Project Two KNN & SVM

Bethany Poulin

12/3/2019

Introduction

This project is designed to evaluate production data from a beverage manufacturing company. Our assignment is to predict `PH`, a Key Performance Indicator (KPI), with a high degree of accuracy through predictive modeling. After thorough examination, we approached this task by splitting the provided data into training and test sets. We evaluated several models on this split and found that **what-ever-worked-best** method yielded the best results.

Each group member worked individually to create their own solution. We built our final submission by collaboratively evaluating and combining each others' approaches. Our introduction should further outline individual responsibilities. For example, **so-and-so** was responsible for **xyz task**.

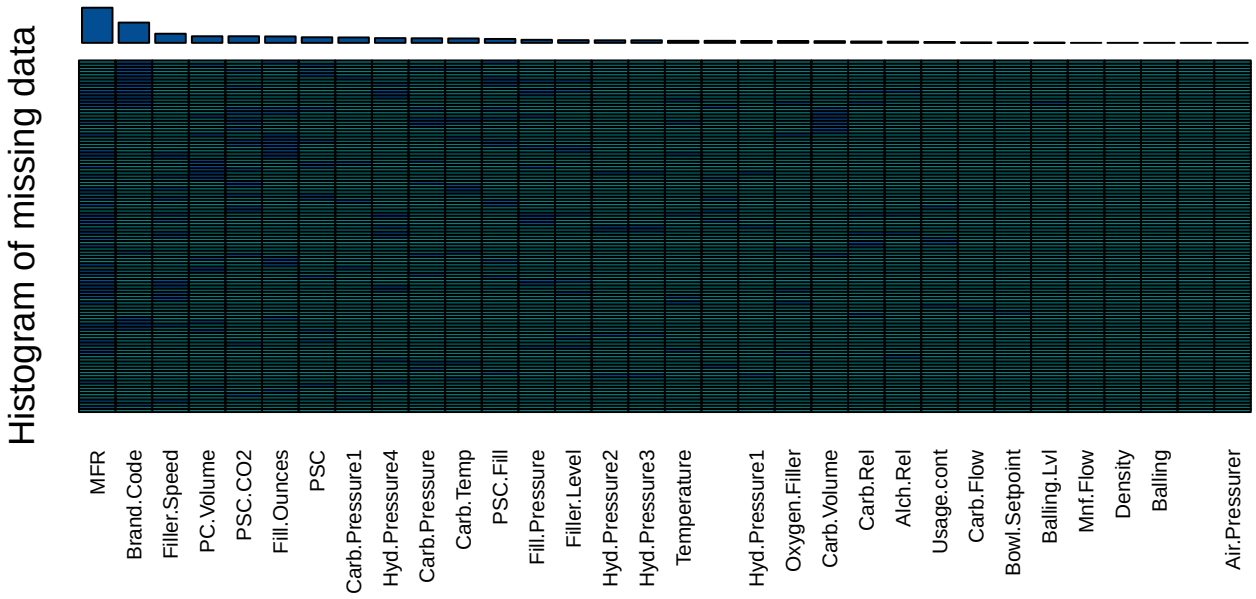
For replication and grading purposes, we made our code avaliable in the appendix section. This code, along with the provided data, score-set results, and individual contributions, can also be accessed through our group github repository:

LINKS
link placeholder
link placeholder
link placeholder
link placeholder
LINKS

Data Exploration

Our initial data had 2571 observations of 32 variables plus our outcome variable PH which we use as a key performance indicator in our manufacturing process as it can be and indicator of both the process health and the ultimate flavor appeal of the final product.

During the exploration, it was clear that most of the variables had some missing values, including Ph, our target variable. MFR and Brand Code have the most (at 212 and 120 respectively) whereas the vast majority had 50 or less missing values. No individual variable had more than 9% missing values and most were under 2% missing as you can see from the following charts.



