort = T)
```

```
## # A tibble: 10 x 2
##    job_industry      n
##    <chr>         <int>
##  1 Financials      203
##  2 Manufacturing   199
##  3 n/a             165
##  4 Health          152
##  5 Retail           78
##  6 Property         64
##  7 IT               51
##  8 Entertainment    37
##  9 Argiculture      26
## 10 Telecomms        25
```

**We hane n/a in job_industry, replace with unknown**

```
new_customers_2$job_industry[new_customers_2$job_industry == "n/a"] <- "unknown"
```

**wealth segment**

```
new_customers_2 %>% count(wealth_segment, sort = T)
```

```
## # A tibble: 3 x 2
##   wealth_segment     n
##   <chr>          <int>
## 1 Mass             508
## 2 High Net         251
## 3 Affluent         241
```

**State**

```
new_customers_2 %>% count(state, sort = T)
```

```
## # A tibble: 3 x 2
##   state     n
##   <chr> <int>
## 1 NSW     506
## 2 VIC     266
## 3 QLD     228
```

**Past 3 years bike related purchase**

```
new_customers_2 %>% count(past_purchase_group, sort = T)
```

```
## # A tibble: 4 x 2
##   past_purchase_group     n
##   <chr>               <int>
## 1 Good                  366
## 2 Better                281
## 3 Bad                   229
## 4 Excellent             124
```

**Property Valuation**

```
new_customers_2 %>% count(pvaluation_group, sort = T)
```

```
## # A tibble: 3 x 2
##   pvaluation_group     n
##   <chr>            <int>
## 1 Wealthy            559
## 2 Average            318
## 3 Minimum            123
```

```
new_customers_3 <- new_customers_2
new_customers_3$age_group <- as.integer(factor(new_customers_3$age_group))
new_customers_3$job_industry <- as.integer(factor(new_customers_3$job_industry))
new_customers_3$wealth_segment <- as.integer(factor(new_customers_3$wealth_segment))
new_customers_3$owns_car <- as.integer(factor(new_customers_3$owns_car))
new_customers_3$state <- as.integer(factor(new_customers_3$state))
new_customers_3$past_purchase_group <- as.integer(factor(new_customers_3$past_purchase_group))
new_customers_3$pvaluation_group <- as.integer(factor(new_customers_3$pvaluation_group))
str(new_customers_3)
```

```
## tibble [1,000 x 19] (S3: tbl_df/tbl/data.frame)
##  $ customer_name      : chr [1:1000] "Chickie Brister" "Morly Genery" "Ardelis Forrester" "I
##  $ gender             : chr [1:1000] "Male" "Male" "Female" "Female" ...
##  $ past_purchases     : num [1:1000] 86 69 10 64 34 39 23 74 50 72 ...
##  $ dob                : Date[1:1000], format: "1957-07-12" "1970-03-22" ...
##  $ job_title          : chr [1:1000] "General Manager" "Structural Engineer" "Senior Cost Ac
##  $ job_industry       : int [1:1000] 6 7 3 6 3 2 3 8 6 5 ...
##  $ wealth_segment     : int [1:1000] 3 3 1 1 1 2 3 3 3 3 ...
##  $ deceased           : chr [1:1000] "N" "N" "N" "N" ...
##  $ owns_car           : int [1:1000] 2 1 1 2 1 2 1 2 2 2 ...
##  $ tenure             : num [1:1000] 14 16 10 5 19 22 8 10 5 17 ...
##  $ address            : chr [1:1000] "45 Shopko Center" "14 Mccormick Park" "5 Colorado Cros
##  $ postcode           : num [1:1000] 4500 2113 3505 4814 2093 ...
##  $ state              : int [1:1000] 2 1 3 2 1 2 1 2 1 2 ...
##  $ country            : chr [1:1000] "Australia" "Australia" "Australia" "Australia" ...
##  $ property_valuation : num [1:1000] 6 11 5 1 9 7 7 5 10 5 ...
##  $ age                : num [1:1000] 66 53 49 44 58 72 47 50 51 38 ...
##  $ age_group          : int [1:1000] 2 1 1 1 2 2 1 1 1 1 ...
##  $ past_purchase_group: int [1:1000] 3 2 1 2 4 4 1 2 4 2 ...
##  $ pvaluation_group   : int [1:1000] 1 3 1 2 3 1 1 1 3 1 ...
```

