­­

Project 1

An in-depth analysis of Orange Juice sales

Dan Lemaire - Gavin Moss - Joni Jensen

Submission Date: June 1st, 2017

Mktg6620 – Machine Learning for Business Applications

Overview 3

Problem Definition 3

Executive Summary 3

Drivers 3

Predictive Model 4

Methods 5

Data Prep 5

Data Cleaning 5

Data Classes 5

Variable Selection 5

Feature Engineering 6

Model Selection 6

Cross-validated Training Scores 6

Confirmation with Bootstrapped Testing Scores 6

Variable importance 6

Logistic regression 6

The problem of collinearity 6

Boruta analysis 6

Recommendations 7

Loyalty 7

Data 7

Pricing 7

Conclusion 7

# Overview

The goal of this analysis is to understand how to increase the revenue from the orange juice category of store. The store sells two brands of orange juice, Minute Maid and Citrus Hill. Since Minute Maid has higher margins than Citrus Hill, this analysis will make recommendations regarding which drivers influence a consumer’s decision to purchase Minute Maid orange juice. This allows our company to leverage those drivers as opportunities to influence Minute Maid sales. It will additionally provide a predictive model for more precise forecasting. This forecasting will be of benefit now, but will be of tremendous benefit later when the company adjusts its marketing to increase Minute Maid sales (since an updated forecast will be required).

## Problem Definition

The stores sell two brands of orange juice, Citrus Hill and Minute Maid. As Minute Maid sales have a lager profit margin we want to find what factors cause customers to purchases Citrus Hill orange juice instead of Minute Maid. We are also interested in building a model that is able to predict with confidence what juice a customer will purchase.

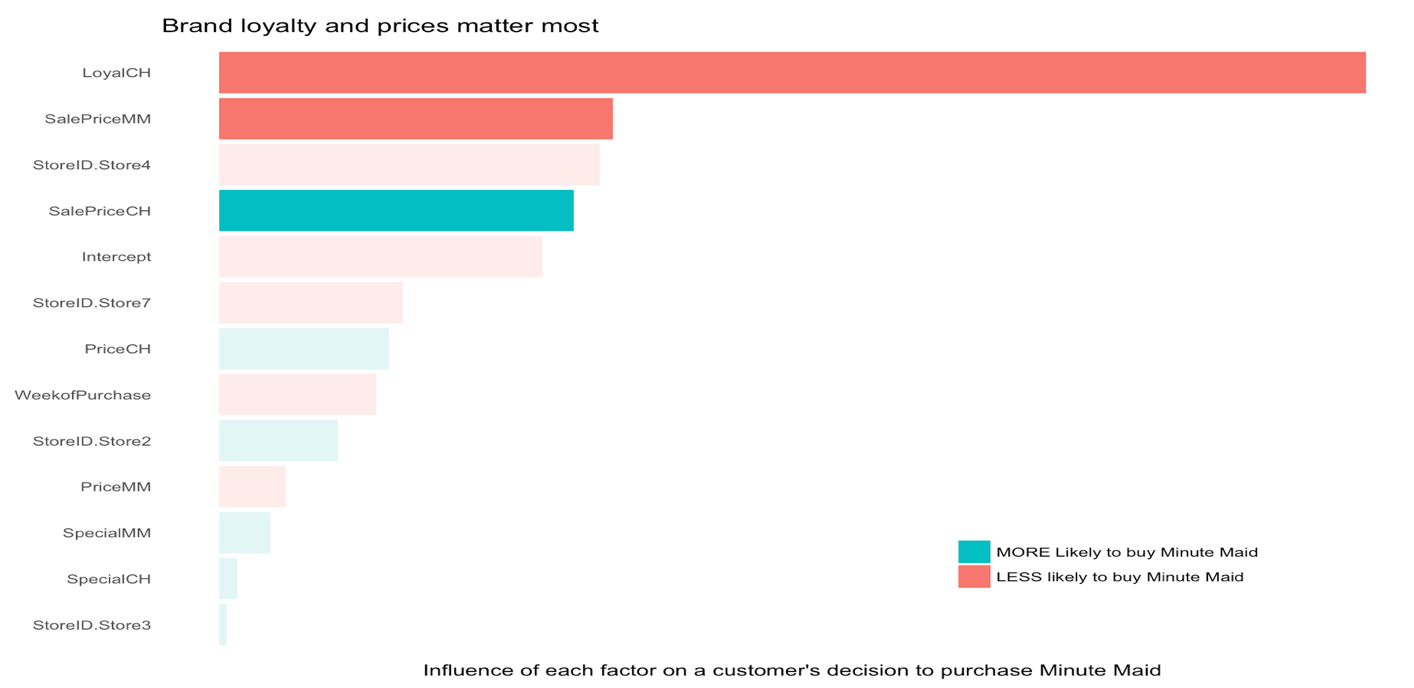
## Executive Summary

By using Logistic Regression, we have arrived at three recommendations given the goals and assumptions of this analysis:

* Erode customer loyalty to Citrus Hill to improve Minute Maid sales
* Test the effect of increased prices on Citrus Hill
* Collect more information to test for seasonality

