Business Understanding:

The frozen food market in the US was valued at over $55 billion in 2021, 40.7% of which can be attributed to ready meals.[1] In order to capture customers it's important to understand who the target segment should include. Key questions to answer include:

1. What does the distribution of frozen dinner expenditures look like?
2. What are the key customer segments of this market and whom should we target?
3. How can we predict whether or not a household belongs to our target segment?

Data Understanding:

The distribution of yearly frozen dinner expenditures is skewed heavily to the right with almost two thirds of households not purchasing frozen meals at all. Discounting the households that didn't purchase frozen dinners, there remain 115 outlier households that spent in excess of $382 on frozen dinners over the course of the year with the maximum spender purchasing over $50,000 worth of frozen dinners (Appendix A).

This abnormal distribution (Image #1) should be taken into consideration when deciding how to segment these households effectively. Along with this, all categorical variables should be transformed into dummy variables in order to be used for modeling purposes.

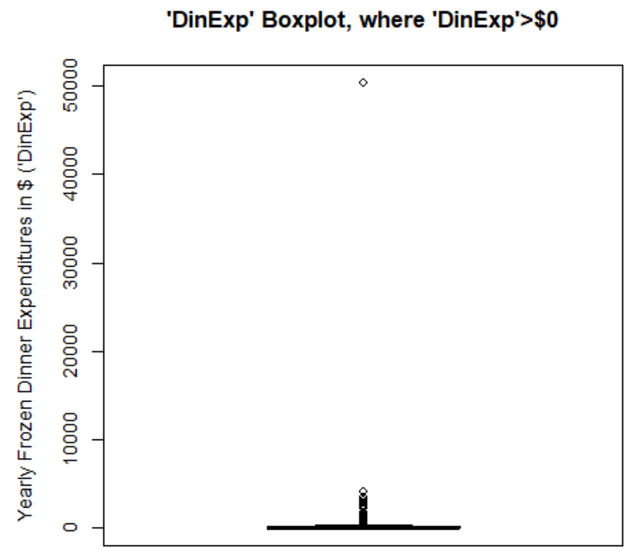


Image #1. 'DinExp' Distribution

Data Preparation:

Assuming that one frozen dinner costs $5, we categorized households into five manually defined bins based on the average number of frozen dinners purchased per week.[2] Scaling of the bins was loosely based on average meals per time period, so meals per year, meals per month, meals per week, and meals per day (Appendix B).

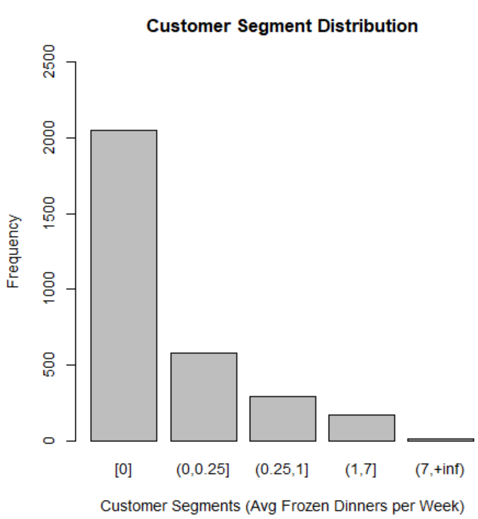


Image #2. Customer Segment Distribution.

The distribution after segmentation is still skewed (Image #2), but much less so than before. In order to see which segment has the greatest TPV and thus which segment we should focus on marketing to, we grouped total frozen dinner expenditures by segment and found the total amount spent for each category (Appendix C).

This visualization (Image #3) shows us that households that purchase between 1 and 7 frozen dinners per week on average have the highest yearly Total Payment Volume (TPV), and thus are the segment we should be targeting. To further prepare our data for modeling, we exclude all observations containing NA values and transform the five categorical variables to binary dummy variables (Appendix D).

---

[1] https://www.grandviewresearch.com/industry-analysis/us-frozen-food-market

[2] https://www.foodnavigator-usa.com/Article/2022/10/19/frozen-food-prices-up-17-vs.-2021-but-consumers-still-believe-it-provides-good-value-says-acosta

