MKTG 6620-001 - MACHINE LEARNING FOR BUSINESS APPLICATIONS

FINAL PROJECT REPORT NOV-20-2023

DEBAYAN DUTTA (U1427957)

THE UNIVERSITY OF UTAH

MKTG 6620-001 - Machine Learning for Business Applications - Final Project

MSBA Fall 2023 - Link to Posit Cloud Code: https://posit.cloud/content/7032286

Debayan Dutta

2023-11-20

Contents

1 BUSINESS PROBLEM DEFINITION (Define the problem) :

The Grocery store chain that sells two types of orange juices, one Citrus Hill(CH) and the other Minute Maid (MM) is challenged with the desire to improve the performance of orange juice category by increasing overall sale, particularly the MM brand sale. It is documented that MM brand has a higher margin, and make more money on per unit sale than CH. The marketing manager is curious to know what factors (predictor variables) associated with the grocery store or each of individual products influence the choice of customer who either buys CH brand or MM brand. In turn the brand manager wants to know the variables that influence the probability of a customer purchasing MM brand orange juice. The manager wishes to understand how they can manipulate these factors to increase the sale of MM brand juice. The Sales manager on the other hand wishes to devise a predictive model that will predict the probability of a customer purchasing the MM brand juice.

The Objectives inferred from business problem description are as follows

1. Identify the most influencing predictor variables that has higher effect on customer purchasing.
2. Identify how the strongly influencing predictors influence the purchase of either CH or MM brand i.e what is relation between predictors and target varible is?
3. Are all predictors important predictors? Are there predictors influencing behavior of other predictors?
4. Build a holistic model that provides evidence of predictors influencing Purchase and also predicts the probability of customer purchasing MM brand juice with higher and better accuracy.

2 APPROACH AND ANALYSIS (Define method (s)) :

2.1 Loading necessary libraries

```
options(warn = -1)

library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(dplyr)
library(ggplot2)
library(tidyr)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8
```

```r
library(carData)
library(car)
```

```
##
## Attaching package: 'car'
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following object is masked from 'package:purrr':
##
##     some
```

```r
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```r
library(parsnip)
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.1.1 --
## v broom        1.0.5     v rsample      1.2.0
## v dials        1.2.0     v tune         1.1.2
## v infer        1.0.5     v workflows    1.1.3
## v modeldata    1.2.0     v workflowsets 1.0.1
## v recipes      1.0.8     v yardstick    1.2.0
## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard()        masks purrr::discard()
## x Matrix::expand()         masks tidyr::expand()
## x dplyr::filter()          masks stats::filter()
## x recipes::fixed()         masks stringr::fixed()
## x dplyr::lag()             masks stats::lag()
## x caret::lift()            masks purrr::lift()
## x Matrix::pack()           masks tidyr::pack()
## x yardstick::precision()   masks caret::precision()
## x yardstick::recall()      masks caret::recall()
## x car::recode()            masks dplyr::recode()
## x yardstick::sensitivity() masks caret::sensitivity()
## x car::some()              masks purrr::some()
## x yardstick::spec()        masks readr::spec()
## x yardstick::specificity() masks caret::specificity()
## x recipes::step()          masks stats::step()
## x Matrix::unpack()         masks tidyr::unpack()
## x recipes::update()        masks Matrix::update(), stats::update()
## * Search for functions across packages at https://www.tidymodels.org/find/
```

```r
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
##
## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(vip)
```

```
##
## Attaching package: 'vip'
##
## The following object is masked from 'package:utils':
##
##     vi
```

2.2 Loading dataset:

```
OJ<-read.csv(url("http://data.mishra.us/files/project/OJ_data.csv"))
OJ[2:14] <- lapply(OJ[2:14], as.numeric)
OJ$Purchase <- as.factor(OJ$Purchase)
sapply(OJ,class)
```

```
##      Purchase       PriceCH       PriceMM        DiscCH        DiscMM
##      "factor"     "numeric"     "numeric"     "numeric"     "numeric"
##     SpecialCH     SpecialMM       LoyalCH   SalePriceMM   SalePriceCH
##     "numeric"     "numeric"     "numeric"     "numeric"     "numeric"
##     PriceDiff     PctDiscMM     PctDiscCH ListPriceDiff
##     "numeric"     "numeric"     "numeric"     "numeric"
```

```
dd <- OJ

# Data set type:
class(dd)
```

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

```
# Structure of the data set:
#str(dd)

# Check class distribution of target variable:
summary(dd$Purchase)
```

```
##   0   1
## 417 653
```