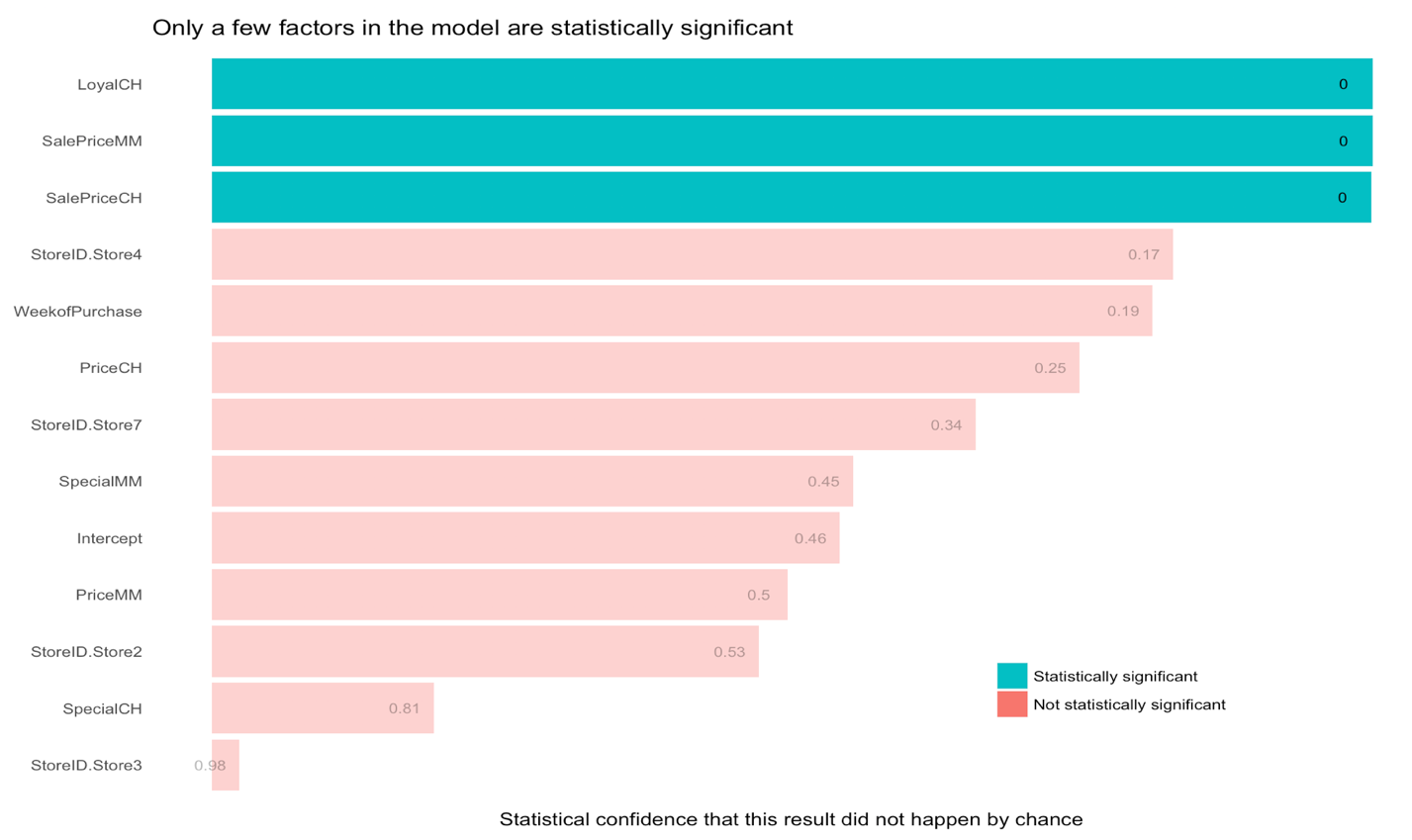
# Drivers

The following chart shows the magnitude of the effect different drivers have on a customer’s propensity to buy Minute Maid and Citrus Hill orange juice.



