Linear Models Wholesale Customer Analysis

Johnny Lee

2025-04-01

Libraries And Seed

```
library(glmnet)
library(tidyverse)
library(GGally)
library(caret)
set.seed(7890682)
options(scipen = 999)
```

Functions

```
# Return the RMSE
getRMSE = function(actual, pred) {
  return (sqrt(mean((actual - pred)^2)))
}

# Z-score scale continuous variables
scaleZ = function(train, test) {
  scaled_train = train[,3:8]
  scaled_test = test[,3:8]
  for (i in 3:8) {
    mu_train = mean(train[,i])
    sd_train = sd(train[,i])
    scaled_train[,i-2] = (train[,i] - mu_train)/sd_train
    scaled_test[,i-2] = (test[,i] - mu_train)/sd_train
  }
  scaled_train = cbind(train[,1:2], scaled_train)
  scaled_test = cbind(test[,1:2], scaled_test)
  return (list(scaled_test = scaled_test, scaled_train = scaled_train))
}

# Create a ridge/lasso regression model and return important information
runGlmnet = function(alpha, X_train, Y_train, X_test, Y_test, groceryTrain) {
  cv_model = cv.glmnet(X_train, Y_train, alpha = alpha, nfolds = 10)

  preds_min_scaled = predict(cv_model, newx = X_test, s = cv_model$lambda.min)
  preds_1se_scaled = predict(cv_model, newx = X_test, s = cv_model$lambda.1se)
```

```r
  rmse_min = getRMSE(Y_test, preds_min_scaled)
  rmse_1se = getRMSE(Y_test, preds_1se_scaled)

  items = list(
    model = cv_model,
    rmse_min = rmse_min * sd(groceryTrain),
    rmse_1se = rmse_1se * sd(groceryTrain),
    lambda_min = cv_model$lambda.min,
    lambda_1se = cv_model$lambda.1se,
    cv_rmse_min = sqrt(min(cv_model$cvm)) * sd(groceryTrain)
  )

  return (items)
}

# Cross validate linear models
runLM = function(train, features) {
  ctrl_linear <- trainControl(
    method = "cv",
    number = 10,
    savePredictions = "final",
  )

  formula = as.formula(paste("Grocery", "~", paste(features, collapse = " + ")))

  lm_model <- train(
    formula,
    data = train,
    method = "lm",
    trControl = ctrl_linear,
    metric = "RMSE"
  )

  return (lm_model$results$RMSE)
}

# Cross validate the base mean of grocery training data model
kFoldBase = function(train, k) {
  rmses = numeric(k)
  n = nrow(train)
  fold_size = floor(n/k)

  shuffled_indices = sample(1:n, n)
  train_shuffled = train[shuffled_indices,]

  for (i in 1:k) {
    start = (i-1) * fold_size + 1

    if (i == k) {
      end = n
    } else {
      end = i*fold_size
    }
```

```
    valid_set = train_shuffled[start:end, ]
    train_set = train_shuffled[-(start:end), ]


    model = mean(train_set$Grocery)
    rmses[i] = getRMSE(model, valid_set$Grocery)
  }

  return (mean(rmses))
}
```

Preparing Data

```
# Reading data
data = read.csv("Wholesale_customers_data.csv")
data$Region = factor(data$Region)
data$Channel = factor(data$Channel)

# Sampling training rows with 70/30 split
trainRows = sample(nrow(data), 0.70*nrow(data))

# Original training and testing data
train = data[trainRows, ]
test = data[-trainRows, ]

# Z-score scaled training and testing data
scaledData = scaleZ(train, test)
trainZ = scaledData$scaled_train
testZ = scaledData$scaled_test
```

Base Model

```
# Model
baseGrocery = mean(train$Grocery)

# RMSE on test
rmse = getRMSE(test$Grocery, baseGrocery)

# RMSE using cv
cv_rmse = kFoldBase(train,10)

c(rmse, cv_rmse)
```

```
## [1] 11522.397  7945.851
```

RMSE on test data: 11522.397

RMSE using cross-validation: 7945.851

Let's try to bring this down and add the term most correlated to Grocery which is Detergents_Paper.

Detergents_Paper Linear Model

```r
# Model
fit = lm(Grocery ~ Detergents_Paper, data = train)

# Predictions
preds = predict(fit, newdata = test)

# RMSE on test
rmse = getRMSE(test$Grocery, preds)

# RMSE using cv
cv_rmse = runLM(train, "Detergents_Paper")

c(rmse, cv_rmse)
```

```
## [1] 3980.050 3499.863
```

RMSE on test data: 3980.050

RMSE using cross-validation: 3499.863

Let's add the term second most correlated to Grocery which is Milk.

Detergents_Paper + Milk Linear Model

```r
# Model
fit = lm(Grocery ~ Detergents_Paper + Milk, data = train)

# Predictions
preds = predict(fit, newdata = test)

# RMSE on test
rmse = getRMSE(test$Grocery, preds)

# RMSE using cv
cv_rmse = runLM(train, c("Detergents_Paper", "Milk"))

c(rmse, cv_rmse)
```

```
## [1] 4143.552 3018.830
```

RMSE on test data: 4143.552

RMSE using cross-validation: 3018.830

Let's add all features.

All Term Linear Model

```r
# Model
fit = lm(Grocery ~ ., data = train)

# Predictions
preds = predict(fit, newdata = test)

# RMSE on test
rmse = getRMSE(test$Grocery, preds)

# RMSE using cv
cv_rmse = runLM(train, ".")

c(rmse, cv_rmse)
```

```
## [1] 3944.449 3092.855
```

RMSE on test data: 3944.449

RMSE using cross-validation: 3092.855

Let's remove Region and Channel.

All Continuous Term Linear Model

```r
# Model
fit = lm(Grocery ~ . -Region - Channel, data = train)

# Predictions
preds = predict(fit, newdata = test)

# RMSE on test
rmse = getRMSE(test$Grocery, preds)

# RMSE using cv
cv_rmse = runLM(train, ". -Region - Channel")

c(rmse, cv_rmse)
```

```
## [1] 3938.570 3061.831
```

