Project 1

An in-depth analysis of orange juice sales

Dan Lemaire - Gavin Moss - Joni Jensen

Submission Date: June 13th, 2017

MKTG6620 – Machine Learning for Business Applications

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.

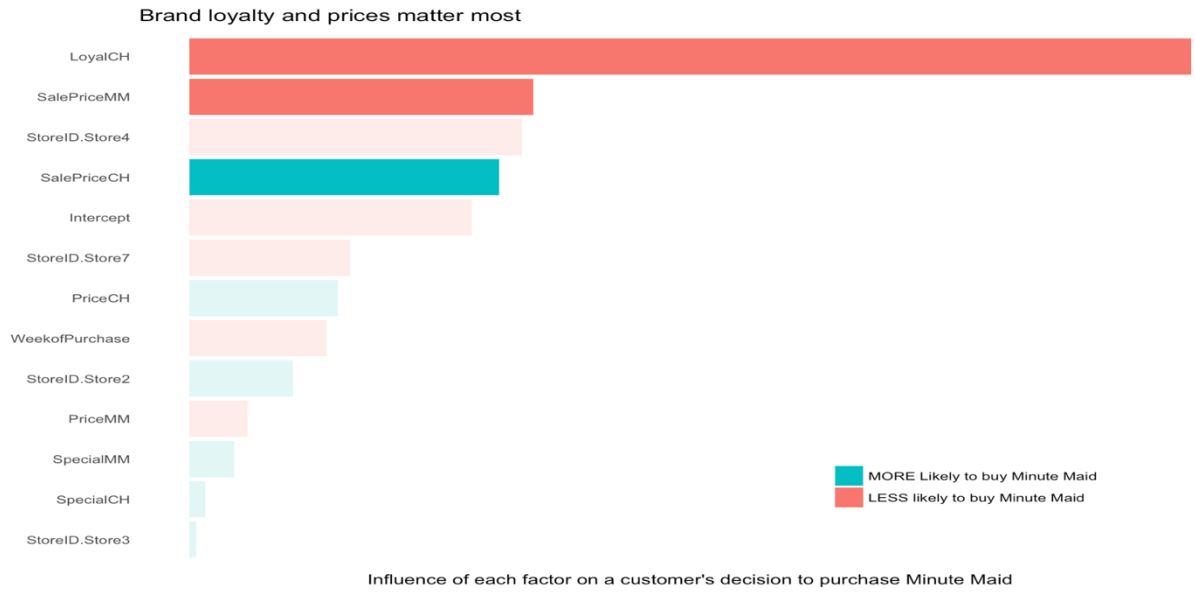
Executive Summary

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

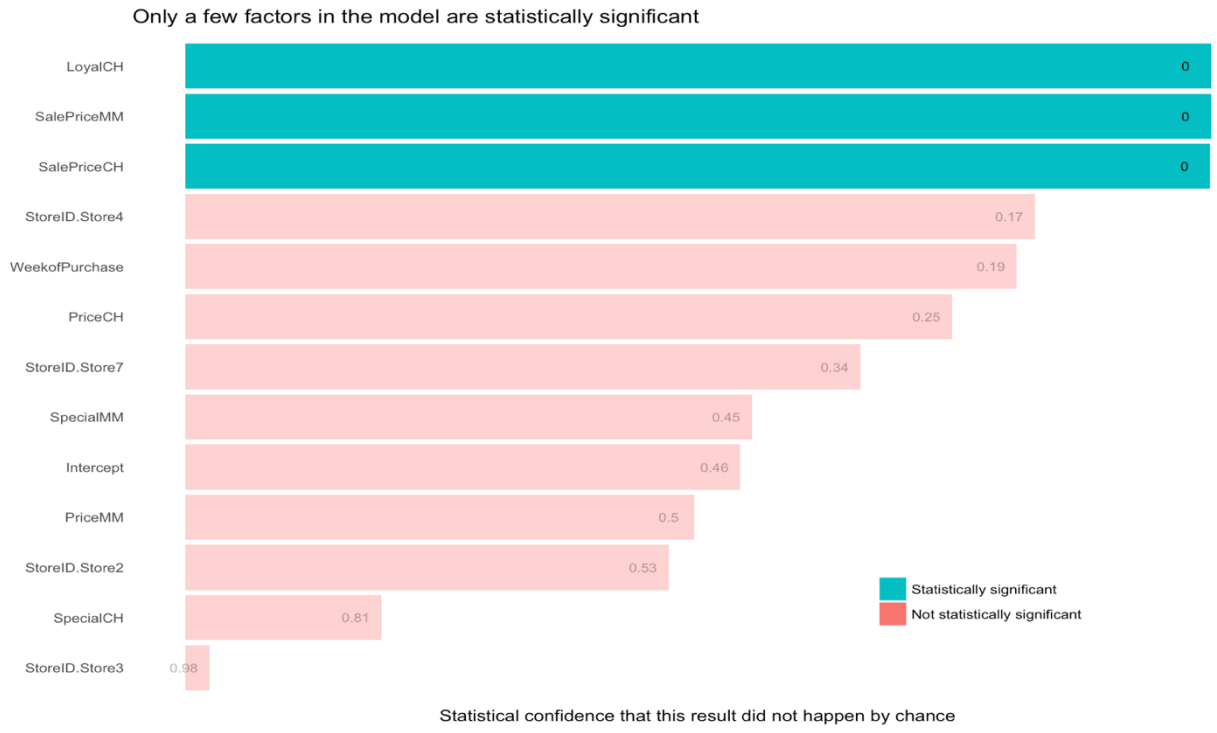
Drivers

