Bookbinders Case Study

Brandi Rodriguez

March 2021

LOAD DATA

```
rm(list = ls())
library(tidyverse)
library(caret)
library(e1071)

#read data
library(readxl)
setwd(getwd())
raw_train = read_excel('BBBC-Train.xlsx')
raw_test = read_excel('BBBC-Test.xlsx')

#make a copy and remove duplicate records
train = distinct(raw_train)
test = distinct(raw_test)

str(train)
```

```
## tibble [1,600 x 12] (S3: tbl_df/tbl/data.frame)
##  $ Observation     : num [1:1600] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Choice          : num [1:1600] 1 1 1 1 1 1 1 1 1 1 ...
##  $ Gender          : num [1:1600] 1 1 1 1 0 1 1 0 1 1 ...
##  $ Amount_purchased: num [1:1600] 113 418 336 180 320 268 198 280 393 138 ...
##  $ Frequency       : num [1:1600] 8 6 18 16 2 4 2 6 12 10 ...
##  $ Last_purchase   : num [1:1600] 1 11 6 5 3 1 12 2 11 7 ...
##  $ First_purchase  : num [1:1600] 8 66 32 42 18 4 62 12 50 38 ...
##  $ P_Child         : num [1:1600] 0 0 2 2 0 0 2 0 3 2 ...
##  $ P_Youth         : num [1:1600] 1 2 0 0 0 0 3 2 0 3 ...
##  $ P_Cook          : num [1:1600] 0 3 1 0 0 0 2 0 3 0 ...
##  $ P_DIY           : num [1:1600] 0 2 1 1 1 0 1 0 0 0 ...
##  $ P_Art           : num [1:1600] 0 3 2 1 2 0 2 0 2 1 ...
```

CONVERT VARIABLES

```
#convert categorical variables to factor
train$Choice = as.factor(train$Choice)
test$Choice = as.factor(test$Choice)
train$Gender = as.factor(train$Gender)
test$Gender = as.factor(test$Gender)

#convert some numeric variables to integer
train$Frequency = as.integer(train$Frequency)
train$P_Child = as.integer(train$P_Child)
train$P_Youth = as.integer(train$P_Youth)
train$P_Cook = as.integer(train$P_Cook)
train$P_DIY = as.integer(train$P_DIY)
train$P_Art = as.integer(train$P_Art)

test$Frequency = as.integer(test$Frequency)
test$P_Child = as.integer(test$P_Child)
test$P_Youth = as.integer(test$P_Youth)
test$P_Cook = as.integer(test$P_Cook)
test$P_DIY = as.integer(test$P_DIY)
test$P_Art = as.integer(test$P_Art)

str(train)
```

```
## tibble [1,600 x 12] (S3: tbl_df/tbl/data.frame)
##  $ Observation     : num [1:1600] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Choice          : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Gender          : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 2 1 2 2 ...
##  $ Amount_purchased: num [1:1600] 113 418 336 180 320 268 198 280 393 138 ...
##  $ Frequency       : int [1:1600] 8 6 18 16 2 4 2 6 12 10 ...
##  $ Last_purchase   : num [1:1600] 1 11 6 5 3 1 12 2 11 7 ...
##  $ First_purchase  : num [1:1600] 8 66 32 42 18 4 62 12 50 38 ...
##  $ P_Child         : int [1:1600] 0 0 2 2 0 0 2 0 3 2 ...
##  $ P_Youth         : int [1:1600] 1 2 0 0 0 0 3 2 0 3 ...
##  $ P_Cook          : int [1:1600] 0 3 1 0 0 0 2 0 3 0 ...
##  $ P_DIY           : int [1:1600] 0 2 1 1 1 0 1 0 0 0 ...
##  $ P_Art           : int [1:1600] 0 3 2 1 2 0 2 0 2 1 ...
```

MISSING VALUES

```
sum(is.na(train))
```

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

The dataset has no missing values.

```
library(Amelia)
```

```
## Warning: package 'Amelia' was built under R version 3.6.3
```

```
## Loading required package: Rcpp
```

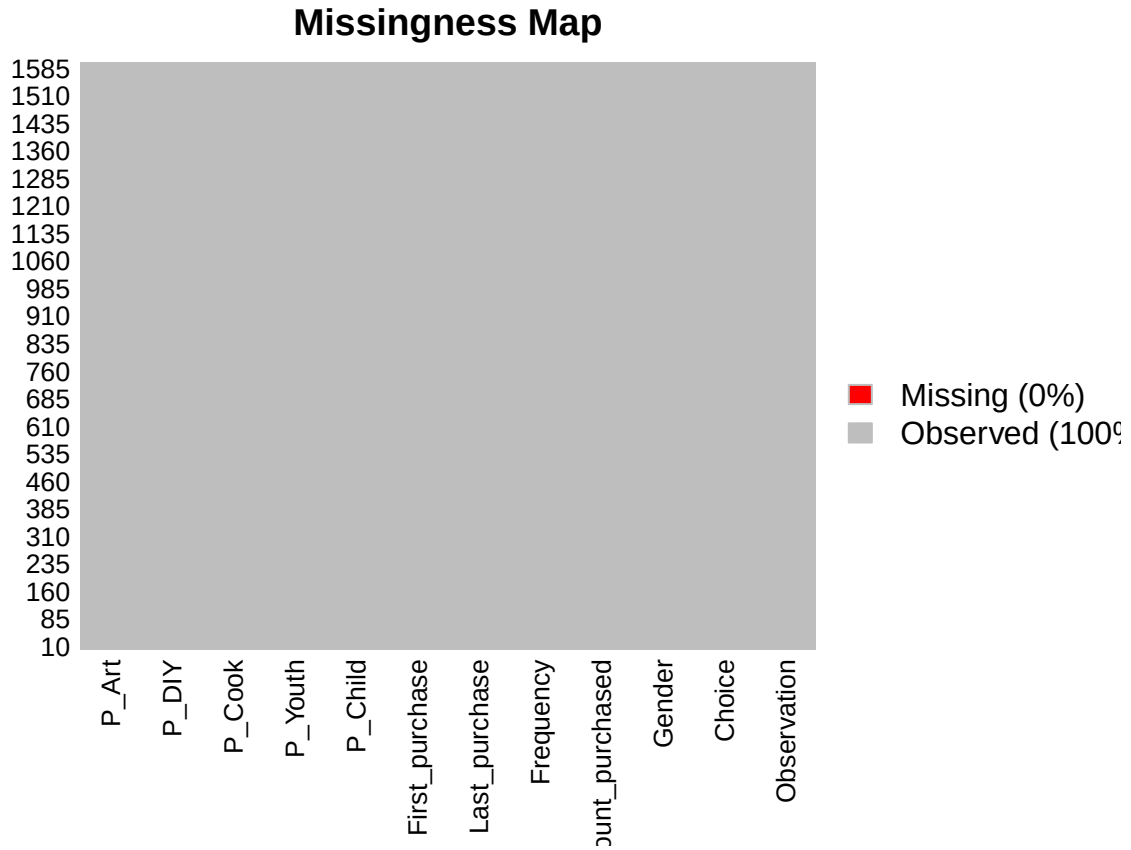
```
## ##
## ## Amelia II: Multiple Imputation
## ## (Version 1.7.6, built: 2019-11-24)
## ## Copyright (C) 2005-2021 James Honaker, Gary King and Matthew Blackwell
## ## Refer to http://gking.harvard.edu/amelia/ for more information
## ##
```

```
missmap(train, col=c("red", "gray"))
```

```
## Warning: Unknown or uninitialised column: `arguments`.
```

```
## Warning: Unknown or uninitialised column: `arguments`.
```

```
## Warning: Unknown or uninitialised column: `imputations`.
```



EXPLORATORY DATA ANALYSIS

```
library(DataExplorer)
```

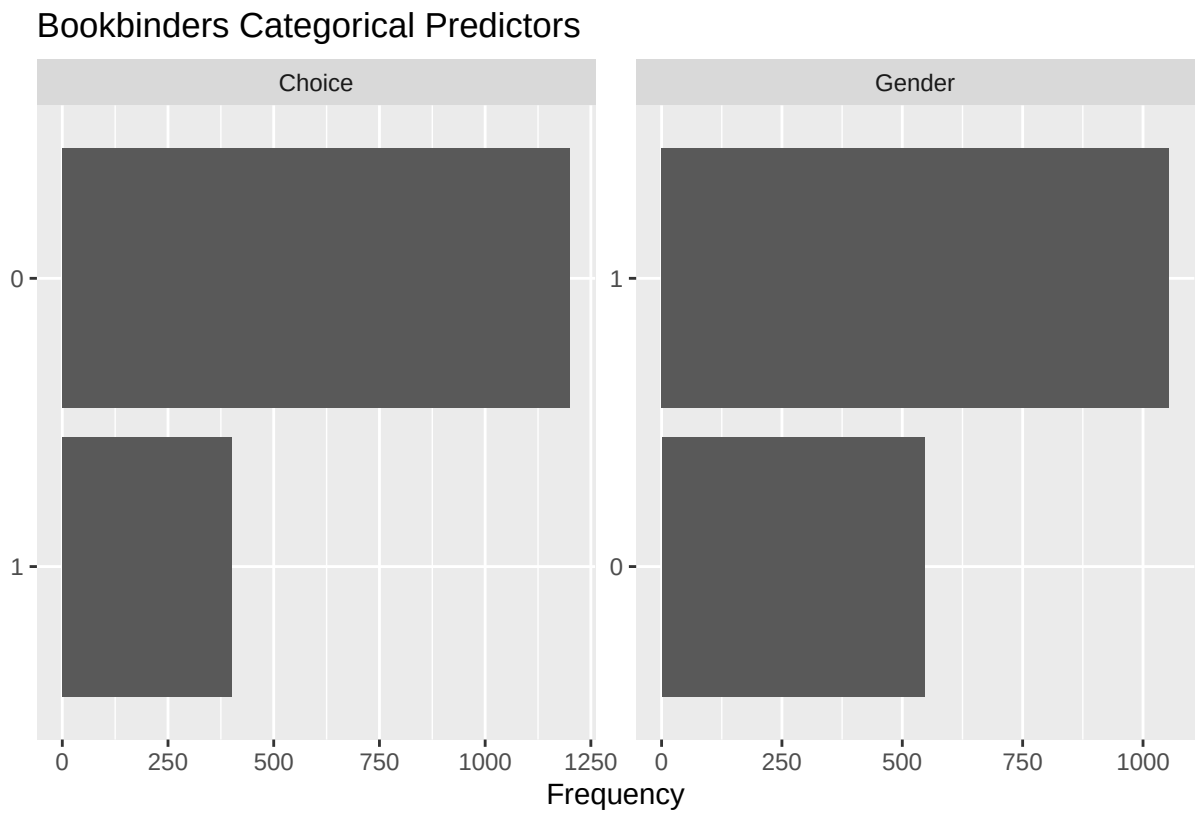
```
## Warning: package 'DataExplorer' was built under R version 3.6.3
```

```
introduce(train)
```