From the data frame summary we can observe that the minority class is PurchaseMM == 0 with 417 count while the majority class is PurchaseCH == 1 with 653. We can also safely assume that a difference of (653 -417) = 236 in count does not contribute towards class imbalance and hence we will not have to perform any downsampling.

```
# Checking descriptive statistics of price columns for MM and CH

variances <- apply(dd[-1], 2, var)
means <- apply(dd[-1], 2, mean)
medians <- apply(dd[-1], 2, median)
```

```r
std_devs <- apply(dd[-1], 2, sd)


ddstats <- data.frame(
  Mean = means,
  Median = medians,
  SD = std_devs,
  Variance = variances
)

print(ddstats)
```

```
##                     Mean Median        SD    Variance
## PriceCH       1.86742056   1.86 0.1019698 0.010397830
## PriceMM       2.08541121   2.09 0.1343855 0.018059466
## DiscCH        0.05185981   0.00 0.1174742 0.013800186
## DiscMM        0.12336449   0.00 0.2138337 0.045724872
## SpecialCH     0.14766355   0.00 0.3549321 0.125976762
## SpecialMM     0.16168224   0.00 0.3683312 0.135667888
## LoyalCH       0.56578233   0.60 0.3078425 0.094767023
## SalePriceMM   1.96204673   2.09 0.2526974 0.063855957
## SalePriceCH   1.81556075   1.86 0.1433836 0.020558853
## PriceDiff     0.14648598   0.23 0.2715627 0.073746293
## PctDiscMM     0.05929844   0.00 0.1017600 0.010355106
## PctDiscCH     0.02731384   0.00 0.0622324 0.003872872
## ListPriceDiff 0.21799065   0.24 0.1075354 0.011563873
```

From the above descriptive analysis we can see, that the price variation for MM is higher than CH. Median price of MM is 2.08 unit while that of CH is 1.86 unit.

This begs a question whether the higher margin of MM is because of its higher median price and not because of the quantities sold. We know from the purchase summary number of customers purchased MM is less than CH hence we can say priceMM is a factor that brings more revenue but does not necessarily confirms that all ordered MM items are sold by the end which can lead to overstocking.

2.3 Check for missing values:

```r
# Missing value count in the data frame

sum(is.na(dd))
```

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

There is no missing values in the dataset hence imputation is not required.

2.4 Split the dataset into train and test set (80:20):

```r
set.seed(1234)

datasplit <- createDataPartition(dd$Purchase, p = 0.8, list=FALSE)
trainData <- dd[datasplit,]
testData <- dd[-datasplit,]


Xtrain <- trainData[-1]
ytrain <- trainData$Purchase

Xtest <- testData[-1]
ytest <- testData$Purchase
```

2.5 Implementing Logistic Regression - using all predictor variables:

```r
modelLR1 <- glm(ytrain ~.,data = Xtrain, family = binomial(link = "logit"))

summary(modelLR1)
```

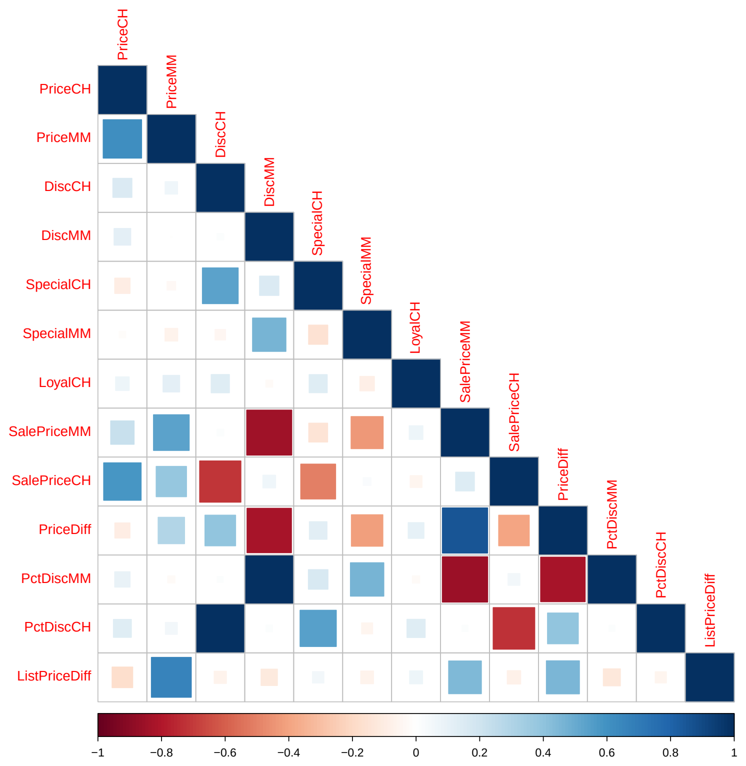
```
##
## Call:
## glm(formula = ytrain ~ ., family = binomial(link = "logit"),
##     data = Xtrain)
##
## Coefficients: (4 not defined because of singularities)
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -5.067392   2.233663  -2.269  0.02329 *
## PriceCH         -3.690282   1.430307  -2.580  0.00988 **
## PriceMM          4.364578   0.998685   4.370 1.24e-05 ***
## DiscCH         -18.965006  20.936871  -0.906  0.36503
## DiscMM         -27.254777   9.822278  -2.775  0.00552 **
## SpecialCH       -0.003476   0.369540  -0.009  0.99250
## SpecialMM       -0.140405   0.300234  -0.468  0.64003
## LoyalCH          6.825682   0.465227  14.672  < 2e-16 ***
## SalePriceMM            NA         NA      NA       NA
## SalePriceCH            NA         NA      NA       NA
## PriceDiff              NA         NA      NA       NA
## PctDiscMM       51.446780  20.530276   2.506  0.01221 *
## PctDiscCH       42.850231  39.723091   1.079  0.28071
## ListPriceDiff          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: 1146.03  on 856  degrees of freedom
## Residual deviance:  639.21  on 847  degrees of freedom
## AIC: 659.21
##
## Number of Fisher Scoring iterations: 5
```

From the prelimianry analysis, after implementing logistic regression, we can see that the predictor variables with highest effect size on purchase are ‘PriceCH’, ‘PriceCH’, ‘DiscMM’, ‘LoyalCH’ and ‘PctDiscMM’.