Variables sorted by number of missings:

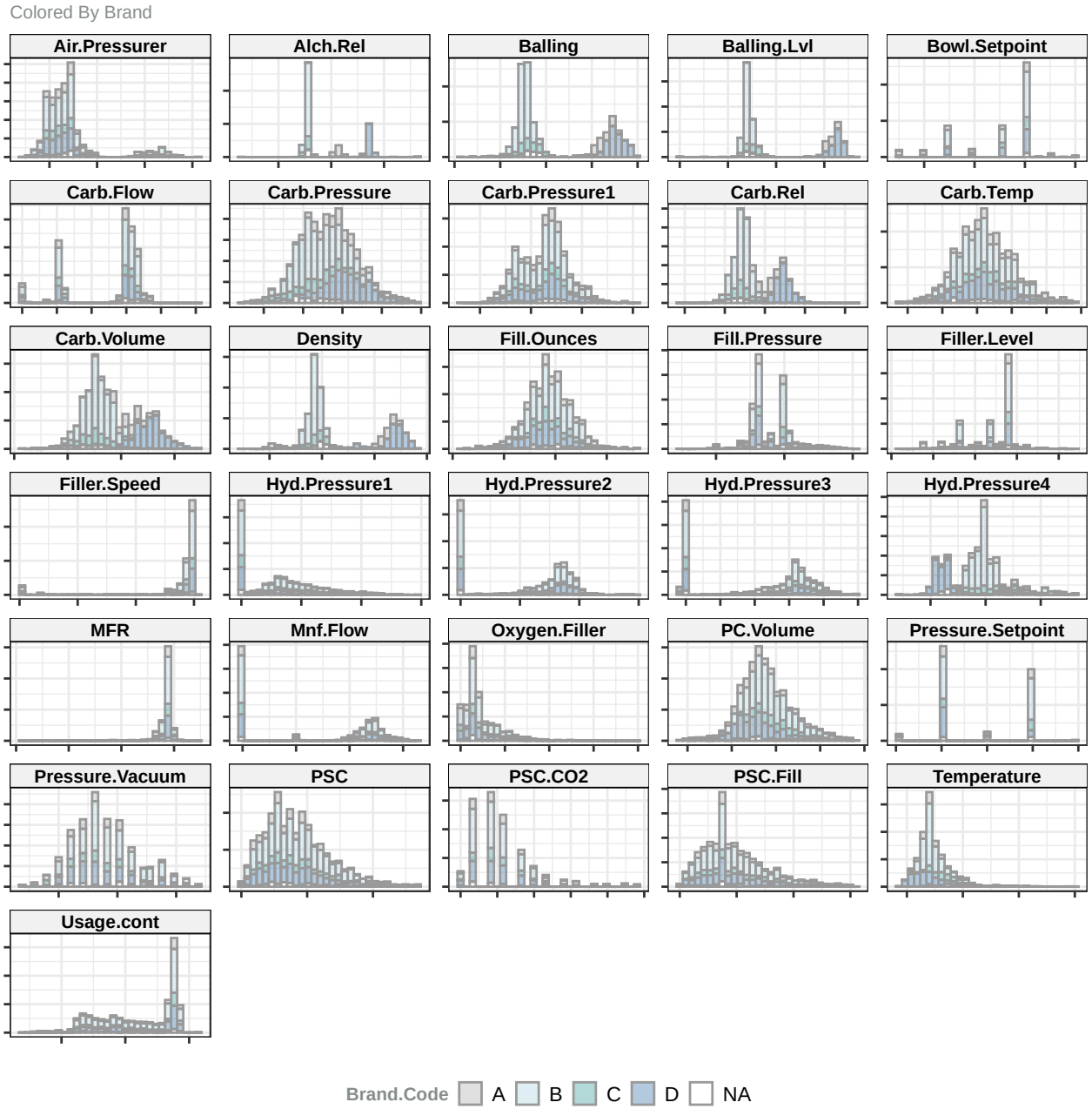
Variable	Count
MFR	208
Brand.Code	120
Filler.Speed	54
PC.Volume	39
PSC.CO2	39
Fill.Ounces	38
PSC	33
Carb.Pressure1	32
Hyd.Pressure4	28
Carb.Pressure	27
Carb.Temp	26
PSC.Fill	23
Fill.Pressure	18
Filler.Level	16
Hyd.Pressure2	15
Hyd.Pressure3	15
Temperature	12
Pressure.Setpoint	12
Hyd.Pressure1	11
Oxygen.Filler	11
Carb.Volume	10
Carb.Rel	8
Alch.Rel	7
Usage.cont	5
Carb.Flow	2
Bowl.Setpoint	2
Balling.Lvl	1
Mnf.Flow	0
Density	0
Balling	0
Pressure.Vacuum	0
Air.Pressurer	0

Likewise the test set, also had missing values leading us to consider methods of handling imputation which would provide the most consistent values for both sets.