```
new_customers_4 <- new_customers_3 %>% dplyr::select(1,2,6,7,9,13,17:19)
```

```
new_customers_4 <- new_customers_4 %>%
  mutate_if(is.numeric, as.factor)
str(new_customers_4)
```

```
## tibble [1,000 x 9] (S3: tbl_df/tbl/data.frame)
##  $ customer_name      : chr [1:1000] "Chickie Brister" "Morly Genery" "Ardelis Forrester" "I
##  $ gender             : chr [1:1000] "Male" "Male" "Female" "Female" ...
##  $ job_industry       : Factor w/ 10 levels "1","2","3","4",..: 6 7 3 6 3 2 3 8 6 5 ...
##  $ wealth_segment     : Factor w/ 3 levels "1","2","3": 3 3 1 1 1 2 3 3 3 3 ...
##  $ owns_car           : Factor w/ 2 levels "1","2": 2 1 1 2 1 2 1 2 2 2 ...
##  $ state              : Factor w/ 3 levels "1","2","3": 2 1 3 2 1 2 1 2 1 2 ...
##  $ age_group          : Factor w/ 4 levels "1","2","3","4": 2 1 1 1 2 2 1 1 1 1 ...
##  $ past_purchase_group: Factor w/ 4 levels "1","2","3","4": 3 2 1 2 4 4 1 2 4 2 ...
##  $ pvaluation_group   : Factor w/ 3 levels "1","2","3": 1 3 1 2 3 1 1 1 3 1 ...
```

**use the Ordinal Logistic Regression model,model_ordinal, that had an accuracy of 36% on training set and 34% on testing set**

```
new_customers_5 <- new_customers_2
new_customers_5$loyalty <- predict(model_ordinal, newdata = new_customers_4)
```

#### We have predicted the classification of the different 1000 new customers

#### Loyalty count

```r
class(new_customers_5)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```r
new_customers_5 <- data.frame(new_customers_5)
class(new_customers_5)
```

```
## [1] "data.frame"
```

```r
write_csv(new_customers_5, "new_customers_listings.csv")
```

```r
new_customers_5 %>% count(loyalty, sort = T)
```

```
##       loyalty   n
## 1  most loyal 591
## 2     regular 407
## 3 hibernating   2
```

```r
most_loyal_customers <- new_customers_5 %>% filter(loyalty == "most loyal")
regular_customers <- new_customers_5 %>% filter(loyalty == "regular")
hibernating_customers <- new_customers_5 %>% filter(loyalty == "hibernating")
```