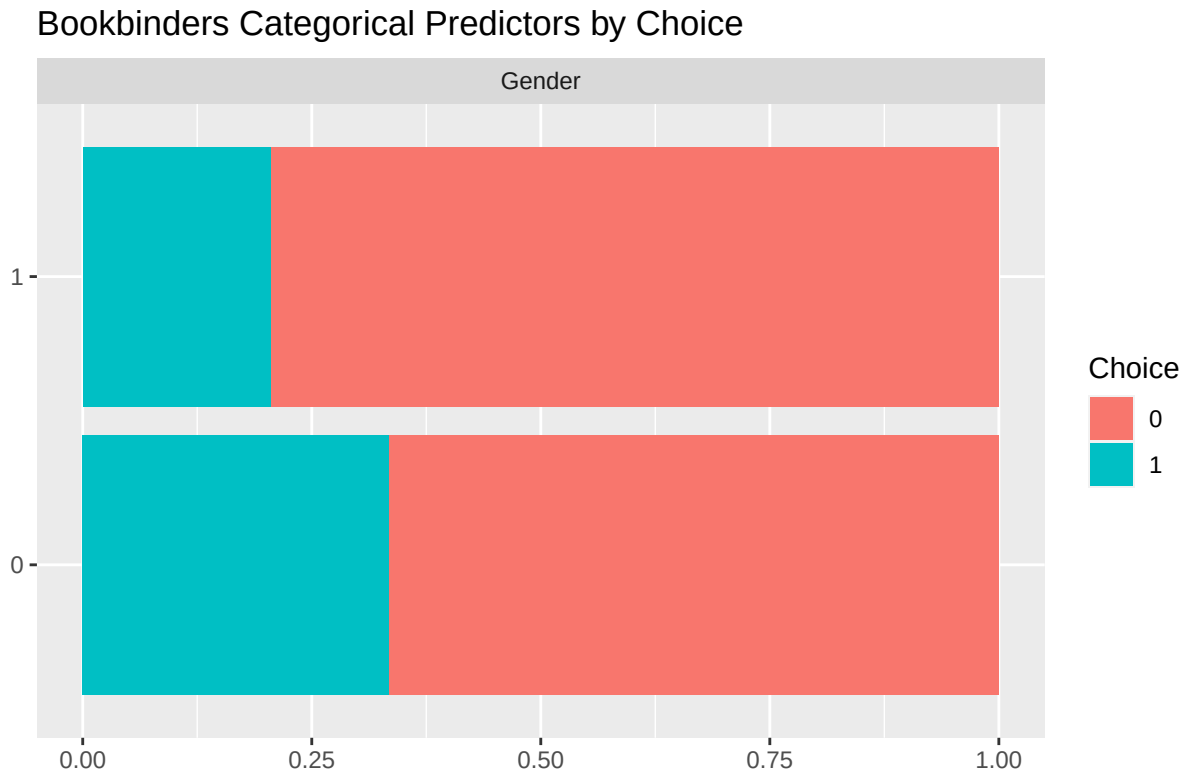
```
## # A tibble: 1 x 9
##    rows columns discrete_columns continuous_colu~ all_missing_col~
##   <int>   <int>            <int>            <int>            <int>
## 1  1600      12                2               10                0
## # ... with 4 more variables: total_missing_values <int>, complete_rows <int>,
## #   total_observations <int>, memory_usage <dbl>
```

The training dataset consists of 1600 observations and 10 variables. There are 2 discrete columns

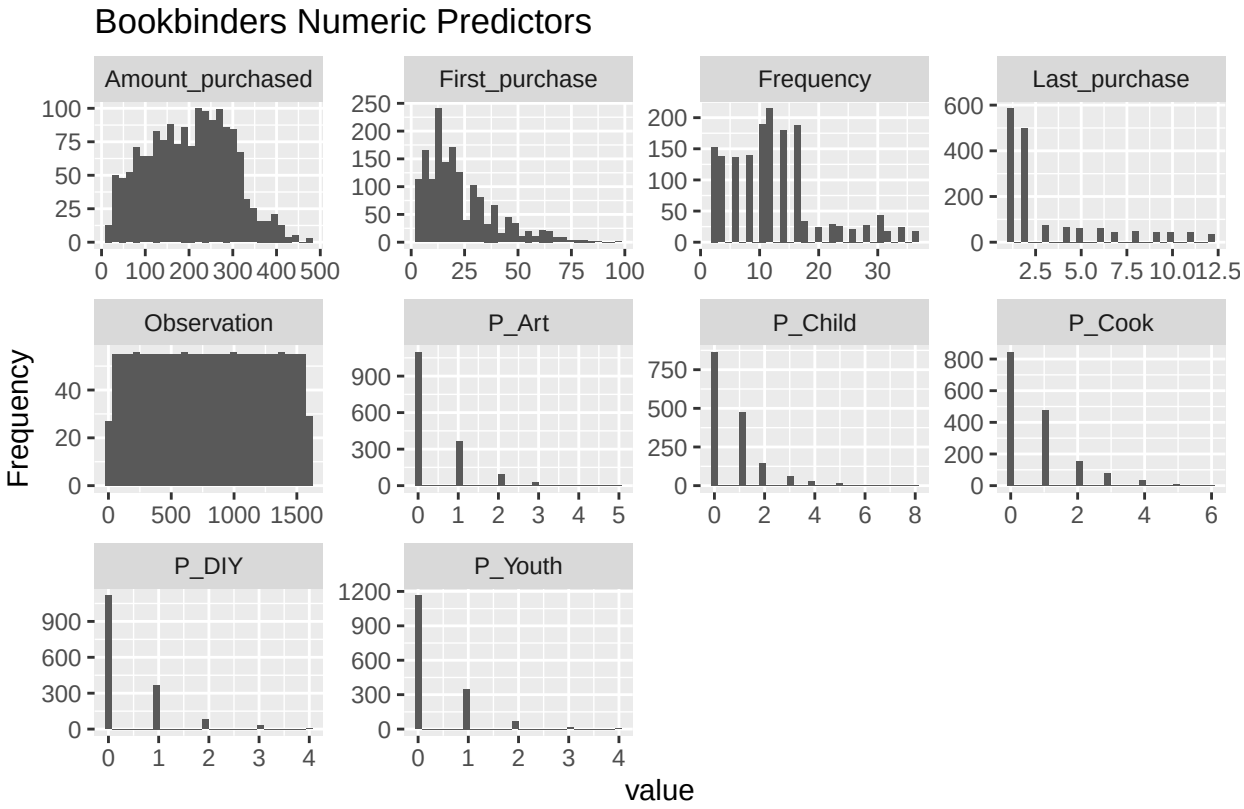
```
plot_bar(train, nrow = 3L, ncol=4L, title = "Bookbinders Categorical Predictors")
```



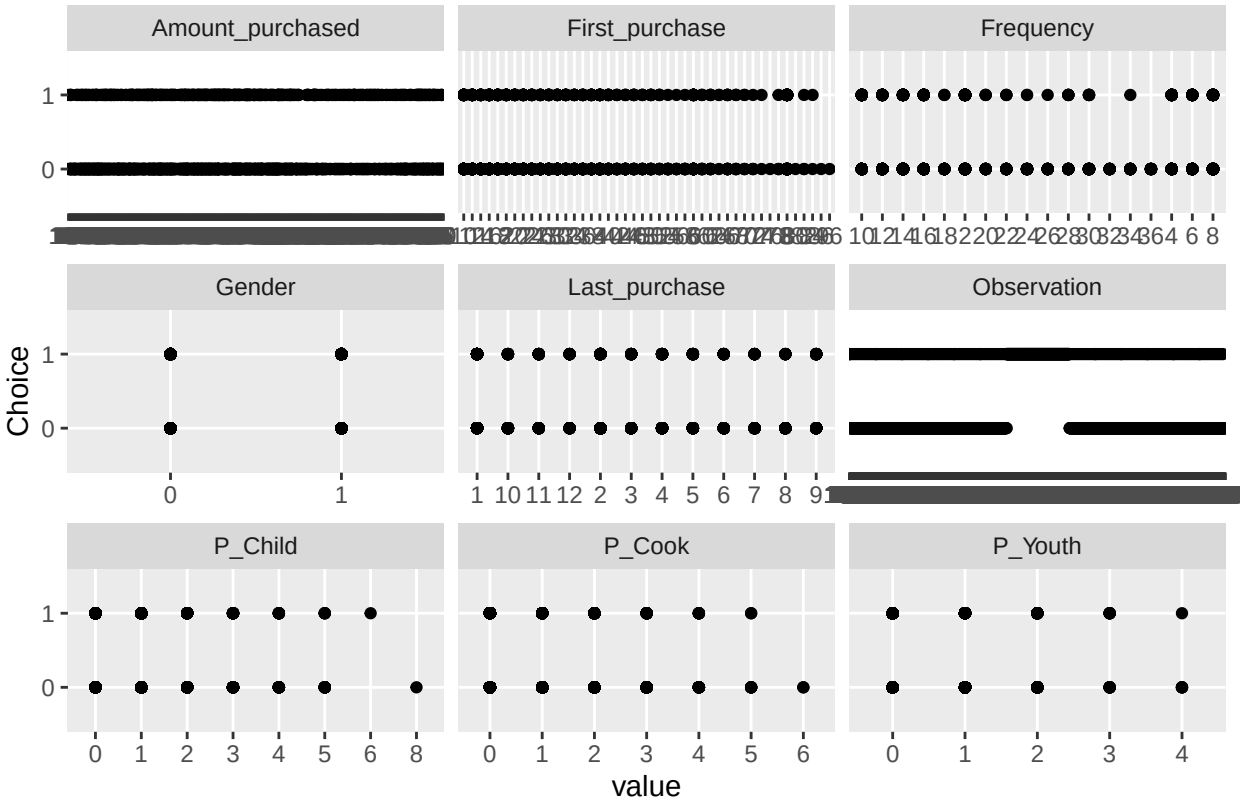

```
plot_bar(train, by="Choice", nrow = 3L, ncol=4L, title = "Bookbinders Categorical Predictors by Choice"
```