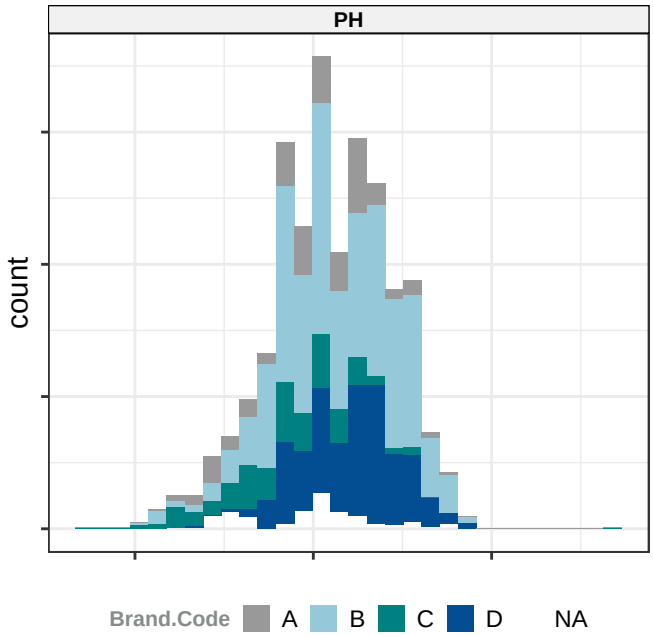
Predictor Variables

There is quite a variety of distributions for each of the predictor variables, some appearing normal, or nearly normaly distributed and other clearly multi-modal or puntuated, which led us to further examine the relationships between the the different brands in `Brand Code` and the predictor variables.

Plotting histograms with colored by brand, it is clear that the other preditors vary relative to their associated brand. However, there are very few areas within the distributions where brand is a fixed dividing line between values in the distributions. `Carb Volume`, `Carb Rel` & `Density` are obviously divided by at least one brand.



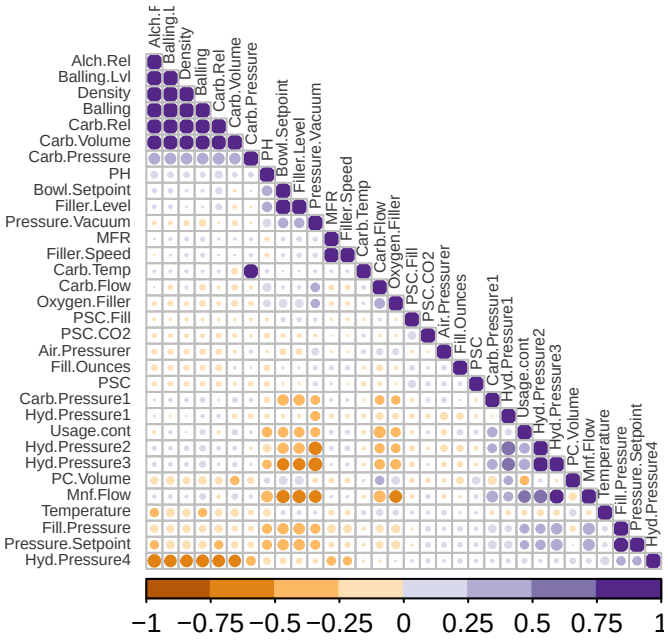
It may be useful to see how Brands relate to Ph ranges.



Clearly there are different quantities of each of the brands, but they are all relatively similarly distributed among the variety of Ph values, including those values which are missing.