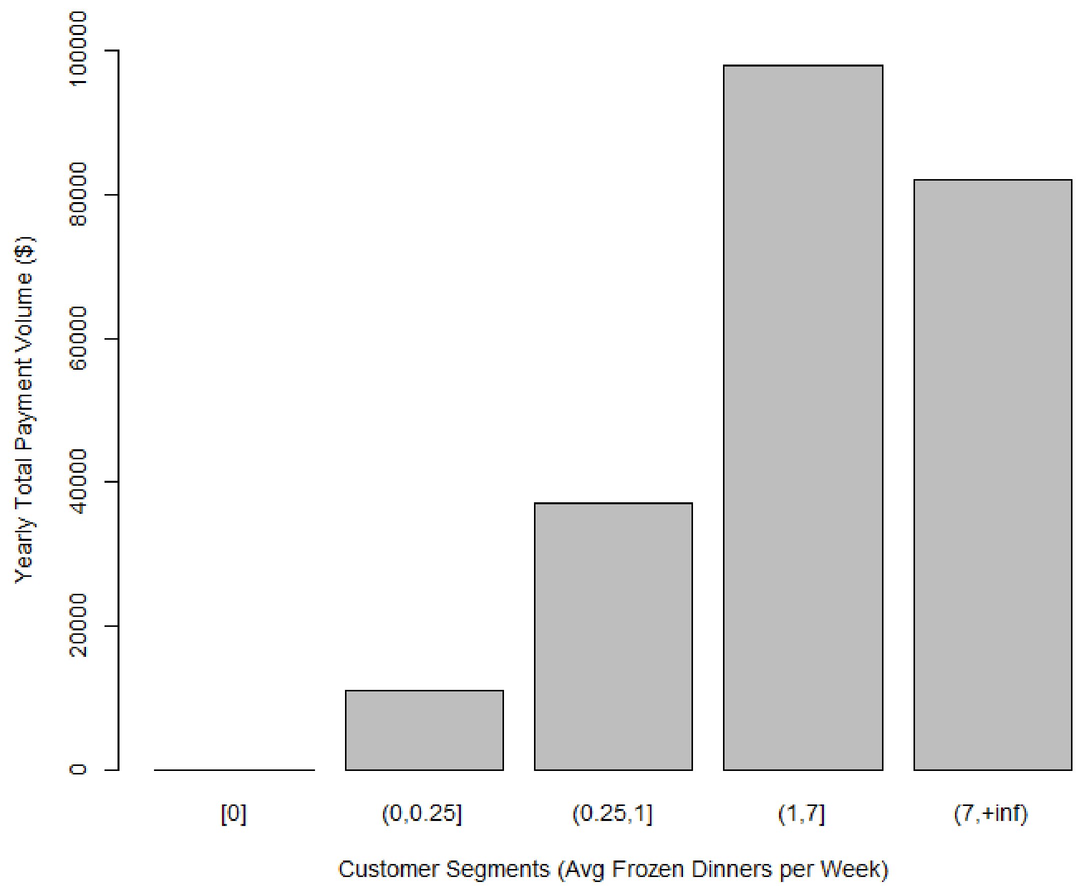


Image #3. TPV by Customer Segment

Modeling:

As we were looking to classify which of the five segments any household may belong to, the first model we built was a KNN model aiming to classify observations into one of the five customer segments. To do this we scaled the explanatory variables accordingly before splitting our data into 60% training data and 40% validation data. To train our first model 'mdf_KNN_fit' we used 10 cross validation folds and tested numbers 1 through 10 for potential values of K. However, after discovering that our resulting overall accuracy score was just over 64% with sensitivity and specificity rates close to 0% and 100% for each category, we decided to try a wider range of values (Appendix E). According to a tutorial found online, a good bet for finding the optimal value of K is to try the square root of the available sample size.[3] As our training data consisted of approximately 1,600 observations, we decided to test between 1 and 40 for possible values of K (Appendix F). The new model 'mdf_KNN_fit1' with K=40 had an incrementally better accuracy score of just over 65%, but its sensitivity and specificity rates were suboptimal at essentially either zero or 100% for each segment yet again. We decided to transform our response variable into a binary categorical variable, where '1' means that the household belongs to our target segment and '0' means the opposite (Appendix G). While the accuracy for this model 'mdf_KNN_fit2' was much higher at ~94%, the results were still inadequate as sensitivity was zero. This essentially told us that our model was predicting every observation to be non-target. In order to adjust for this, we changed the default cutoff for this same model to one equal to the proportion of positive observations in the target class (Appendix H). While the resulting accuracy score was lower at ~49%, the sensitivity and specificity scores were much more realistic and actionable (69.6% and 47.4% respectively). With sensitivity having this value, we know that 69.6% of positive classifications are true.

---

[3] https://quantdev.ssri.psu.edu/sites/qdev/files/kNN_tutorial.html#:~:text=Note%3A%20We%20use%20k%2DNN,when%20predicting%20a%20continuous%20outcome.

Following this KNN model ('mdf_KNN_fit2'), we built a logistic regression model 'm1' to compare performance statistics, once again using the binary classification of target segment vs. non-target segment as our response variable (Appendix I). For the first iteration of the model we included as many explanatory variables as possible. Results showed that a significant number of our explanatory variables were perfectly correlated, so we removed all variables with a correlation coefficient higher than 0.75 (Appendix J). As this new model 'm2' had higher accuracy, sensitivity, and specificity scores (70.0%, 69.8% and 70.1% respectively) than our best KNN model 'mdf_KNN_fit2', we continued testing variations of this logistic regression model. In the next variation we removed all variables describing male and female education levels to see how it would affect the model's performance (Appendix K). The resulting performance scores of 'm3' were lower across the board at 61.8%, 65.1% and 61.7% (accuracy, sensitivity and specificity), so we removed the housing income dummy variables and the dummy variable 'OthResStatus' due to their high standard errors (Appendices K, L). All performance measures remained relatively the same as 'm3' for this new model, 'm4'. We then returned to model 'm2' as our best performing logistic regression model and compared it to the best performing KNN model 'mdf_KNN_fit2' with a cumulative lift chart (Appendix M). A cumulative lift chart is a visualization showing the performance of a selected model versus random selection over different numbers of observations.