```r
write_csv(most_loyal_customers, "most_loyal_customers.csv")
write_csv(regular_customers, "regular_customers.csv")
write_csv(hibernating_customers, "hibernating_customers.csv")
```

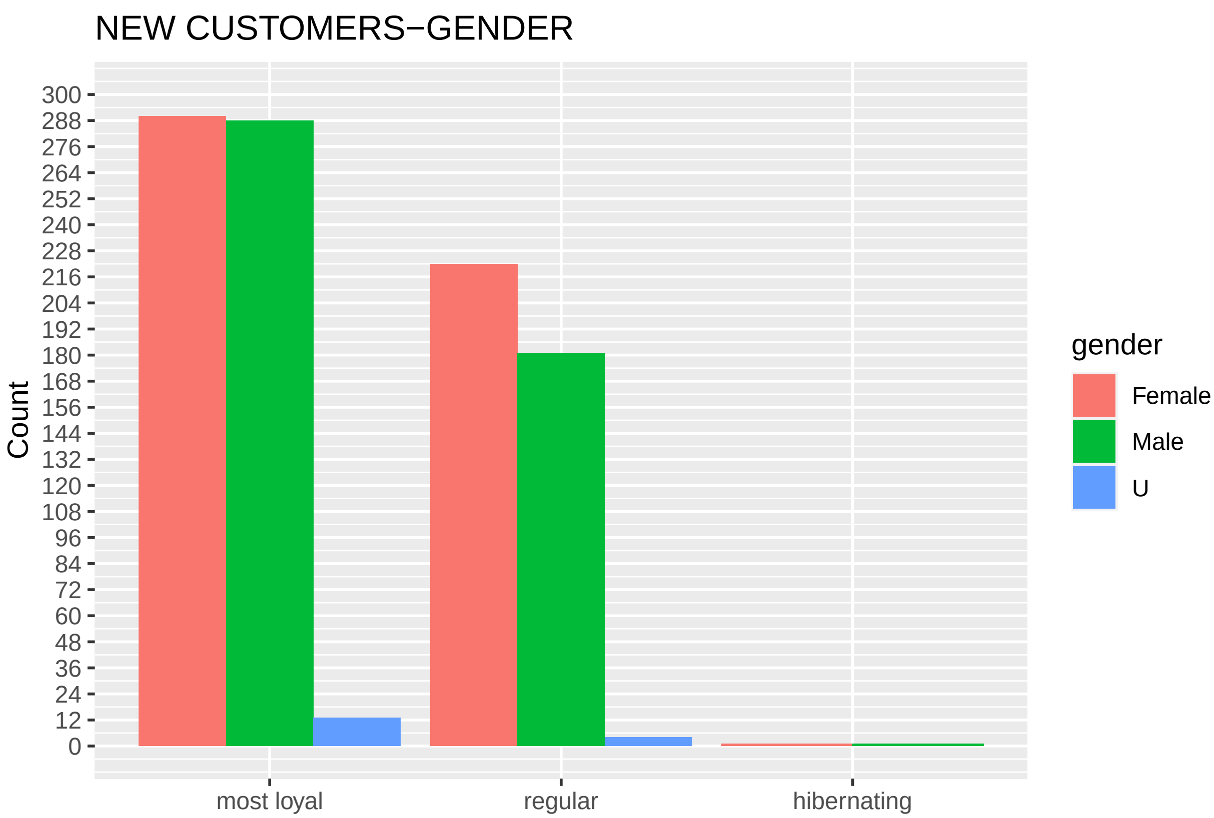
```r
library(ggrepel)
library(ggfortify)
```

- Most Loyal-These customers have the potential of being the most loyal. They can be targeted differently with potential royalty rewards and there branding needs understood.
- Regular-These are customers from the new list that have the potential of being regular customers. They have the potential of being the most loyal too.
- Hibernating-They are the fewest and some work can be put on them with several offers.

#### Different customer behavior

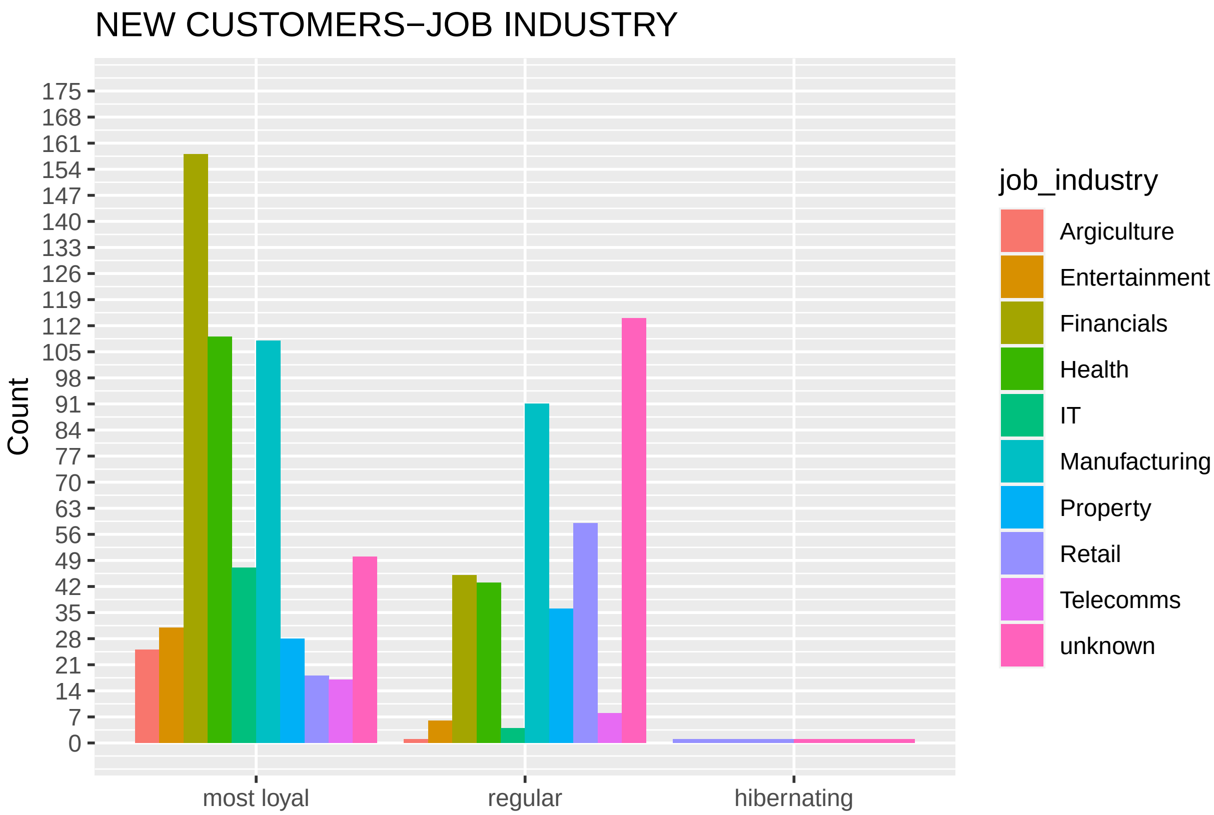
```r
new_customers_data <- new_customers_5
```