It is also observed that coefficients of some of the predictors are ‘NA’ indicating these might be redundant for the fact that they are highly correlated. These predictors are ‘SalePriceMM’, ‘SalePriceCH’, ‘PriceDiff’, ‘ListPriceDiff’. We further investigate the collinearity among the variables.

2.6 Implementing Logistic Regression - Checking Collinearity among predictor variables:

```
correlation_data = dd[,-1]

correlation <- cor(correlation_data)

corrplot(correlation, method = "square", type = "lower")
```



2.7 Applying Penalized Regression - Lasso to determin the important variables:

```
set.seed(1234)

# Lasso Regression

#perform k-fold cross-validation to find optimal lambda value
```

```
cv_model_Lasso <- cv.glmnet(as.matrix(Xtrain),
                            as.numeric(ytrain == 0), alpha = 1)

#find optimal lambda value that minimizes test MSE
best_lambda <- cv_model_Lasso$lambda.min
best_lambda
```

```
## [1] 0.006361714
```

```
best_lasso <- glmnet(as.matrix(Xtrain),as.numeric(ytrain == 0),
                     alpha = 1, lambda = best_lambda)

best_lasso
```

```
##
## Call:  glmnet(x = as.matrix(Xtrain), y = as.numeric(ytrain == 0), alpha = 1,      lambda = best_lamb
##
##   Df  %Dev   Lambda
## 1  5 47.13 0.006362
```

```
coefficients_lasso <- coef(best_lasso)

important_variables_lasso <- names(coefficients_lasso[coefficients_lasso != 0])

coefficients_lasso
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)    0.98005309
## PriceCH        .
## PriceMM        .
## DiscCH         .
## DiscMM         .
## SpecialCH      .
## SpecialMM      0.02214050
## LoyalCH       -0.97621202
## SalePriceMM    .
## SalePriceCH    0.01663748
## PriceDiff     -0.33656840
## PctDiscMM      .
## PctDiscCH      .
## ListPriceDiff -0.11735137
```

2.8 Implementing Logistic Regression - Remodeling logistic regression with only variables obtained from lasso regression output:

```
modelLR2 <- glm(ytrain ~ LoyalCH + PriceDiff + ListPriceDiff,data = Xtrain,
                family = binomial(link = "logit"))

summary(modelLR2)
```

```
##
## Call:
## glm(formula = ytrain ~ LoyalCH + PriceDiff + ListPriceDiff, family = binomial(link = "logit"),
##     data = Xtrain)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)    
## (Intercept)    -3.5511     0.3282 -10.821  < 2e-16 ***
## LoyalCH         6.7353     0.4527  14.879  < 2e-16 ***
## PriceDiff       2.9405     0.4301   6.836 8.13e-12 ***
## ListPriceDiff   0.7469     1.0251   0.729    0.466    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1146.03  on 856  degrees of freedom
## Residual deviance:  648.51  on 853  degrees of freedom
## AIC: 656.51
##
## Number of Fisher Scoring iterations: 5
```

AIC score of model_LR1 = 659.21 AIC score of model_LR2 = 656.51 We can observe that when we regressed the target variable Purchase against the predictor variables obtained from the lasso regression there is drop in the AIC score which is healthy when comparing two models.

2.9 Implementing Logistic Regression - to identify how much does the predictors only affect Purchase of MM

```r
modelLR3 <- glm(ytrain == 0 ~ LoyalCH + PriceDiff + ListPriceDiff,
                data = Xtrain, family = binomial(link = "logit"))

summary(modelLR3)
```

```
##
## Call:
## glm(formula = ytrain == 0 ~ LoyalCH + PriceDiff + ListPriceDiff,
##     family = binomial(link = "logit"), data = Xtrain)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)    
## (Intercept)     3.5511     0.3282  10.821  < 2e-16 ***
## LoyalCH        -6.7353     0.4527 -14.879  < 2e-16 ***
## PriceDiff      -2.9405     0.4301  -6.836 8.13e-12 ***
## ListPriceDiff  -0.7469     1.0251  -0.729    0.466    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1146.03  on 856  degrees of freedom
## Residual deviance:  648.51  on 853  degrees of freedom
## AIC: 656.51
##
## Number of Fisher Scoring iterations: 5
```

It is observed that the two major predictors affecting the purhase of MM are LoyalCH and PriceDiff, both being inversely proportional to MM purchasing.

This implies that the for every 6.7258 unit decrease in loyalty of CH customer there is 1 unit rise in purchase of MM. Loyal CH customer tends to buy CH brand orange juice irrespective of price.

Again for every 2.8490 unit decrease in price difference between CH price and MM price, increases the purchase of MM price. This difference can be both increase in of CH/MM or decrease in price of CH/MM.