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.



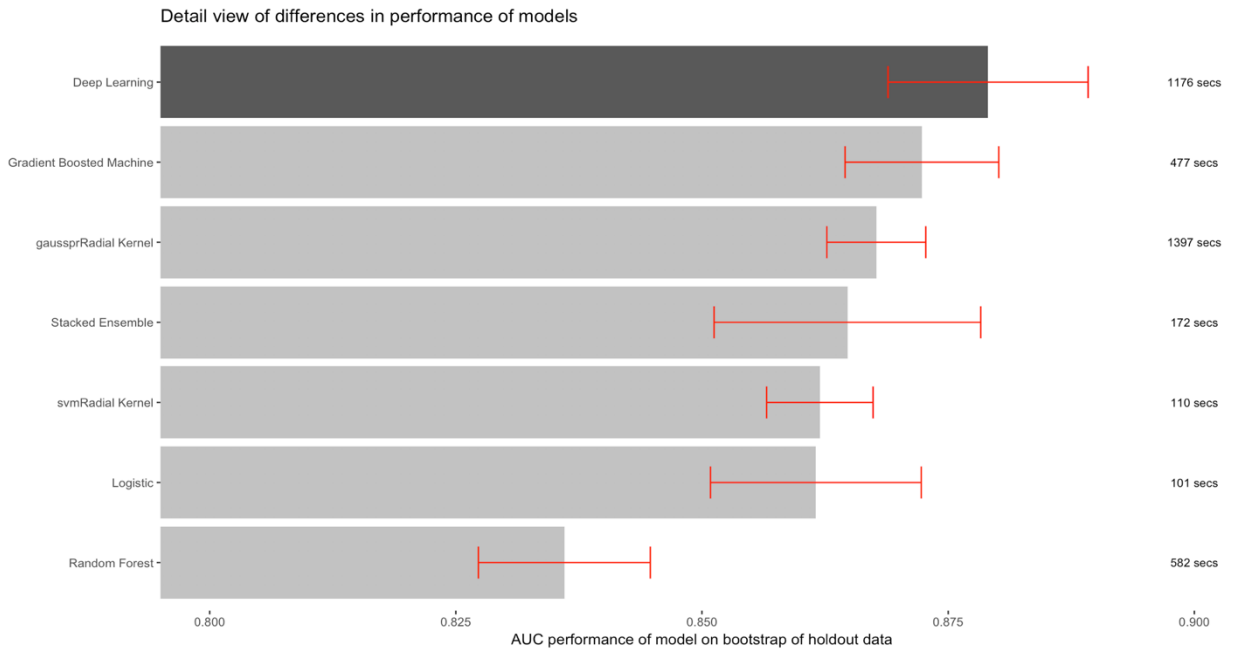
The most prominent drivers of the purchase decision are brand loyalty and the price of each orange juice. Increased brand loyalty to Citrus Hill decreases the likelihood that someone will purchase Minute Maid, as does a higher price on Minute Maid. A higher price on Citrus Hill increase the likelihood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