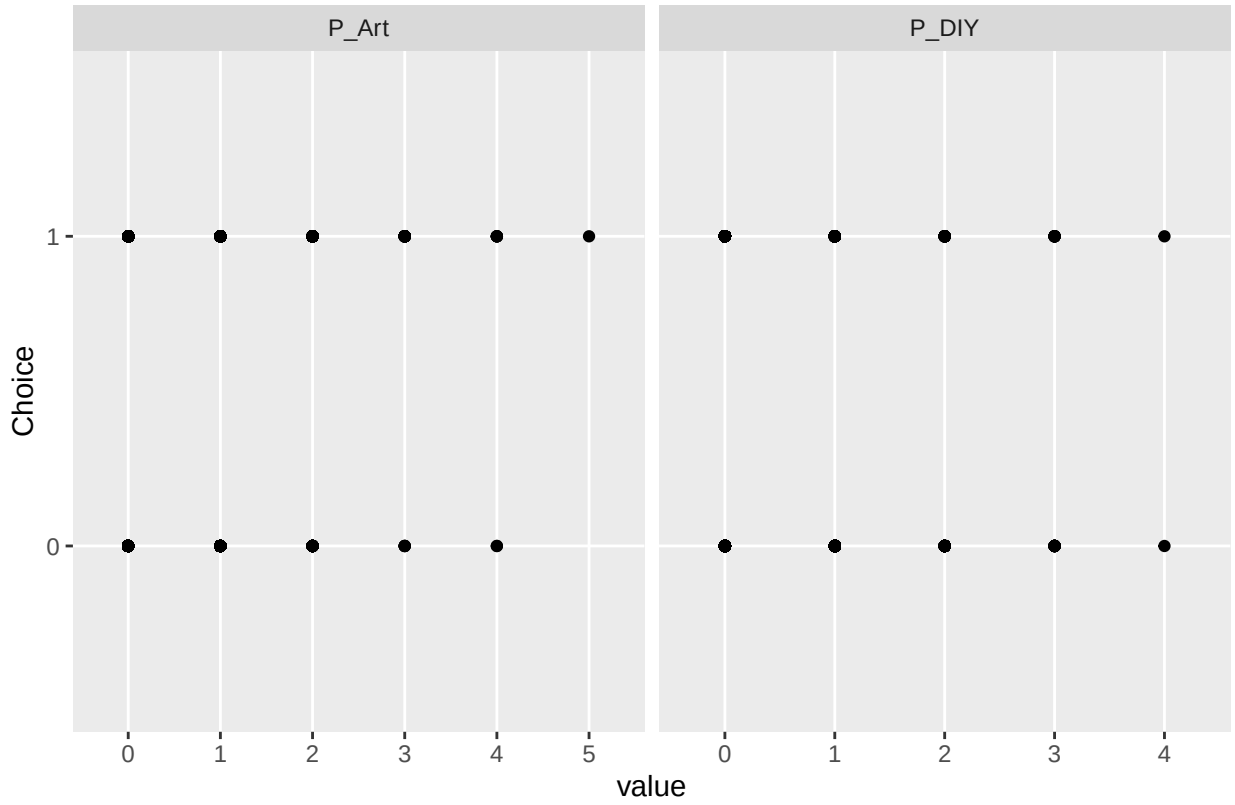



```
plot_histogram(train, title = "Bookbinders Numeric Predictors")
```




```
plot_scatterplot(train, by="Choice")
```


Amount_purchased
First_purchase
Frequency
Gender
Last_purchase
Observation
P_Child
P_Cook
P_Youth
Choice
value



Majority of customers did not purchase the book and there is a larger number of males sampled (gender = 1), however a larger proportion of females ended up purchasing the book. 'Observation' is just indexing the sample and provides no real insight, so it needs to be removed. All numeric variables have a right skewed distribution. Scatterplots would show if there's a linear relationship between the variables and the response variable and if a data transformation may be needed, but unsurprisingly, they are ineffective in this case since the response is a binary categorical variable.

DROP 'OBSERVATION'

The variable "Observation" is not a useful predictor. It's just indexing the observations.

```
train = train[c(2:12)]
str(train)
```

```
## tibble [1,600 x 11] (S3: tbl_df/tbl/data.frame)
##  $ Choice          : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Gender          : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 2 1 2 2 ...
##  $ Amount_purchased: num [1:1600] 113 418 336 180 320 268 198 280 393 138 ...
##  $ Frequency       : int [1:1600] 8 6 18 16 2 4 2 6 12 10 ...
##  $ Last_purchase   : num [1:1600] 1 11 6 5 3 1 12 2 11 7 ...
##  $ First_purchase  : num [1:1600] 8 66 32 42 18 4 62 12 50 38 ...
##  $ P_Child         : int [1:1600] 0 0 2 2 0 0 2 0 3 2 ...
##  $ P_Youth         : int [1:1600] 1 2 0 0 0 0 3 2 0 3 ...
##  $ P_Cook          : int [1:1600] 0 3 1 0 0 0 2 0 3 0 ...
##  $ P_DIY           : int [1:1600] 0 2 1 1 1 0 1 0 0 0 ...
```

```
##  $ P_Art            : int [1:1600] 0 3 2 1 2 0 2 0 2 1 ...
```

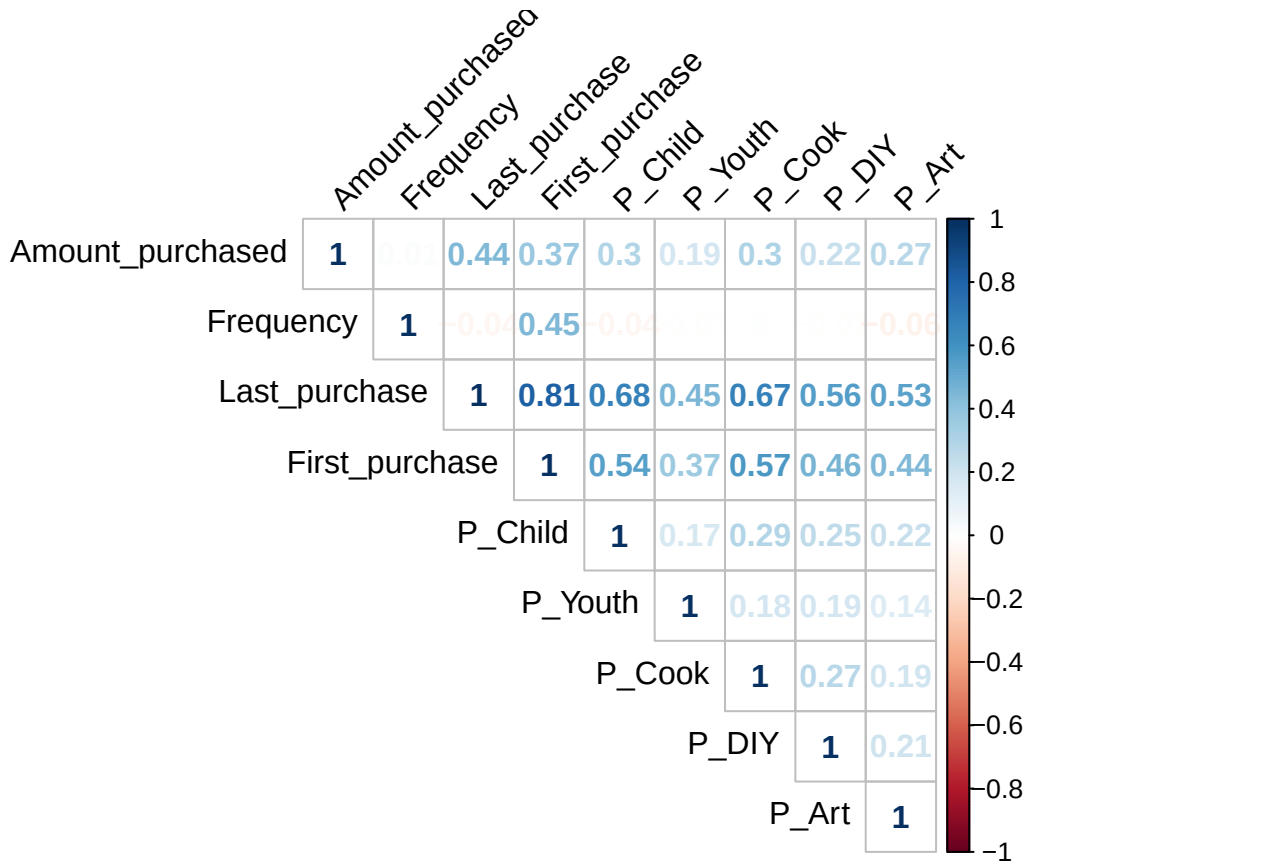
CORRELATION

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.6.3
```

```
## corrplot 0.84 loaded
```

```
corrplot(cor(train[, 3:11]), method = "number", type = "upper", tl.col = "black", tl.srt=45)
```



The data exhibits no extremely strong correlations. Last_purchase and First_purchase have the strongest correlation (.81).

LINEAR REGRESSION 1

*Linear regression doesn't work if the response variable is a factor, so coerce it back to a numeric variable before running the regression model.