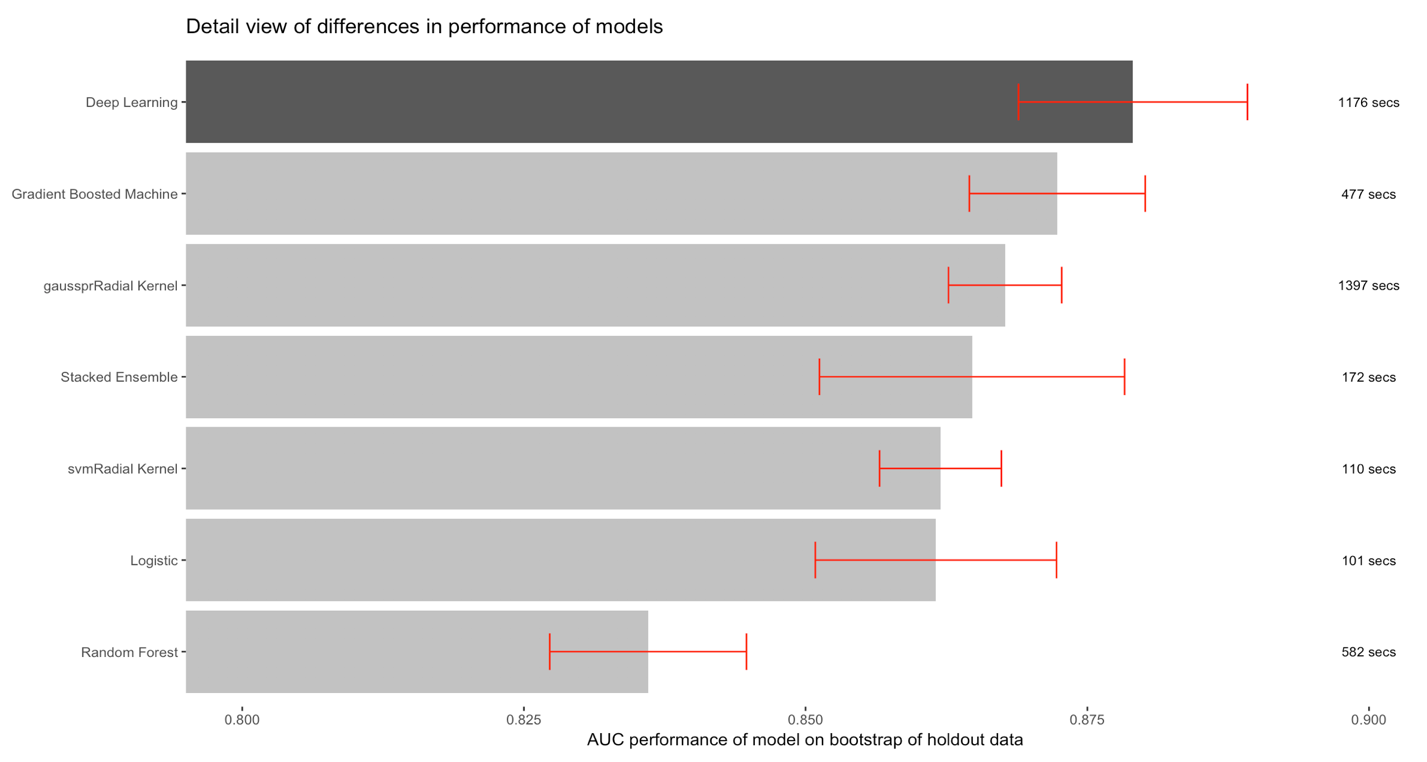
The most prominent drivers of the purchase decision are brand loyalty and the price of each orange juice. Increased brand loyalty to Citrus Hill decreases the likelihood that someone will purchase Minute Maid, as does a higher price on Minute Maid. A higher price on Citrus Hill increase the likelihood that a customer will purchase Minute Maid.

Other effects are not statistically significant at a confidence level of 95%, as can be seen in the chart below.



# Predictive Model

For the purpose of prediction, we evaluated 7 models with a wide range of tuning enhancements. The chart below shows how each of these models scores. All models performed well with reliable predictive power and different strengths and weaknesses. The basic logistic regression takes minimal computing resources and tends to generalize well. A deep learning model performs the best of all, but takes the most computing resources. If this model is likely to be used in the future and the resources to re-train the model are not a factor, then the deep learning model will continue to improve as it collects more information. However, if the intent is to run a simple model on a regular basis with minimal resources, then the logistic regression is cheap and “good enough”. The model choice relies on how the model will be implemented and if the run time is a concern.



# Methods

## Data Prep

### Data Cleaning

This dataset was already very clean with no missing values and no outliers. We found 24 duplicated rows (out of 1070 observations) and found that this did effect the training results of the models. These duplicates were removed.

### Data Classes

With some variables (such as the Week of Purchase), we could treat the variable as a factor or a numeric variable. First, we converted the week numbers to 1 through 52 for clarity. Next, we tested the models with these variables as factors and as continuous. A factor will allow the model to find non-linear relationships (such as certain months being different than other months) but is prone to overfitting. In this data, the numeric variable was much less prone to overfitting and produces the best results, though data with more than one year of observations may yield different results. If more than one year were present, autocorrelation (times-series approach) would be preferred.

### Variable Selection

We noticed that several of the columns in this dataset were simply linear combinations of other columns. We removed columns that were linear combinations of other columns and tested the performance of the model to confirm. We used the Caret function findLinearCombos(). We used variance inflation factor analysis to test for strong multicollinearity and found none (with the exception of an engineered feature that will be discussed in the next section).

### Feature Engineering

For the purpose of understanding which drivers influence a customer to purchase Minute Maid brand, we did not add any additional features to keep the message strong and simple. However, we found that by engineering a “MonthofPurchase” feature to supplement the “WeekofPurchase” variable, we were able to improve the predictive power of the model over keeping only one of the features. There must be a signal in Month that is hidden in the noise of Week that improving the model.

## Model Selection

### Cross-validated Training Scores

To score each model, we used the area under the ROC curve as our metric. Each model was trained with 10-fold cross validation to arrive at a confidence interval of expected performance against new data. The deep learning model performed the best, and its increased performance over the parsimonious logistic regression was statistically significant. We recognized that there may be additional user requirements (such as model training resources, frequency of re-training improvements, and parsimony) that will guide the choice of predictive model. The deep learning model performed the best and would improve the most with more data, but could not be interpreted for inference and consumed the most resources to train. The support vector machine with the Gaussian Process Radial kernel performed as well as the Deep Learning model, but with a narrower confidence interval so could be interpreted as more reliable (it took nearly as long to train). The logistic regression performed noticeably worse but the difference in performance may not be noteworthy and it trained in the shortest runtime.

### Confirmation with Bootstrapped Testing Scores

To report the most accurate expected performance of each model against new data, we looked at the performance of each model on a holdout dataset that we called “test”. We bootstrapped the test data to arrive at a 95% confidence interval and all figures were within the range of the expected performance scores from the training set, confirming that our choices and analysis did not have surprising results against new data.

