PREDICTING SALES REVENUE

AN ANALYSIS BASED ON STATISTICAL DATA MODELING

GUNNAR MARTIN

JANUARY 8, 2017

MOCK DATA
FOR DEMONSTRATION ONLY

OBJECTIVES
THE GOAL AND THE PROCESS

This statistical analysis is based on a mock data set containing hourly sales revenue for 14 stores across 44 continuous months of data spanning from 2013 to 2017.

The primary objectives of this analysis:

1. Create a model that predicts sales revenue based on certain explanatory variables
2. Identify the top predictors of sales revenue

In order to meet the objectives of this analysis, the following process was executed:

1. Analyze the data set in order to establish the explanatory variables and response variable
2. Choose a statistical software tool and determine which functions to use for creating models
3. Determine and execute an approach for building the best predictive model
4. Use the best predictive model to make a prediction based on certain explanatory variables
5. Use the model to determine the top predictors of sales revenue

VARIABLE ANALYSIS
UNDERSTANDING THE DATA SET

The provided data set included 10 fields of data. The first task was to determine the type of data in each field and its role in building the data model. The fields are as follows:

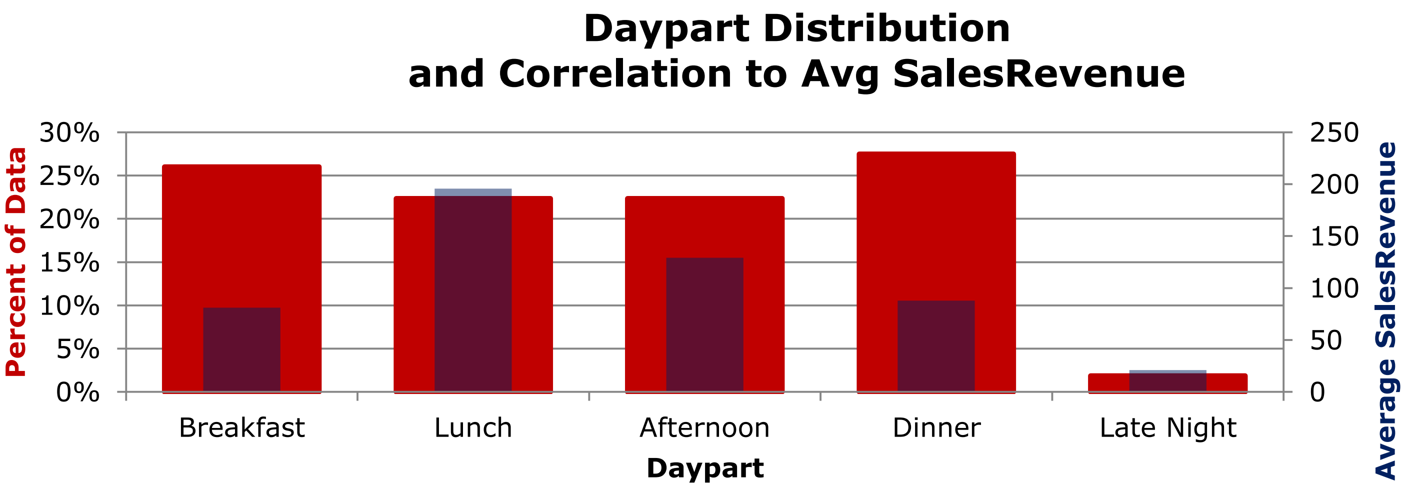
1. **SalesRevenue**: Response variable
   - Data type: Numerical
2. **Daypart**: Explanatory variable
   - Data type: Categorical (5 possible values)
3. **Fiscal_dayofWk**: Explanatory variable
   - Data type: Categorical (7 possible values) or numerical
   - The new variable **DOW_AsChar** was created to use in order for this field to be treated as categorical by the software
4. **HourlyWeather**: Explanatory variable
   - Data type: Categorical (9 possible values)
5. **Hour**: Explanatory variable
   - Data type: Numerical
6. **AvgHourlyTemp**: Explanatory variable
   - Data type: Numerical