```r
train_linear = train
test_linear = test
train_linear$Choice = as.numeric(train_linear$Choice)
test_linear$Choice = as.numeric(test_linear$Choice)


linear1 = lm(Choice~., data=train_linear)
summary(linear1)
```

```
##
## Call:
## lm(formula = Choice ~ ., data = train_linear)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.9603 -0.2462 -0.1161  0.1622  1.0588
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1.3642284  0.0307411  44.378  < 2e-16 ***
## Gender1          -0.1309205  0.0200303  -6.536 8.48e-11 ***
## Amount_purchased  0.0002736  0.0001110   2.464   0.0138 *
## Frequency        -0.0090868  0.0021791  -4.170 3.21e-05 ***
## Last_purchase     0.0970286  0.0135589   7.156 1.26e-12 ***
## First_purchase   -0.0020024  0.0018160  -1.103   0.2704
## P_Child          -0.1262584  0.0164011  -7.698 2.41e-14 ***
## P_Youth          -0.0963563  0.0201097  -4.792 1.81e-06 ***
## P_Cook           -0.1414907  0.0166064  -8.520  < 2e-16 ***
## P_DIY            -0.1352313  0.0197873  -6.834 1.17e-11 ***
## P_Art             0.1178494  0.0194427   6.061 1.68e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3788 on 1589 degrees of freedom
## Multiple R-squared:  0.2401, Adjusted R-squared:  0.2353
## F-statistic:  50.2 on 10 and 1589 DF,  p-value: < 2.2e-16
```

```r
predict_linear1 = predict(linear1, newdata=test_linear) #make predictions
mean((predict_linear1 - test_linear$Choice)^2)
```

```
## [1] 0.09255105
```

The full linear regression model has a low mean square error of .0925, but an r-square of .24, so only 24% of the variance in Choice is predictable by the model.

```r
library(car)
vif(linear1)
```

```
##           Gender Amount_purchased        Frequency    Last_purchase
##         1.005801         1.248066         3.253860        18.770402
##   First_purchase          P_Child          P_Youth            P_Cook
##         9.685333         3.360349         1.775022         3.324928
```

```
##            P_DIY            P_Art
##         2.016910         2.273771
```

Last_purchase had an 81% correlation to First_purchase and has a high VIF of 18.77.

LINEAR REGRESSION 2

Linear regression model, dropping First_Purchase because it had a p-value > .05 in linear1, indicating it is an insignificant predictor.

```
linear2 = lm(Choice~. - First_purchase, data=train_linear)
summary(linear2)
```

```
##
## Call:
## lm(formula = Choice ~ . - First_purchase, data = train_linear)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.9802 -0.2452 -0.1157  0.1655  1.0595
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1.3727367  0.0297590  46.128  < 2e-16 ***
## Gender1          -0.1316464  0.0200208  -6.575 6.56e-11 ***
## Amount_purchased  0.0002742  0.0001110   2.470   0.0136 *
## Frequency        -0.0110830  0.0012128  -9.138  < 2e-16 ***
## Last_purchase     0.0894288  0.0116772   7.658 3.25e-14 ***
## P_Child          -0.1275991  0.0163571  -7.801 1.11e-14 ***
## P_Youth          -0.0973642  0.0200903  -4.846 1.38e-06 ***
## P_Cook           -0.1433497  0.0165218  -8.676  < 2e-16 ***
## P_DIY            -0.1365578  0.0197520  -6.914 6.82e-12 ***
## P_Art             0.1150034  0.0192719   5.967 2.97e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3788 on 1590 degrees of freedom
## Multiple R-squared:  0.2395, Adjusted R-squared:  0.2352
## F-statistic: 55.63 on 9 and 1590 DF,  p-value: < 2.2e-16
```

```
predict_linear2 = predict(linear2, newdata=test_linear) #make predictions
mean((predict_linear2 - test_linear$Choice)^2)
```

```
## [1] 0.0923713
```

```
vif(linear2)
```

```
##           Gender Amount_purchased        Frequency    Last_purchase
##         1.004715         1.248033         1.007855        13.920175
##          P_Child          P_Youth            P_Cook             P_DIY
##         3.341880         1.771355         3.290657         2.009454
##            P_Art
##         2.233698
```

LINEAR REGRESSION 3

Linear regression model, dropping Last_Purchase because of it's high VIF causing mutlicollinearity.

```
linear3 = lm(Choice~. - Last_purchase, data=train_linear)
summary(linear3)
```

```
##
## Call:
## lm(formula = Choice ~ . - Last_purchase, data = train_linear)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.0018 -0.2482 -0.1277  0.1567  1.1035
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1.3926595  0.0309609  44.981  < 2e-16 ***
## Gender1          -0.1290720  0.0203424  -6.345 2.89e-10 ***
## Amount_purchased  0.0003518  0.0001122   3.135 0.001753 **
## Frequency        -0.0157943  0.0019980  -7.905 4.97e-15 ***
## First_purchase    0.0046036  0.0015884   2.898 0.003803 **
## P_Child          -0.0502183  0.0126891  -3.958 7.90e-05 ***
## P_Youth          -0.0225339  0.0175326  -1.285 0.198888
## P_Cook           -0.0667467  0.0131127  -5.090 4.00e-07 ***
## P_DIY            -0.0606486  0.0170835  -3.550 0.000396 ***
## P_Art             0.1916012  0.0167447  11.443  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3847 on 1590 degrees of freedom
## Multiple R-squared:  0.2156, Adjusted R-squared:  0.2111
## F-statistic: 48.55 on 9 and 1590 DF,  p-value: < 2.2e-16
```

```
predict_linear3 = predict(linear3, newdata=test_linear) #make predictions
mean((predict_linear3 - test_linear$Choice)^2)
```

```
## [1] 0.0929021
```

```
vif(linear3)
```

```
##           Gender Amount_purchased        Frequency   First_purchase
##         1.005634         1.235982         2.651820         7.182666
##          P_Child          P_Youth           P_Cook            P_DIY
##         1.949849         1.307915         2.009609         1.457362
##            P_Art
##         1.634878
```

Linear regression model, dropping Last_Purchase and P_Youth

```
linear4 = lm(Choice~. - Last_purchase - P_Youth, data=train_linear)
summary(linear4)
```

```
##
## Call:
## lm(formula = Choice ~ . - Last_purchase - P_Youth, data = train_linear)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.9775 -0.2486 -0.1267  0.1493  1.1085
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1.387614   0.030717  45.174  < 2e-16 ***
## Gender1          -0.128517   0.020342  -6.318 3.44e-10 ***
## Amount_purchased  0.000343   0.000112   3.061 0.002239 **
## Frequency        -0.014952   0.001888  -7.920 4.42e-15 ***
## First_purchase    0.003758   0.001446   2.599 0.009440 **
## P_Child          -0.046933   0.012431  -3.775 0.000166 ***
## P_Cook           -0.063342   0.012845  -4.931 9.02e-07 ***
## P_DIY            -0.058483   0.017004  -3.439 0.000598 ***
## P_Art             0.195593   0.016457  11.885  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3848 on 1591 degrees of freedom
## Multiple R-squared:  0.2148, Adjusted R-squared:  0.2108
## F-statistic: 54.39 on 8 and 1591 DF,  p-value: < 2.2e-16
```

```
predict_linear4 = predict(linear4, newdata=test_linear) #make predictions
mean((predict_linear4 - test_linear$Choice)^2)
```

```
## [1] 0.09304435
```

```
vif(linear4)
```

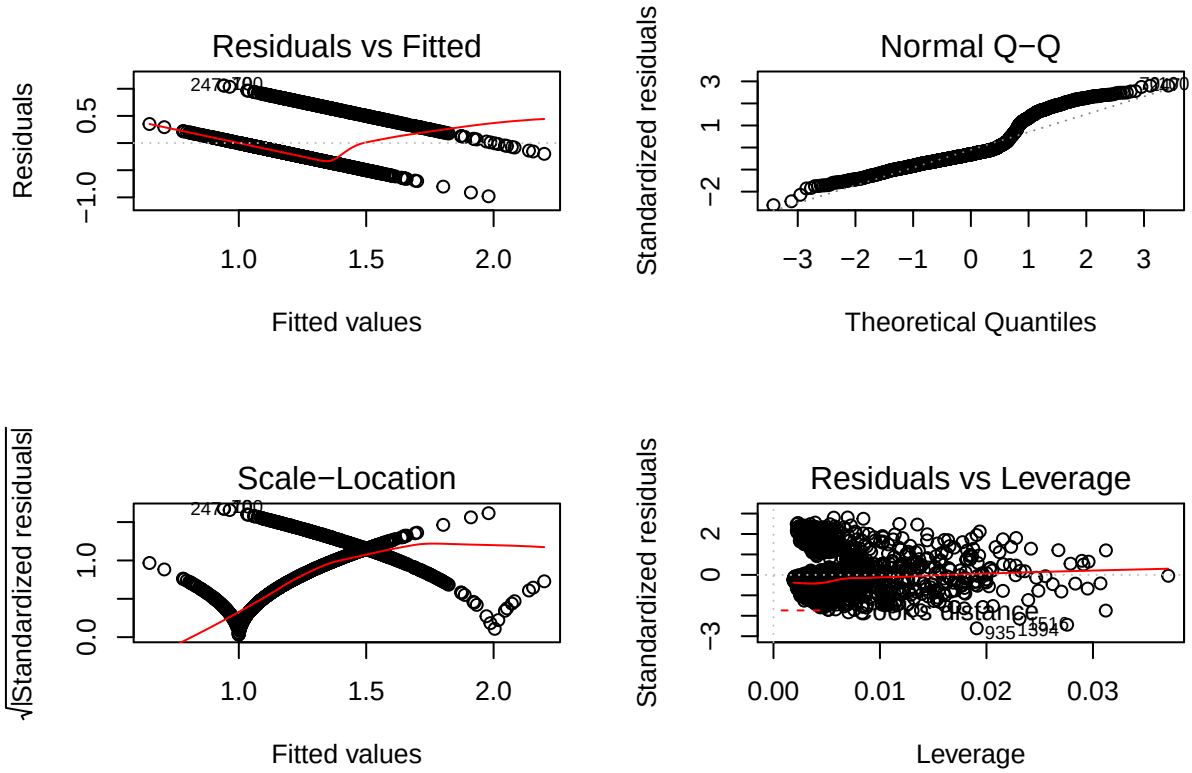
```
##           Gender Amount_purchased        Frequency   First_purchase
##         1.005181         1.231417         2.366704         5.950106
##          P_Child           P_Cook            P_DIY            P_Art
##         1.870721         1.927593         1.443183         1.578636
```

FINAL LINEAR REGRESSION MODEL