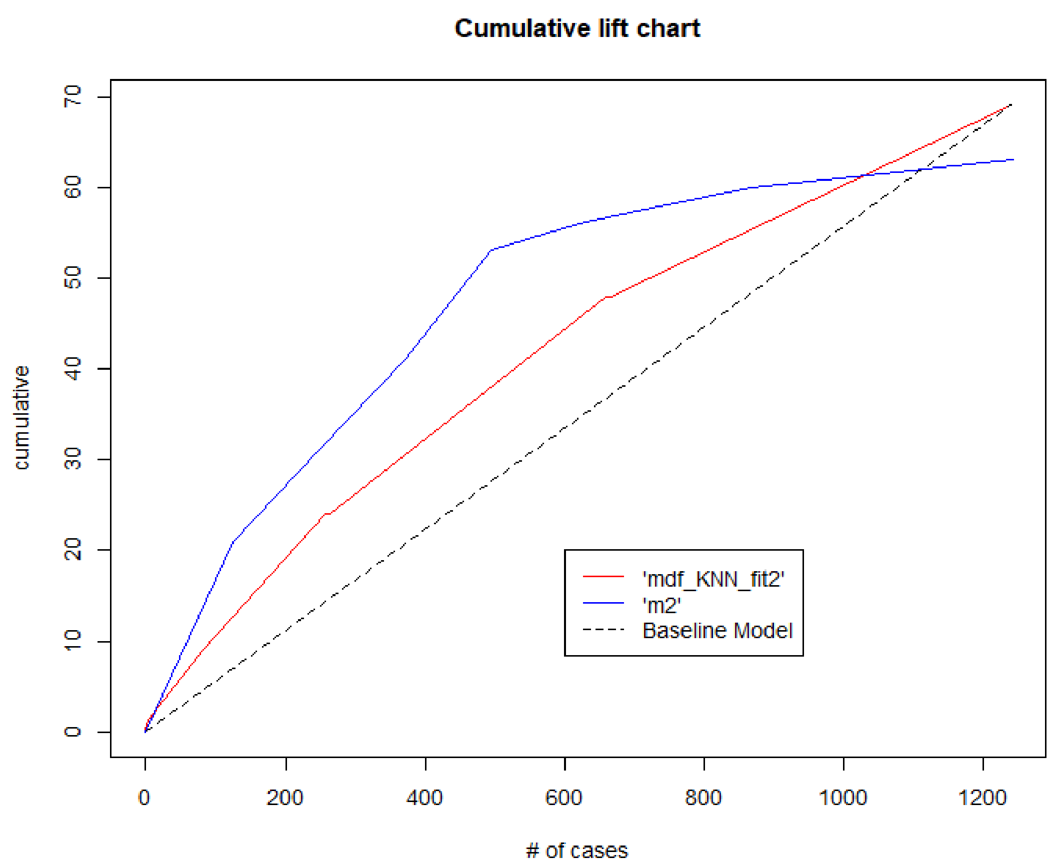


Image #4. Cumulative lift chart

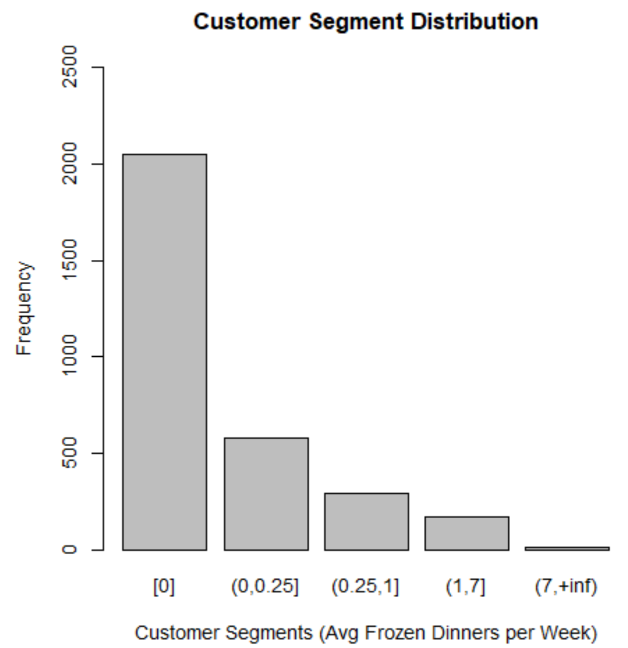
As seen in the chart above (Image #4), 'm2' performs much better than the baseline and 'mdf_KNN_fit2' when we select under 1000 observations. As the number of observations approaches 1200, the performance of 'm2' drops below that of random selection while that of 'mdf_KNN_fit2' approaches the same performance as random selection.

Evaluation:

By using the 'm2' model, we are able to predict and identify whether or not households belong to our target segment accurately over 70% of the time. Although the model is not 100% accurate, it moderately achieves the business objectives we outlined in the business understanding phase. We recommend that in future expansion efforts, strategy executives focus on acquiring customers from households that purchase an average of 1 to 7 frozen dinners per week, or households with total yearly frozen dinner spending between $240 and $1680 as these customers are part of the segment with the largest total payment volume. While the 'm2' logistic regression model is appropriate for our current usage of loosely classifying target segment status, we recommend further improvements in accuracy, sensitivity and specificity for future utilization. Additionally, the model should not be used on its own to classify households; rather, it should be used in conjunction with other data and analyses that provide a holistic picture of whether or not households belong to our target segment.

Deployment:
The data provided describing demographic and purchasing information for households in the midwestern US presents us with strong evidence for potential business opportunities in the US frozen food market. The frozen food market in the US was valued at over $55 billion in 2021, 40.7% of which can be attributed to ready meals.[1] In order to capture customers from this market it's important to understand what the current market status looks like. Key insights from the data include:

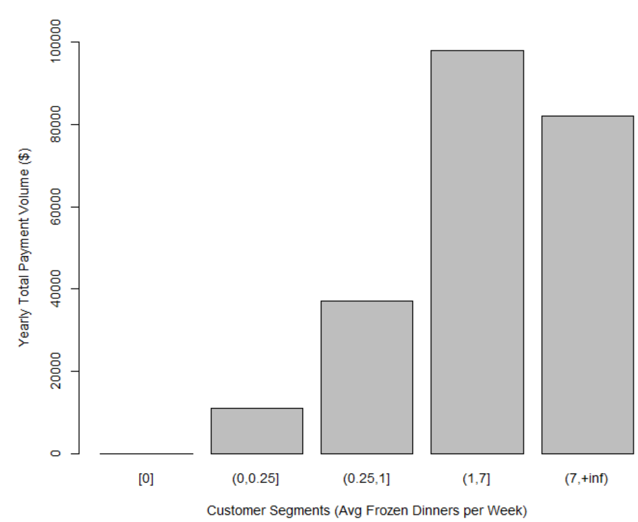
- Almost <u>2/3 of surveyed households purchased no frozen dinners</u>
- Of the third that did purchase frozen dinners, the <u>most common purchase frequency was <=1 meal/month</u>
- The customer segment with the <u>highest yearly TPV</u> was <u>households who purchase anywhere from 1 to 7 frozen dinners per week</u>