VARIABLE ANALYSIS (cont.)
UNDERSTANDING THE DATA SET

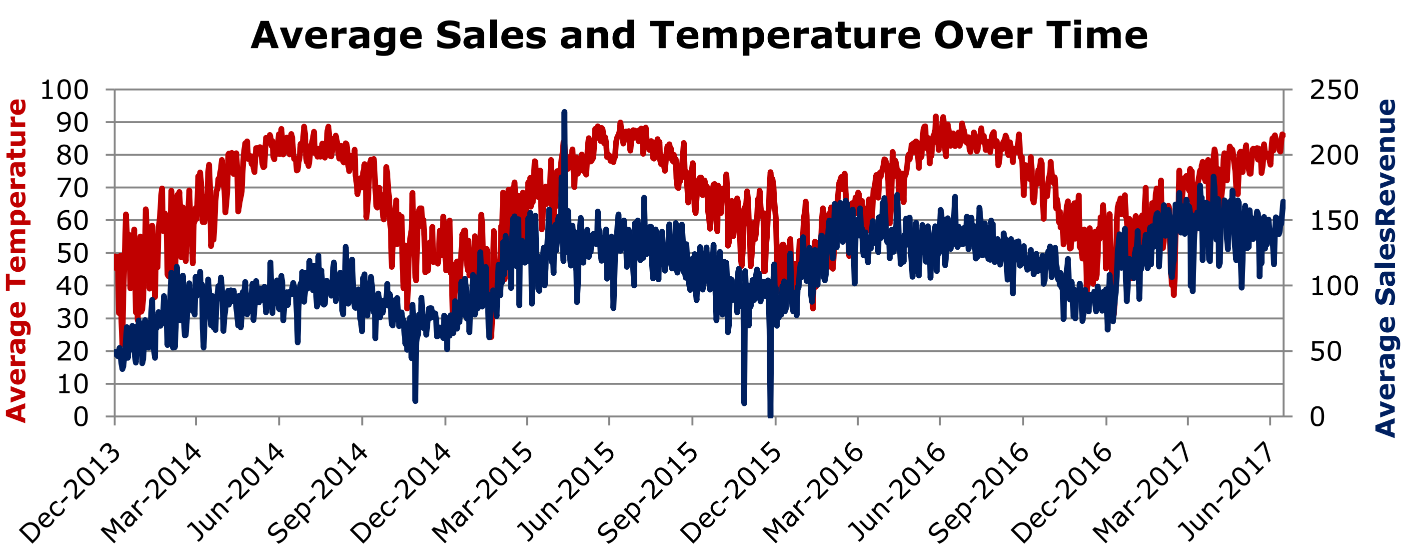
7. **Fiscal_Quarter**: Explanatory variable
   - Data type: Numerical or categorical
   - The new variable **Qtr_AsChar** was created to use in order for this field to be tested as categorical by the software
8. **DateStringYYYYMMDD**: Explanatory variable
   - Data type: Numerical
   - The new variable **Data_AsDate** was created to use, so that this field would be treated as numerical by the software
9. **Store_ID**: Ignored
   - This variable was not available for making predictions, so it was not utilized to help train the model
10. **AvgHourlySales**: Ignored
    - This variable was not available for making predictions, so it was not utilized to help train the model

INITIAL OBSERVATIONS
VISUALIZING THE DATA SET

Before the modeling process began, graphs of the entire data set were rendered in order to aid in understanding the data set and to help guide the modeling process.