```
linear_final = linear2
```

Review assumptions of final linear regression model selected:

```
par(mfrow = c(2,2))
plot(linear_final)
```



The final linear regression model's diagnostics plots prove that it is an inappropriate model for BBBC's classification task. The scatterplots previously plotted showed the independent variables did not display linear relationships with the response variable. The first plot in the diagnostics plots is useful for checking the assumption of linearity and homoscedasticity. Instead of randomly scattered residuals with a straight and horizontal line centered around y = 0, which is characteristic of linearity, the residuals form a very distinctive pattern - two downward sloping lines and a bent line. To assess if the homoscedasticity assumption is met, the residuals should be equally spread around the y = 0 line, but they are not. The normality assumption can be evaluated by looking at the QQ plot. The normality assumption is violated, as the residuals do not follow closely along the 45-degree line. The third plot is useful for checking homoscedasticity. Ideally, the red line will be flat and horizontal with equally and randomly scattered data points, so clearly the homoscedasticity assumption is not satisfied. The fourth plot tells us there are a few influential points based on Cook's distance.

LOGISTIC REGRESSION 1

Normally, dummy variables would be created for categorical variables, but in this case it's not necessary and would be redundant. Gender already has the same two levels (1 if male, 0 if not) that dummy-coding would produce.

```
levels(train$Gender)
```

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

```r
logit1 = glm(Choice ~., data = train, family = "binomial")
summary(logit1)
```

```
##
## Call:
## glm(formula = Choice ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.38586  -0.66728  -0.43696  -0.02242   2.72238
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -0.3515281  0.2143839  -1.640   0.1011
## Gender1          -0.8632319  0.1374499  -6.280 3.38e-10 ***
## Amount_purchased  0.0018641  0.0007918   2.354   0.0186 *
## Frequency        -0.0755142  0.0165937  -4.551 5.35e-06 ***
## Last_purchase     0.6117713  0.0938127   6.521 6.97e-11 ***
## First_purchase   -0.0147792  0.0128027  -1.154   0.2483
## P_Child          -0.8112489  0.1167067  -6.951 3.62e-12 ***
## P_Youth          -0.6370422  0.1433778  -4.443 8.87e-06 ***
## P_Cook           -0.9230066  0.1194814  -7.725 1.12e-14 ***
## P_DIY            -0.9058697  0.1437025  -6.304 2.90e-10 ***
## P_Art             0.6861124  0.1270176   5.402 6.60e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1799.5  on 1599  degrees of freedom
## Residual deviance: 1392.2  on 1589  degrees of freedom
## AIC: 1414.2
##
## Number of Fisher Scoring iterations: 5
```

```r
#Which predictors are signifcant and calculate model fit statistics
significant_if = summary(logit1)$coeff[-1,4]<.05
logit1.significant = names(significant_if)[significant_if ==TRUE]

logit1.significant
```

```
## [1] "Gender1"          "Amount_purchased" "Frequency"        "Last_purchase"
## [5] "P_Child"          "P_Youth"          "P_Cook"           "P_DIY"
## [9] "P_Art"
```

```r
AIC = AIC(logit1)
BIC = BIC(logit1)
cbind(AIC, BIC)
```

```
##           AIC      BIC
## [1,] 1414.159 1473.315
```

```r
#make predictions
library(caret)
test$PredProb = predict.glm(logit1, newdata=test, type = 'response')
test$Pred.Choice = ifelse(test$PredProb >= .5,1,0)
caret::confusionMatrix(as.factor(test$Pred.Choice), as.factor(test$Choice))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1968  125
##          1  128   79
##
##                Accuracy : 0.89
##                  95% CI : (0.8765, 0.9025)
##     No Information Rate : 0.9113
##     P-Value [Acc > NIR] : 0.9998
##
##                   Kappa : 0.324
##
##  Mcnemar's Test P-Value : 0.8999
##
##             Sensitivity : 0.9389
##             Specificity : 0.3873
##          Pos Pred Value : 0.9403
##          Neg Pred Value : 0.3816
##              Prevalence : 0.9113
##          Detection Rate : 0.8557
##    Detection Prevalence : 0.9100
##       Balanced Accuracy : 0.6631
##
##        'Positive' Class : 0
##
```

```r
#calculate auc
library(ROCR)
library(pROC)
library(car)
pred1 = prediction(predict(logit1, test, type = "response"), test$Choice)
auc1 = round(as.numeric(performance(pred1, measure = "auc")@y.values), 3)
auc1
```

```
## [1] 0.8
```

```r
vif(logit1)
```

```
##           Gender Amount_purchased        Frequency    Last_purchase
##         1.023359         1.232172         2.490447        17.706670
##   First_purchase          P_Child          P_Youth            P_Cook
##         9.247748         2.992269         1.761546         3.229097
##            P_DIY            P_Art
##         1.992698         1.938089
```

Last_Purchase has a high VIF.

LOGISTIC REGRESSION 2

Fitted model excluding First_purchase, the least significant predictor in the full logistic model.

```
logit2 = glm(Choice ~.-First_purchase, data = train, family = "binomial")
summary(logit2)
```

```
##
## Call:
## glm(formula = Choice ~ . - First_purchase, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.44132  -0.66647  -0.43745  -0.01855   2.72460
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -0.2833949  0.2062721  -1.374   0.1695
## Gender1          -0.8660575  0.1373268  -6.307 2.85e-10 ***
## Amount_purchased  0.0018357  0.0007908   2.321   0.0203 *
## Frequency        -0.0903261  0.0106304  -8.497  < 2e-16 ***
## Last_purchase     0.5536689  0.0784519   7.057 1.70e-12 ***
## P_Child          -0.8181807  0.1163377  -7.033 2.02e-12 ***
## P_Youth          -0.6424923  0.1432548  -4.485 7.29e-06 ***
## P_Cook           -0.9330131  0.1190073  -7.840 4.51e-15 ***
## P_DIY            -0.9101106  0.1433591  -6.348 2.17e-10 ***
## P_Art             0.6643371  0.1255243   5.292 1.21e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1799.5  on 1599  degrees of freedom
## Residual deviance: 1393.5  on 1590  degrees of freedom
## AIC: 1413.5
##
## Number of Fisher Scoring iterations: 5
```

```
#Which predictors are signifcant and calculate model fit statistics
significant_if = summary(logit2)$coeff[-1,4]<.05
logit2.significant = names(significant_if)[significant_if ==TRUE]

logit2.significant
```

```
## [1] "Gender1"          "Amount_purchased" "Frequency"        "Last_purchase"
## [5] "P_Child"          "P_Youth"          "P_Cook"           "P_DIY"
## [9] "P_Art"
```

```
AIC = AIC(logit2)
BIC = BIC(logit2)
cbind(AIC, BIC)
```

```
##           AIC      BIC
## [1,] 1413.496 1467.273
```

```
#make predictions
library(caret)
test$PredProb = predict.glm(logit2, newdata=test, type = 'response')
test$Pred.Choice = ifelse(test$PredProb >= .5,1,0)
caret::confusionMatrix(as.factor(test$Pred.Choice), as.factor(test$Choice))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1973  127
##          1  123   77
##
##                Accuracy : 0.8913
##                  95% CI : (0.8779, 0.9037)
##     No Information Rate : 0.9113
##     P-Value [Acc > NIR] : 0.9995
##
##                   Kappa : 0.3216
##
##  Mcnemar's Test P-Value : 0.8495
##
##             Sensitivity : 0.9413
##             Specificity : 0.3775
##          Pos Pred Value : 0.9395
##          Neg Pred Value : 0.3850
##              Prevalence : 0.9113
##          Detection Rate : 0.8578
##    Detection Prevalence : 0.9130
##       Balanced Accuracy : 0.6594
##
##        'Positive' Class : 0
##
```

```
#calculate auc
library(ROCR)
library(pROC)
library(car)
pred2 = prediction(predict(logit2, test, type = "response"), test$Choice)
auc2 = round(as.numeric(performance(pred2, measure = "auc")@y.values), 3)
auc2
```

```
## [1] 0.801
```

LOGISTIC REGRESSION 3

Fitted model excluding Last_purchase because of its high VIF.