```r
new_gender <- new_customers_data %>% group_by(loyalty) %>% count(gender)
ggplot(new_gender, aes(loyalty, n, fill = gender)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_y_continuous("Count",
    breaks = seq(0, 300, by = 12),
    limits = c(0, 300)) +
  labs(title = "NEW CUSTOMERS-GENDER", x = "")
```



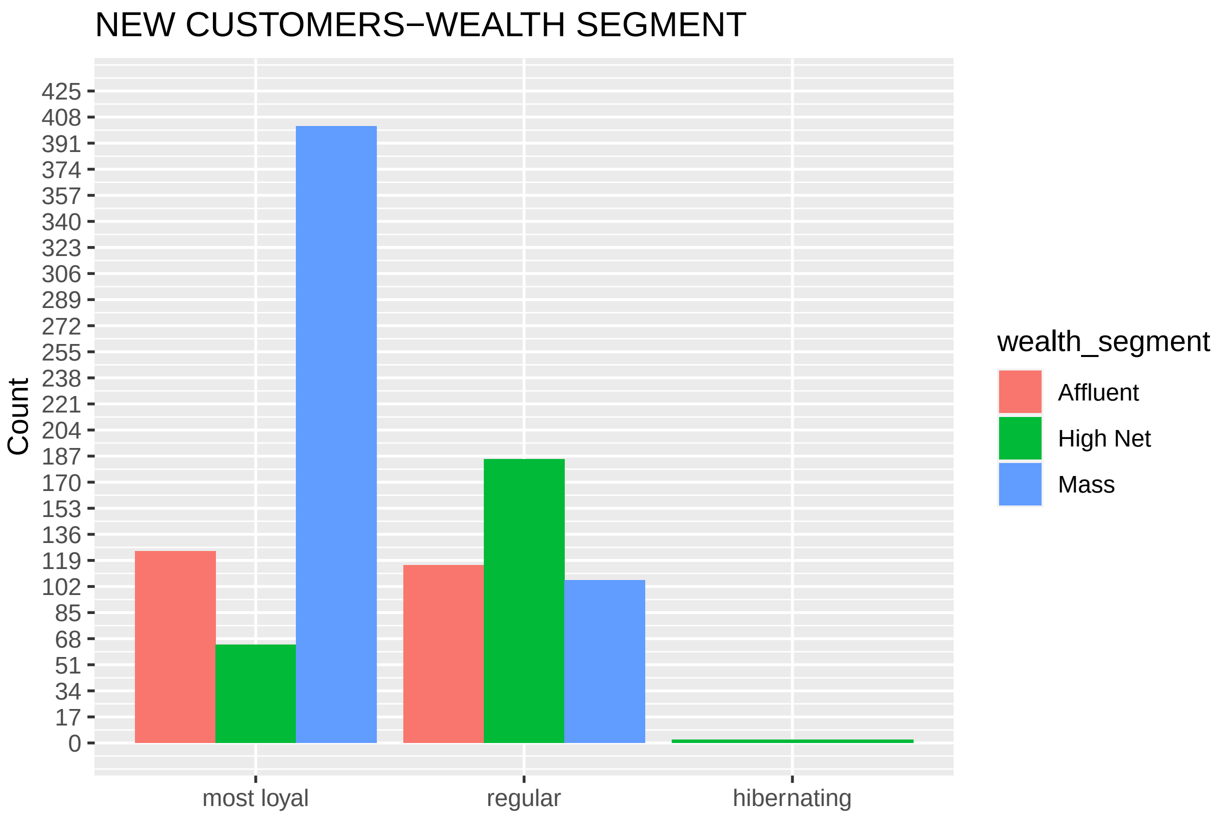

**Gender should not be a focus in trying to get new customers to be consistent buyers.**