2.10 Implementing Logistic Regression - Creating new train and test set with predictor variables obtained from Lasso Regression:

```
Xtrain_modelLR2 <- Xtrain %>% select(LoyalCH,PriceDiff,ListPriceDiff)

ytrain_modelLR2 <- ytrain

Xtest_modelLR2 <- Xtest %>% select(LoyalCH,PriceDiff,ListPriceDiff)

ytest_modelLR2 <- ytest
```

2.11 Implementing Logistic Regression - Prediction using modelLR2 with train set and test set:

```
prediction_modelLR2_train_prob <- predict(modelLR2, Xtrain_modelLR2, type = "response")
prediction_modelLR2_train <- ifelse(prediction_modelLR2_train_prob > 0.5,1,0)
prediction_modelLR2_train <- as.factor(prediction_modelLR2_train)

prediction_modelLR2_test_prob <- predict(modelLR2, Xtest_modelLR2, type = "response")
prediction_modelLR2_test <- ifelse(prediction_modelLR2_test_prob > 0.5,1,0)
prediction_modelLR2_test <- as.factor(prediction_modelLR2_test)
```

2.12 Implementing Logistic Regression - Performance Evaluation of modelLR2 on train set:

```
t(confusionMatrix(prediction_modelLR2_train,ytrain_modelLR2)$table)
```

```
##          Prediction
## Reference   0   1
##         0 260  74
##         1  67 456
```

```
(confusionMatrix(prediction_modelLR2_train,ytrain_modelLR2)$byClass)
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.7784431            0.8718929            0.7951070
##       Neg Pred Value            Precision               Recall
##            0.8603774            0.7951070            0.7784431
##                   F1           Prevalence       Detection Rate
##            0.7866868            0.3897316            0.3033839
## Detection Prevalence    Balanced Accuracy
##            0.3815636            0.8251680
```

2.13 Implementing Logistic Regression - Performance Evaluation of modelLR2 on test set:

```
t(confusionMatrix(prediction_modelLR2_test,ytest_modelLR2)$table)
```

```
##          Prediction
## Reference   0   1
##         0  56  27
##         1  13 117
```

```
(confusionMatrix(prediction_modelLR2_test,ytest_modelLR2)$byClass)
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.6746988            0.9000000            0.8115942
##       Neg Pred Value            Precision               Recall
##            0.8125000            0.8115942            0.6746988
##                   F1           Prevalence       Detection Rate
##            0.7368421            0.3896714            0.2629108
## Detection Prevalence    Balanced Accuracy
##            0.3239437            0.7873494
```

From the first confusion matrix we can see that the model has correctly predicted 456 instances meaning it is the True Positive and 74 instances are False Positive for the train data set.

From the second confusion matrix we can see that the model has correctly predicted 117 instances meaning it is the True Positive and 27 instances are False Positive for the test data set.

2.14 Implementing Logistic Regression- Evaluating AUC-ROC of modelLR2:

```
roc_curve <- roc(response = ytest_modelLR2, predictor = prediction_modelLR2_test_prob)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
auc_roc <- auc(roc_curve)
print(auc_roc)
```

```
## Area under the curve: 0.8697
```

2.15 Implementing XGBoost Model - Initial data split:

For this we will split the original data into traindata and testdata. We are not scaling or standardizing the data for XGBoost model as this model will intuitively take care of any kind of scaling requirements.

```
set.seed(123)

datapart <- initial_split(dd, prop = 0.8, strata = Purchase)

traindata <- training(datapart)
testdata <- testing(datapart)
```

2.16 Implementing XGBoost Model - model formulation:

```
rec_Purchase <- recipe(Purchase ~ ., traindata) %>% prep(training = traindata)
```

2.17 Implementing XGBoost Model - Define XGBoost algorithm:

```
modelXGB1 <- boost_tree(
trees = tune(),
tree_depth = tune(),
learn_rate = tune()) %>%
set_engine("xgboost", verbosity = 0) %>%
set_mode("classification")
```

2.18 Implementing XGBoost Model - Hyperparameter tuning and Cross-Validation using GridSearch:

```
hyper_grid <- grid_regular(
trees(),
tree_depth(),
learn_rate(),
levels = 4)

CVfolds <- vfold_cv(traindata, v=5)
```

XGboost is a blackbox model with higher predictive power and the most tuned model can be obtained by tuning the hyperparameters (also known as the model performance parameters) to achive the best fit. For our model we have incorporated 3 parameters number of trees in the model, tree depth, and learning rate. We use grid search technique to find the best hyperparameter combination that helps us achieve the best XGBoost model.