Correlations

The plot below shows that `BallingAlch`, `RelBalling`, `LvlDensityCarb`, `RelBrand`, `CodeDCarb.`, `VolumeCarb`, `Pressure` are all highly correlated with each other, but not particularly highly correlated with the outcome, `PH` variable. They are all 25% or less correlated with `PH`, as `PH` is with other variables both positive and negative.



Data Pre-Processing

Imputation

Because the data is missing values in both the training and test sets, it is important to maintain a consistent imputational method between the Student data and the test data so that identical observations in both sets would recieve the same imputed missing values. The bigger issue in imputation is how to manage the missing `Brand Code` observations.

It could be useful to impute them based on the other variables, but it may also be meaningful to leave consider them unknown, allowing the know classes to differentiate well, as they seem to be highly related to the outcome variable.

Because the models under evaluation both require, scaling, centering and are not robust with categorical variables, it makes sense to use carets knn imputation for this as it will allow us to apply this method to the test data. The outcome variable, `PH` had four missing observations, those rows were simply dropped.

Test data is imputed with this model and `PH` is removed from the set.

Model Preprocessing

Both K-nearest neighbors and Support Vector machine Regressions require centering and scaling to reduce the bias of magnitude between variables, we also removed the variable `Hyd Pressure1` due to near zero variance and applied Box-Cox transformations to the models to satify the near-normalcy requirements.

Because neither model is robust to categorical data, the `Brand Code` was converted to dummy variables with no category for the unknowns. This seemed like the most effective way of sussing out the true influence of categories without introducing bias of imputation.

Cross Validation

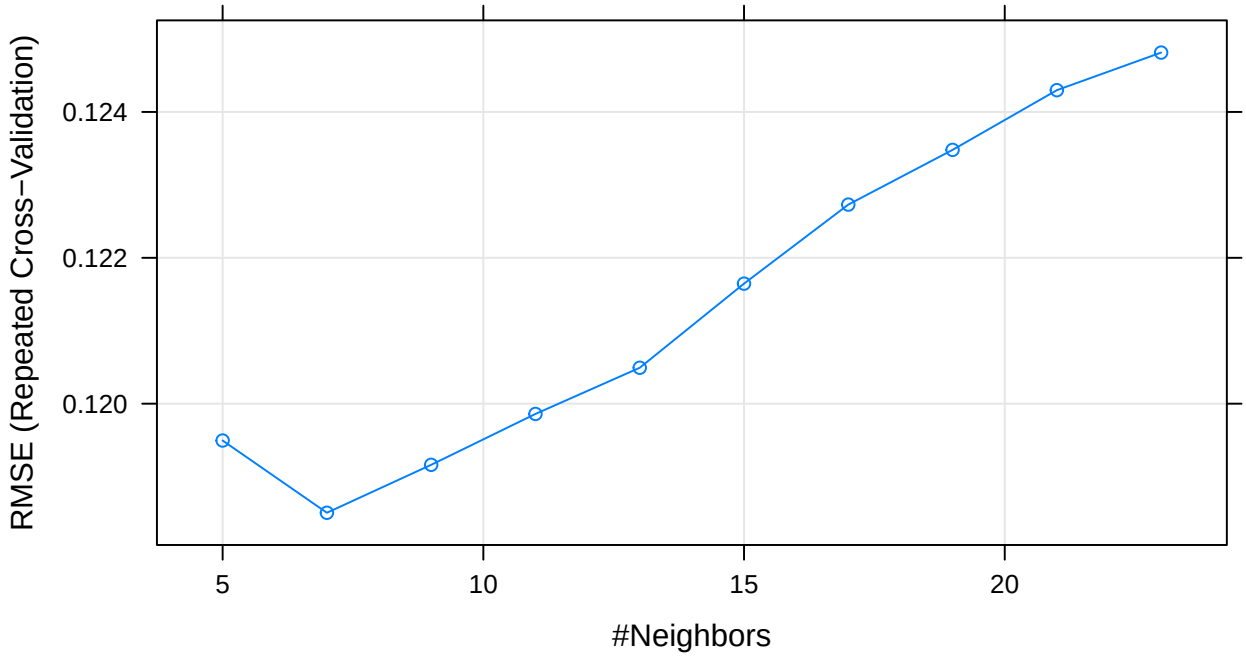
Instead of doing a train-test-split, we used repeated cross validation, with 6 model repetitions and 10 folds of cross validation per model. This allows us to train over all of the observation enhancing resolution for brands with less observations.