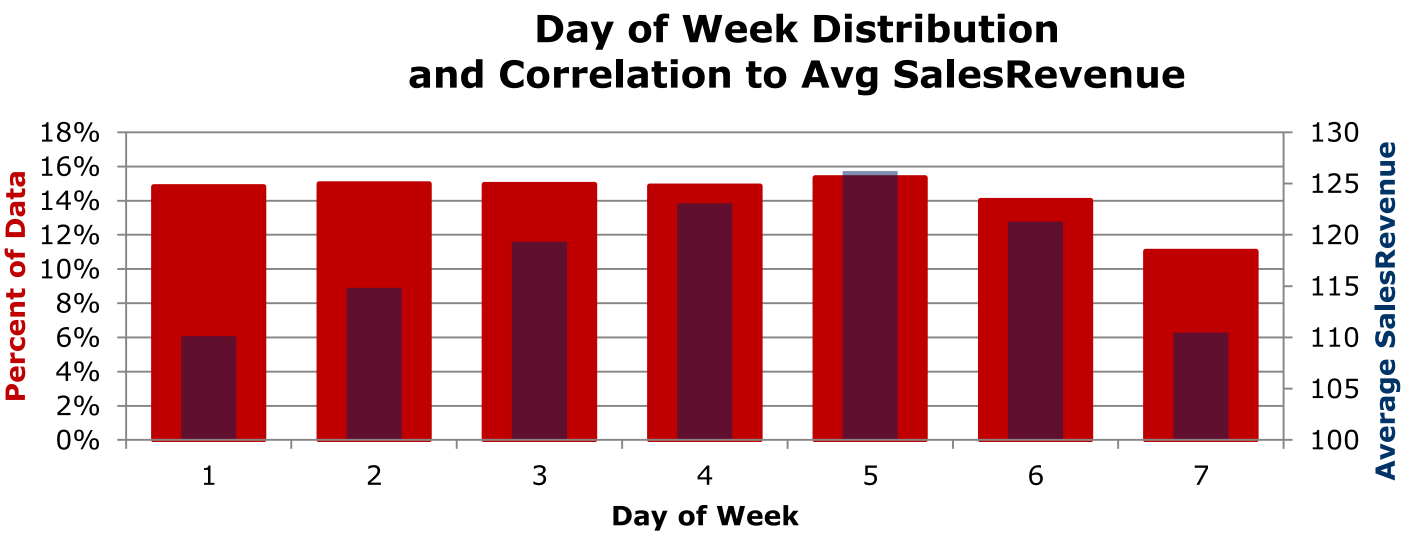
The data for Daypart was relatively well distributed, with the exception of Late Night. Sales were highest around Lunch and in the Afternoon.



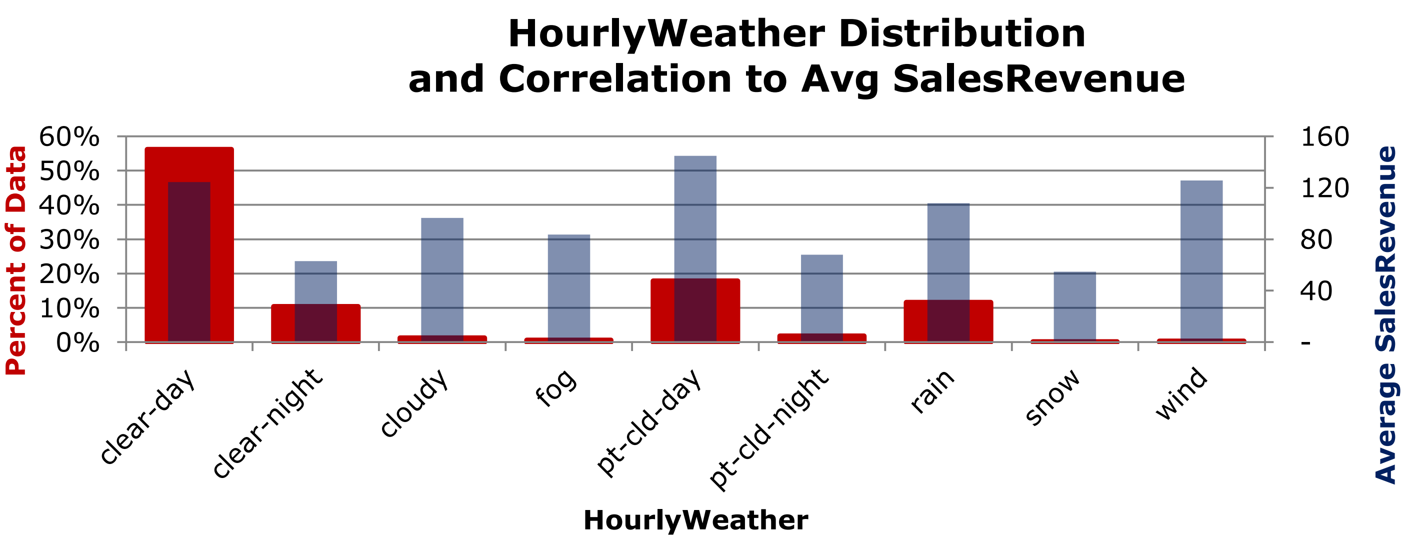
Average SalesRevenue appeared to trend upward over time. The averaged temperature moved predictable with the seasons.

INITIAL OBSERVATIONS (cont.)