## Variable importance

### Logistic regression

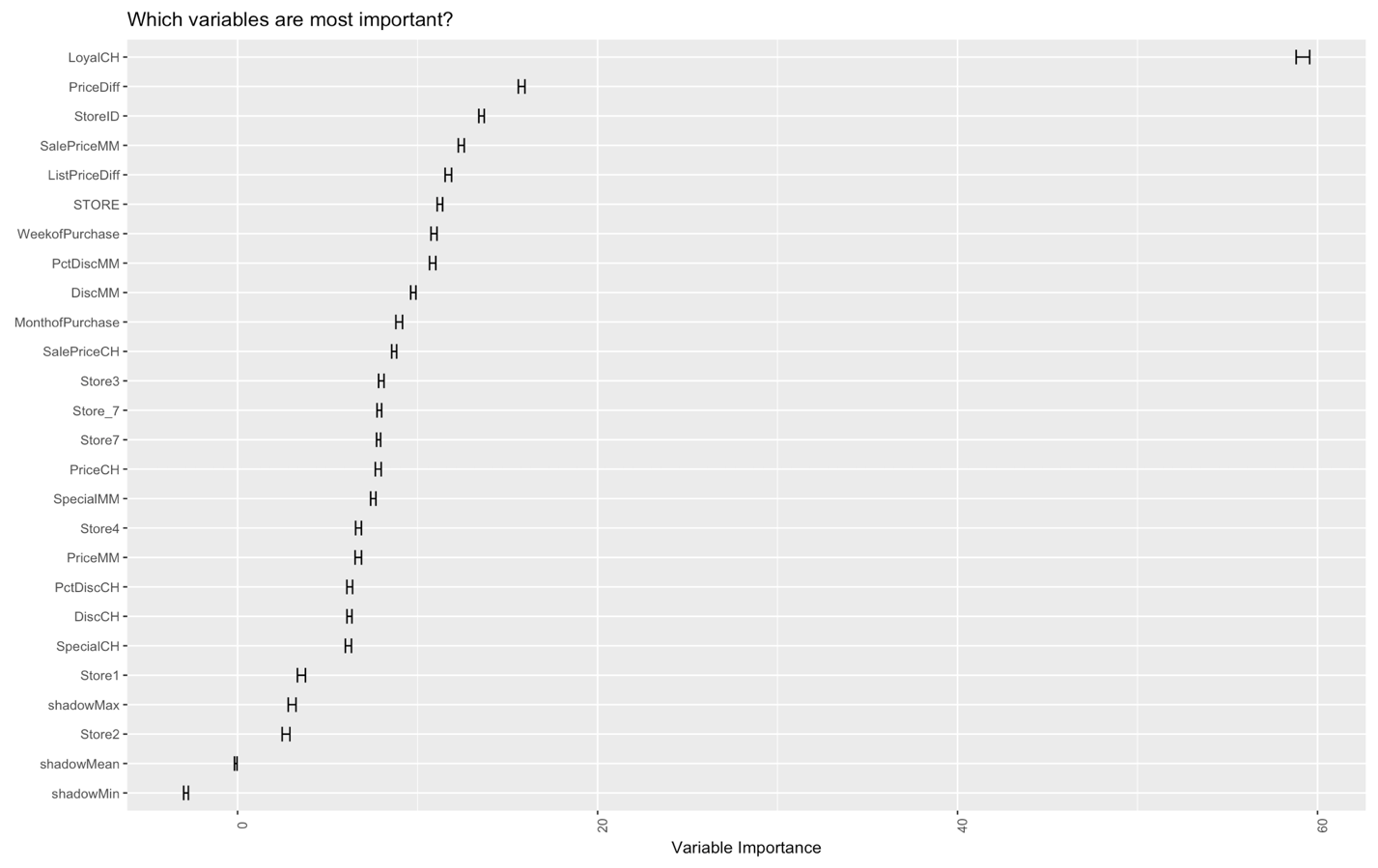
To understand what drives a consumer’s decision to purchase each brand or orange juice, we considered the standardized coefficients of the model. Since the ranges of each variable were different, it was necessary to standardize the variable importances to be able to compare apples to apples (or oranges to oranges in this case). We could be statistically confident at 95% in only three of the variables: customer loyalty to Citrus Hill, and the prices of each brand of orange juice at checkout.

### The problem of collinearity

Unfortunately, some collinearity does exist between the features that were kept, and linear combinations exist among the features that were removed. We wanted to ask more specific questions like: “How does the difference in prices matter compared to the prices themselves?” To do this, we applied a Boruta analysis to understand each specific driver.

### Boruta analysis

Boruta analysis assesses variable importance by adding randomized features called “shadows” and applying repeated random forests to them. Over several iterations, one can measure how much each specific variable impacts the total performance of a model apart from its correlation with other variables. The chart below should be used only as a reference to supplement understanding of the variable importances noted previously. A 95% confidence interval for each variable importance is included.