2.19 Implementing XGBoost Model - Model fit and Prediction:

```
modelXGB1_wf <- workflow() %>%
add_model(modelXGB1) %>%
add_recipe(rec_Purchase)
```

2.20 Implementing XGBoost Model - Computing accuracy & aucroc score of modelXGB1:

```
set.seed(1234)

modelXGB1_tune <- modelXGB1_wf %>%
tune_grid(
resamples = CVfolds,
grid = hyper_grid,
metrics = metric_set(accuracy,roc_auc)
)
```

2.21 Implementing XGBoost Model - Seleting the best hyperparameter combinations:

```
best_modelXGB1 <- modelXGB1_tune %>%
select_best("accuracy","roc_auc")

best_modelXGB1
```

2.22 Implementing XGBoost Model - model with best hyperparameter:

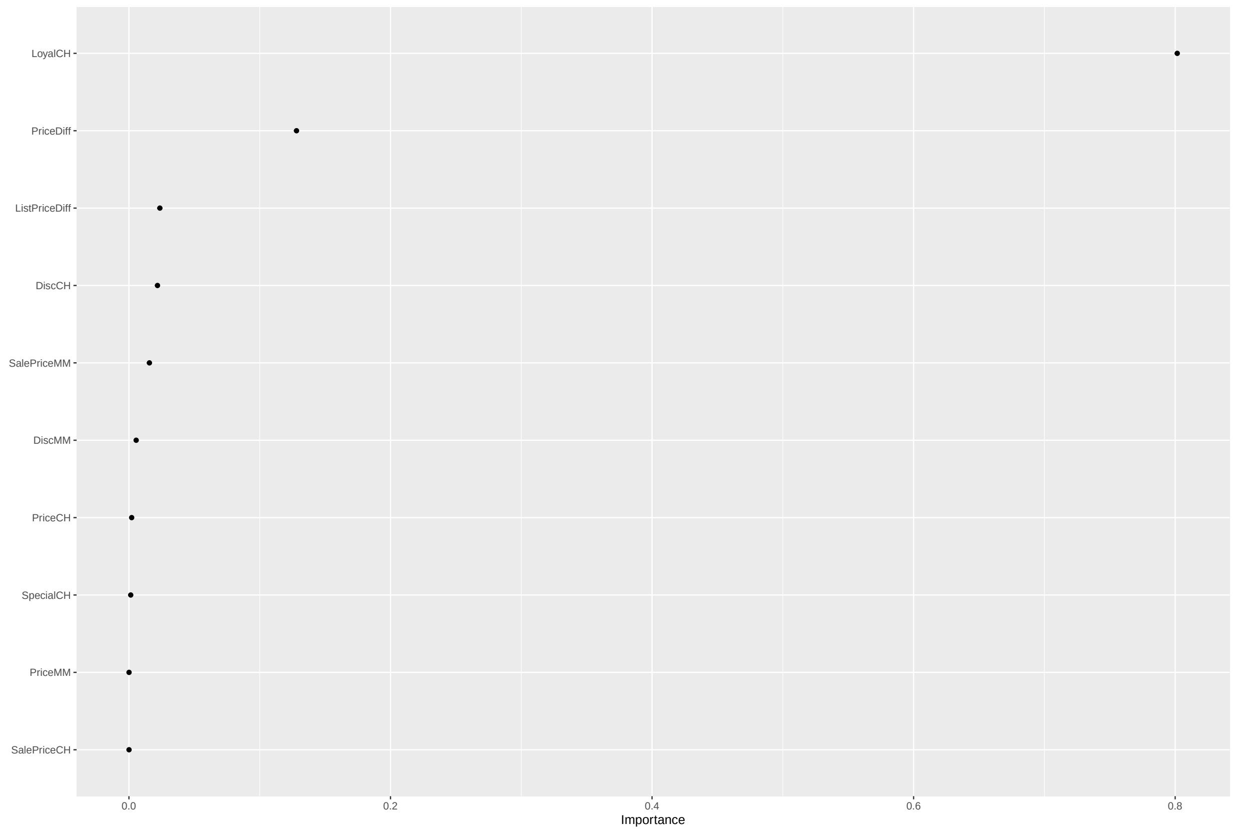
```
modelXGB1_finalWF <- modelXGB1_wf %>% finalize_workflow(best_modelXGB1)
```

2.23 Implementing XGBoost Model - Fitting the best model:

```
modelXGB_fit <- modelXGB1_finalWF %>%

  last_fit(split = datapart)

modelXGB_fit %>%

  collect_metrics()
```

2.24 Implementing XGBoost Model - Identifying most important predictors visually:

```
modelXGB1_finalWF %>%
fit(data = traindata) %>%
extract_fit_parsnip() %>%
vip(geom = "point")
```



We can observe from the feature importance plot that the most important predictor variable is LoyalCH, followed by PriceDiff and ListPriceDiff that influences the purchase of MM over CH.

2.25 Implementing XGBoost Model - Inspecting how the important predictor variables influence the target variable using PDP plots:

```
library(DALEXtra)
```

```
## Loading required package: DALEX
```