```r
new_industry <- new_customers_data %>% group_by(loyalty) %>% count(job_industry)
ggplot(new_industry, aes(loyalty, n, fill = job_industry)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_y_continuous("Count",
    breaks = seq(0, 175, by = 7),
    limits = c(0, 175)) +
  labs(title = "NEW CUSTOMERS-JOB INDUSTRY", x = "")
```



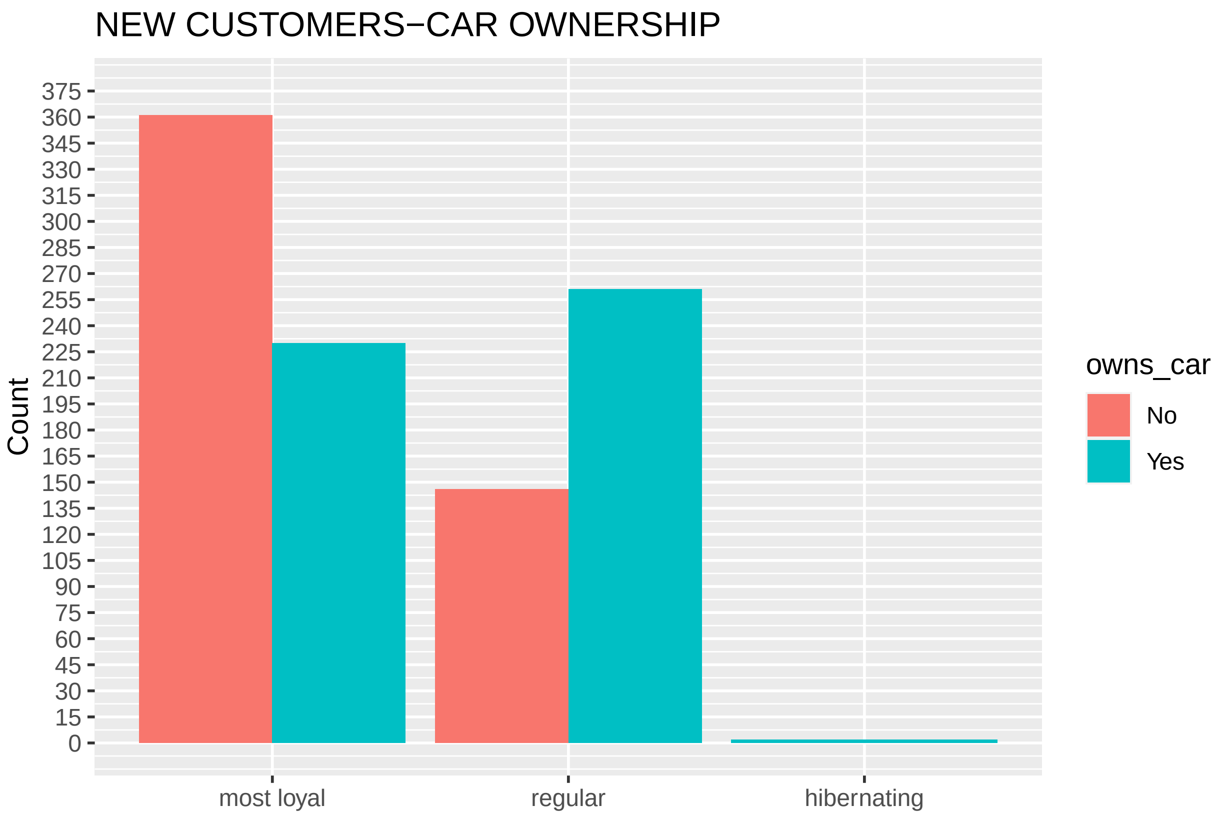

**Customers working in the Financial Services, Health and Manufacturing have the largets potential of being very loyal and regular. Those in the retail and those whose industry is unknown or are unwilling to disclose their job industry category are less likely to be regular or loyal.**