These insights provide two unique opportunities, for which we have prepared recommendations:

1. If we are looking to enter the frozen dinner market, our first target consumer segment should be the one with the most present cash flow, even if it doesn't have the highest concentration of customers. In order to determine which households belong to this segment, we have <u>developed a statistical model based on demographic and purchasing data that can predict whether or not a household is in our target segment with over 70% accuracy</u>. As it is not 100% accurate, this model is designed to be used in conjunction with other data and analyses to ensure that money spent on marketing towards target customers is not wasted.

2. If we are looking to gain a sustainable competitive advantage in the frozen dinner market, next steps should include <u>brainstorming ways to convert the 2/3 of surveyed households that purchased no frozen dinners into paying customers</u>. We could begin this by building a model to identify which non-paying households are most similar to paying households, and thus are most likely to become paying customers. Additionally, if we were to come into possession of data describing household purchasing behavior over time we could analyze adoption patterns to further develop the strength of this model.



Thank you for your time and please feel free to contact us if you would like more detailed information regarding this report.

Denise Ho & Ryan Innamorati

Appendix

A.

```
#Final Project
#Packages
library(tidyverse)
library(readxl)
library(caret)
library(gains)
library(pROC)
library(ROCR)
options(scipen=999)

#EDA (Exploratory Data Analysis)
df <- read_excel("ERIMData.xlsx") %>%
  as_tibble()
head(df)
df$DinExp %>%
  summary()
# we can see that this distribution is heavily skewed towards the right, and yet a majority of
# households spend $0 on frozen dinners. We separate the data from those who spend $0 to see what
# the distribution looks like.
df[df$DinExp > 0,] %>%
  count() / count(df)
# it appears that almost 2/3 of the data did not purchase frozen meals over the course of a year.
# let's look only at those that did.
purchases <- df[df$DinExp > 0,]
bp <- purchases$DinExp %>%
  boxplot()

bp$out %>%
  as_tibble() %>%
  arrange(desc(-value))
# there are 115 outliers >$382
```

```
bp$out %>%
  as_tibble() %>%
  arrange(desc(-value))
A tibble: 115 x 1
  value
  <dbl>
1  382.
2  385.
3  390.
4  390.
5  397.
6  400.
7  410.
8  411.
9  412.
0  415.
```

```
df$DinExp %>%
  summary()
  Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
  0.00     0.00     0.00    73.27    11.90 50312.60
```

B.

```
# Purchasing behavior bins (dinPurBins):
# Bin 1: [0] Spend $0/week (0 mpy)
# Bin 2: (0,60] (<=1 meal per month)
# Bin 3: (60,240] (<=1 meal per week)
# Bin 4: (240,1680] (<=1 meal per day)
# Bin 5: (1680,] (>1 mpd)

dinPurBins <- c(0,0.01,60,240,1680,55000)
df$mpwBins <- cut(
  df$DinExp,
  breaks = dinPurBins,
  labels = c("[0]","(0,0.25]","(0.25,1]","(1,7]","(7,+inf)"),
  include.lowest = TRUE,
  right = FALSE
)
table(df$mpwBins) %>%
  barplot(
    main="Customer Segment Distribution",
    ylim=c(0,2500),
    xlab="Customer Segments (Avg Frozen Dinners per Week)",
    ylab="Frequency"
  )
# The distribution after segmentation is still skewed, but much less so. Let's look and see which
# segment has the greatest TPV to see which segment we should focus on marketing to.
```

C.

```r
grouped_tpv <- df %>%
  group_by(mpwBins) %>%
  summarise(tpv = sum(DinExp))
grouped_tpv$tpv %>%
  barplot(
    names.arg = grouped_tpv$mpwBins,
    ylim = c(0,100000),
    ylab = "Yearly Total Payment Volume ($)",
    xlab = "Customer Segments (Avg Frozen Dinners per Week)"
  )
```