# Recommendations

## Loyalty

## Data

## Pricing

# Conclusion

# Appendix

## Complete Code

For complete code, see github repository at: <https://github.com/danlemaire/orange_juice.git>

## Minimum Reproducible Code

#Load necessary packages

packages <- list("magrittr", "tidyverse", "caret", "broom", "h2o", "scales", "corrplot", "GGally", "stringr", "Boruta", "ROCR", "kernlab")

lapply(packages, require, character.only = T)

#Set seed now before training split, and again before training each model to compare with others

set.seed(444)

#Load data, remove variables that are linear or highly correlated per vif and cor

df <- read\_csv("oj.csv") %>%

#map\_at(c("Purchase", "Store7", "STORE", "StoreID"), as.factor) %>%

as\_tibble %>%

mutate(WeekofPurchase = WeekofPurchase - min(WeekofPurchase) + 1,

MonthofPurchase = ceiling(WeekofPurchase/(52/12)),

Purchase = factor(Purchase, levels = c("MM", "CH"), labels = c("MinuteMaid", "CitrusHill")),

StoreID = factor(StoreID, labels = c("Store1", "Store2", "Store3", "Store4", "Store7"))

) %>%

select(-STORE, -Store7, -PctDiscMM, -PctDiscCH, -PriceDiff, -ListPriceDiff, -DiscMM, -DiscCH)

#Build train and test set

df <- df[!duplicated(df),] #Found some duplicates in rows

train <- df %>% sample\_frac(.8)

test <- df %>% setdiff(train)

#Remove variables athat are linear combinations of other variables

#train %>% findLinearCombos

#Use Boruta to find variable importances independent of collinearity

df\_boruta <- read\_csv("oj.csv") %>%

as\_tibble %>%

mutate(WeekofPurchase = WeekofPurchase - min(WeekofPurchase) + 1,

MonthofPurchase = ceiling(WeekofPurchase/(52/12)),

Purchase = factor(Purchase, levels = c("MM", "CH"), labels = c("MinuteMaid", "CitrusHill")),

#StoreID = factor(StoreID, labels = c("Store1", "Store2", "Store3", "Store4", "Store7")),

Store\_7 = Store7,

Store1 = ifelse(StoreID == 1, 1, 0),

Store2 = ifelse(StoreID == 2, 1, 0),

Store3 = ifelse(StoreID == 3, 1, 0),

Store4 = ifelse(StoreID == 4, 1, 0),

Store7 = ifelse(StoreID == 7, 1, 0)

)

boruta.train <- Boruta(Purchase ~ ., data = df\_boruta, doTrace = 2, pValue = .05)

#plot(boruta.train)

boruta.train$ImpHistory %>%

as\_tibble %>%

gather(key, value) %>%

ggplot(aes(x = reorder(key, value), y = value, group = key)) +

stat\_summary(fun.data = mean\_cl\_normal, geom = "errorbar", width = .5) +

#geom\_boxplot() +

labs(title = "Which variables are most important?",

x = element\_blank(),

y = "Variable Importance") +

coord\_flip() +

theme(axis.text.x = element\_text(angle = 90),

legend.position = "none")

#Initiate h2o cluster

h2o.init(nthreads = -1)

#Split data

h2o\_train <- as.h2o(train)

h2o\_test <- as.h2o(test)

train\_x <- setdiff(colnames(h2o\_train), "Purchase")

train\_y <- "Purchase"

#Build default GLM model

default\_glm <- h2o.glm(y = train\_y,

x = train\_x,

training\_frame = h2o\_train,

family = "binomial",

nfolds = 10,

model\_id = "default\_glm",

fold\_assignment = "Modulo",

keep\_cross\_validation\_predictions = T,

lambda = 0,

compute\_p\_values = T,

seed = 444

)

#Bootstrap performance of default glm model on test set

default\_glm\_performance <- c()

for (i in 1:100) {

default\_glm\_performance[i] <- h2o\_test %>%

as\_data\_frame %>%

sample\_frac(1, replace = T) %>%

as.h2o %>%

h2o.performance(default\_glm, .) %>%

h2o.auc

}

#Test alphas for tuned glm model

tuned\_glm <- h2o.grid("glm",

grid\_id = "tuned\_glm",

hyper\_params = list(alpha = c(seq(0,1,.1), .01)),

x = train\_x,

y = train\_y,

training\_frame = h2o\_train,

family = "binomial",

lambda\_search = TRUE,

max\_iterations = 100,

stopping\_metric = "AUC",

stopping\_tolerance = 0.00001,

stopping\_rounds = 4,

seed = 444

)

#Impact of alpha on tuned glm model: best alpha is 0 (pure ridge, no lasso)

tuned\_glm@summary\_table %>%

as\_data\_frame %>%

mutate(alpha = (str\_replace(.$alpha, ".", "") %>% str\_replace(".$", "") %>% as.numeric)) %>%

arrange(alpha) %>%

ggplot(aes(alpha, logloss)) +

geom\_point() +

labs(title = "Best alpha is 0",

subtitle = "(Even though performance generally improves with higher alpha)",

y = "Performance of model (lower is better)") +

theme(axis.text.y = element\_blank(),

axis.ticks.y = element\_blank())

best\_tuned\_glm <- h2o.getModel("tuned\_glm\_model\_0")