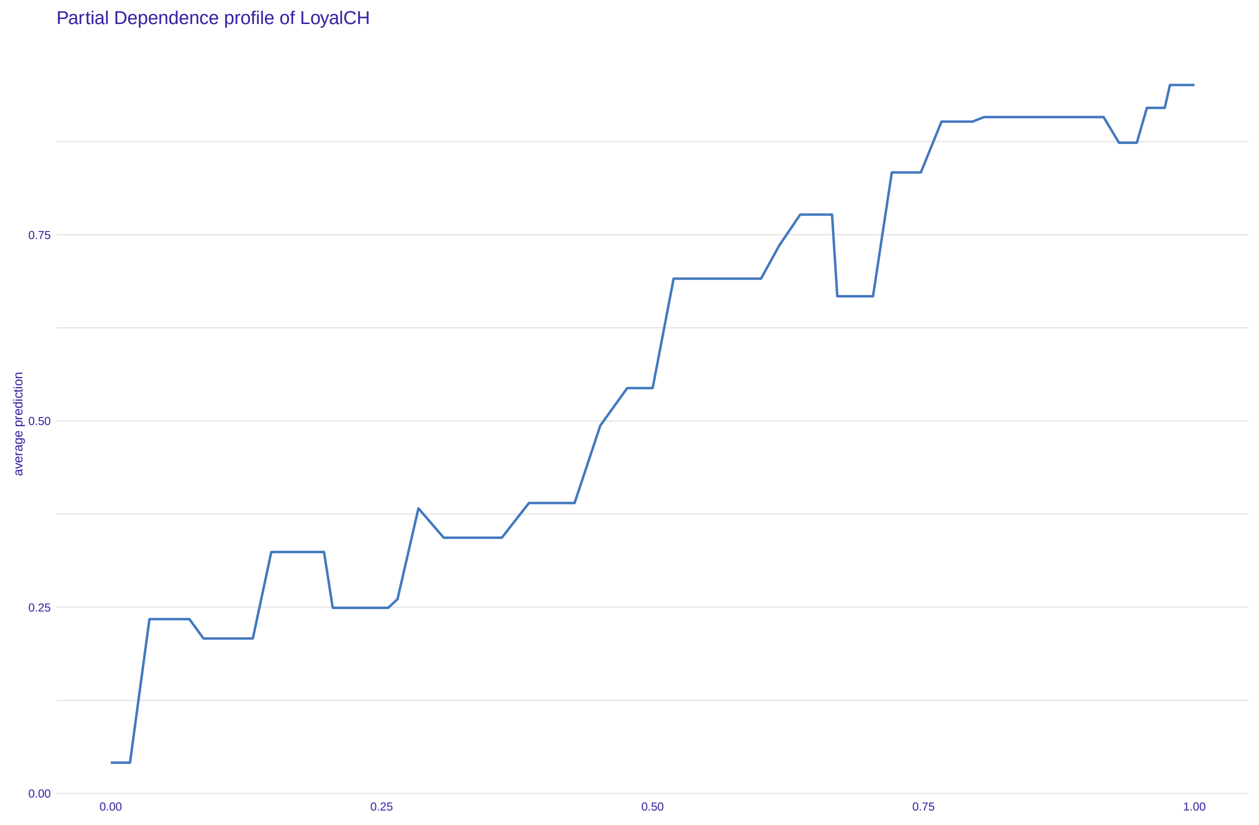
```
## Welcome to DALEX (version: 2.4.3).
## Find examples and detailed introduction at: http://ema.drwhy.ai/
## Additional features will be available after installation of: ggpubr.
## Use 'install_dependencies()' to get all suggested dependencies
```

```
##
## Attaching package: 'DALEX'

## The following object is masked from 'package:vip':
##
##     titanic

## The following object is masked from 'package:dplyr':
##
##     explain
```

```
modelXGB2 <- modelXGB1_finalWF %>% fit(data = traindata)

exp <- DALEXtra::explain_tidymodels(modelXGB2,
data = traindata[,-1],
y = traindata$Purchase==1, prob = TRUE,
type = "pdp",verbose = FALSE)

pdpLoyalCH <- model_profile(exp,
variables = "LoyalCH")

pdpPriceDiff <- model_profile(exp,
variables = "PriceDiff")

pdpListPriceDiff <- model_profile(exp,
variables = "ListPriceDiff")
```