```
new_wealth <- new_customers_data %>% group_by(loyalty) %>% count(wealth_segment)
ggplot(new_wealth, aes(loyalty, n, fill = wealth_segment)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_y_continuous("Count",
    breaks = seq(0, 425, by = 17),
    limits = c(0, 425)) +
  labs(title = "NEW CUSTOMERS-WEALTH SEGMENT", x = "")
```



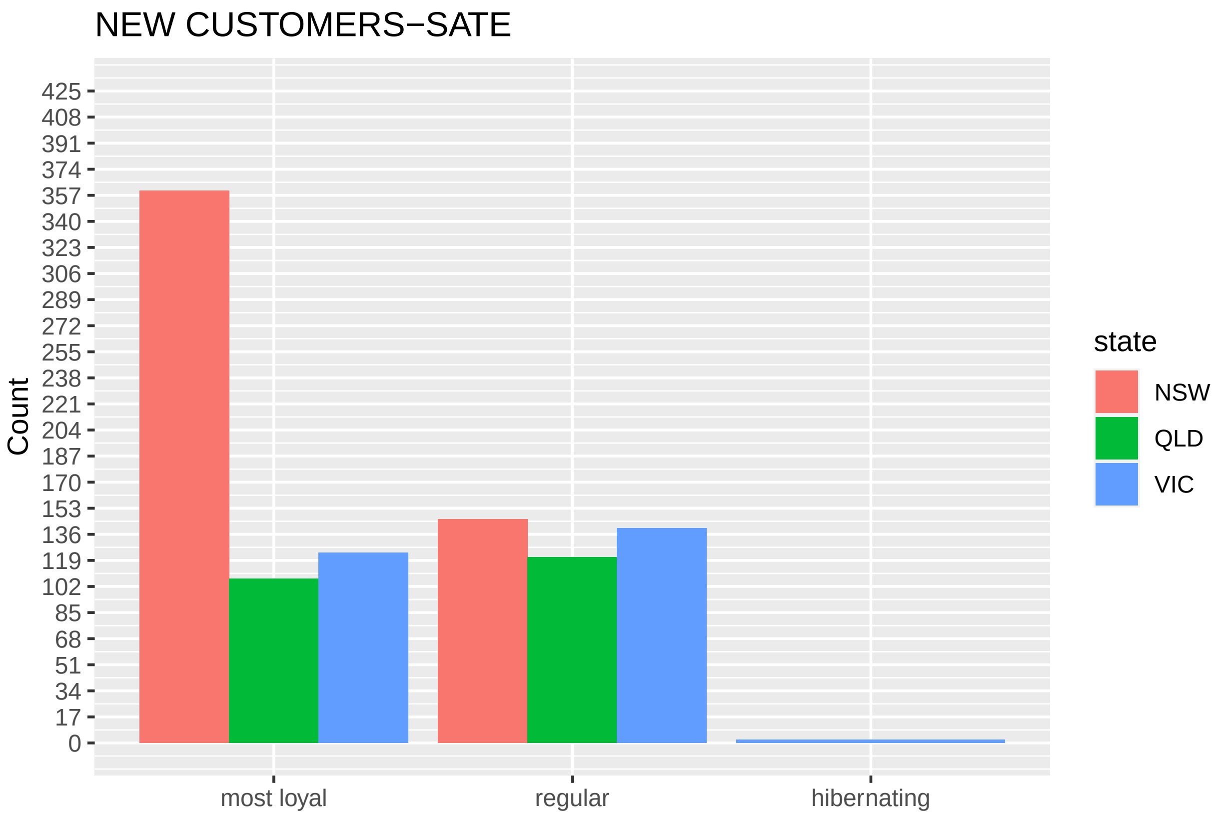

**Mass customers are good customers.**