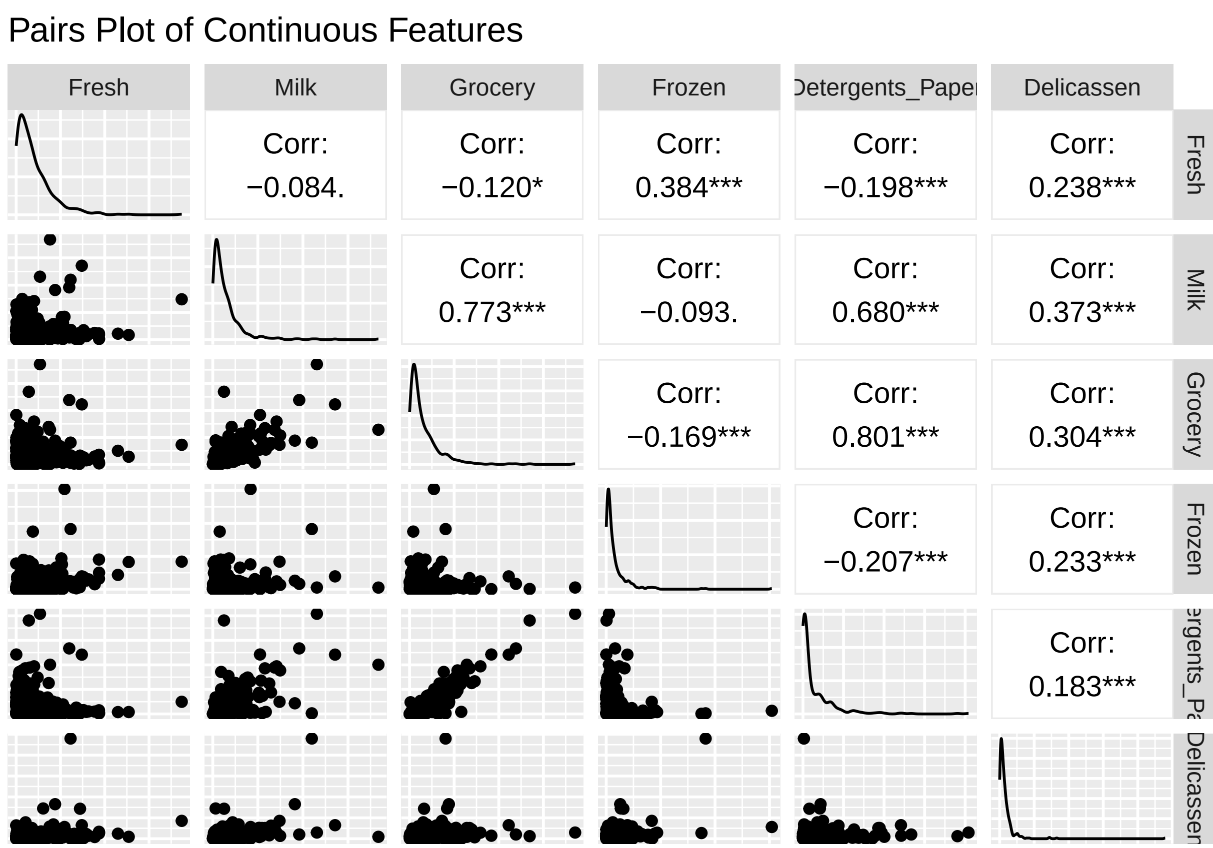
RMSE on test data: 3938.570

RMSE using cross-validation: 3061.831

Let's do a pairs plot to see the relationship between our variables.

Pair plot

```r
ggpairs(data[,-c(1,2)],
        upper = list(continuous = wrap("cor",
                                       method = "spearman",
                                       size = 4,
                                       color = "black"))) +
  theme(axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks = element_blank()) +
  labs(title = "Pairs Plot of Continuous Features")
```



Interaction: Detergent and Milk, Frozen and Fresh, Milk and Delicassen. Lets add those interaction terms in the model.

Non-Linear relationship Possibility: Grocery and Fresh, Frozen, and Delicassen.

Let's add these terms to the model.

Max Feature Model

```r
# Model
fit = lm(Grocery ~ poly(Fresh, degree = 3) +
                poly(Frozen, degree = 2) +
                poly(Delicassen, degree = 2) +
                Milk +
                Detergents_Paper +
```

```r
                 Channel +
                 Region +
                 (Fresh * Frozen) +
                 (Milk * Detergents_Paper) +
                 (Milk * Delicassen),
                 data = train)

# Predictions
preds = predict(fit, newdata = test)

# RMSE on test
rmse = getRMSE(preds, test$Grocery)

# RMSE using cv
cv_rmse = fullModel= runLM(train, c("poly(Fresh, degree = 3)",
                 "poly(Frozen, degree = 2)",
                 "poly(Delicassen, degree = 2)",
                 "Milk",
                 "Detergents_Paper",
                 "Channel",
                 "Region",
                 "(Fresh * Frozen)",
                 "(Milk * Detergents_Paper)",
                 "(Milk * Delicassen)"))
```

```
## Warning in predict.lm(modelFit, newdata): prediction from rank-deficient fit;
## attr(*, "non-estim") has doubtful cases
## Warning in predict.lm(modelFit, newdata): prediction from rank-deficient fit;
## attr(*, "non-estim") has doubtful cases
## Warning in predict.lm(modelFit, newdata): prediction from rank-deficient fit;
## attr(*, "non-estim") has doubtful cases
## Warning in predict.lm(modelFit, newdata): prediction from rank-deficient fit;
## attr(*, "non-estim") has doubtful cases
```

```r
c(rmse, cv_rmse)
```

```
## [1] 4760.82 4487.64
```

RMSE on test data: 4760.82

RMSE using cross-validation: 4487.64

Lots of multicollinearity now, let's remove the interaction terms.

Max Feature Model No Interaction Terms

```r
# Model
fit = lm(Grocery ~ poly(Fresh, degree = 3) +
                 poly(Frozen, degree = 2) +
                 poly(Delicassen, degree = 2) +
                 Milk +
```

```r
                Detergents_Paper +
                Channel +
                Region,
                data = train)

# Predictions
preds = predict(fit, newdata = test)

# RMSE on test
rmse = getRMSE(preds, test$Grocery)

# RMSE using cv
cv_rmse = runLM(train, c("poly(Fresh, degree = 3)",
                "poly(Frozen, degree = 2)",
                "poly(Delicassen, degree = 2)",
                "Milk",
                "Detergents_Paper",
                "Channel",
                "Region"))

c(rmse, cv_rmse)
```

```
## [1] 4034.895 3539.790
```

RMSE on test data: 4034.895

RMSE using cross-validation: 3539.790

Let's remove Region and Channel.

Continuous and Polynomial Term Model

```r
# Model
fit = lm(Grocery ~ poly(Fresh, degree = 3) +
                poly(Frozen, degree = 2) +
                poly(Delicassen, degree = 2) +
                Milk +
                Detergents_Paper,
                data = train)

# Predictions
preds = predict(fit, newdata = test)

# RMSE on test
rmse = getRMSE(preds, test$Grocery)

# RMSE using cv
cv_rmse = runLM(train, c("poly(Fresh, degree = 3)",
                "poly(Frozen, degree = 2)",
                "poly(Delicassen, degree = 2)",
                "Milk",
                "Detergents_Paper"))
```

```r
c(rmse, cv_rmse)
```

```
## [1] 4001.833 3975.031
```

RMSE on test data: 4001.833

RMSE using cross-validation: 3975.031

Let's try ridge regression on our max feature model.

Max Feature Ridge Model

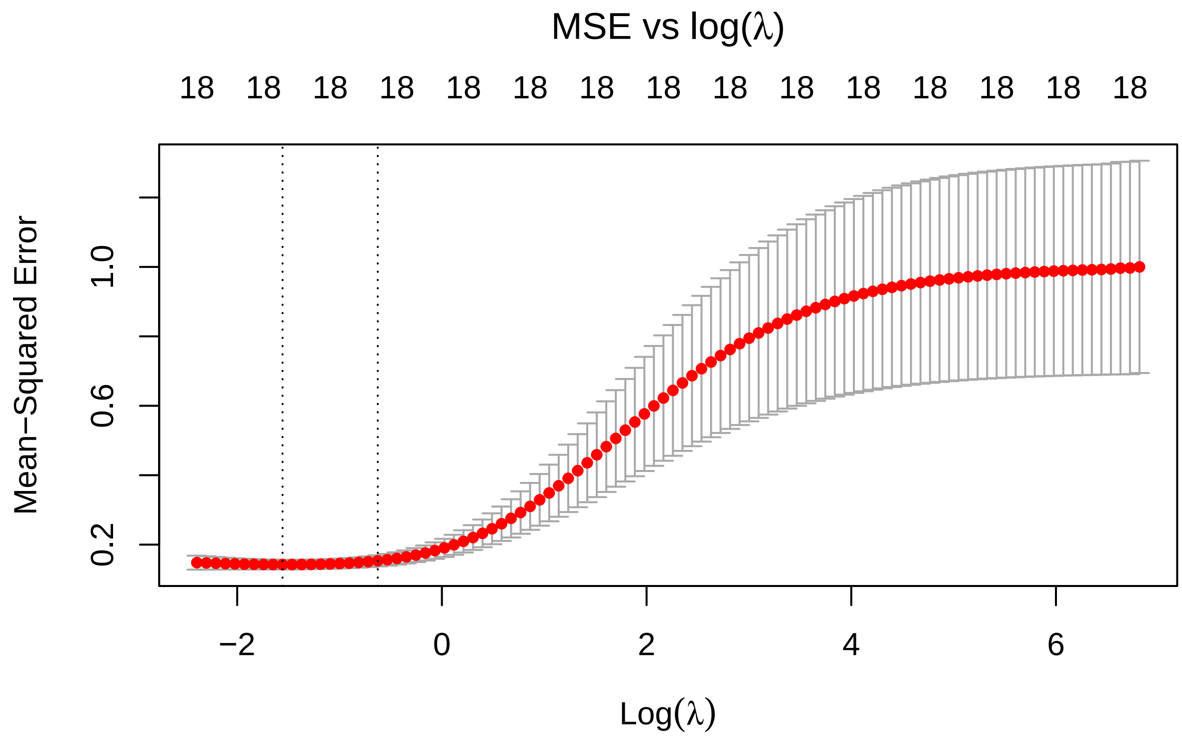
```r
# Model
fit = lm(Grocery ~ poly(Fresh, degree = 3) +
                poly(Frozen, degree = 2) +
                poly(Delicassen, degree = 2) +
                Milk +
                Detergents_Paper +
                Channel +
                Region +
                (Fresh * Frozen) +
                (Milk * Detergents_Paper) +
                (Milk * Delicassen),
                data = trainZ)

# Training data
X_train = model.matrix(fit)[,-1]
Y_train = trainZ$Grocery

# Testing data
X_test = model.matrix(terms(fit), data = testZ)[, -1]
Y_test = testZ$Grocery

# Running cv on model
ridge_items = runGlmnet(0, X_train, Y_train, X_test, Y_test, train$Grocery)

# Plotting lambdas
par(mar = c(5,4,6,2))
plot(ridge_items$model,
     main = expression("MSE vs log("*lambda*")"),
     cex.main = 1.2)
```