```r
logit3 = glm(Choice ~.-Last_purchase, data = train, family = "binomial")
summary(logit3)
```

```
##
## Call:
## glm(formula = Choice ~ . - Last_purchase, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.46171  -0.68074  -0.46620  -0.00855   2.80519
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -0.1489829  0.2095375  -0.711 0.477079
## Gender1          -0.8302649  0.1350384  -6.148 7.83e-10 ***
## Amount_purchased  0.0022691  0.0007747   2.929 0.003399 **
## Frequency        -0.1194992  0.0152620  -7.830 4.89e-15 ***
## First_purchase    0.0306235  0.0108454   2.824 0.004748 **
## P_Child          -0.3456948  0.0908420  -3.805 0.000142 ***
## P_Youth          -0.1789417  0.1226235  -1.459 0.144489
## P_Cook           -0.4578299  0.0950443  -4.817 1.46e-06 ***
## P_DIY            -0.4265209  0.1209960  -3.525 0.000423 ***
## P_Art             1.0778036  0.1144995   9.413  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1799.5  on 1599  degrees of freedom
## Residual deviance: 1437.0  on 1590  degrees of freedom
## AIC: 1457
##
## Number of Fisher Scoring iterations: 5
```

```r
#Which predictors are signifcant and calculate model fit statistics
significant_if = summary(logit3)$coeff[-1,4]<.05
logit3.significant = names(significant_if)[significant_if ==TRUE]

logit3.significant
```

```
## [1] "Gender1"          "Amount_purchased" "Frequency"        "First_purchase"
## [5] "P_Child"          "P_Cook"           "P_DIY"            "P_Art"
```

```r
AIC = AIC(logit3)
BIC = BIC(logit3)
cbind(AIC, BIC)
```

```
##           AIC      BIC
## [1,] 1456.978 1510.756
```

```r
#make predictions
library(caret)
test$PredProb = predict.glm(logit3, newdata=test, type = 'response')
test$Pred.Choice = ifelse(test$PredProb >= .5,1,0)
caret::confusionMatrix(as.factor(test$Pred.Choice), as.factor(test$Choice))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1987  131
##          1  109   73
##
##                Accuracy : 0.8957
##                  95% CI : (0.8824, 0.9079)
##     No Information Rate : 0.9113
##     P-Value [Acc > NIR] : 0.9956
##
##                   Kappa : 0.3215
##
##  Mcnemar's Test P-Value : 0.1752
##
##             Sensitivity : 0.9480
##             Specificity : 0.3578
##          Pos Pred Value : 0.9381
##          Neg Pred Value : 0.4011
##              Prevalence : 0.9113
##          Detection Rate : 0.8639
##    Detection Prevalence : 0.9209
##       Balanced Accuracy : 0.6529
##
##        'Positive' Class : 0
##
```

```r
#calculate auc
library(ROCR)
library(pROC)
library(car)
pred3 = prediction(predict(logit3, test, type = "response"), test$Choice)
auc3 = round(as.numeric(performance(pred3, measure = "auc")@y.values), 3)
auc3
```

```
## [1] 0.796
```

LOGISTIC REGRESSION 4

```r
logit4 = glm(Choice ~.-Last_purchase - P_Youth, data = train, family = "binomial")
summary(logit4)
```

```
##
```

```
## Call:
## glm(formula = Choice ~ . - Last_purchase - P_Youth, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.38384  -0.68970  -0.46632  -0.01251   2.81619
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -0.1948240  0.2065891  -0.943 0.345655
## Gender1          -0.8269170  0.1349290  -6.129 8.87e-10 ***
## Amount_purchased  0.0022126  0.0007718   2.867 0.004145 **
## Frequency        -0.1122481  0.0143722  -7.810 5.72e-15 ***
## First_purchase    0.0237578  0.0097616   2.434 0.014942 *
## P_Child          -0.3190401  0.0886869  -3.597 0.000321 ***
## P_Cook           -0.4274362  0.0920025  -4.646 3.39e-06 ***
## P_DIY            -0.4061888  0.1198011  -3.391 0.000698 ***
## P_Art             1.1071561  0.1129446   9.803  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1799.5  on 1599  degrees of freedom
## Residual deviance: 1439.1  on 1591  degrees of freedom
## AIC: 1457.1
##
## Number of Fisher Scoring iterations: 5
```

```
#Which predictors are signifcant and calculate model fit statistics
significant_if = summary(logit4)$coeff[-1,4]<.05
logit4.significant = names(significant_if)[significant_if ==TRUE]

logit4.significant
```

```
## [1] "Gender1"          "Amount_purchased" "Frequency"        "First_purchase"
## [5] "P_Child"          "P_Cook"           "P_DIY"            "P_Art"
```

```
AIC = AIC(logit4)
BIC = BIC(logit4)
cbind(AIC, BIC)
```

```
##           AIC      BIC
## [1,] 1457.149 1505.549
```

```
#make predictions
library(caret)
test$PredProb = predict.glm(logit4, newdata=test, type = 'response')
test$Pred.Choice = ifelse(test$PredProb >= .5,1,0)
caret::confusionMatrix(as.factor(test$Pred.Choice), as.factor(test$Choice))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1988  132
##          1  108   72
##
##                Accuracy : 0.8957
##                  95% CI : (0.8824, 0.9079)
##     No Information Rate : 0.9113
##     P-Value [Acc > NIR] : 0.9956
##
##                   Kappa : 0.3183
##
##  Mcnemar's Test P-Value : 0.1376
##
##             Sensitivity : 0.9485
##             Specificity : 0.3529
##          Pos Pred Value : 0.9377
##          Neg Pred Value : 0.4000
##              Prevalence : 0.9113
##          Detection Rate : 0.8643
##    Detection Prevalence : 0.9217
##       Balanced Accuracy : 0.6507
##
##        'Positive' Class : 0
##
```

```
#calculate auc
library(ROCR)
library(pROC)
library(car)
pred4 = prediction(predict(logit4, test, type = "response"), test$Choice)
auc4 = round(as.numeric(performance(pred4, measure = "auc")@y.values), 3)
auc4
```

```
## [1] 0.795
```

FINAL LOGISTIC REGRESSION MODEL

logit3 will be used as the final logistic regression model because it had the highest accuracy rate (89.57%). Logit4 had the same accuracy rate, but a lower AUC.

```
logit_final = logit3
```

```
odds_ratio = exp(logit_final$coefficients)
round(odds_ratio, 3)
```

```
##      (Intercept)          Gender1 Amount_purchased        Frequency
##            0.862            0.436            1.002            0.887
##   First_purchase          P_Child          P_Youth           P_Cook
```

```
##           1.031           0.708           0.836           0.633
##           P_DIY           P_Art
##           0.653           2.938
```

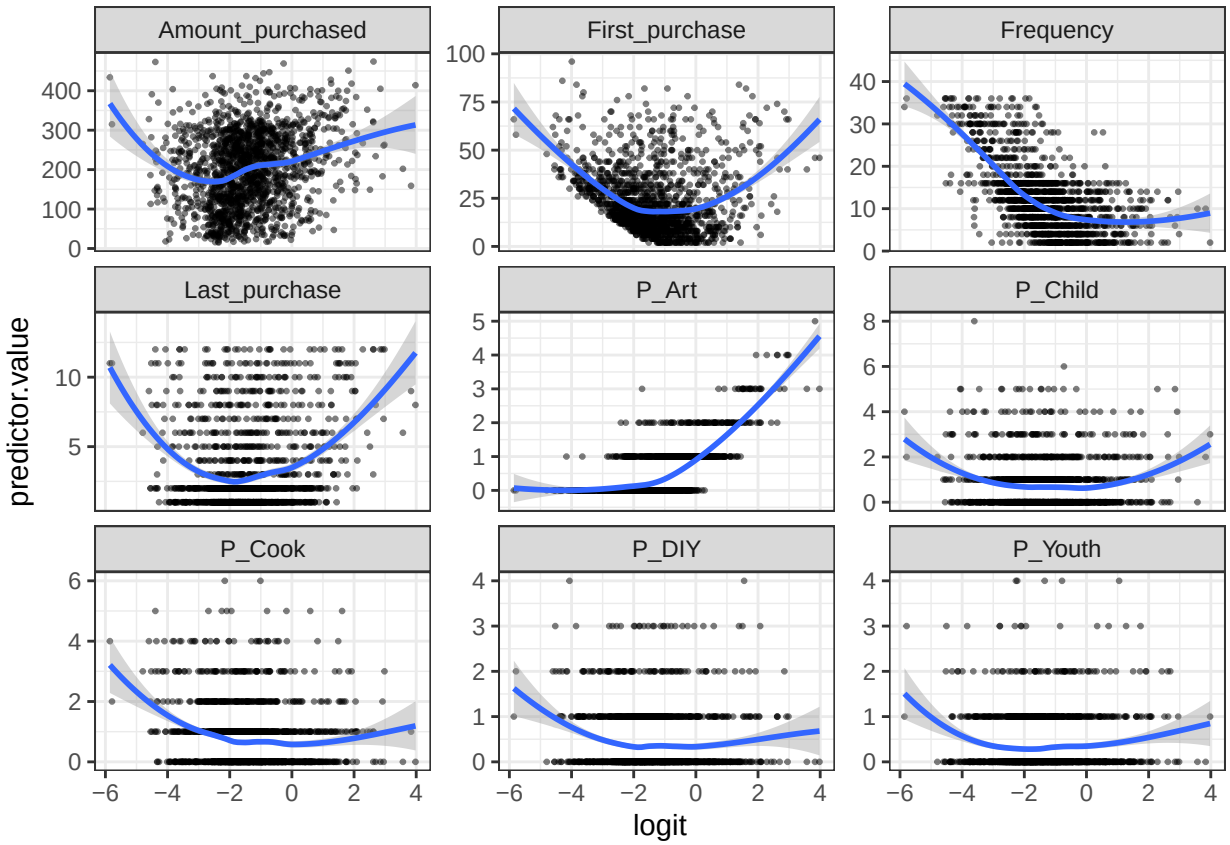
The coefficients of logistic models are not intuitive to interpret, so it's more common to use odds ratio for interpretation instead. An odds ratio less than 1 means that an increase in x leads to a decrease in the odds that y = 1. An odds ratio greater than 1 means that na increase in x leads to an increase in the odds that y = 1.

The odds of a purchase are 100(.436 - 1) = 56.4% lower for males than females. Each increase in Amount_purchased leads to a 100(1.002-1) = 20% increase in the odds of a purchase. Each additional purchase in the chosen period leads to a 100(.887-1) = 11.3% decrease in the odds of purchasing. For each additional month since the first purchase was made, the odds of purchasing increase by 100(1.031-1) = 3.1%. For each additional children's book purchased, the odds of purchase decrease by 100(.708-1) = 29.2%. For each additional youth book purchased, the odds of purchase decrease by 100(.836-1) = 16.4%, each additional cook book reduces the odds by 36.7%, each additional DIY book decreases the odds by 34.7%, and each additional art book purchased increases the odds of purchase by 193%!

This suggests BBBC would have better luck targeting their female customers and those who have purchased art books. Review assumptions of final logistic regression model selected:

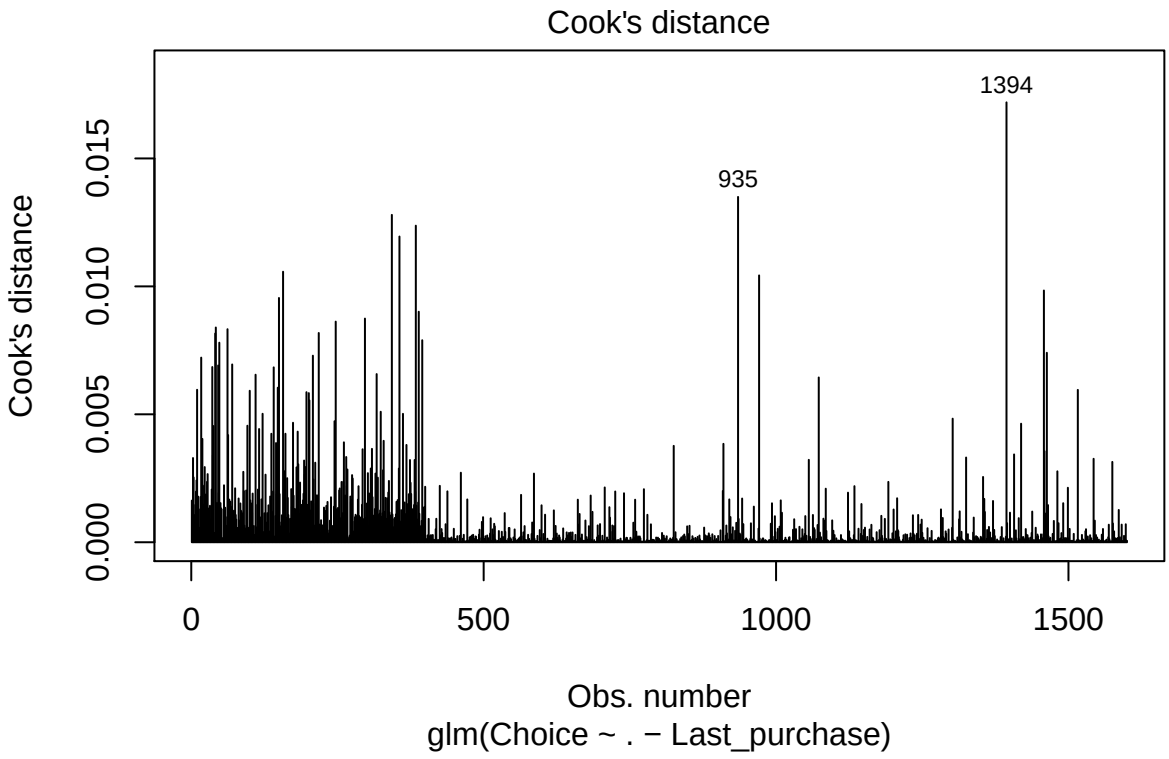
```
#linearity assumption