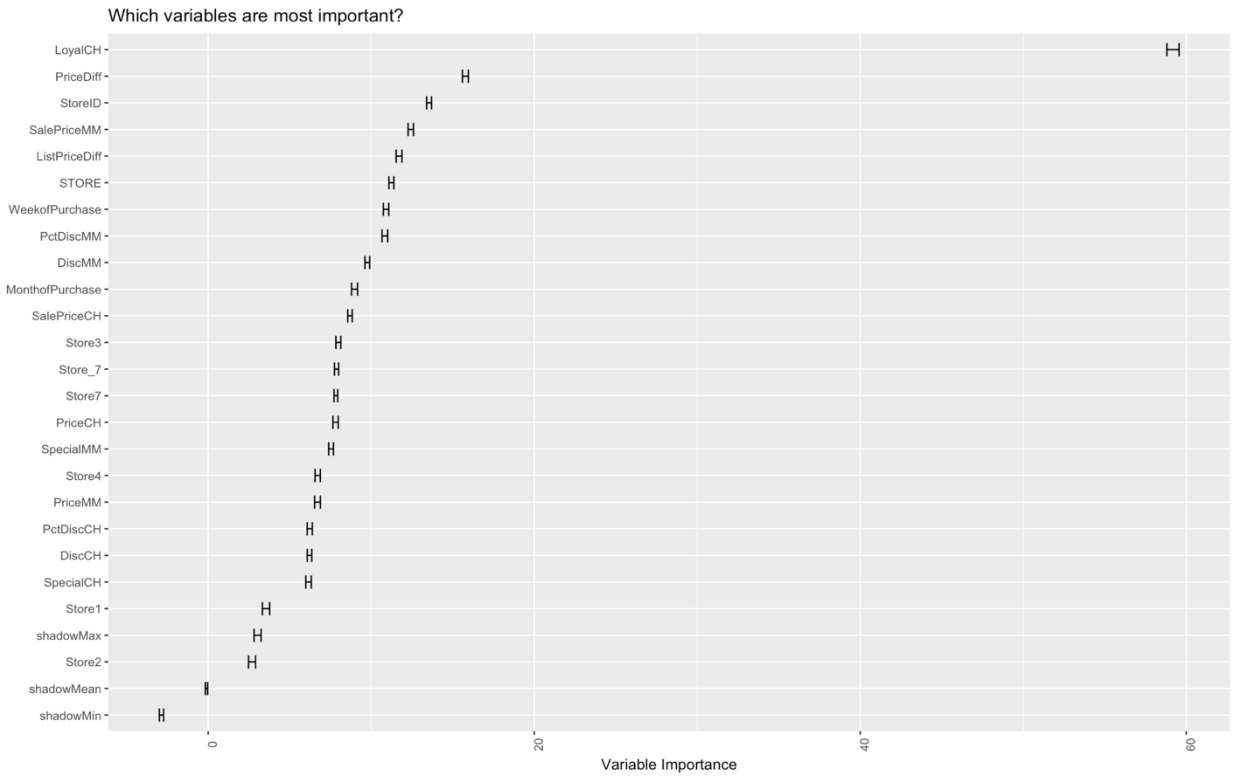
The problem of collinearity

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

As detailed above, loyalty to Citrus Hill (CH) was a primary inverse predictor of Minute Maid (MM) purchases. Orange juice is considered a utilitarian product; product attribute and purchase specific information are probably used in decision making. Given brand loyalty, consumers may be expending little cognitive effort on their OJ purchasing decision, relying instead on Heuristics:

- Habit: “I bought CH last time”
- Price: “CH is cheaper”
- Affect-referral: “I feel good about serving CH to my family”
- Surrogate indicators: “It is made by P&G, which is a reputable company”