VISUALIZING THE DATA SET



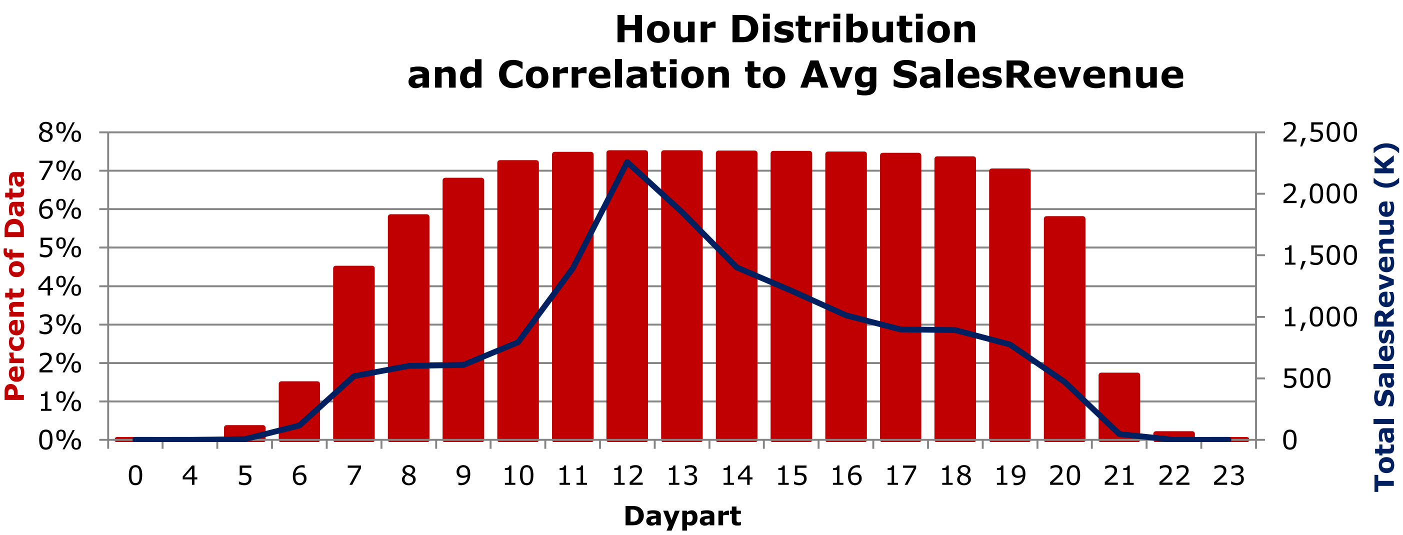
The data for Day of Week was relatively evenly distributed over the week. Sales averaged highest on days 4 through 6 (Thursday, Friday, Saturday).



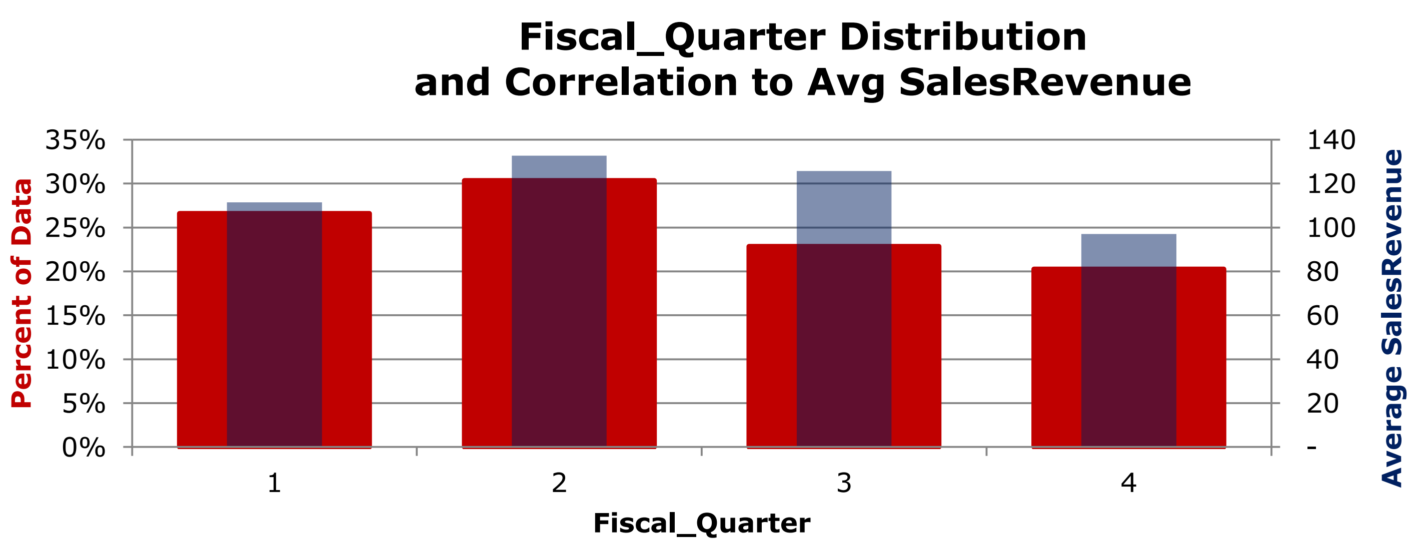
The data HourlyWeather was poorly distributed, with over half the data belonging to clear-day. Sales averaged highest on clear-day, partly cloudy day and wind.