Model preforms best for log(s) < 0. Let's see the RMSE values.

```
c(ridge_items$rmse_min,
  ridge_items$rmse_1se,
  ridge_items$cv_rmse_min,
  ridge_items$lambda_min,
  ridge_items$lambda_1se)
```

```
## [1] 5353.0180816 5606.2259668 3216.6982265    0.2108156    0.5344943
```

RMSE with min lambda on test data: 5353.018

RMSE with 1se lambda on test data: 5606.226

RMSE using cross-validation: 3216.698

The model didn't preform great on the great test data but did well in cross validation. Let's check significant coefficients.

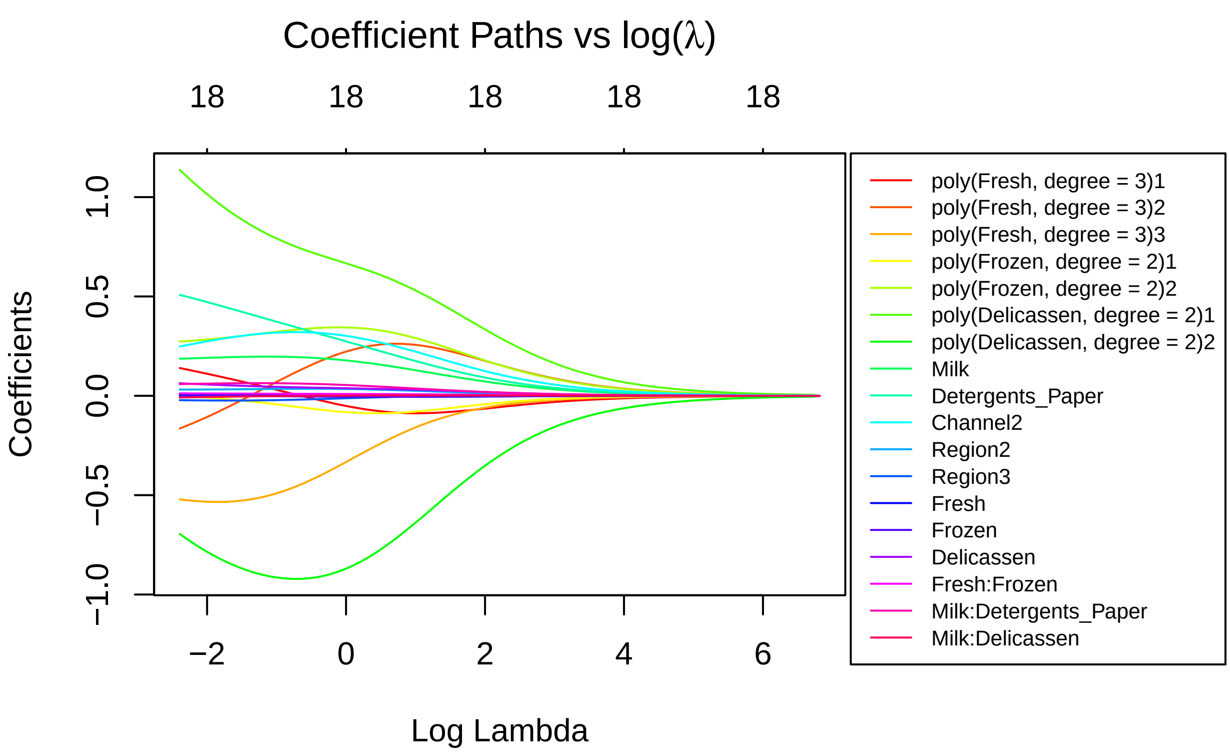
```
# Model
fit = glmnet(X_train, Y_train, alpha = 0)

# Adjusting Plot Margins
par(mar = c(5,4,6,10.5))

# Plot
plot(fit, xvar = "lambda", col = rainbow(ncol(X_train)),
```

```r
     main = expression("Coefficient Paths vs log("*lambda*")"),
     cex.main = 1.2)

legend("topright",
  legend = colnames(X_train),
  col = rainbow(ncol(X_train)),
  lty = 1,
  xpd = TRUE,
  inset = c(-0.55,0),
  cex = 0.7)
```