D.

```r
#exclude all NA values:
df <- na.exclude(df)

# dummy variables:
df$AptResType <- ifelse(df$ResType==1,1,0)
df$ConResType <- ifelse(df$ResType==2,1,0)
df$SFResType <- ifelse(df$ResType==3,1,0)
df$MFResType <- ifelse(df$ResType==4,1,0)
df$MobResType <- ifelse(df$ResType==5,1,0)
df$OthResType <- ifelse(df$ResType==6,1,0)

df$OwnResStatus <- ifelse(df$ResStatus==1,1,0)
df$RenResStatus <- ifelse(df$ResStatus==2,1,0)
df$OthResStatus <- ifelse(df$ResStatus==3,1,0)

df$HHInc1 <- ifelse(df$HHInc==1,1,0)
df$HHInc2 <- ifelse(df$HHInc==2,1,0)
df$HHInc3 <- ifelse(df$HHInc==3,1,0)
df$HHInc4 <- ifelse(df$HHInc==4,1,0)
df$HHInc5 <- ifelse(df$HHInc==5,1,0)
df$HHInc6 <- ifelse(df$HHInc==6,1,0)
df$HHInc7 <- ifelse(df$HHInc==7,1,0)
df$HHInc8 <- ifelse(df$HHInc==8,1,0)
df$HHInc9 <- ifelse(df$HHInc==9,1,0)
df$HHInc10 <- ifelse(df$HHInc==10,1,0)
df$HHInc11 <- ifelse(df$HHInc==11,1,0)
df$HHInc12 <- ifelse(df$HHInc==12,1,0)
df$HHInc13 <- ifelse(df$HHInc==13,1,0)
df$HHInc14 <- ifelse(df$HHInc==14,1,0)
```

```r
df$MEdu0 <- ifelse(df$MEdu==0,1,0)
df$MEdu1 <- ifelse(df$MEdu==1,1,0)
df$MEdu2 <- ifelse(df$MEdu==2,1,0)
df$MEdu3 <- ifelse(df$MEdu==3,1,0)
df$MEdu4 <- ifelse(df$MEdu==4,1,0)
df$MEdu5 <- ifelse(df$MEdu==5,1,0)
df$MEdu6 <- ifelse(df$MEdu==6,1,0)
df$MEdu7 <- ifelse(df$MEdu==7,1,0)
df$MEdu8 <- ifelse(df$MEdu==8,1,0)
df$MEdu9 <- ifelse(df$MEdu==9,1,0)
df$MEdu10 <- ifelse(df$MEdu==10,1,0)
df$MEdu11 <- ifelse(df$MEdu==11,1,0)

df$FEdu0 <- ifelse(df$MEdu==0,1,0)
df$FEdu1 <- ifelse(df$MEdu==1,1,0)
df$FEdu2 <- ifelse(df$MEdu==2,1,0)
df$FEdu3 <- ifelse(df$MEdu==3,1,0)
df$FEdu4 <- ifelse(df$MEdu==4,1,0)
df$FEdu5 <- ifelse(df$MEdu==5,1,0)
df$FEdu6 <- ifelse(df$MEdu==6,1,0)
df$FEdu7 <- ifelse(df$MEdu==7,1,0)
df$FEdu8 <- ifelse(df$MEdu==8,1,0)
df$FEdu9 <- ifelse(df$MEdu==9,1,0)
df$FEdu10 <- ifelse(df$MEdu==10,1,0)
df$FEdu11 <- ifelse(df$MEdu==11,1,0)
```

```r
mdf <- df[,c(
  "mpwBins",
  "HHNbr",
  "MWrkHrs",
  "FWrkHrs",
  "FBirth",
  "MBirth",
  "Cable",
  "Cats",
  "Dogs",
  "YogExp",
  "AptResType", "ConResType", "SFResType", "MFResType", "MobResType", "OthResType",
  "OwnResStatus", "RenResStatus", "OthResStatus",
  "HHInc1","HHInc2","HHInc3","HHInc4","HHInc5","HHInc6","HHInc7","HHInc8","HHInc9","HHInc10",
    "HHInc11","HHInc12","HHInc13","HHInc14",
  "MEdu0","MEdu1","MEdu2","MEdu3","MEdu4","MEdu5","MEdu6","MEdu7","MEdu8","MEdu9","MEdu10","MEdu11",
  "FEdu0","FEdu1","FEdu2","FEdu3","FEdu4","FEdu5","FEdu6","FEdu7","FEdu8","FEdu9","FEdu10","FEdu11"
)]
```

E.

```r
#modeling
mdf1 <- scale(mdf[2:57])
mdf1 <- data.frame(
  mdf1,
  mdf$mpwBins
)
colnames(mdf1)[57] <- 'mpwBins'

set.seed(1)
mdf_idx <- createDataPartition(mdf1$mpwBins, p=0.6, list=FALSE)
train_mdf <- mdf1[mdf_idx,]
valid_mdf <- mdf1[-mdf_idx,]

mdf_ctrl <- trainControl(method="cv", number=10)
mdf_grid <- expand.grid(.k=c(1:10))
set.seed(1)
mdf_KNN_fit <- train(
  mpwBins~.,
  data=train_mdf,
  method="knn",
  trControl=mdf_ctrl,
  tuneGrid=mdf_grid
)
mdf_KNN_fit

mdf_KNN_pred <- predict(mdf_KNN_fit, newdata=valid_mdf)
confusionMatrix(mdf_KNN_pred, valid_mdf$mpwBins, positive='1')
```

```
Confusion Matrix and Statistics

          Reference
Prediction [0] (0,0.25] (0.25,1] (1,7] (7,+inf)
  [0]       785      219      109    65        3
  (0,0.25]   25        7        5     1        0
  (0.25,1]    8        5        3     3        0
  (1,7]       0        0        1     0        1
  (7,+inf)    0        0        0     0        0

Overall Statistics

               Accuracy : 0.6411
                 95% CI : (0.6137, 0.6679)
    No Information Rate : 0.6597
    P-Value [Acc > NIR] : 0.9201

                  Kappa : 0.0153

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: [0] Class: (0,0.25] Class: (0.25,1] Class: (1,7] Class: (7,+inf)
Sensitivity             0.95966        0.030303        0.025424     0.000000        0.000000
Specificity             0.06161        0.969277        0.985740     0.998292        1.000000
Pos Pred Value          0.66469        0.184211        0.157895     0.000000             NaN
Neg Pred Value          0.44068        0.813644        0.905815     0.944265        0.996774
Prevalence              0.65968        0.186290        0.095161     0.055645        0.003226
Detection Rate          0.63306        0.005645        0.002419     0.000000        0.000000
Detection Prevalence    0.95242        0.030645        0.015323     0.001613        0.000000
Balanced Accuracy       0.51063        0.499790        0.505582     0.499146        0.500000
```