```
new_car <- new_customers_data %>% group_by(loyalty) %>% count(owns_car)
ggplot(new_car, aes(loyalty, n, fill = owns_car)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_y_continuous("Count",
    breaks = seq(0, 375, by = 15),
    limits = c(0, 375)) +
  labs(title = "NEW CUSTOMERS-CAR OWNERSHIP", x = "")
```



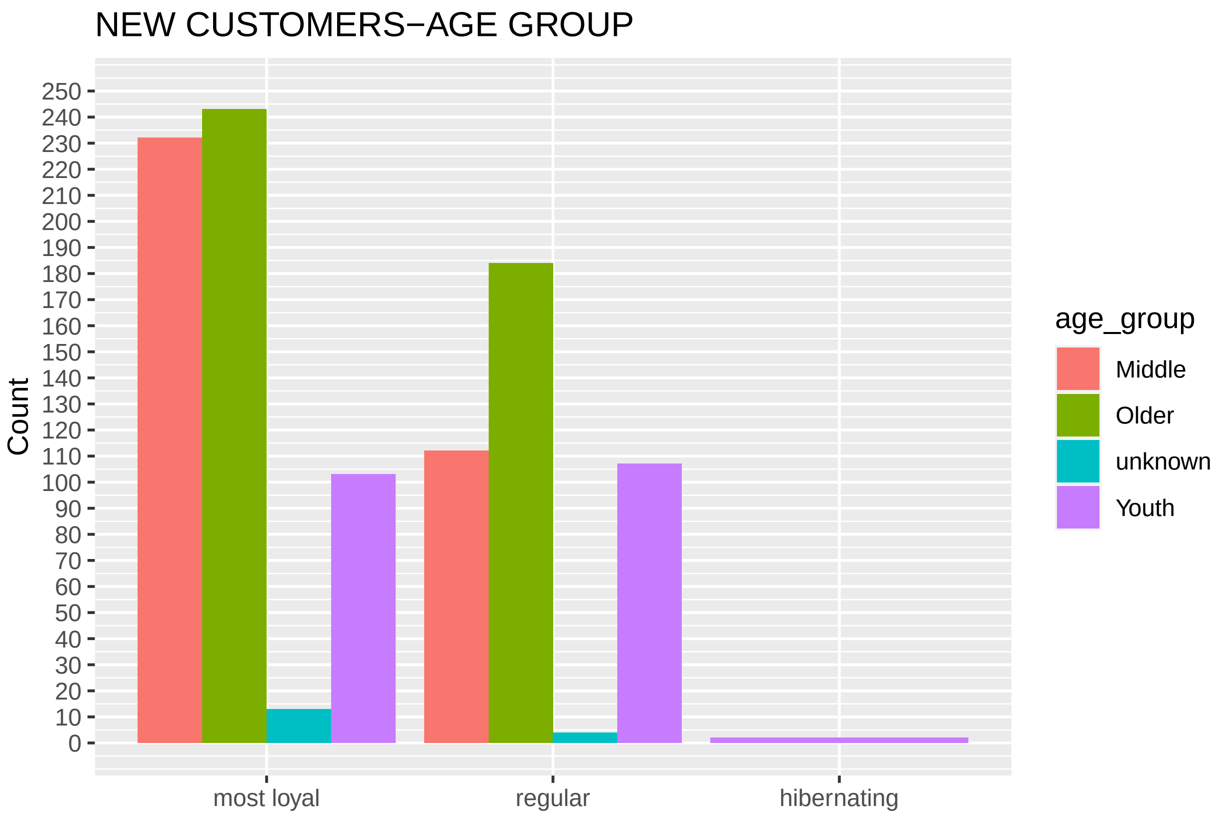

**It seems like car ownership might not affect the visits, thus customers should be targeted whether they own or do not own a car.**