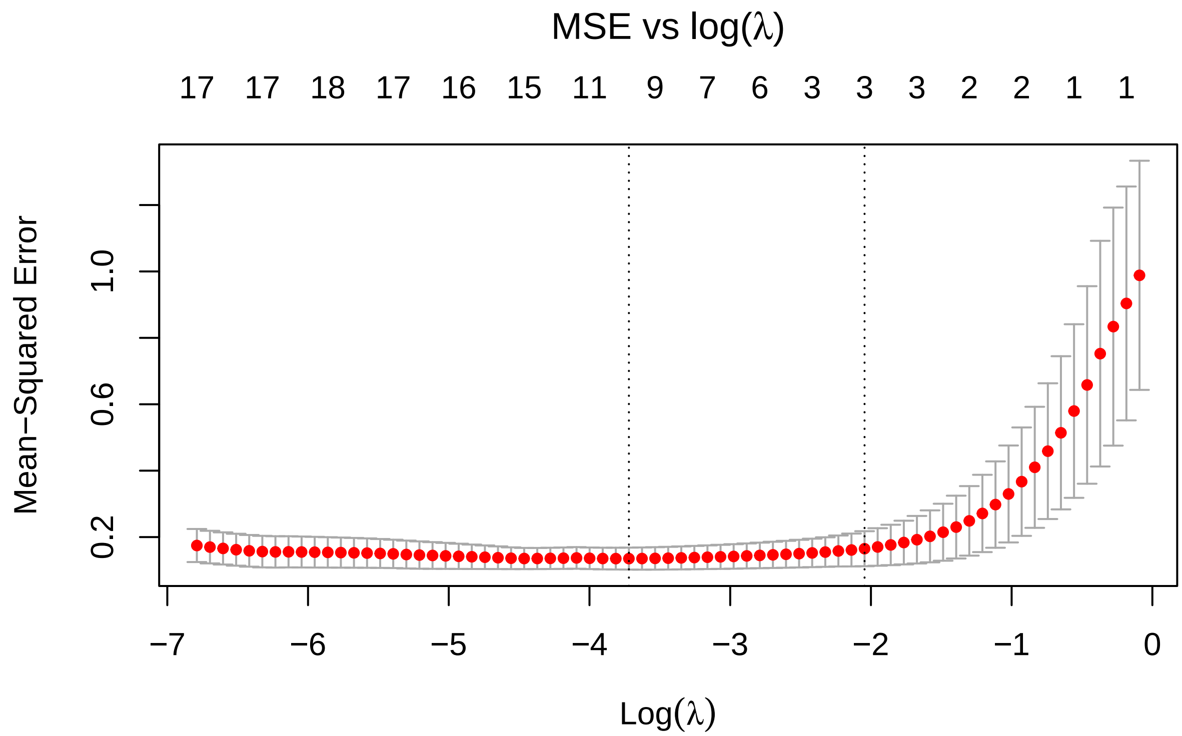


The polynomials terms for Delicassen are the most significant. Given the large discrepancy in RMSE values this indicates the polynomials terms are over fitting the model leading to smaller RMSE during cross validation.

Max Feature Lasso Model

```r
# Model
lasso_items = runGlmnet(1, X_train, Y_train, X_test, Y_test, train$Grocery)

# Plotting lambdas
par(mar = c(5,4,6,2))
plot(lasso_items$model,
     main = expression("MSE vs log("*lambda*")"),
     cex.main = 1.2)
```




```
# Saving Models
saveRDS(
  list(ridge_max = ridge_items, lasso_max = lasso_items),
  "regression_max_models.rds"
)
```

It appears lower levels of lambda correspond to lower MSE, meaning most terms are significant. Let's look at the RMSE and lambda values.

```
c(lasso_items$rmse_min,
  lasso_items$rmse_1se,
  lasso_items$cv_rmse_min,
  lasso_items$lambda_min,
  lasso_items$lambda_1se)
```

```
## [1] 4443.75541665 4300.74183944 3131.54892786    0.02423867    0.12935448
```

RMSE with min lambda on test data: 4443.755

RMSE with 1se lambda on test data: 4300.742

RMSE using cross-validation: 3131.549

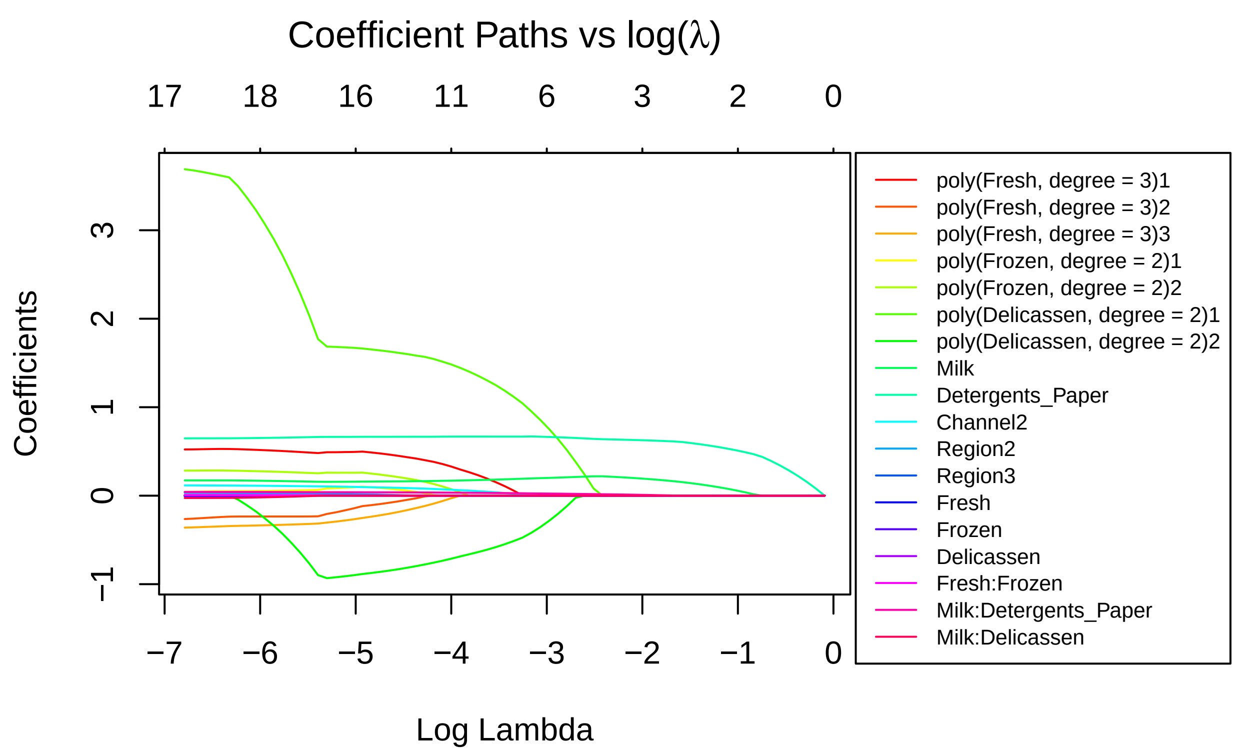
The model didn't preform great on the great test data but did well in cross validation. Let's check significant coefficients.

```r
# Model
fit = glmnet(X_train, Y_train, alpha = 1)

# Adjusting Plot Margins
par(mar = c(5,4,6,10.5))

# Plot
plot(fit, xvar = "lambda", col = rainbow(ncol(X_train)),
     main = expression("Coefficient Paths vs log("*lambda*")"),
     cex.main = 1.2)

legend("topright",
  legend = colnames(X_train),
  col = rainbow(ncol(X_train)),
  lty = 1,
  xpd = TRUE,
  inset = c(-0.55,0),
  cex = 0.7)
```



The polynomials terms for Delicassen are the most significant. Given the large discrepancy in RMSE values this indicates the polynomials terms are over fitting the model leading to smaller RMSE during cross validation. This is the same situation we had with ridge regression. Let's remove the polynomial terms and go back to the continuous feature model only.