F.

```r
mdf_grid1 <- expand.grid(.k=c(1:40))
set.seed(1)
mdf_KNN_fit1 <- train(
  mpwBins~.,
  data=train_mdf,
  method="knn",
  trControl=mdf_ctrl,
  tuneGrid=mdf_grid1
)
mdf_KNN_fit1

mdf_KNN_pred1 <- predict(mdf_KNN_fit1, newdata=valid_mdf)
confusionMatrix(mdf_KNN_pred1, valid_mdf$mpwBins, positive='1')
```

```
Confusion Matrix and Statistics

          Reference
Prediction [0] (0,0.25] (0.25,1] (1,7] (7,+inf)
  [0]       818      231      117    69        4
  (0,0.25]    0        0        0     0        0
  (0.25,1]    0        0        0     0        0
  (1,7]       0        0        1     0        0
  (7,+inf)    0        0        0     0        0

Overall Statistics

               Accuracy : 0.6597
                 95% CI : (0.6325, 0.686)
    No Information Rate : 0.6597
    P-Value [Acc > NIR] : 0.5132

                  Kappa : 0.0014

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: [0] Class: (0,0.25] Class: (0.25,1] Class: (1,7] Class: (7,+inf)
Sensitivity             1.00000          0.0000         0.00000    0.0000000        0.000000
Specificity             0.00237          1.0000         1.00000    0.9991460        1.000000
Pos Pred Value          0.66021             NaN             NaN    0.0000000             NaN
Neg Pred Value          1.00000          0.8137         0.90484    0.9443099        0.996774
Prevalence              0.65968          0.1863         0.09516    0.0556452        0.003226
Detection Rate          0.65968          0.0000         0.00000    0.0000000        0.000000
Detection Prevalence    0.99919          0.0000         0.00000    0.0008065        0.000000
Balanced Accuracy       0.50118          0.5000         0.50000    0.4995730        0.500000
```

G.

```r
bin_mdf <- mdf
bin_mdf$mpwBins <- ifelse(mdf$mpwBins=='(1,7]',1,0)
colnames(bin_mdf)[1] <- 'target'

mdf2 <- scale(bin_mdf[2:57])
mdf2 <- data.frame(
  mdf2,
  bin_mdf$target
)
colnames(mdf2)[57] <- 'target'
mdf2$target <- as.factor(mdf2$target)

set.seed(1)
mdf_idx2 <- createDataPartition(mdf2$target, p=0.6, list=FALSE)
train_mdf2 <- mdf2[mdf_idx2,]
valid_mdf2 <- mdf2[-mdf_idx2,]

#We use the same control and grid as before.
set.seed(1)
mdf_KNN_fit2 <- train(
  target~.,
  data=train_mdf2,
  method="knn",
  trControl=mdf_ctrl,
  tuneGrid=mdf_grid1
)
mdf_KNN_fit2

mdf_KNN_pred2 <- predict(mdf_KNN_fit2, newdata=valid_mdf2)
confusionMatrix(mdf_KNN_pred2, valid_mdf2$target, positive='1')
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1172   69
         1    0    0

               Accuracy : 0.9444
                 95% CI : (0.9302, 0.9565)
    No Information Rate : 0.9444
    P-Value [Acc > NIR] : 0.532

                  Kappa : 0

 Mcnemar's Test P-Value : 0.0000000000000002695

            Sensitivity : 0.0000
            Specificity : 1.0000
         Pos Pred Value :    NaN
         Neg Pred Value : 0.9444
             Prevalence : 0.0556
         Detection Rate : 0.0000
   Detection Prevalence : 0.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : 1
```

H.

```r
cutoff <- ifelse(mdf2$target=='1',1,0) %>%
  sum() / count(mdf2) %>%
  as.numeric()

mdf_KNN_pred_prob <- predict(mdf_KNN_fit2, newdata=valid_mdf2, type='prob')

confusionMatrix(
  as.factor(ifelse(mdf_KNN_pred_prob[,2]>cutoff, '1', '0')),
  valid_mdf2$target,
  positive = '1'
)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 555  21
         1 617  48

               Accuracy : 0.4859
                 95% CI : (0.4577, 0.5141)
    No Information Rate : 0.9444
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0334

 Mcnemar's Test P-Value : <0.0000000000000002

            Sensitivity : 0.69565
            Specificity : 0.47355
         Pos Pred Value : 0.07218
         Neg Pred Value : 0.96354
             Prevalence : 0.05560
         Detection Rate : 0.03868
   Detection Prevalence : 0.53586
      Balanced Accuracy : 0.58460

       'Positive' Class : 1
```