Models

For both KNN and SVM models a grid of seeds was created from our original seed to ensure that out repeated cross validation would be repeateable. The same seeds were used in both the SVM and KNN.

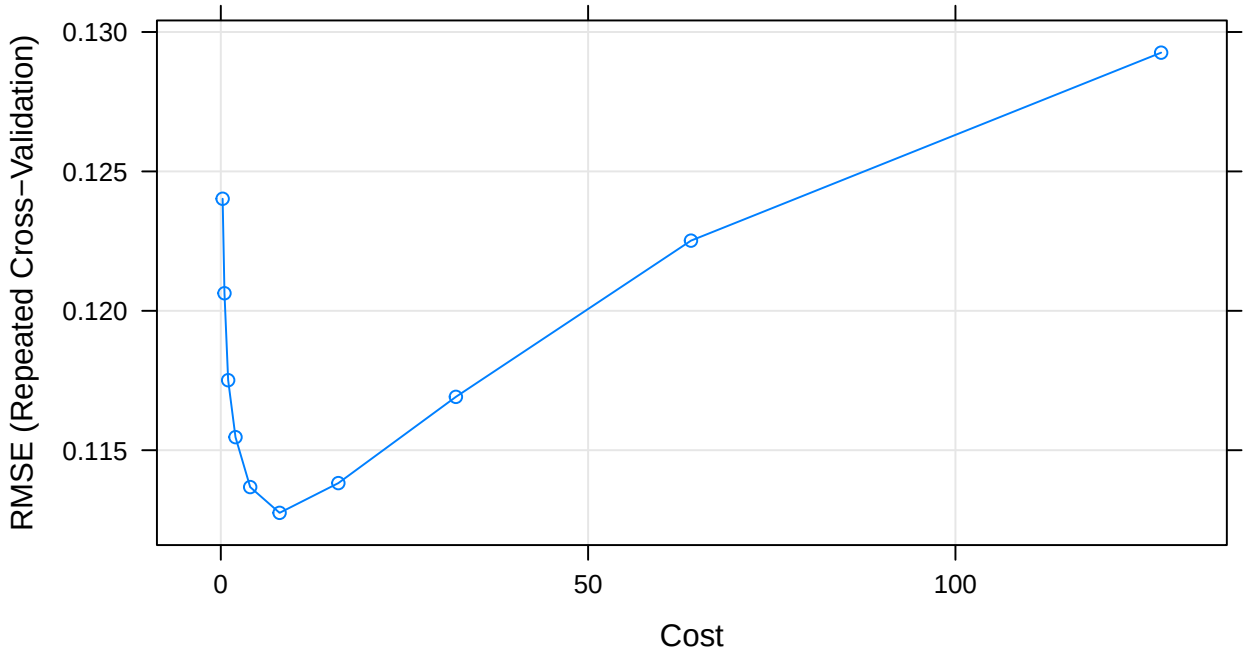
K-nearest Neighbor

The K-nearest Neighbor was tuned across a variety of neighbors where we settled on a 7 neigbors as the optimal number based on Root Mean Squared Error of `0.1185`.



Support Vector Machine

The support vector machine, although less efficient than the k-nearest neighbor to train, provided a much more robust final model using a radial kernel with a cost of 10, passed as the tune length settling on $\sigma = 0.020$ and $cost = 8$ returning a $RMSE = 0.1127$