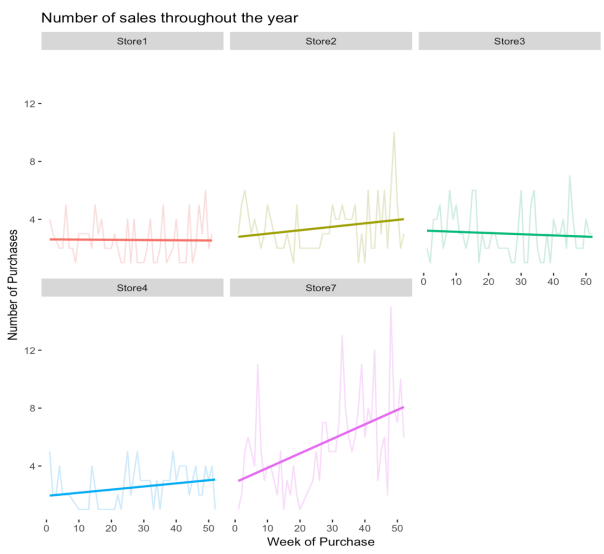
In order to improve sales of MM, the stores need to erode some of consumer’s brand loyalty to CH. This could be accomplished through a variety of means including:

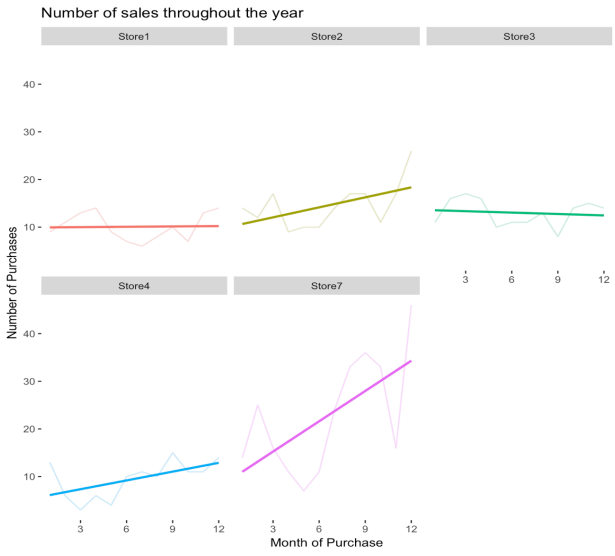
- In-store sampling with a coupon to purchase MM (demonstrate superior quality of MM). This would move the product into the awareness set and likely the consideration set as well.

- Customized promotions for CH buyers such as checkout or in-store smart phone coupons for MM. This could encourage them to try MM and move it from the consideration set to the specific alternative purchased. By only offering coupons/discounts to CH buyers, the stores are protecting their margins as existing MM customers will not be receiving the discounts on a product they already purchase.
- Consider placing a cooler with just MM near the point of purchase to influence stimulus-based consideration.