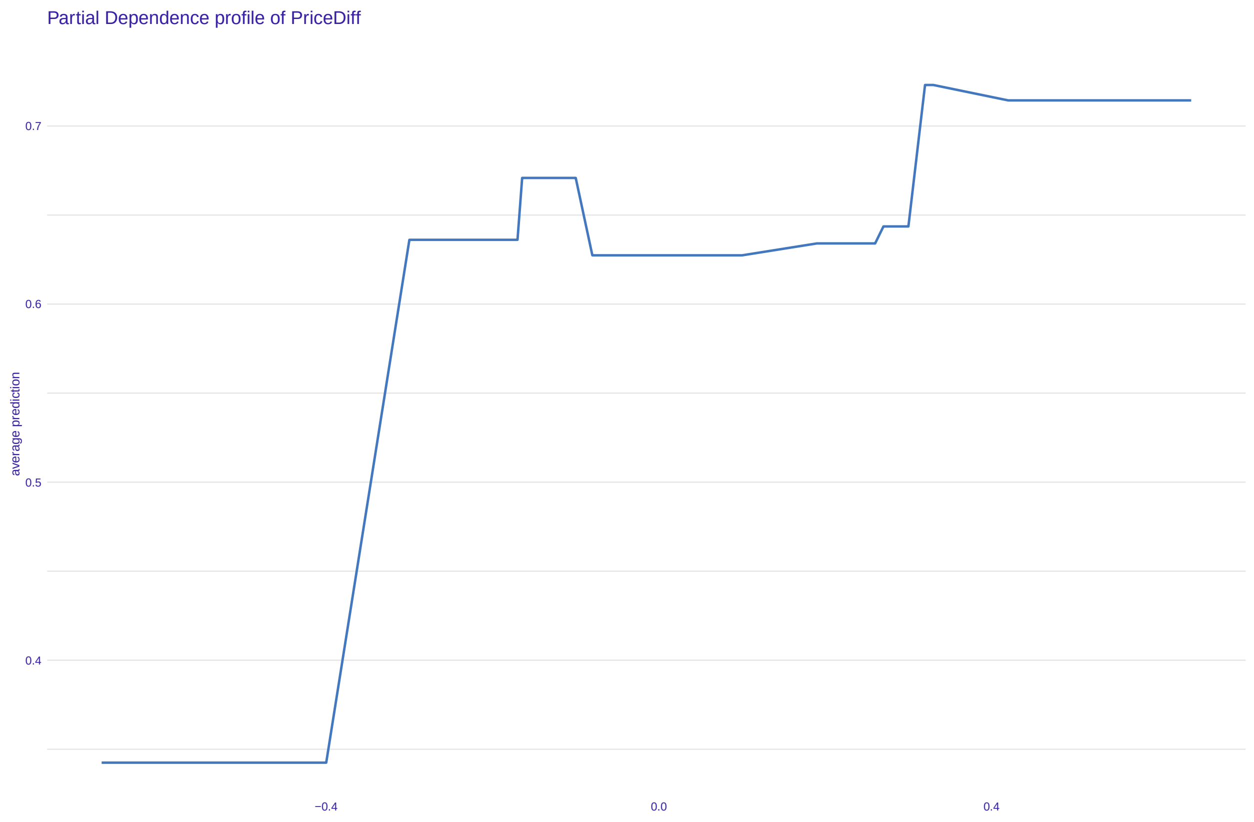
2.26 Implementing XGBoost Model - LoyalCH influence on Purchase:

```
plot(pdpLoyalCH, title = 'Partial Dependence profile of LoyalCH')
```



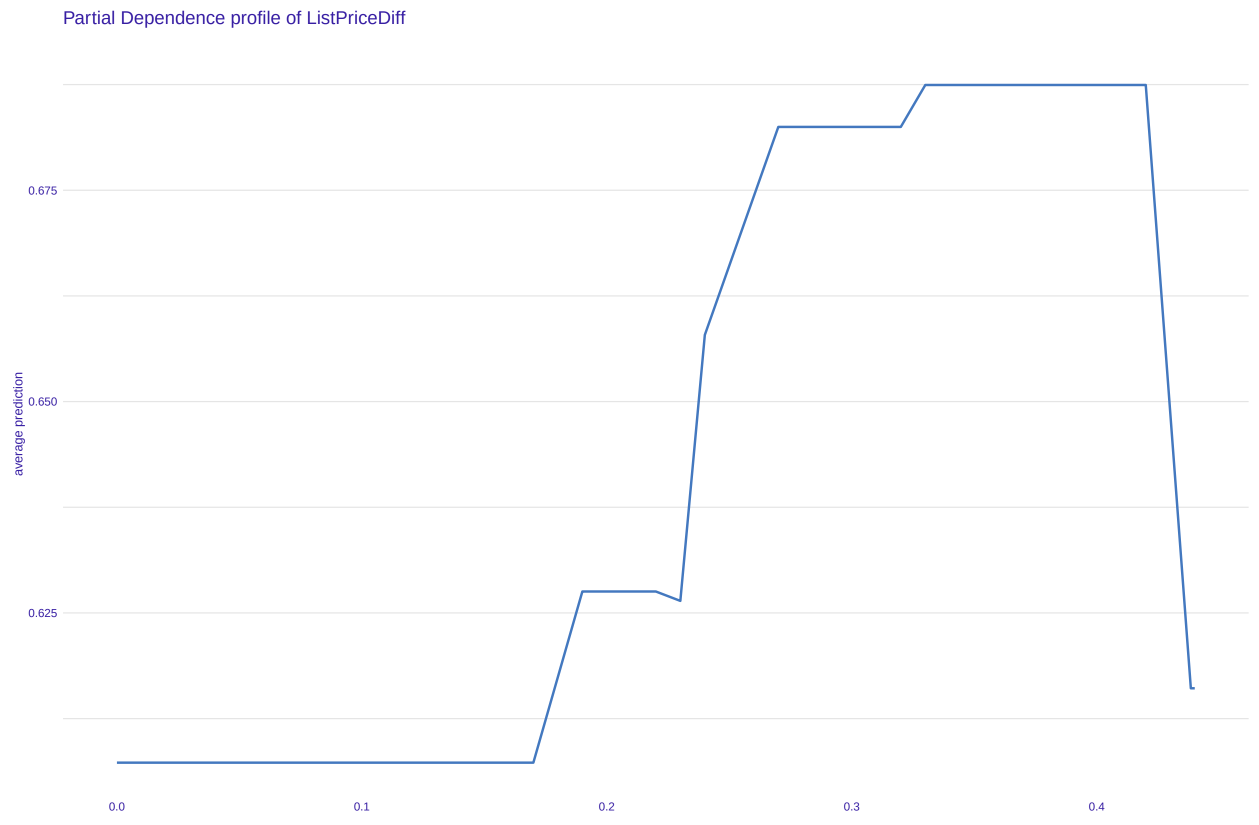
2.27 Implementing XGBoost Model - PriceDiff influence on Purchase:

```r
plot(pdpPriceDiff, title = 'Partial Dependence profile of PriceDiff')
```



2.28 Implementing XGBoost Model - ListPriceDiff influence on Purchase:

```
plot(pdpListPriceDiff, title = 'Partial Dependence profile of ListPriceDiff')
```



2.29 SUMMARY OF ANALYTICAL APPROACH:

Data: The data provided to us has 14 columns and 1070 records. The "Purchase" column is the target variable with binary values of 1 representing a customer purchasing CH brand and 0 representing MM brand.