#Compare model performance on train and test sets for tuned and default model

default\_glm %>% h2o.performance(h2o\_test) %>% h2o.confusionMatrix

best\_tuned\_glm %>% h2o.performance(h2o\_test) %>% h2o.confusionMatrix

default\_glm %>% h2o.confusionMatrix

best\_tuned\_glm %>% h2o.confusionMatrix

#Bootstrap performance of tuned glm model on test data

tuned\_glm\_performance <- c()

for (i in 1:100) {

tuned\_glm\_performance[i] <- h2o\_test %>%

as\_data\_frame %>%

sample\_frac(1, replace = T) %>%

as.h2o %>%

h2o.performance(best\_tuned\_glm, .) %>%

h2o.auc

}

#Build stacked ensemble of models

rf <- h2o.randomForest(train\_x, train\_y, h2o\_train, model\_id = "rf\_default", seed = 444,

nfolds = 10, fold\_assignment = "Modulo", keep\_cross\_validation\_predictions = T)

gbm <- h2o.gbm(train\_x, train\_y, h2o\_train, model\_id = "gbm\_default", seed = 444,

nfolds = 10, fold\_assignment = "Modulo", keep\_cross\_validation\_predictions = T)

dl <- h2o.deeplearning(train\_x, train\_y, h2o\_train, model\_id = "dl\_default", seed = 444,

nfolds = 10, fold\_assignment = "Modulo", keep\_cross\_validation\_predictions = T)

stacked <- h2o.stackedEnsemble(train\_x, train\_y, h2o\_train,

base\_models = list("default\_glm", "gbm\_default", "dl\_default", "rf\_default"),

selection\_strategy = "choose\_all")

h2o.performance(stacked, h2o\_test) %>% h2o.auc

h2o.performance(default\_glm, h2o\_test) %>% h2o.auc

h2o.performance(dl, h2o\_test) %>% h2o.auc

h2o.performance(rf, h2o\_test) %>% h2o.auc

h2o.performance(gbm, h2o\_test) %>% h2o.auc

stacked\_model\_results <- c()

for (i in 1:30) {

boot <- h2o\_test %>% as\_tibble %>% sample\_frac(1, replace = T)

stacked\_model\_results[i] <- ((boot %>% as.h2o %>% h2o.predict(stacked, .))$MinuteMaid %>% as\_tibble %>%

prediction(., labels = boot$Purchase) %>% performance("auc"))@y.values[[1]][1]

}

default\_glm\_results <- c()

for (i in 1:30) {

boot <- h2o\_test %>% as\_tibble %>% sample\_frac(1, replace = T)

default\_glm\_results[i] <- ((boot %>% as.h2o %>% h2o.predict(default\_glm, .))$MinuteMaid %>% as\_tibble %>%

prediction(., labels = boot$Purchase) %>% performance("auc"))@y.values[[1]][1]

}

dl\_results <- c()

for (i in 1:30) {

boot <- h2o\_test %>% as\_tibble %>% sample\_frac(1, replace = T)

dl\_results[i] <- ((boot %>% as.h2o %>% h2o.predict(dl, .))$MinuteMaid %>% as\_tibble %>%

prediction(., labels = boot$Purchase) %>% performance("auc"))@y.values[[1]][1]

}

rf\_results <- c()

for (i in 1:30) {

boot <- h2o\_test %>% as\_tibble %>% sample\_frac(1, replace = T)

rf\_results[i] <- ((boot %>% as.h2o %>% h2o.predict(rf, .))$MinuteMaid %>% as\_tibble %>%

prediction(., labels = boot$Purchase) %>% performance("auc"))@y.values[[1]][1]

}

gbm\_results <- c()

for (i in 1:30) {

boot <- h2o\_test %>% as\_tibble %>% sample\_frac(1, replace = T)

gbm\_results[i] <- ((boot %>% as.h2o %>% h2o.predict(gbm, .))$MinuteMaid %>% as\_tibble %>%

prediction(., labels = boot$Purchase) %>% performance("auc"))@y.values[[1]][1]

}

final\_test\_results <- data\_frame(model = "Logistic",

ymin = default\_glm\_results %>%

median - qt(0.975, df = length(default\_glm\_results) - 1) \*

sd(default\_glm\_results)/sqrt(length(default\_glm\_results)),

y = default\_glm\_results %>%

median,

ymax = default\_glm\_results %>%

median + qt(0.975, df = length(default\_glm\_results) - 1) \*

sd(default\_glm\_results)/sqrt(length(default\_glm\_results))

) %>%

bind\_rows(data\_frame(model = "Deep Learning",

ymin = dl\_results %>%

median - qt(0.975, df = length(dl\_results) - 1) \*

sd(dl\_results)/sqrt(length(dl\_results)),

y = dl\_results %>%

median,

ymax = dl\_results %>%

median + qt(0.975, df = length(dl\_results) - 1) \*

sd(dl\_results)/sqrt(length(dl\_results))

)) %>%

bind\_rows(data\_frame(model = "Random Forest",

ymin = rf\_results %>%

median - qt(0.975, df = length(rf\_results) - 1) \*

sd(rf\_results)/sqrt(length(rf\_results)),

y = rf\_results %>%

median,

ymax = rf\_results %>%

median + qt(0.975, df = length(rf\_results) - 1) \*

sd(rf\_results)/sqrt(length(rf\_results))

)) %>%

bind\_rows(data\_frame(model = "Gradient Boosted Machine",

ymin = gbm\_results %>%

median - qt(0.975, df = length(gbm\_results) - 1) \*

sd(gbm\_results)/sqrt(length(gbm\_results)),

y = gbm\_results %>%

median,

ymax = gbm\_results %>%

median + qt(0.975, df = length(gbm\_results) - 1) \*

sd(gbm\_results)/sqrt(length(gbm\_results))