I.

```
Call:
glm(formula = target ~ ., family = binomial(link = "logit"),
    data = train_bin_mdf)

Deviance Residuals:
   Min       1Q   Median       3Q      Max
 -8.49     0.00     0.00     0.00     8.49

Coefficients: (16 not defined because of singularities)
```


```
Call:
glm(formula = target ~ ., family = binomial(link = "logit"),
    data = train_bin_mdf)

Deviance Residuals:
   Min       1Q   Median       3Q      Max
 -8.49     0.00     0.00     0.00     8.49

Coefficients: (16 not defined because of singularities)
```


```
                       Estimate        Std. Error    z value              Pr(>|z|)
(Intercept)     480324473818904         341063790    1408313 <0.0000000000000002 ***
HHNbr           185280831519425           1625306  113997524 <0.0000000000000002 ***
MWrkHrs           -300312447974            125587   -2391277 <0.0000000000000002 ***
FWrkHrs           -515112070824             91602   -5623363 <0.0000000000000002 ***
FBirth             311266120860              4786   65044252 <0.0000000000000002 ***
MBirth           -3059469710891            176563  -17327917 <0.0000000000000002 ***
Cable          -447795930712512           3413350 -131189574 <0.0000000000000002 ***
Cats             41930958069699           2009634   20864977 <0.0000000000000002 ***
Dogs             79148552959101           2206027   35878329 <0.0000000000000002 ***
YogExp            1258725345035              5127  245492740 <0.0000000000000002 ***
AptResType      950371718768239          23933543   39708777 <0.0000000000000002 ***
ConResType     1677394686071419          28604613   58640706 <0.0000000000000002 ***
SFResType      2106589894059118          22671041   92919857 <0.0000000000000002 ***
MFResType      2329763570915103          24328150   95764109 <0.0000000000000002 ***
MobResType     3633848408876591          25309513  143576384 <0.0000000000000002 ***
OthResType                   NA                NA         NA                  NA
OwnResStatus   -631906923104231          13514696  -46757021 <0.0000000000000002 ***
RenResStatus   -450754955635972          14209122  -31722928 <0.0000000000000002 ***
OthResStatus                 NA                NA         NA                  NA
HHInc1          162422611629988          19412077    8367091 <0.0000000000000002 ***
HHInc2           22172086555207          17905744    1238267 <0.0000000000000002 ***
HHInc3          905327987606612          17462850   51843083 <0.0000000000000002 ***
HHInc4          912614199821031          17253819   52893461 <0.0000000000000002 ***
HHInc5          -49228524821936          17072262   -2883539 <0.0000000000000002 ***
HHInc6          334378517698690          17074379   19583641 <0.0000000000000002 ***
HHInc7          -57942653112142          17173303   -3373996 <0.0000000000000002 ***
HHInc8          -47018010647917          17465447   -2692059 <0.0000000000000002 ***
HHInc9           67253992048315          17622478    3816375 <0.0000000000000002 ***
HHInc10         297096373566139          17956559   16545284 <0.0000000000000002 ***
HHInc11          81111744557794          17703074    4581789 <0.0000000000000002 ***
HHInc12         989005119627092          19449566   50849727 <0.0000000000000002 ***
HHInc13        1231308903115158          23518762   52354324 <0.0000000000000002 ***
HHInc14                      NA                NA         NA                  NA
MEdu0         -5345544094468087         342595054  -15603098 <0.0000000000000002 ***
MEdu1           347125611931218          48305078    7186110 <0.0000000000000002 ***
MEdu2           737055983544846          19620176   37566227 <0.0000000000000002 ***
MEdu3            34745920096372          10420443    3334400 <0.0000000000000002 ***
MEdu4            72798699256390           9748675    7467548 <0.0000000000000002 ***
MEdu5            -6574080036817           7806818    -842095 <0.0000000000000002 ***
MEdu6           -68889034350120          10028924   -6869035 <0.0000000000000002 ***
MEdu7           451286651046153           9088901   49652500 <0.0000000000000002 ***
MEdu8           468437312689262           7627307   61415822 <0.0000000000000002 ***
MEdu9           153009029718217           7751283   19739833 <0.0000000000000002 ***
MEdu10          468446380458889          11070956   42313094 <0.0000000000000002 ***
MEdu11                       NA                NA         NA                  NA
FEdu0                        NA                NA         NA                  NA
FEdu1                        NA                NA         NA                  NA
FEdu2                        NA                NA         NA                  NA
FEdu3                        NA                NA         NA                  NA
FEdu4                        NA                NA         NA                  NA
FEdu5                        NA                NA         NA                  NA
FEdu6                        NA                NA         NA                  NA
FEdu7                        NA                NA         NA                  NA
FEdu8                        NA                NA         NA                  NA
FEdu9                        NA                NA         NA                  NA
FEdu10                       NA                NA         NA                  NA
FEdu11                       NA                NA         NA                  NA
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

J.

```r
nums <- cor(train_bin_mdf)%>%
  findCorrelation(cutoff=0.75) %>%
  sort()
train_bin_mdf[-nums]

m2 <- glm(
  target~.,
  family=binomial(link="logit"),
  data=train_bin_mdf[-nums]
)
summary(m2)

phat2 <- predict(m2, valid_bin_mdf, type = "response")
yhat2 <- ifelse(phat2 >= cutoff, 1, 0)

ytp2 <- ifelse(yhat2 == 1 & valid_bin_mdf$target == 1, 1, 0)
ytn2 <- ifelse(yhat2 == 0 & valid_bin_mdf$target == 0, 1, 0)
# Accuracy score
mean(valid_bin_mdf$target == yhat2)
# Sensitivity:
sum(ytp2) / sum(valid_bin_mdf$target == 1)
# Specificity:
sum(ytn2) / sum(valid_bin_mdf$target == 0)
```

```
> # Accuracy score
> mean(valid_bin_mdf$target == yhat2)
[1] 0.7004831
> # Sensitivity:
> sum(ytp2) / sum(valid_bin_mdf$target == 1)
[1] 0.6984127
> # Specificity:
> sum(ytn2) / sum(valid_bin_mdf$target == 0)
[1] 0.7005937
>
```

K.