EDA: Upon exploring the data frame, it is observed that there were no missing/Null values and hence we have not performed any imputation. All predictor variables (13 out 14 columns excluding the Purchase column) are "numeric" datatype. We observed that the predictor variables are all same data type and scaled and doesn't have different units associated with them in addition to that, we have obtained the descriptive statistics of the predictor variables and the variance among the recorded data in the columns are not high, hence we have decided not to pursue any scaling/standardization, as the data points are not largely diverse.

Class Imbalance: The distribution of target variable does not show any significant class imbalance as the majority class (Purchase==1) is 653 count and minority class (Purchase==0) is 417. Hence we have chosen not to down-sample the majority class.

Train-Test Split: We have split the data set into 80:20 train-test split for our model fit and prediction. This is essential for evaluating model performance, estimating different features and to inspect and avoid over-fitting and under-fitting of our model.

Predictive Model Selection and Implementation:

Logistic Regression: The first logistic regression that we use is with all predictor variables regressed against the target variable purchase. We do notice the effect size of some variables being more than others, however, we also notice that the coefficients of some variables appear to be NA (refer modelLR1). This suggest that there is redundancy among variable due to multicollinearity.

Multicollinearity: To address the issue of multicollinearity we have implemented penalized regression technique, a Lasso regression model, which gives us 5 variables that are influencing t he P urchase, t arget variable while penalizes other redundant variables to 0. The Lasso regression, also known as L1 regularization, helps to achieve model feature selection and to avoid model over-fitting.

Variable Selection: The 3 most important predictor variables obtained from lasso regression are 'LoyalCH', 'PriceDiff', 'ListPriceDiff'. We model another Logistic regression model with these 5 predictors (refer modelLR2) and we obtaine the AIC score to be 656.51 (less than AIC of modelLR1, 659.21). The model, modelLR2 obtains a ROCAUC score of 0.8697, implying decent performance. Hence we can conclude that the using all predictor variables (13) is not advisable rather use only the important once to obtain better results for logistic regression modeling.

Gradient Boosted Tree: We further implement a black-box model, XGBoost, to investigate if our performance can be further perfected. The Xgboot model has been hyper-parameter tuned with 3 parameter, viz. number of decision trees in the ensemble model, tree depth and learning rate. We obatin the best combination of hyper-parameter using grid search. This model provides a ROCAUC score of 0.9267, which is indicates better performance than the logistic regression model. Since Xgboost intuitively takes care of redundancy among predictor features and over-fitting there is less investment on pre-processing. The Xgboost model also pro-vides us with same number of important features which includes include 'LoyalCH','PriceDiff','ListPriceDiff'. 'LoyalCH' being the strongest predictor.

The partial dependency plots obtained for 'LoyalCH' predictor variable, inferred as a strong predictor from the XGBoost model, shows a rather fluctuating relation between target variable and the predictor. This implies that the behavior of Loyal CH brand customers have been rather sporadic than constant. Seasonality and external events might have influenced so. The fluctuation in Loyal CH customer behavior has either helped increase MM brand sales or decreases MM brand sales time to time.

In conclusion, the XGBoost model performs better and is recommended as the model that can be used to predict new customer purchase pattern.

3 RESULTS AND RECOMMENDATIONS:

3.1 BRAND MANAGER RECOMMENDATIONS:

The predictor variables that can be of marketing managers interest are 'LoyalCH','PriceDiff','ListPriceDiff'. These predictor variables highly influence the Purchase pattern of customers.To increase the sale of MM brand a few recommended steps would be to firstly, build a wider and loyal customer base for MM, understand the loyal customer of CH brand loyal and what drives them to be comfortable with purchasing CH brand again and again. New customers can be incentivized by free samples and one on one purchase, provide special discounts and offers to attract old and new customers. Implement a more robust SpecialMM offers that can appeal emotionally to the customer building a sense of importance of being a loyal MM customer. Secondly, focus on the price difference between the CH and MM brand, and reason with the customer base why the price difference exists if it is not possible to minimize the price difference.

3.2 SALES MAANAGER RECOMMENDATIONS:

We implemented two models both which can be used to provide the probability of a new customer purchasing MM brand. We recommend using XGboost model as it has a better performance (ROCAUC ~ 0.93). Since there an obvious expectation of more data to be sourced in course of time, more customer purchase pattern data can be fed into the model along side more new predictors can be incorporated in the model to make it further accurate in prediction, XGboost model is highly scalable and will fit the task perfectly. XGBoost incorporates regularization techniques to prevent over-fitting and improve generalization performance.

Link to Posit Cloud Code: https://posit.cloud/content/7032286