)) %>%

bind\_rows(data\_frame(model = "Stacked Ensemble",

ymin = stacked\_model\_results %>%

median - qt(0.975, df = length(stacked\_model\_results) - 1) \*

sd(stacked\_model\_results)/sqrt(length(stacked\_model\_results)),

y = stacked\_model\_results %>%

median,

ymax = stacked\_model\_results %>%

median + qt(0.975, df = length(stacked\_model\_results) - 1) \*

sd(stacked\_model\_results)/sqrt(length(stacked\_model\_results))

))

final\_test\_results

# Build SVM model with svmRadial kernel and see performance results

set.seed(444)

svm\_radial\_model <- train(Purchase ~ .,

data = train,

method = 'svmRadial',

trControl = trainControl(method = "cv",

number = 10,

summaryFunction = twoClassSummary,

classProbs = TRUE,

returnResamp = "final"

),

preProc = c("center","scale"),

metric = "ROC",

verbose = FALSE,

probability = TRUE,

tuneGrid = expand.grid(sigma = .016, C = 1.95)

)

svm\_radial\_model\_results <- c()

for (i in 1:100) {

idx <- sample\_frac(test, 1, replace = T)

svm\_radial\_model\_results[i] <- (predict(svm\_radial\_model, idx, type = "prob") %>%

select(MinuteMaid) %>%

prediction(idx$Purchase) %>%

performance("auc"))@y.values[[1]][1]

}

# Build SVM model with gaussprRadial kernel and see performance results

set.seed(444)

svm\_gaussprRadial\_model <- train(Purchase ~ .,

data = train,

method = "gaussprRadial",

trControl = trainControl(method = "cv",

number = 10,

summaryFunction = twoClassSummary,

classProbs = TRUE,

returnResamp = "final"

),

preProc = c("center","scale"),

metric = "ROC",

verbose = FALSE,

probability = TRUE,

tuneGrid = expand.grid(sigma = .0255556)

)

svm\_gaussprRadial\_model\_results <- c()

for (i in 1:100) {

idx <- sample\_frac(test, 1, replace = T)

svm\_gaussprRadial\_model\_results[i] <- (predict(svm\_gaussprRadial\_model, idx, type = "prob") %>%

select(MinuteMaid) %>%

prediction(idx$Purchase) %>%

performance("auc"))@y.values[[1]][1]

}

final\_test\_results %<>%

bind\_rows(data\_frame(model = "svmRadial Kernel",

ymin = svm\_radial\_model\_results %>%

median - qt(0.975, df = length(svm\_radial\_model\_results) - 1) \*

sd(svm\_radial\_model\_results)/sqrt(length(svm\_radial\_model\_results)),

y = svm\_radial\_model\_results %>%

median,

ymax = svm\_radial\_model\_results %>%

median + qt(0.975, df = length(svm\_radial\_model\_results) - 1) \*

sd(svm\_radial\_model\_results)/sqrt(length(svm\_radial\_model\_results))

)) %>%

bind\_rows(data\_frame(model = "gaussprRadial Kernel",

ymin = svm\_gaussprRadial\_model\_results %>%

median - qt(0.975, df = length(svm\_gaussprRadial\_model\_results) - 1) \*

sd(svm\_gaussprRadial\_model\_results)/sqrt(length(svm\_gaussprRadial\_model\_results)),

y = svm\_gaussprRadial\_model\_results %>%

median,

ymax = svm\_gaussprRadial\_model\_results %>%

median + qt(0.975, df = length(svm\_gaussprRadial\_model\_results) - 1) \*

sd(svm\_gaussprRadial\_model\_results)/sqrt(length(svm\_gaussprRadial\_model\_results))

))

list(svm\_gaussprRadial\_model, svm\_radial\_model) %>%

resamples %>%

summary

list(svm\_gaussprRadial\_model, svm\_radial\_model) %>%

resamples %>%

bwplot(metric = "ROC", ylab = c("gaussian kernel", "radial kernel"))

final\_test\_results <- bind\_cols(final\_test\_results, data\_frame(runtime = c(default\_glm@model$run\_time, dl@model$run\_time, rf@model$run\_time, gbm@model$run\_time, stacked@model$run\_time, svm\_radial\_model$times$everything[3] \* 60, svm\_gaussprRadial\_model$times$everything[3] \* 60)))

final\_test\_results %>%

mutate(isWinner = ifelse(y == max(y), 1, 0) %>% factor,

runtime = paste0(round(runtime), " secs")) %>%

ggplot(aes(reorder(model, y), ymin = ymin, ymax = ymax, y = y, label = runtime)) +

geom\_bar(aes(alpha = isWinner), stat = "identity") +

geom\_text(y = .90, size = 3) +

geom\_errorbar(color = "red", width = .4) +

coord\_flip(ylim = c(.8, .9)) +

labs(title = "Detail view of differences in performance of models",

y = "AUC performance of model on bootstrap of holdout data",

x = element\_blank()) +

theme(panel.background = element\_blank(),

legend.position = "none") +

scale\_alpha\_discrete(range = c(.4,1))

#Plot variable importances using standardized coefficients from glm, remove Month