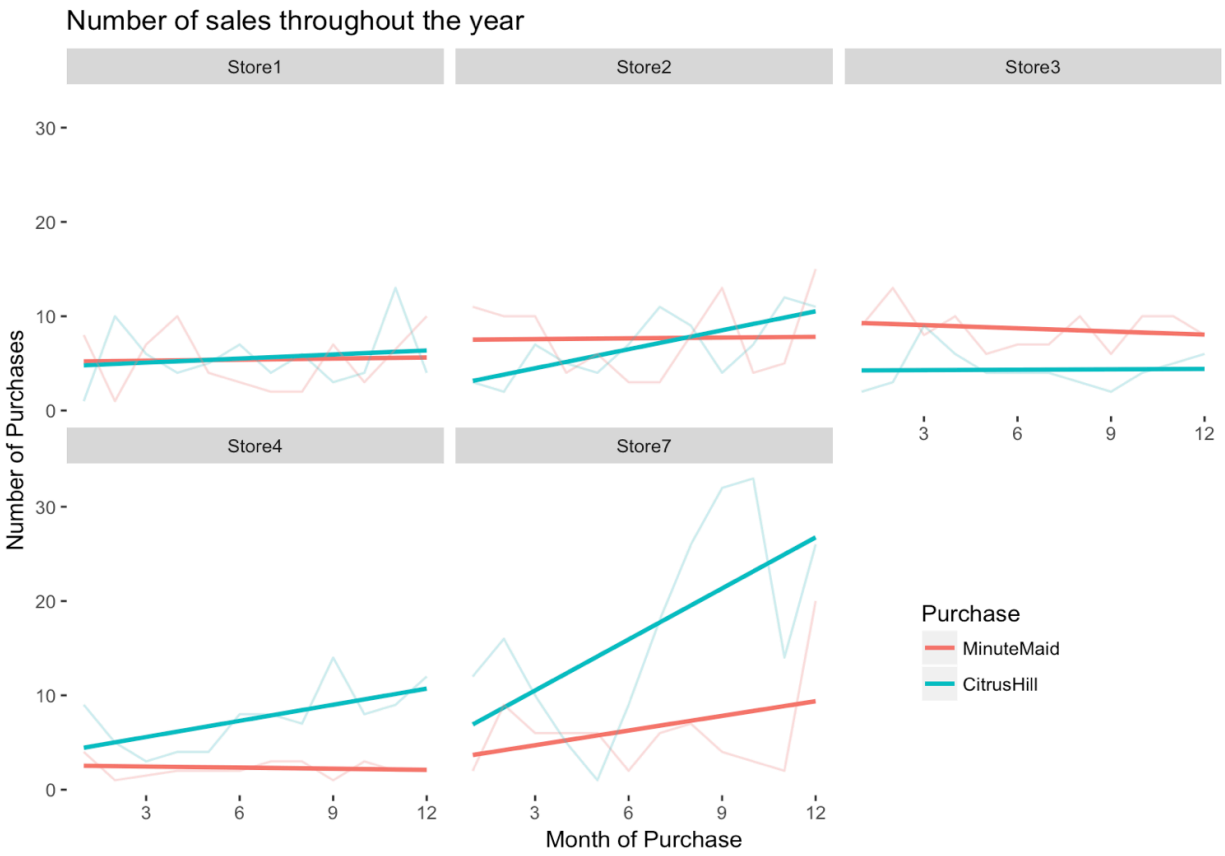
Data

The model showed some variation over the course of the year, which showed different purchasing pattern by store. We would recommend that additional data be gathered so that we could assess if there is any seasonality overall and on a store-by-store basis. We could also use this to see if there are seasonal periods at particular stores that lead changes in behavior that impact the CH/MM purchasing decision. Based on the outcome of the additional analysis, promotions could be structured to encourage the purchase of MM. See visualizations below for time-effect of sales.





Additionally, any possible trends differ depending on the store AND the brand of orange juice. See below:



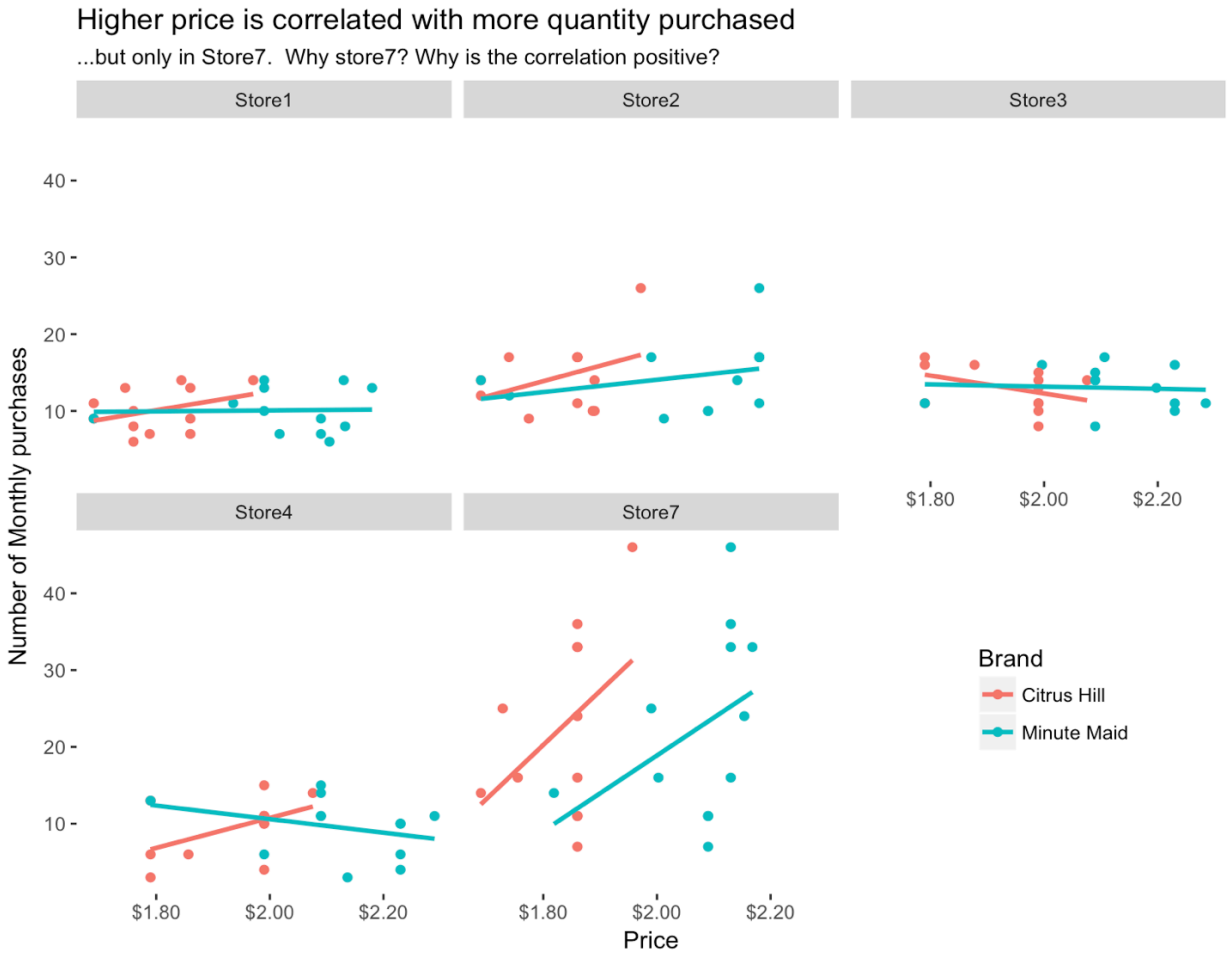
Pricing

We recommend additional pricing analysis. As CH is never more expensive than MM, is it possible to raise the price of CH and increase overall OJ revenue? Or could a CH price increase be used to push sales to higher-margin MM? We think additional testing of the price elasticity of both CH and MM could yield some insight.

Consider store specific pricing:

- Seasonality: tailor to each specific store's annual purchasing patterns
- Evaluate store-by-store price elasticity
- Create a Shiny App for each store

There seems to be no price elasticity except for Store7, where the effect is in the opposite direction of what would be expected (see plot below). There may be other explanations (such as an excellent existing pricing model).



Conclusion

We evaluated eight models. Each performed well with AUC scores between .83 and .88. Our best model was a deep learning model, though several performed nearly as well.

The model indicated that seasonality could be an issue, but more data and analysis is required to make this determination. CH loyalty and the sale price of both minute Maid and Citrus Hill were also important drivers.

Our three recommendations for increasing sales of Minute Maid included:

- Erode Citrus Hill loyalty
- Further evaluate the pricing strategy for both CH and MM
- Check for seasonality

Citrus Hill Postscript

P&G introduced Citrus Hill in the fall of 1982 and waged a heated battle with rival juice makers with its heavily promoted brand. In 1992, P&G announced the closure of its Citrus Hill orange juice division. According to the CEO, Citrus Hill was unable to gain on its two main rivals, Seagram Co. Ltd.'s Tropicana and Coca-Cola Co.'s Minute Maid. Citrus Hill was unable to become a market leader since it didn't have a competitive advantage.

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")
```