Continuous Ridge Model

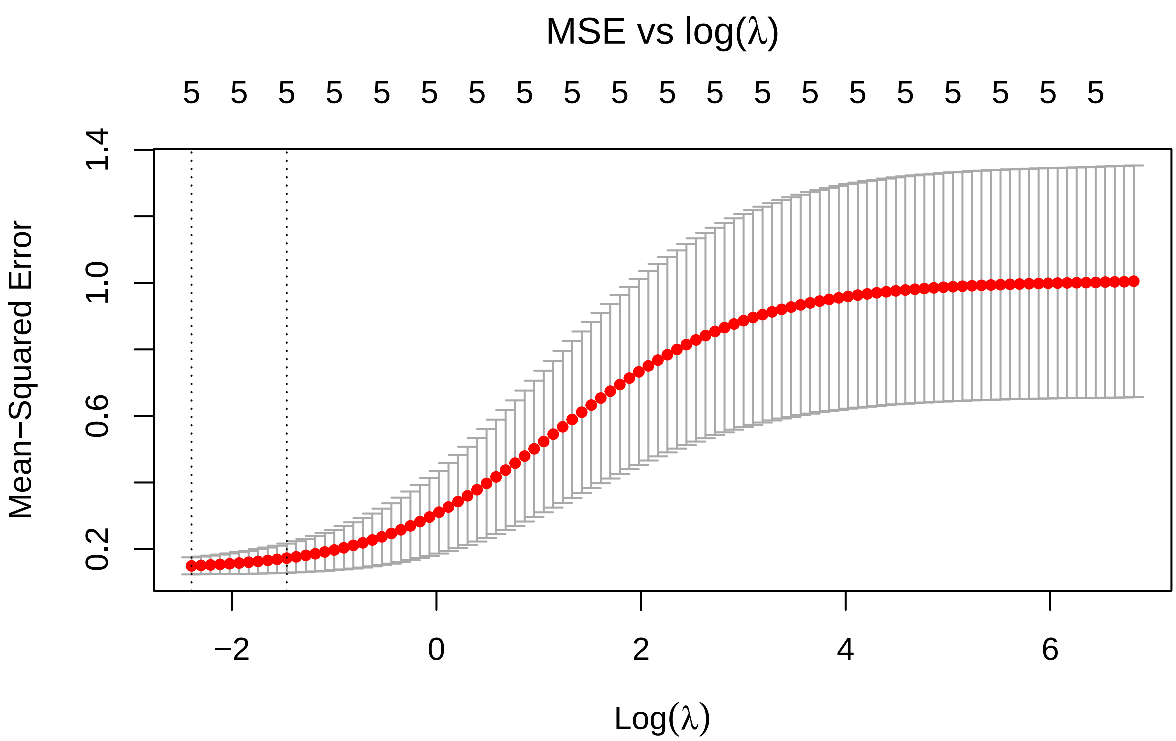
```
# Model
fit = lm(Grocery ~. -Region - Channel,
                data = trainZ)

# Training data
X_train = model.matrix(fit)[,-1]
Y_train = trainZ$Grocery

# Testing data
X_test = model.matrix(terms(fit), data = testZ)[, -1]
Y_test = testZ$Grocery

# Running cv on model
ridge_items = runGlmnet(0, X_train, Y_train, X_test, Y_test, train$Grocery)

# Plotting lambdas
par(mar = c(5,4,6,2))
plot(ridge_items$model,
     main = expression("MSE vs log("*lambda*")"),
     cex.main = 1.2)
```



Model preforms best for log(s) < 0. Let's see the RMSE values.

```r
c(ridge_items$rmse_min,
  ridge_items$rmse_1se,
  ridge_items$cv_rmse_min,
  ridge_items$lambda_min,
  ridge_items$lambda_1se)
```

```
## [1] 4115.01364356 4400.43456238 3292.08764323    0.09125703    0.23136982
```

RMSE with min lambda on test data: 4115.0137

RMSE with 1se lambda on test data: 4400.435

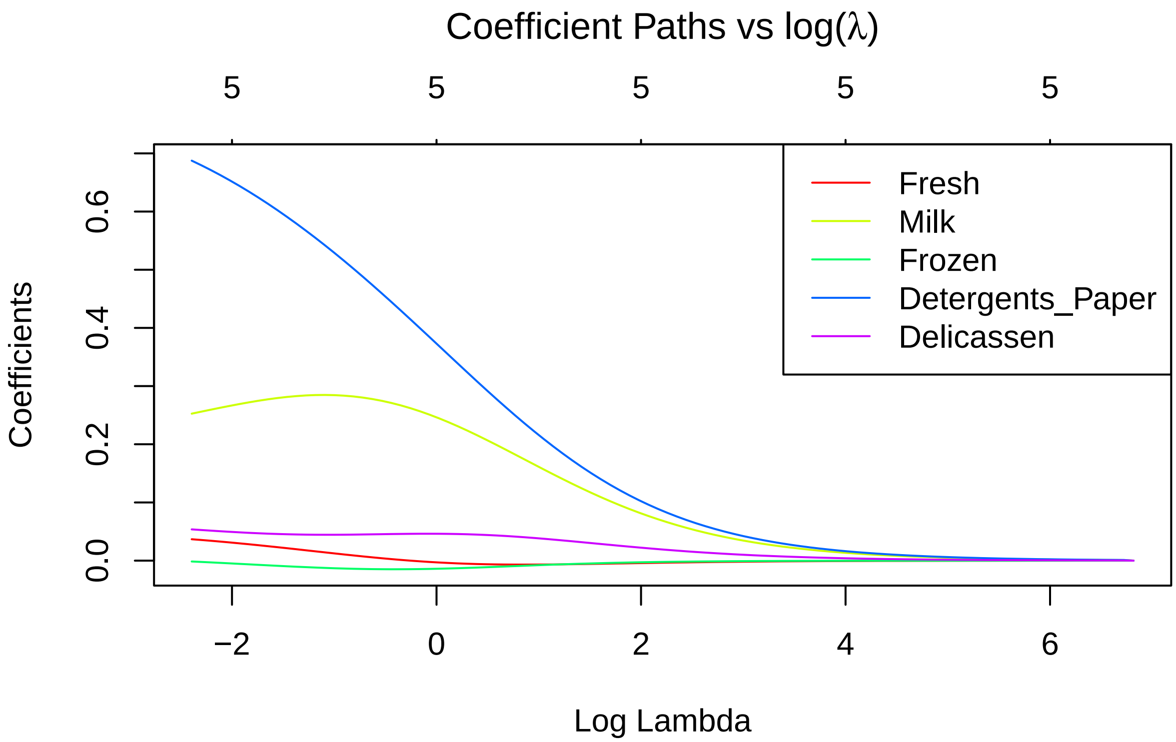
RMSE using cross-validation: 3292.087

```r
# Model
ridge_fit = glmnet(X_train, Y_train, alpha = 0)

# Adjusting Plot Margins
par(mar = c(5,4,6,2))

# Plot
plot(ridge_fit, xvar = "lambda", col = rainbow(ncol(X_train)),
     main = expression("Coefficient Paths vs log("*lambda*")"),
     cex.main = 1.2)

legend("topright",
  legend = colnames(X_train),
  col = rainbow(ncol(X_train)),
  lty = 1)
```

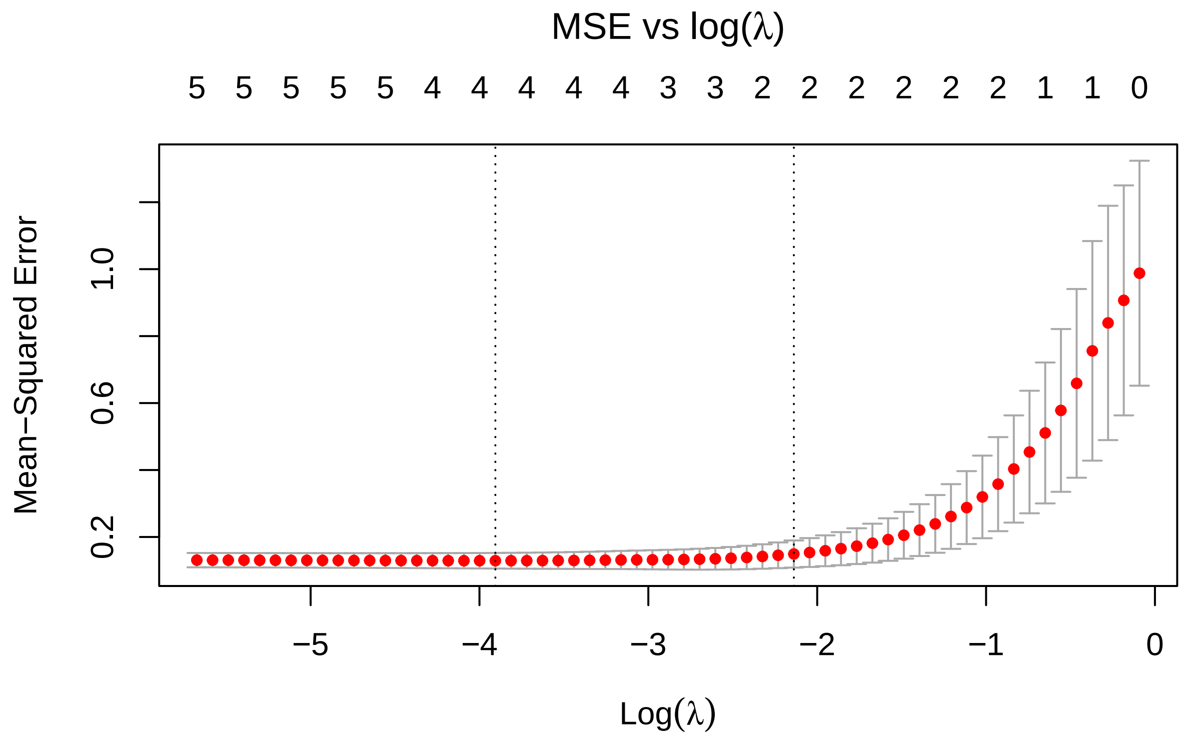


As expected Milk and Detergents_Paper are the best predictors of Grocery. Let's try Lasso regression.