#predict probability of y
probabilities = predict(logit_final, type = "response")
predicted.classes = ifelse(probabilities > .5, 1, 0)

#select only numeric predictors
mydata=dplyr::select_if(train, is.numeric)
predictors = colnames(mydata)

#bind the logit and tidy the data for plotting
mydata = mutate(mydata, logit = log(probabilities / (1 - probabilities)))
mydata = gather(mydata, key = "predictors", value = "predictor.value", -logit)

#plot
ggplot(mydata, aes(logit, predictor.value))+
  geom_point(size = .5, alpha = .5) +
  geom_smooth(method = "loess") +
  theme_bw() +
  facet_wrap(~predictors, scales = "free_y")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



The plots above can be used to visually inspect if there is a linear relationship between the continuous predictor variables and the logit of the outcome. However, in this case most of the numeric variables are not continuous. Instead, they are discrete integer variables measuring counts (i.e. Frequency, P_Art, P_Child, P_Cook, etc.). Amount_purchased shows a roughly linear association with the Choice outcome in logit scale, aside from the points farthest to the left in the plot.

```
#influential observations
plot(logit_final, which = 4, id.n = 2) #cook's distance
```




```
#extract model results to compute std. residuals
library(broom)
```

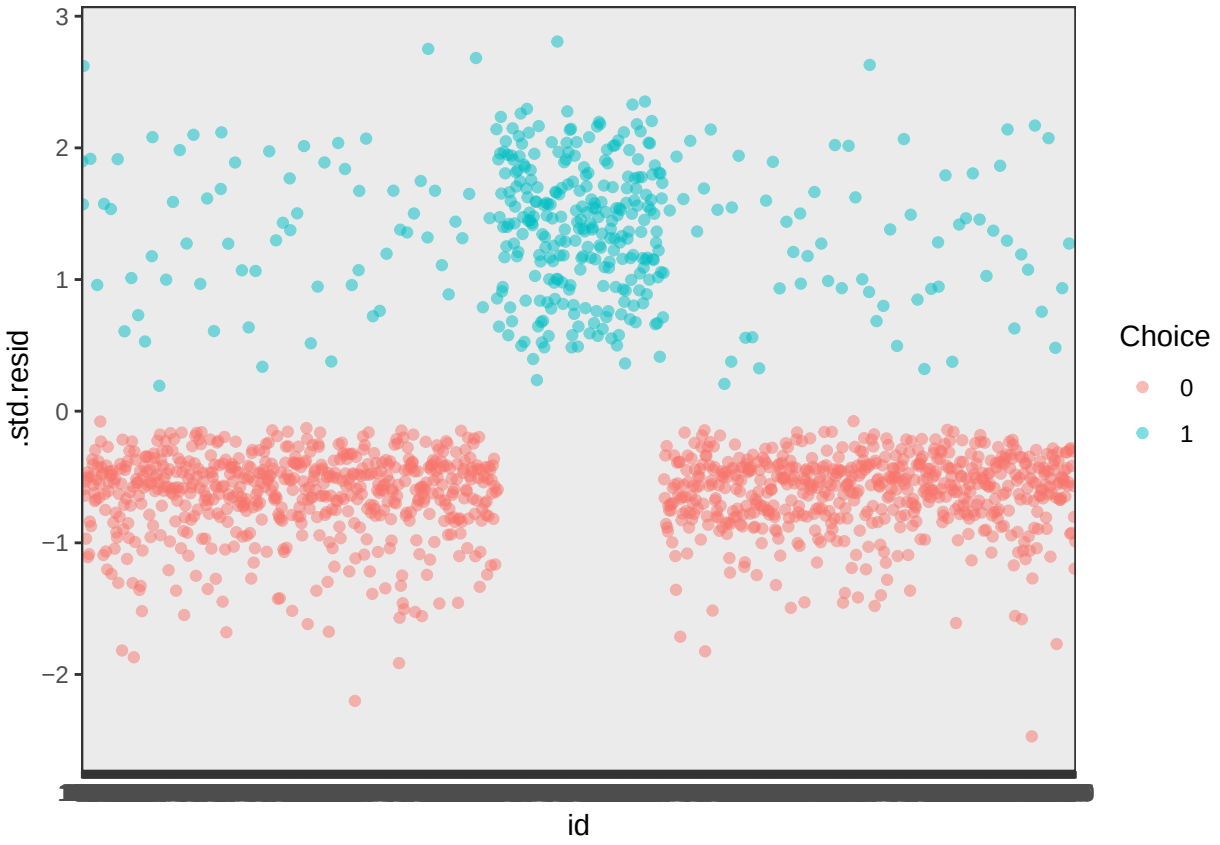
```
## Warning: package 'broom' was built under R version 3.6.3
```

```
logit_final.data = augment(logit_final) %>%
  mutate(index = 1:n())

#display the top largest values according to Cook's distance
logit_final.data %>% top_n(2, .cooksd)
```

```
## # A tibble: 2 x 18
##   Choice Gender Amount_purchased Frequency Last_purchase First_purchase P_Child
##   <fct>  <fct>             <dbl>     <int>         <dbl>          <dbl>   <int>
## 1 0      1                   263         4            12             66       2
## 2 0      1                   267        14            12             72       5
## # ... with 11 more variables: P_Youth <int>, P_Cook <int>, P_DIY <int>,
## #   P_Art <int>, .fitted <dbl>, .resid <dbl>, .std.resid <dbl>, .hat <dbl>,
## #   .sigma <dbl>, .cooksd <dbl>, index <int>
```

```
#add a column to identify rows
id = rownames(logit_final.data)
logit_final.data = cbind(id=id, logit_final.data)
```

```
#plot the standardized residuals
ggplot(logit_final.data, aes(id, .std.resid)) +
  geom_point(aes(color = Choice), alpha = .5) +
  theme_bw()
```




```
#filter potential influential data points with abs(.std.res) > 3
logit_final.data %>%
  filter(abs(.std.resid)>3)
```

```
##  [1] id                Choice            Gender            Amount_purchased
##  [5] Frequency         Last_purchase     First_purchase    P_Child
##  [9] P_Youth           P_Cook            P_DIY             P_Art
## [13] .fitted           .resid            .std.resid        .hat
## [17] .sigma            .cooksd           index
## <0 rows> (or 0-length row.names)
```

All absolute standardized residuals were below 3, indicating there are no outliers. If there were, they could be removed, the data could be transformed to a log scale, or a nonparametric method could be used instead. http://www.sthda.com/english/articles/36-classification-methods-essentials/148-logistic-regression-assumptions-and-diagnostics-in-r/

SUPPORT VECTOR MACHINE 1

```
library(e1071)
```

```
form1 = Choice ~ .
```

Declaring gamma ranging from .01 to .1 in increments of .01, which will return 10 values of gamma. Then iterate cost from 0.1 to 1 in increments of .1, which will return 10 values for cost.

```
set.seed(2021)
tuned = tune.svm(form1, data=train, gamma = seq(.01, .1, by = .01), cost = seq(.1, 1, by = .1))
```

```
#optimal parameters within our 10 by 10 grid
tuned$best.parameters
```

```
##    gamma cost
## 65  0.05  0.7
```

```
#run an SVM using the values of the best parameters using the radial kernel
svm1 = svm(form1, data=train, kernel = "radial", gamma = tuned$best.parameters$gamma, cost = tuned$best
summary(svm1)
```

```
##
## Call:
## svm(formula = form1, data = train, kernel = "radial", gamma = tuned$best.parameters$gamma,
##     cost = tuned$best.parameters$cost)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.7
##
## Number of Support Vectors:  785
##
##  ( 372 413 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
#make predictions
svm1_predict = predict(svm1, test, type = "response")
table(pred = svm1_predict, true = test$Choice)
```

```
##     true
## pred    0    1
##    0 2054  166
##    1   42   38
```

```
caret::confusionMatrix(svm1_predict, test$Choice)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2054  166
##          1   42   38
##
##                Accuracy : 0.9096
##                  95% CI : (0.8971, 0.921)
##     No Information Rate : 0.9113
##     P-Value [Acc > NIR] : 0.6327
##
##                   Kappa : 0.2291
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9800
##             Specificity : 0.1863
##          Pos Pred Value : 0.9252
##          Neg Pred Value : 0.4750
##              Prevalence : 0.9113
##          Detection Rate : 0.8930
##    Detection Prevalence : 0.9652
##       Balanced Accuracy : 0.5831
##
##        'Positive' Class : 0
##
```

```
#caret::confusionMatrix(test$Choice, svm1_predict)
```