(h2o.glm(y = train\_y,

x = (h2o\_train %>% colnames %>% setdiff("Purchase") %>% setdiff("MonthofPurchase")),

training\_frame = h2o\_train,

family = "binomial",

nfolds = 10,

model\_id = "default\_glm",

fold\_assignment = "Modulo",

keep\_cross\_validation\_predictions = T,

lambda = 0,

compute\_p\_values = T,

seed = 444

))@model$coefficients\_table %>%

as\_tibble %>%

mutate(sign = factor(ifelse(standardized\_coefficients >= 0, 1, 0)),

significant = (ifelse(p\_value <= .05, 1, 0) %>% factor)) %>%

filter(!is.na(p\_value)) %>%

ggplot(aes(x = reorder(names, abs(standardized\_coefficients)),

y = abs(standardized\_coefficients),

fill = sign,

alpha = significant)) +

geom\_bar(stat = "identity") +

coord\_flip() +

labs(title = "Brand loyalty and prices matter most",

y = "Influence of each factor on a customer's decision to purchase Minute Maid",

x = element\_blank()) +

theme(axis.text.x = element\_blank(),

axis.ticks.x = element\_blank(),

axis.ticks.y = element\_blank(),

legend.position = c(.75, 0.15),

legend.title = element\_blank(),

panel.background = element\_blank()

) +

scale\_fill\_discrete(breaks = c(1, 0),

labels = c("MORE Likely to buy Minute Maid", "LESS likely to buy Minute Maid")

) +

scale\_alpha\_discrete(guide = "none", range = c(.15, 1))

#Plot significance of each standardized coefficient using glm model

(h2o.glm(y = train\_y,

x = (h2o\_train %>% colnames %>% setdiff("Purchase") %>% setdiff("MonthofPurchase")),

training\_frame = h2o\_train,

family = "binomial",

nfolds = 10,

model\_id = "default\_glm",

fold\_assignment = "Modulo",

keep\_cross\_validation\_predictions = T,

lambda = 0,

compute\_p\_values = T,

seed = 444

))@model$coefficients\_table %>%

as\_tibble %>%

transmute(names = names,

p\_value = p\_value,

significant = (ifelse(p\_value <= .05, 1, 0) %>% factor),

removed = ifelse(is.na(p\_value), 1, 0)) %>%

filter(!is.na(significant)) %>%

ggplot(aes(reorder(names, -abs(p\_value)),

1 - p\_value,

fill = significant,

alpha = significant,

label = round(p\_value, 2))) +

geom\_bar(stat = "identity") +

coord\_flip() +

labs(title = "Only a few factors in the model are statistically significant",

y = "Statistical confidence that this result did not happen by chance",

x = element\_blank()) +

theme(axis.text.x = element\_blank(),

axis.ticks.x = element\_blank(),

axis.ticks.y = element\_blank(),

legend.position = c(.75, 0.15),

legend.title = element\_blank(),

panel.background = element\_blank()

) +

geom\_text(nudge\_y = -.025, size = 3) +

scale\_fill\_discrete(breaks = c(01,0), labels = c("Statistically significant", "Not statistically significant")) +

scale\_alpha\_discrete(guide = "none", range = c(.35, 1))

#Brand loyalty viz

train %>%

select(Purchase, StoreID, LoyalCH) %>%

group\_by(StoreID) %>%

ggplot(aes(StoreID, LoyalCH, fill = Purchase)) +

geom\_boxplot() +

labs(title = "Brand loyalty affects purchase differently in each store",

x = element\_blank(),

y = "Loyalty to Citrus Hill") +

scale\_y\_continuous(labels = percent) +

theme(panel.background = element\_blank())

#Price sensitivity viz

train %>%

select(MonthofPurchase, PriceCH, PriceMM, StoreID) %>%

gather(brand, price, -MonthofPurchase, -StoreID) %>%

group\_by(MonthofPurchase, brand, StoreID) %>%

summarise(price = mean(price),

purchases = n()) %>%

ggplot(aes(price, purchases, color = brand)) +

geom\_point() +

geom\_smooth(method = "lm", se = F) +

facet\_wrap(~StoreID) +

labs(title = "Higher price is correlated with more quantity purchased",

subtitle = "...but only in Store7. Why store7? Why is the correlation positive?",

x = "Price",

y = "Number of Monthly purchases",

color = "Brand") +

scale\_x\_continuous(labels = dollar) +

scale\_color\_discrete(labels = c("Citrus Hill", "Minute Maid")) +

theme(legend.position = c(.85, .25),

panel.background = element\_blank())

#Sales over time viz (week and month to compare)

train %>%

group\_by(WeekofPurchase, StoreID) %>%

summarise(purchases = n()) %>%

ggplot(aes(WeekofPurchase, purchases, color = StoreID)) +

geom\_smooth(method = "lm", se = F) +

geom\_line(alpha = .25) +

facet\_wrap(~StoreID) +

theme(panel.background = element\_blank(),

legend.position = "none") +

labs(title = "Number of sales throughout the year",

x = "Week of Purchase",

y = "Number of Purchases")

train %>%

group\_by(MonthofPurchase, StoreID) %>%

summarise(purchases = n()) %>%

ggplot(aes(MonthofPurchase, purchases, color = StoreID)) +

geom\_smooth(method = "lm", se = F) +

geom\_line(alpha = .25) +

facet\_wrap(~StoreID) +

theme(panel.background = element\_blank(),

legend.position = "none") +

labs(title = "Number of sales throughout the year",

x = "Month of Purchase",

y = "Number of Purchases")