Continuous Lasso Model

```r
# Model
lasso_items = runGlmnet(1, X_train, Y_train, X_test, Y_test, train$Grocery)

# Plotting lambdas
par(mar = c(5,4,6,2))
plot(lasso_items$model,
     main = expression("MSE vs log("*lambda*")"),
     cex.main = 1.2)
```




```r
# Saving Models
saveRDS(
  list(ridge_cont = ridge_items, lasso_cont = lasso_items),
  "regression_cont_models.rds"
)
```

It appears lower levels of lambda correspond to lower MSE, meaning most terms are significant. Let's look at the RMSE and lambda values.

```r
c(lasso_items$rmse_min,
  lasso_items$rmse_1se,
  lasso_items$cv_rmse_min,
  lasso_items$lambda_min,
  lasso_items$lambda_1se)
```

```
## [1] 3937.98711947 4134.72930644 3060.75098960    0.02012337    0.11786298
```

RMSE with min lambda on test data: 3937.987

RMSE with 1se lambda on test data: 4134.729

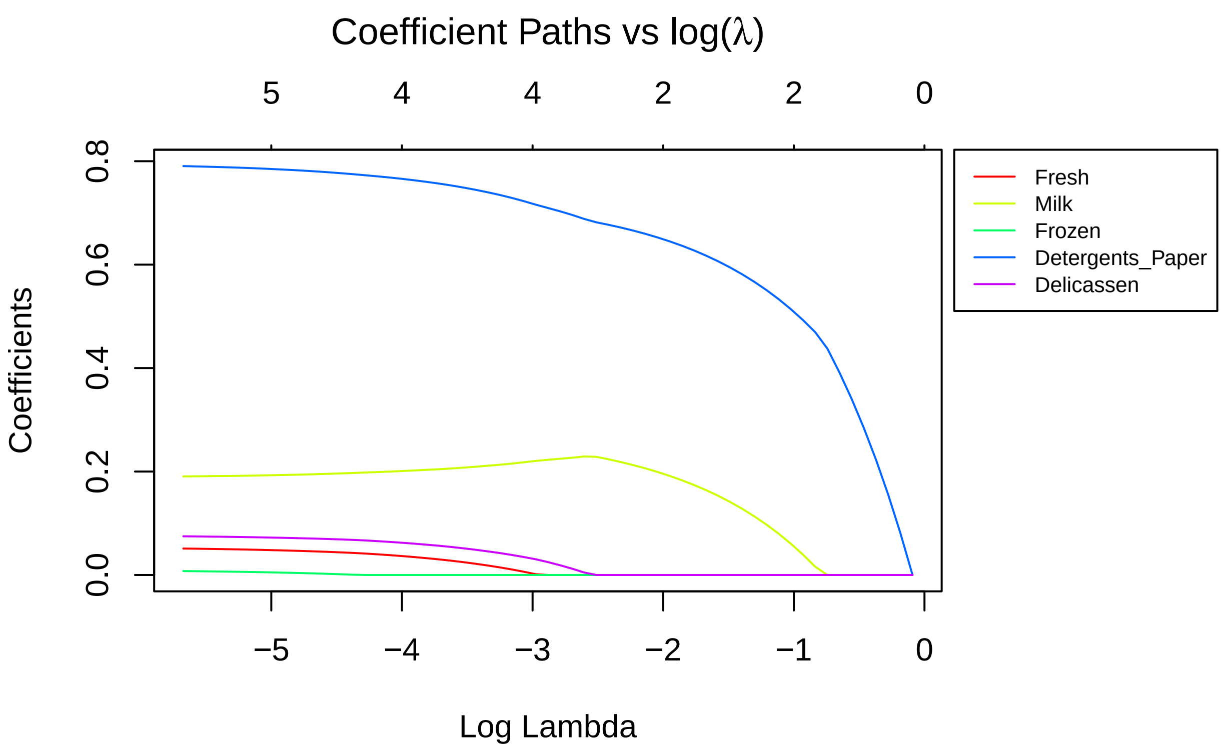
RMSE using cross-validation: 3060.751

```r
# Model
lasso_fit = glmnet(X_train, Y_train, alpha = 1)
```

```r
# Adjusting Plot Margins
par(mar = c(5,4,6,8))

# Plot
plot(lasso_fit, xvar = "lambda", col = rainbow(ncol(X_train)),
     main = expression("Coefficient Paths vs log("*lambda*")"),
     cex.main = 1.2)

legend("topright",
  legend = colnames(X_train),
  col = rainbow(ncol(X_train)),
  lty = 1,
  xpd = TRUE,
  inset = c(-0.35,0),
  cex = 0.7)
```




```r
# Saving Models
saveRDS(
  list(ridge_coefs = ridge_fit, lasso_coefs = lasso_fit),
  "regression_cont_coefs.rds"
)
```

Detergents_Paper is our best model. Let's try ridge regression with our continuous polynomial model.

Continuous Polynomial Ridge Model

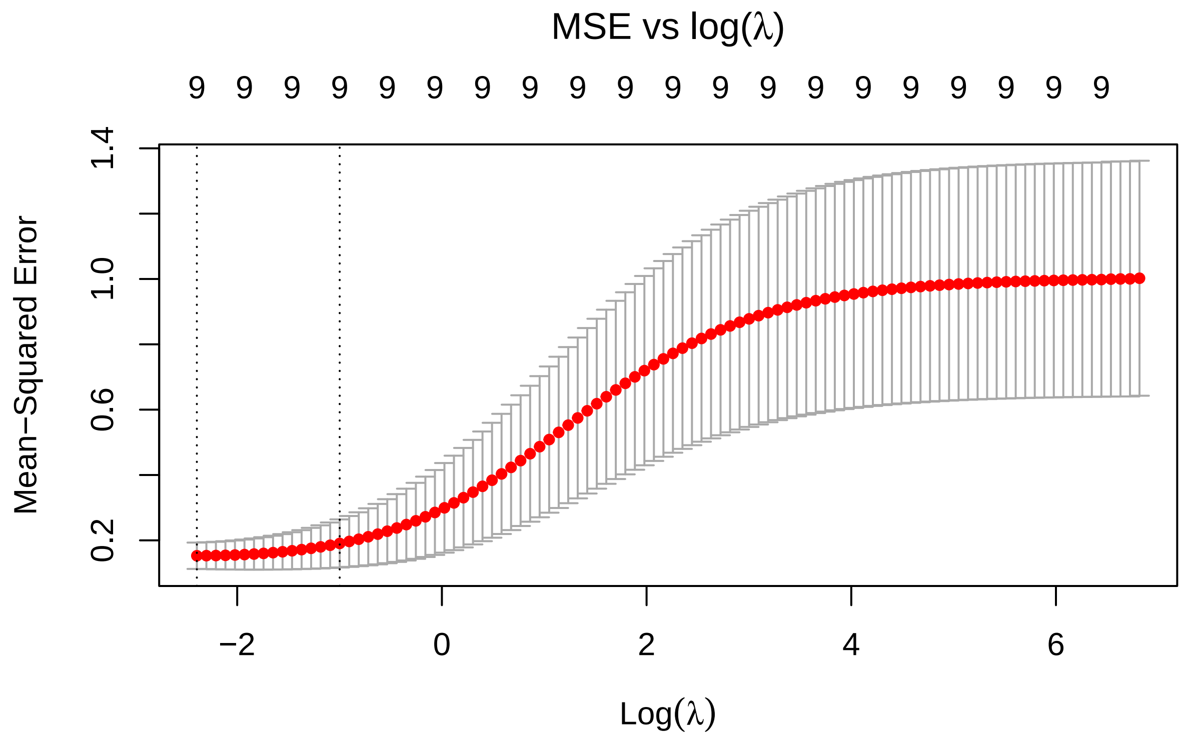
```
# Model
fit = lm(Grocery ~ poly(Fresh, degree = 3) +
                poly(Frozen, degree = 2) +
                poly(Delicassen, degree = 2) +
                Milk +
                Detergents_Paper,
                data = trainZ)

# Training data
X_train = model.matrix(fit)[,-1]
Y_train = trainZ$Grocery

# Testing data
X_test = model.matrix(terms(fit), data = testZ)[, -1]
Y_test = testZ$Grocery

# Running cv on Model
ridge_items = runGlmnet(0, X_train, Y_train, X_test, Y_test, train$Grocery)

# Plotting lambdas
par(mar = c(5,4,6,2))
plot(ridge_items$model,
     main = expression("MSE vs log("*lambda*")"),
     cex.main = 1.2)
```