```r
colnames(train_bin_mdf[-nums])

m3 <- glm(
  target~HHNbr+MWrkHrs+FWrkHrs+Cable+RenResStatus+OthResStatus+
    HHInc1+HHInc2+HHInc3+HHInc4+HHInc5+HHInc6+HHInc7+HHInc8+
    HHInc9+HHInc10+HHInc11+HHInc12+HHInc13+HHInc14,
  family=binomial(link="logit"),
  data=train_bin_mdf
)
summary(m3)

phat3 <- predict(m3, valid_bin_mdf, type = "response")
yhat3 <- ifelse(phat3 >= cutoff, 1, 0)

ytp3 <- ifelse(yhat3 == 1 & valid_bin_mdf$target == 1, 1, 0)
ytn3 <- ifelse(yhat3 == 0 & valid_bin_mdf$target == 0, 1, 0)
# Accuracy score
mean(valid_bin_mdf$target == yhat3)
# Sensitivity:
sum(ytp3) / sum(valid_bin_mdf$target == 1)
# Specificity:
sum(ytn3) / sum(valid_bin_mdf$target == 0)
```

```
> # Accuracy score
> mean(valid_bin_mdf$target == yhat3)
[1] 0.6183575
> # Sensitivity:
> sum(ytp3) / sum(valid_bin_mdf$target == 1)
[1] 0.6507937
> # Specificity:
> sum(ytn3) / sum(valid_bin_mdf$target == 0)
[1] 0.6166243
```

```
> summary(m3)

Call:
glm(formula = target ~ HHNbr + MWrkHrs + FWrkHrs + Cable + RenResStatus +
    OthResStatus + HHInc1 + HHInc2 + HHInc3 + HHInc4 + HHInc5 +
    HHInc6 + HHInc7 + HHInc8 + HHInc9 + HHInc10 + HHInc11 + HHInc12 +
    HHInc13 + HHInc14, family = binomial(link = "logit"), data = train_bin_mdf)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.0784  -0.3841  -0.2846  -0.2157   2.9928

Coefficients: (1 not defined because of singularities)
               Estimate Std. Error z value       Pr(>|z|)
(Intercept)   -4.668127   1.106748  -4.218 0.000024661280 ***
HHNbr          0.525194   0.082208   6.389 0.000000000167 ***
MWrkHrs       -0.003980   0.006179  -0.644          0.519
FWrkHrs        0.006259   0.005648   1.108          0.268
Cable          0.111588   0.230306   0.485          0.628
RenResStatus  -0.642902   0.425618  -1.511          0.131
OthResStatus -13.878415 435.236270  -0.032          0.975
HHInc1         1.002219   1.238756   0.809          0.418
HHInc2         0.492732   1.154803   0.427          0.670
HHInc3        -0.769859   1.211442  -0.635          0.525
HHInc4        -0.130262   1.115420  -0.117          0.907
HHInc5         0.419178   1.079147   0.388          0.698
HHInc6         0.356447   1.078024   0.331          0.741
HHInc7         0.341440   1.085050   0.315          0.753
HHInc8         0.320133   1.101696   0.291          0.771
HHInc9         0.057401   1.114415   0.052          0.959
HHInc10        0.304510   1.123106   0.271          0.786
HHInc11       -0.126613   1.132533  -0.112          0.911
HHInc12       -0.189762   1.279732  -0.148          0.882
HHInc13        1.045930   1.305006   0.801          0.423
HHInc14              NA         NA      NA             NA
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 835.86  on 1862  degrees of freedom
Residual deviance: 761.90  on 1843  degrees of freedom
AIC: 801.9

Number of Fisher Scoring iterations: 15
```

L.

```
m4 <- glm(
  target~HHNbr+MWrkHrs+FWrkHrs+Cable+RenResStatus,
  family=binomial(link="logit"),
  data=train_bin_mdf
)
summary(m4)

phat4 <- predict(m4, valid_bin_mdf, type = "response")
yhat4 <- ifelse(phat4 >= cutoff, 1, 0)

ytp4 <- ifelse(yhat4 == 1 & valid_bin_mdf$target == 1, 1, 0)
ytn4 <- ifelse(yhat4 == 0 & valid_bin_mdf$target == 0, 1, 0)
# Accuracy score
mean(valid_bin_mdf$target == yhat4)
# Sensitivity:
sum(ytp4) / sum(valid_bin_mdf$target == 1)
# Specificity:
sum(ytn4) / sum(valid_bin_mdf$target == 0)
```

```
> # Accuracy score
> mean(valid_bin_mdf$target == yhat4)
[1] 0.6127214
> # Sensitivity:
> sum(ytp4) / sum(valid_bin_mdf$target == 1)
[1] 0.6190476
> # Specificity:
> sum(ytn4) / sum(valid_bin_mdf$target == 0)
[1] 0.6123834
```

M.

```
confusionMatrix(
  as.factor(ifelse(phat2 >= cutoff, '1', '0')),
  as.factor(valid_bin_mdf$target),
  positive='1'
)

gains_valid_mdf2 <- valid_mdf2
gains_valid_mdf2$target <- as.numeric(as.character(gains_valid_mdf2$target))
gains_table <- gains(gains_valid_mdf2$target, mdf_KNN_pred_prob[,2])
gains_table

gains_valid_bin <- valid_bin_mdf
gains_valid_bin$target <- as.numeric(as.character(gains_valid_bin$target))
gains_table_bin <- gains(gains_valid_bin$target, phat2)
gains_table

plot(
  c(0, gains_table$cume.pct.of.total*sum(gains_valid_mdf2$target)) ~ c(0, gains_table$cume.obs),
  xlab="# of cases",
  ylab="cumulative",
  main="Cumulative lift chart",
  type="l",
  col="red"
)
lines(
  c(0, sum(gains_valid_mdf2$target)) ~ c(0, dim(gains_valid_mdf2)[1]),
  lty=2
)
lines(
  c(0, gains_table_bin$cume.pct.of.total*sum(gains_valid_bin$target)) ~ c(0, gains_table_bin$cume.obs),
  col="blue"
)
legend(
  600,
  20,
  legend=c("'mdf_KNN_fit2'", "'m2'", "Baseline Model"),
  col=c("red", "blue", "black"),
  lty=c(1,1,2)
)
```