SUPPORT VECTOR MACHINE 2

Train a SVM after dropping Last_purchase because it exhibited a high VIF in prior models.

```
form2 = Choice ~ . -Last_purchase
```

Declaring gamma ranging from .01 to .1 in increments of .01, which will return 10 values of gamma. Then iterate cost from 0.1 to 1 in increments of .1, which will return 10 values for cost.

```
set.seed(2021)
tuned2 = tune.svm(form2, data=train, gamma = seq(.01, .1, by = .01), cost = seq(.1, 1, by = .1))
```

```
#optimal parameters within our 10 by 10 grid
tuned2$best.parameters
```

```
##    gamma cost
## 91  0.01    1
```

```r
#run an SVM using the values of the best parameters using the radial kernel
svm2 = svm(form2, data=train, kernel = "radial", gamma = tuned2$best.parameters$gamma, cost = tuned2$bes
summary(svm2)
```

```
##
## Call:
## svm(formula = form2, data = train, kernel = "radial", gamma = tuned2$best.parameters$gamma,
##     cost = tuned2$best.parameters$cost)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  782
##
##  ( 387 395 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```r
#make predictions
svm2_predict = predict(svm2, test, type = "response")
table(pred = svm2_predict, true = test$Choice)
```

```
##     true
## pred    0    1
##    0 2058  175
##    1   38   29
```

```r
caret::confusionMatrix(svm2_predict, test$Choice)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2058  175
##          1   38   29
##
##                Accuracy : 0.9074
##                  95% CI : (0.8948, 0.9189)
##     No Information Rate : 0.9113
##     P-Value [Acc > NIR] : 0.7586
##
##                   Kappa : 0.178
##
##  Mcnemar's Test P-Value : <2e-16
##
```

```
##             Sensitivity : 0.9819
##             Specificity : 0.1422
##          Pos Pred Value : 0.9216
##          Neg Pred Value : 0.4328
##              Prevalence : 0.9113
##          Detection Rate : 0.8948
##    Detection Prevalence : 0.9709
##       Balanced Accuracy : 0.5620
##
##        'Positive' Class : 0
##
```

SUPPORT VECTOR MACHINE 3

Train a SVM after dropping First_purchase because it exhibited a high VIF in prior models.

```
form3 = Choice ~ . - First_purchase
```

Declaring gamma ranging from .01 to .1 in increments of .01, which will return 10 values of gamma. Then iterate cost from 0.1 to 1 in increments of .1, which will return 10 values for cost.

```
set.seed(2021)
tuned3 = tune.svm(form3, data=train, gamma = seq(.01, .1, by = .01), cost = seq(.1, 1, by = .1))
```

```
#optimal parameters within our 10 by 10 grid
tuned3$best.parameters
```

```
##    gamma cost
## 80   0.1  0.8
```

```
#run an SVM using the values of the best parameters using the RBF kernel
svm3 = svm(form3, data=train, kernel = "radial", gamma = tuned3$best.parameters$gamma, cost = tuned3$bes
summary(svm3)
```

```
##
## Call:
## svm(formula = form3, data = train, kernel = "radial", gamma = tuned3$best.parameters$gamma,
##     cost = tuned3$best.parameters$cost)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.8
##
## Number of Support Vectors:  804
##
##  ( 370 434 )
##
##
```

```
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
#make predictions
svm3_predict = predict(svm3, test, type = "response")
table(pred = svm3_predict, true = test$Choice)
```

```
##     true
## pred    0    1
##    0 2035  154
##    1   61   50
```

```
caret::confusionMatrix(svm3_predict, test$Choice)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2035  154
##          1   61   50
##
##                Accuracy : 0.9065
##                  95% CI : (0.8939, 0.9181)
##     No Information Rate : 0.9113
##     P-Value [Acc > NIR] : 0.8013
##
##                   Kappa : 0.272
##
##  Mcnemar's Test P-Value : 3.511e-10
##
##             Sensitivity : 0.9709
##             Specificity : 0.2451
##          Pos Pred Value : 0.9296
##          Neg Pred Value : 0.4505
##              Prevalence : 0.9113
##          Detection Rate : 0.8848
##    Detection Prevalence : 0.9517
##       Balanced Accuracy : 0.6080
##
##        'Positive' Class : 0
##
```

SUPPORT VECTOR MACHINE 4

Using form1 from svm1, but with a linear kernel

```
form4 = form1
```

Declaring gamma ranging from .01 to .1 in increments of .01, which will return 10 values of gamma. Then iterate cost from 0.1 to 1 in increments of .1, which will return 10 values for cost.

```r
set.seed(2021)
tuned4 = tune.svm(form4, data=train, gamma = seq(.01, .1, by = .01), cost = seq(.1, 1, by = .1))
```

```r
#optimal parameters within our 10 by 10 grid
tuned4$best.parameters
```

```
##    gamma cost
## 65  0.05  0.7
```

```r
#run an SVM using the values of the best parameters using the linear kernel
svm4 = svm(form4, data=train, kernel = "linear", gamma = tuned4$best.parameters$gamma, cost = tuned4$bes
summary(svm4)
```

```
##
## Call:
## svm(formula = form4, data = train, kernel = "linear", gamma = tuned4$best.parameters$gamma,
##     cost = tuned4$best.parameters$cost)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.7
##
## Number of Support Vectors:  739
##
##  ( 367 372 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```r
#make predictions
svm4_predict = predict(svm4, test, type = "response")
table(pred = svm4_predict, true = test$Choice)
```

```
##     true
## pred    0    1
##    0 2011  147
##    1   85   57
```

```r
caret::confusionMatrix(svm4_predict, test$Choice)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2011  147
```

```
##          1   85   57
##
##                Accuracy : 0.8991
##                  95% CI : (0.8861, 0.9111)
##     No Information Rate : 0.9113
##     P-Value [Acc > NIR] : 0.9802
##
##                   Kappa : 0.2768
##
##  Mcnemar's Test P-Value : 6.206e-05
##
##             Sensitivity : 0.9594
##             Specificity : 0.2794
##          Pos Pred Value : 0.9319
##          Neg Pred Value : 0.4014
##              Prevalence : 0.9113
##          Detection Rate : 0.8743
##    Detection Prevalence : 0.9383
##       Balanced Accuracy : 0.6194
##
##        'Positive' Class : 0
##
```

FINAL SVM MODEL

```
svm_final = svm1
```

```
#recall the confusion matrix to get TP, TN, FP, TN
#the confusion matrix for svm_final is the same as svm_1
caret::confusionMatrix(svm1_predict, test$Choice)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2054  166
##          1   42   38
##
##                Accuracy : 0.9096
##                  95% CI : (0.8971, 0.921)
##     No Information Rate : 0.9113
##     P-Value [Acc > NIR] : 0.6327
##
##                   Kappa : 0.2291
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9800
##             Specificity : 0.1863
##          Pos Pred Value : 0.9252
##          Neg Pred Value : 0.4750
```

```
##              Prevalence : 0.9113
##          Detection Rate : 0.8930
##    Detection Prevalence : 0.9652
##       Balanced Accuracy : 0.5831
##
##        'Positive' Class : 0
##
```

```
TN = 2054 #true negatives
FN = 166  #false negatives
FP = 42   #false positives
TP = 38   #true positives
```

MAXIMIZING PROFITABILITY

BBBC is considering a similar mail compaign in the Midwest where it has data for 50,000 customers. They want to know which customers to target and how much more profit coudl they expect to generate using the models prepared, compared to sending the mailer to the entire list.

Compare cost of a mass campaign vs. a targeted campaign

```
cost_no_purchase = 0.65
cost_yes_purchase = .65+(1.45*15)
revenue_per_purchase = 31.95
```

Estimate profit from a mass mailing campaign

```
Mass_Total_Cost = ((TP+FN)*cost_yes_purchase)+((FP+TN)*cost_no_purchase)
Mass_Total_Revenue = (TP+FN)*revenue_per_purchase
Mass_Profit = Mass_Total_Revenue - Mass_Total_Cost
Mass_Profit_per_Mailer = Mass_Profit / (TP+FN+FP+TN)

Mass_Total_Cost
```

```
## [1] 5932
```

```
Mass_Total_Revenue
```

```
## [1] 6517.8
```

```
Mass_Profit
```

```
## [1] 585.8
```

```
Mass_Profit_per_Mailer
```

```
## [1] 0.2546957
```

Estimate profit from a targeted mailing campaign based on SVM model

```r
#only send mailer to those predicted to be positive
Targeted_Total_Cost = (TP*cost_yes_purchase)+(FP*cost_no_purchase)
Targeted_Total_Revenue = (TP*revenue_per_purchase)
Targeted_Profit = Targeted_Total_Revenue - Targeted_Total_Cost
Targeted_Mailers = TP + FP
Target_Profit_per_Mailer = Targeted_Profit / Targeted_Mailers

Targeted_Total_Cost
```

```
## [1] 878.5
```

```r
Targeted_Total_Revenue
```

```
## [1] 1214.1
```

```r
Targeted_Profit
```

```
## [1] 335.6
```

```r
Targeted_Mailers
```

```
## [1] 80
```

```r
Target_Profit_per_Mailer
```

```
## [1] 4.195
```