Model preforms best for log(s) < 0. Let's see the RMSE values.

```
c(ridge_items$rmse_min,
  ridge_items$rmse_1se,
  ridge_items$cv_rmse_min,
  ridge_items$lambda_min,
  ridge_items$lambda_1se)
```

```
## [1] 4164.68975820 4695.67329771 3329.67661646     0.09125703     0.36840619
```

RMSE with min lambda on test data: 4164.690

RMSE with 1se lambda on test data: 4695.673

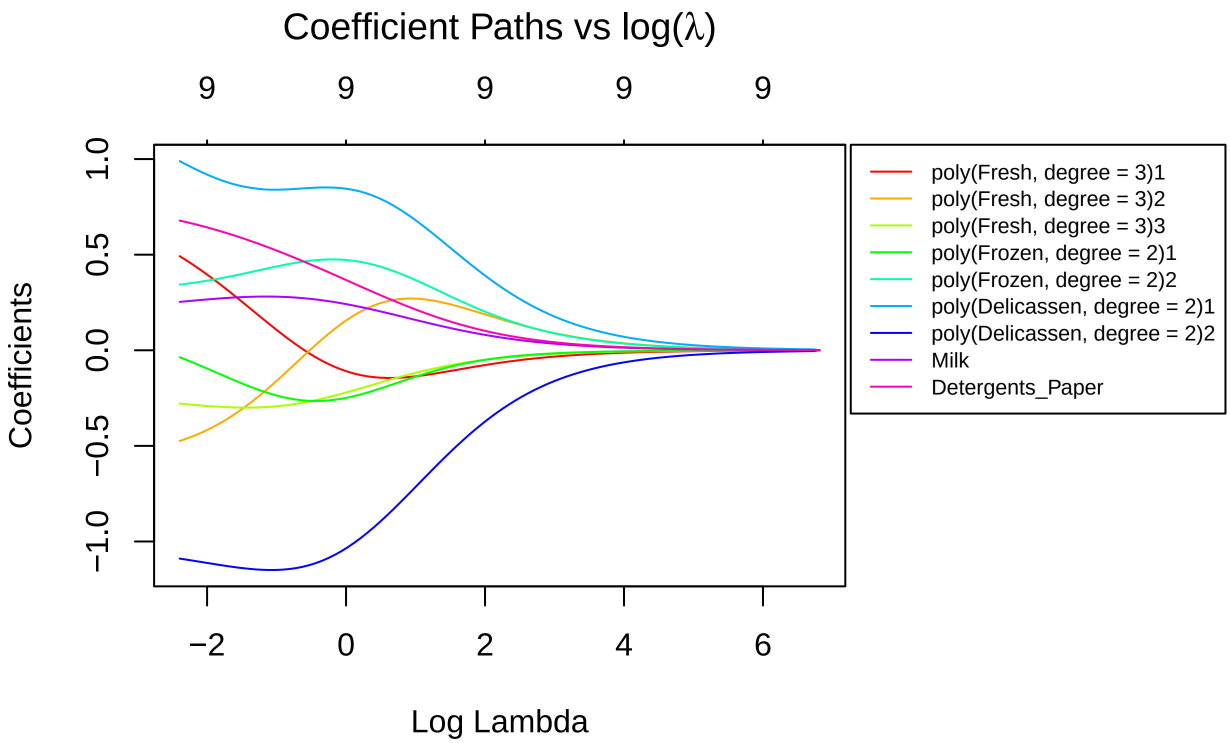
RMSE using cross-validation: 3329.677

```
# Model
fit = glmnet(X_train, Y_train, alpha = 0)

# Adjusting Plot Margins
par(mar = c(5,4,6,10.5))

# Plot
plot(fit, xvar = "lambda", col = rainbow(ncol(X_train)),
     main = expression("Coefficient Paths vs log("*lambda*")"),
     cex.main = 1.2)
```

```r
legend("topright",
  legend = colnames(X_train),
  col = rainbow(ncol(X_train)),
  lty = 1,
  xpd = TRUE,
  inset = c(-0.55,0),
  cex = 0.7)
```

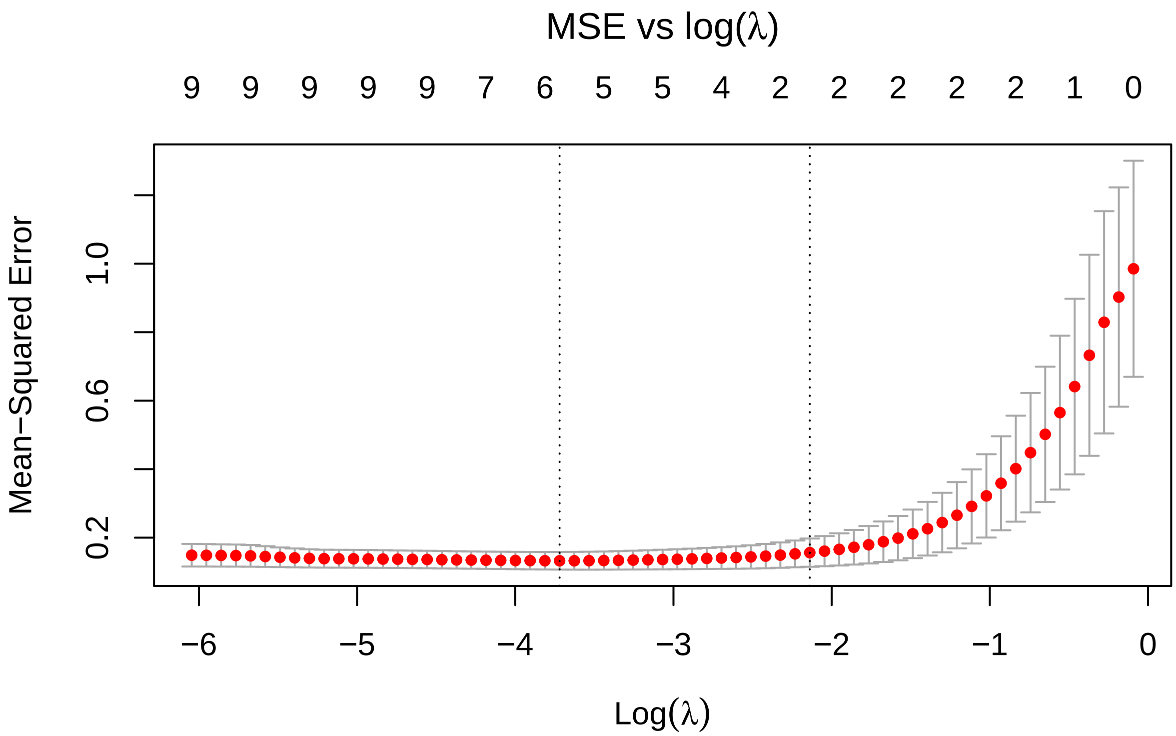


The polynomials terms for Delicassen are the most significant.