```
new_state <- new_customers_data %>% group_by(loyalty) %>% count(state)
ggplot(new_state, aes(loyalty, n, fill = state)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_y_continuous("Count",
    breaks = seq(0, 425, by = 17),
    limits = c(0, 425)) +
  labs(title = "NEW CUSTOMERS-SATE", x = "")
```



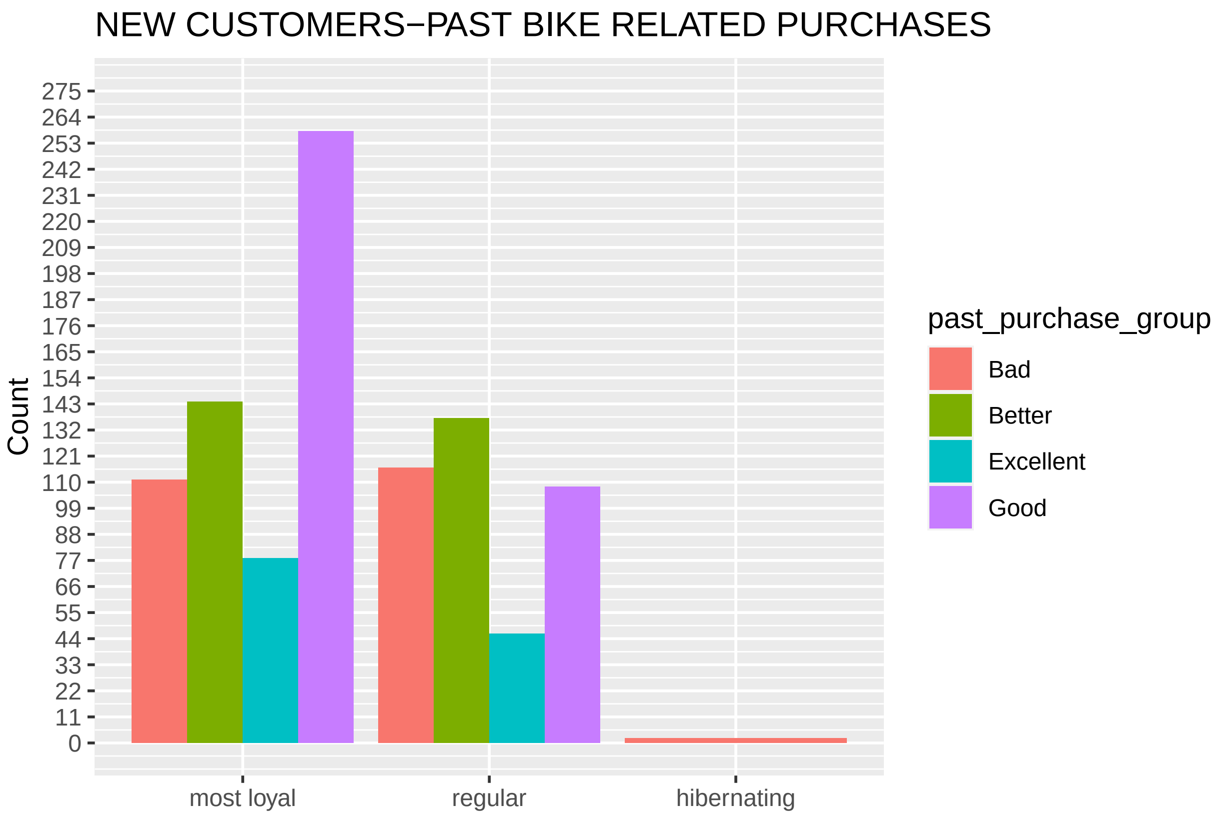

**Customers from New South Wales state are more likely to be regular and very loyal customers.**

```
new_agegroup <- new_customers_data %>% group_by(loyalty) %>% count(age_group)
ggplot(new_agegroup, aes(loyalty, n, fill = age_group)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_y_continuous("Count",
    breaks = seq(0, 250, by = 10),
    limits = c(0, 250)) +
  labs(title = "NEW CUSTOMERS-AGE GROUP", x = "")
```




**Individuals who are over 50 are more likely to be regular and very loyal.**

```
new_past <- new_customers_data %>% group_by(loyalty) %>%
  count(past_purchase_group)
ggplot(new_past, aes(loyalty, n, fill = past_purchase_group)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_y_continuous("Count",
    breaks = seq(0, 275, by = 11),
    limits = c(0, 275)) +
  labs(title = "NEW CUSTOMERS-PAST BIKE RELATED PURCHASES", x = "")
```




**Those who have purchased bike related items in the last 3 years more than 25 times are more likely to be very loyal and regular.**

```
new_valuation <- new_customers_data %>% group_by(loyalty) %>%
  count(pvaluation_group)
ggplot(new_valuation, aes(loyalty, n, fill = pvaluation_group)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_y_continuous("Count",
    breaks = seq(0, 425, by = 17),
    limits = c(0, 425)) +
  labs(title = "NEW CUSTOMERS-PROPERTY VALUATION", x = "")
```


**Customers with property valuation of 4 and above are more likely to be regular and very loyal customers.**

- Gender should not be a focus in trying to get new customers to be consistent buyers.
- Customers working in the Financial Services, Health and Manufacturing have the largets potential of being very loyal and regular. Those in the retail and those whose industry is unknown or are unwilling to disclose their job industry category are less likely to be regular or loyal. Different industry shopuld be targetted different in advertising.
- Mass customers can very loyal while high net worth customers can be regular.
- Car ownership should not be considered while targetting the new customers even though customers who own cars are more likely to be regular customers.
- Customers from NWS can be converted to very loyal customers.
- Individuals who are over 50 are more likely to be regular and very loyal. Special marketing for the Youths should be done.
- Those who have purchased bike related items in the last 3 years more than 25 times are more likely to be very loyal and regular.
- Customers with property valuation of 4 and above are more likely to be regular and very loyal customers.