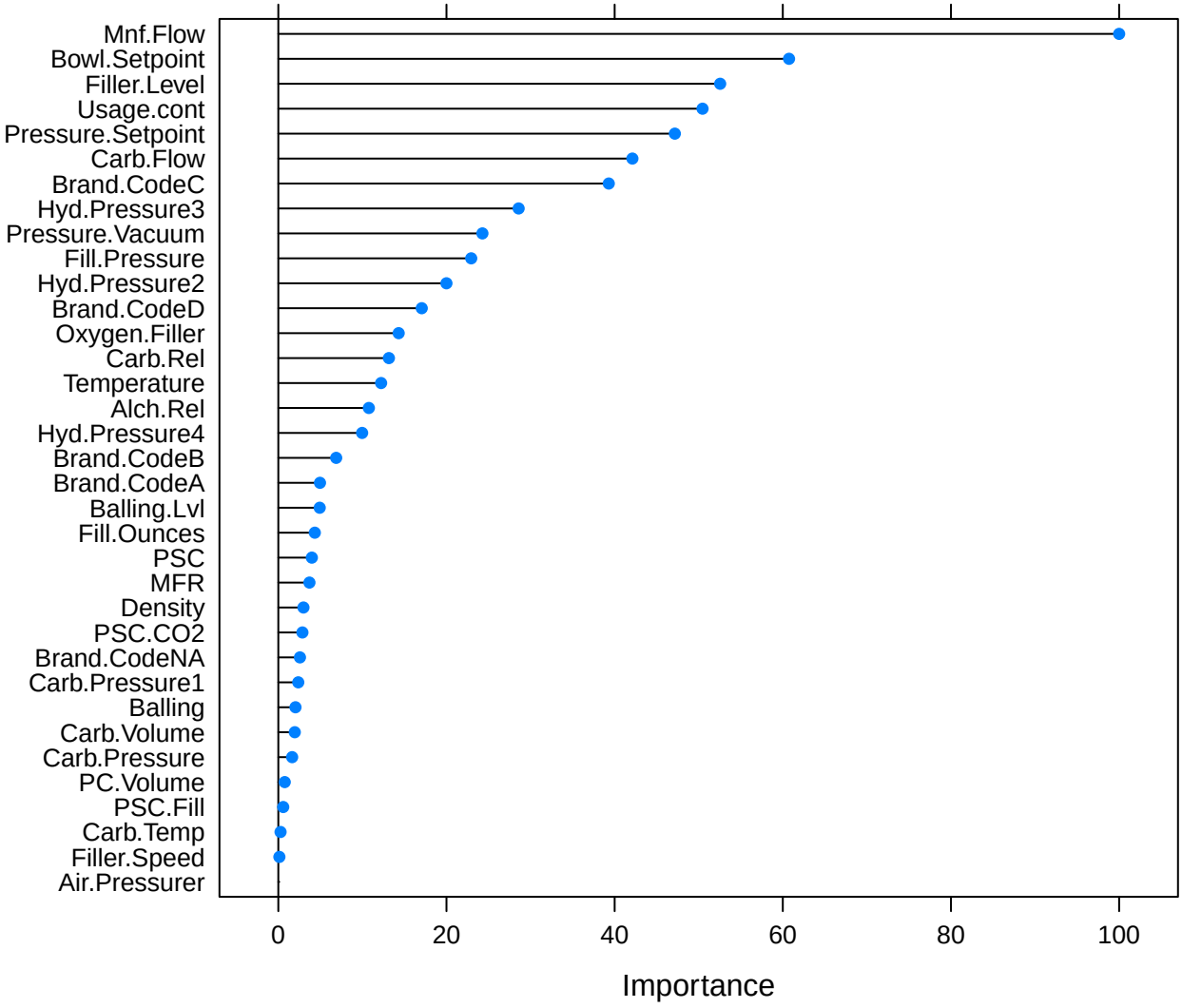


Comparing Our Training Models

Using the 6-repeat 10-fold cross validation the best of the two models, base on the Mean Absolute Percent of Error is the Support Vector regression of .48%, which is almost two times as good as the K-nearest neighbor model.

	RMSE	Rsquared	MAE	MAPE
6	0.1127538	0.5768736	0.0826887	0.004850238
2	0.1185047	0.5329933	0.0884563	0.008826196

As neither SVM nor KNN are able to provide meaningful variable importances, it is hard to compare them by variable. However, caret does provide a comparison based on the variables themselves outside of the models. IT would be interesting to compare this other model importances.