Continuous Polynomial Lasso Model

```r
# Model
lasso_items = runGlmnet(1, X_train, Y_train, X_test, Y_test, train$Grocery)

# Plotting lambdas
par(mar = c(5,4,6,2))
plot(lasso_items$model,
     main = expression("MSE vs log("*lambda*")"),
     cex.main = 1.2)
```



It appears lower levels of lambda correspond to lower MSE, meaning most terms are significant. Let's look at the RMSE and lambda values.

```
c(lasso_items$rmse_min,
  lasso_items$rmse_1se,
  lasso_items$cv_rmse_min,
  lasso_items$lambda_min,
  lasso_items$lambda_1se)
```

```
## [1] 3955.92043612 4134.72930644 3100.47734596    0.02423867    0.11786298
```

RMSE with min lambda on test data: 3955.920

RMSE with 1se lambda on test data: 4134.729

RMSE using cross-validation: 3100.477

The model didn't preform great on the great test data but did well in cross validation. Let's check significant coefficients.

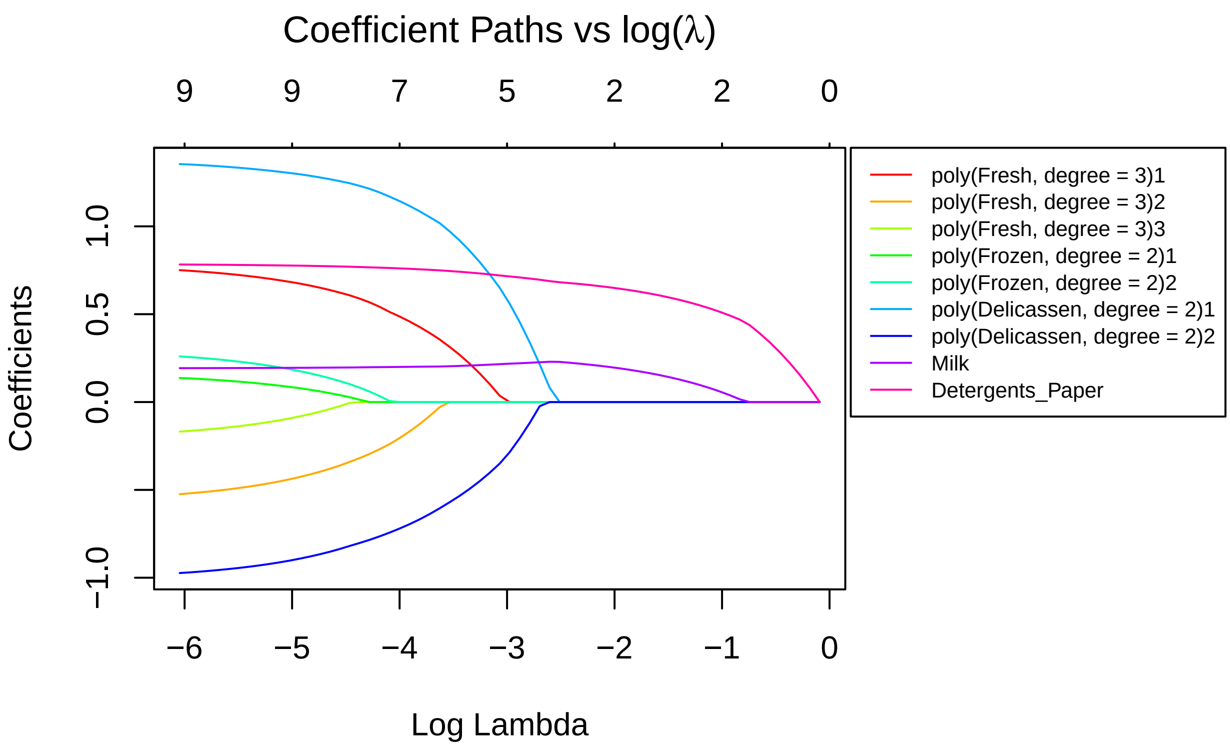
```
# Model
fit = glmnet(X_train, Y_train, alpha = 1)

# Adjusting Plot Margins
par(mar = c(5,4,6,10.5))

# Plot
```

```
plot(fit, xvar = "lambda", col = rainbow(ncol(X_train)),
     main = expression("Coefficient Paths vs log("*lambda*")"),
     cex.main = 1.2)

legend("topright",
  legend = colnames(X_train),
  col = rainbow(ncol(X_train)),
  lty = 1,
  xpd = TRUE,
  inset = c(-0.55,0),
  cex = 0.7)
```



The polynomials terms for Delicassen are the most significant.

Overall Linear Models Conclusion:

```
models <- tribble(
  ~Model, ~`Test RMSE`, ~`CV RMSE`,
  "1. Base (Mean)", 11522, 7945,
  "2. Detergents_Paper Only", 3980, 3500,
  "3. Detergents_Paper + Milk", 4144, 3019,
  "4. All Features", 3944, 3093,
  "5. Continuous Features Only", 3939, 3062,
  "6. Polynomial + Interactions", 4761, 4488,
```

```r
  "7. Polynomial (No Interactions)", 4035, 3540,
  "8. Continuous + Polynomial Terms", 4002, 3975,
  "9. Max Feature Ridge", 5353, 3217,
  "10. Max Feature Lasso", 4444, 3132,
  "11. Continuous Ridge", 4115, 3292,
  "12. Continuous Lasso", 3938, 3061,
  "13. Continuous Polynomial Ridge", 4165, 3330,
  "14. Continuous Polynomial Lasso", 3956, 3100
)

knitr::kable(
  models,
  align = c("l", "r", "r"),
  caption = "Model Performance Comparison",
)
```

Table 1: Model Performance Comparison

Model	Test RMSE	CV RMSE
1. Base (Mean)	11522	7945
2. Detergents_Paper Only	3980	3500
3. Detergents_Paper + Milk	4144	3019
4. All Features	3944	3093
5. Continuous Features Only	3939	3062
6. Polynomial + Interactions	4761	4488
7. Polynomial (No Interactions)	4035	3540
8. Continuous + Polynomial Terms	4002	3975
9. Max Feature Ridge	5353	3217
10. Max Feature Lasso	4444	3132
11. Continuous Ridge	4115	3292
12. Continuous Lasso	3938	3061
13. Continuous Polynomial Ridge	4165	3330
14. Continuous Polynomial Lasso	3956	3100

Summary

Best preforming model: #12 Continuous Lasso (Test RMSE = 3938, CV RMSE = 3061)

Most Stable Model: #8 Continuous + Polynomial Terms ($|\Delta|$RMSE = 27)

Worst Model: #1 Base Mean (Test RMSE = 11522)

Most Overfit Model: #9 Max Feature Ridge ($|\Delta|$RMSE = 2162)

Final Model Selection

We select Model #12 (Continuous Lasso) due to its:

Best RMSE performance

Full intepretability with all retained coefficients

Future-proof design that automatically suppresses irrelevant features if new ones are added

Built-in multicollinearity management critical for our correlated variables