Conclusions

To come later

Appendix

```
require(tidyverse)
require(caret)
require(readxl)
require(XLConnect)
require(naniar)
require(caret)
require(mice)
require(VIM)
require(corrplot)
require(RColorBrewer)
require(gridExtra)
require(ggplot2)
require(readxl)
require(RANN)
require(MLmetrics)
require(dummies)
require(XLConnectJars)
require(openxlsx)
# Importing Data
beverage <- read_excel("/data/StudentData.xlsx", sheet = 1)
test <- read_excel("/data/StudentEvaluation- TO PREDICT.xlsx")

# Creating Useful Names
colnames(beverage) <- make.names(colnames(beverage), unique = TRUE)
colnames(test) <- colnames(beverage)

# Estimating Missing Data
sapply(beverage, function(x) sum(is.na(x))) %>% knitr::kable(col.names = c("Missing Values"),
 caption = "Missing Values Student Data")
sapply(test, function(x) sum(is.na(x))) %>% knitr::kable(col.names = c("Missing Values"),
 caption = "Missing Values Test Data")

# Remove observations with no outcome
beverage <- beverage %>% drop_na(PH)
# training Data from Beverage with Missing PH removed
y_train <- beverage %>% select(PH)

beverage_2 <- beverage %>% mutate(Brand.Code = factor(beverage$Brand.Code)) %>%
 drop_na(PH)

test <- test %>% mutate(Brand.Code = factor(test$Brand.Code))

# Looking at Histograms by Brand
gather_if <- function(data, FUN, key = "key", value = "value",
 na.rm = FALSE, convert = FALSE, factor_key = FALSE) {
 data %>% {
  gather(., key = key, value = value, names(.)[sapply(.,
   FUN = FUN)], na.rm = na.rm, convert = convert,
   factor_key = factor_key)
 }
}
```

```r
hist_data <- beverage_2 %>% mutate(PH = as.factor(PH)) %>%
 gather_if(is.numeric, "key", "value") %>% filter(!is.na(value)) %>%
 group_by(key)


bev_hist <- ggplot(hist_data, aes(value)) + geom_histogram(aes(fill = Brand.Code),
 color = "#999999", alpha = 0.3) + labs(subtitle = "Colored By Brand",
 x = "", y = "") + facet_wrap(~key, scales = "free", nrow = 7) +
 theme_bw() + theme(axis.text.y = element_blank(), axis.title.x = element_blank(),
 axis.text.x = element_blank(), legend.position = "bottom",
 legend.key.size = unit(0.4, "cm")) + scale_fill_manual()

# Histograms by Brand vs. Ph
ph_brand <- beverage_2 %>% select("PH", "Brand.Code") %>%
 mutate(Brand.Code = as.factor(Brand.Code)) %>% gather_if(is.numeric,
 "key", "value") %>% filter(!is.na(value)) %>% group_by(key)

y_hist <- ggplot(ph_brand, aes(value)) + facet_wrap(~key,
 scales = "free") + geom_histogram(aes(fill = Brand.Code)) +
 theme_bw() + theme(axis.text.y = element_blank(), axis.title.x = element_blank(),
 axis.text.x = element_blank(), legend.position = "bottom",
 legend.key.size = unit(0.4, "cm")) + scale_fill_manual()

# Correlations
bev_cor <- cor(na.omit(beverage_2[2:ncol(beverage_2)]))
correlations <- corrplot(bev_cor, method = "circle", type = "lower",
 order = "FPC", col = brewer.pal(n = 8, name = "PuOr"))

# Estimating Near Zero Variance
nzv_bev <- nearZeroVar(beverage_2, saveMetrics = TRUE) %>%
 knitr::kable(caption = "Near Zero Estimates Student Data")  # Just HydPressure 1


# Making brands a factor
beverage$Brand.Code <- factor(beverage$Brand.Code)

# Missing Values By Variable
aggr(beverage, col = c("darkorange", "darkorchid"), numbers = TRUE,
 sortVars = TRUE, labels = names(beverage), cex.axis = 0.7,
 gap = 3, ylab = c("Histogram of missing data", "Pattern"))


# Beverage TRaining Imputed using knn in Caret so applies
# to Test, converting missing Brand to Unknown
ivals <- preProcess(beverage_2, method = "knnImpute", k = 10)
beverage_2 <- predict(ivals, newdata = beverage_2)

# Test Imputed
test_clean <- predict(ivals, test)

# Create Training Dummies b_dummies
# <-dummy(beverage_2$Brand.Code)
# beverage_clean<-cbind(b_dummies, beverage_2)%>%
```

```r
# select(-Brand.Code, -Hyd.Pressure1)

# Create Test Data Dummies
test_clean <- test_clean %>% mutate(Brand.Code = ifelse(is.na(Brand.Code),
 "Unknown", Brand.Code), Brand.Code = factor(Brand.Code))


t_dummies <- dummy(test_clean$Brand.Code)
test_clean <- cbind(t_dummies, test_clean) %>% select(-Brand.Code,
 -Hyd.Pressure1)







# Creating Training & Testing Data
x_train <- beverage_clean %>% select(-PH)
y_train <- beverage %>% select(PH)

# Creating Training & Testing Data

x_test <- test_clean %>% select(-PH)

# Setting up crossvalidation controls

# set.seed(58677) seeds <- vector(mode = 'list', length =
# 61) for(i in 1:60) seeds[[i]] <- sample.int(1000, 10)
# seeds[[61]] <- sample.int(1000, 1) control <-
# trainControl( method = 'repeatedcv', number = 10, ##
# repeated ten times repeats = 6, seeds = seeds)




# Train and Tune the SVM set.seed(58677) svm.tune <-
# train(x=x_train, y= y_train$PH, method = 'svmRadial',
# tuneLength = 10, preProc = c('center', 'scale', 'nzv',
# 'BoxCox'), metric='MAPE', trControl=control) #See SVM
# Output
svm_results <- svm.tune$results %>% knitr::kable()

# SVM Training Predictions & Error
svm_train_preds <- predict(svm.tune, new_data = x_train)
svm_metrics <- postResample(svm_train_preds, obs = y_train$PH) %>%
 knitr::kable(col.names = c("Metric"), caption = "Training Performance Metrics for SVM")
svm_train_mape <- MAPE(svm_train_preds, y_train$PH)

# SVM Training Predictions & Error
svm_test_preds <- predict(svm.tune, new_data = x_test)

# To Come varImp(svm.tune$finalModel)
```

```r
# Traine & Tune KNN



# set.seed(58677) knn.tune <- train(x = x_train, y =
# y_train$PH, method = 'knn', preProc = c('center',
# 'scale', 'nzv', 'BoxCox'), tuneLength = 10, trControl =
# control)

# Evaluate Knn models
knn_results <- knn.tune$results %>% knitr::kable()




train_knn_pred <- predict(knn.tune, newdata = x_train)
knn_metric <- postResample(pred = train_knn_pred, obs = y_train$PH) %>%
 knitr::kable(col.names = c("Metric"), caption = "Training Performance Metrics for knn")

knn_train_mape <- MAPE(train_knn_pred, y_train$PH)
# Test Preds
knn_test_preds <- predict(knn.tune, new_data = x_test)



# TO Come on Single Models varImp(knn.tune)
# write.xlsx(svm_test_preds, 'group_2_predictions.xlsx',
# sheet='SVM') write.xlsx(knn_test_preds,
# 'group_2_predictions.xlsx', sheet='KNN')



# saveRDS(knn.tune, 'knn.rds') saveRDS(svm.tune,
# 'svm.rds') saveRDS(beverage_clean,
# 'final_beverage.rds') # knn.tune<- readRDS('knn.rds') #
# svm.tune < readRDS('svm.rds') # beverage_clean <-
# readRDS('final_beverage.rds')

# varImp(knn.tune$finalModel)

svm_best <- svm.tune$results[6, 3:5]
knn_best <- knn.tune$results[2, 2:4]
best_models <- rbind(svm_best, knn_best)
best_models$MAPE <- rbind(svm_train_mape, knn_train_mape)
```