INITIAL OBSERVATIONS (cont.)

VISUALIZING THE DATA SET



The data for Hour was well distributed between 7 AM and 8 PM. Sales averaged highest around midday, which is consistent with the Daypart data.



Fiscal Quarter distribution was relatively even across the quarters and sales peaked in the spring and summer quarters.

APPROACH
MODEL FUNCTIONS IN R

The programming language R was chosen as the statistical software tool to develop the models. Within this language, 2 different functions were utilized: one to evaluate model fit for a linear regression and one for recursive partitioning.

1. Linear Regression
    - The lm() function in R was chosen to evaluate linear models. Since the response variable is a continuous numerical value, linear regression was assumed to be the most appropriate approach.
2. Recursive Partitioning
    - The rpart() function in R was chosen to evaluate models via recursive partitioning. Since recursive partitioning works by constructing decision trees, it is often used in scenarios involving a categorical response variable. It can, however, also be used to in models seeking a continuous numerical response variable.

MODEL COMPARISON
MSE AND K-FOLD CROSS VALIDATION

The models in this analysis were built through repeated process of comparing the accuracy of models against each other. At the heart of this process was k-fold cross validation. The process for this cross validation was executed as follows.

1. Randomly split the entire set of data into a number of subsets (5 were used in this analysis).
2. Each model was first trained on 80% of the data (4 combined subsets , the "training set"), then predicted SalesRevenue for the remaining 20% of the data (the "test set").
3. The predicted SalesRevenue for the test set was subtracted from the actual SalesRevenue for the test set to get the "prediction error".
4. The prediction errors were squared (to assure a positive number), then averaged to get a single value for this test (the "Mean Square Error" or "MSE").
5. The model is then run through 4 more iterations of testing, using each of the other 4 subsets as the test set during each iteration.
6. After all 5 iterations are done, calculate an average of the 5 MSE scores to get a final MSE for the model.

APPROACH
GREEDILY ADDING EXPLANATORY VARIABLES

The algorithm used to build the “best model” was built by adding explanatory variables to the model one-by-one based on their ability to improve the MSE of the current model in its current state. The algorithm ran as follows:

1. Establish a list of explanatory variables.
2. Cycle through the list of explanatory variables and get the MSE for a model using only this explanatory variable.
3. Whichever variable has the lowest MSE becomes the first explanatory variable for the base model.
4. Cycle through the remaining variables and get the MSE for the base model it addition to each remaining variable.
5. The remaining variable that is added that results in the lowest MSE for the model now becomes part of the base model. The algorithm goes back to step 4.
    - If no additional variable is able to lower the MSE from the base model, the algorithm is complete.
    - If all available variables are used up, then the algorithm is complete.

RESULTS
LINEAR AND RECURISVE PARTITION

For the *linear* function, the model created by the algorithm used the following variables to predict **SalesRevenue**:

Daypart, **Date_AsDate**, **AvgHourlyTemp**, **Qtr_AsChar**, **DOW_AsChar**

The model achieved an MSE of **6,890.3** and had an R-squared value of **0.291** against the original set of data.

For the *recursive partition* function, the model created by the algorithm used the following variables to predict **SalesRevenue**:

Hour, **Date_AsDate**

The model completed an MSE of **7,113.8** and had an R-squared value of **0.032** against the original set of data.

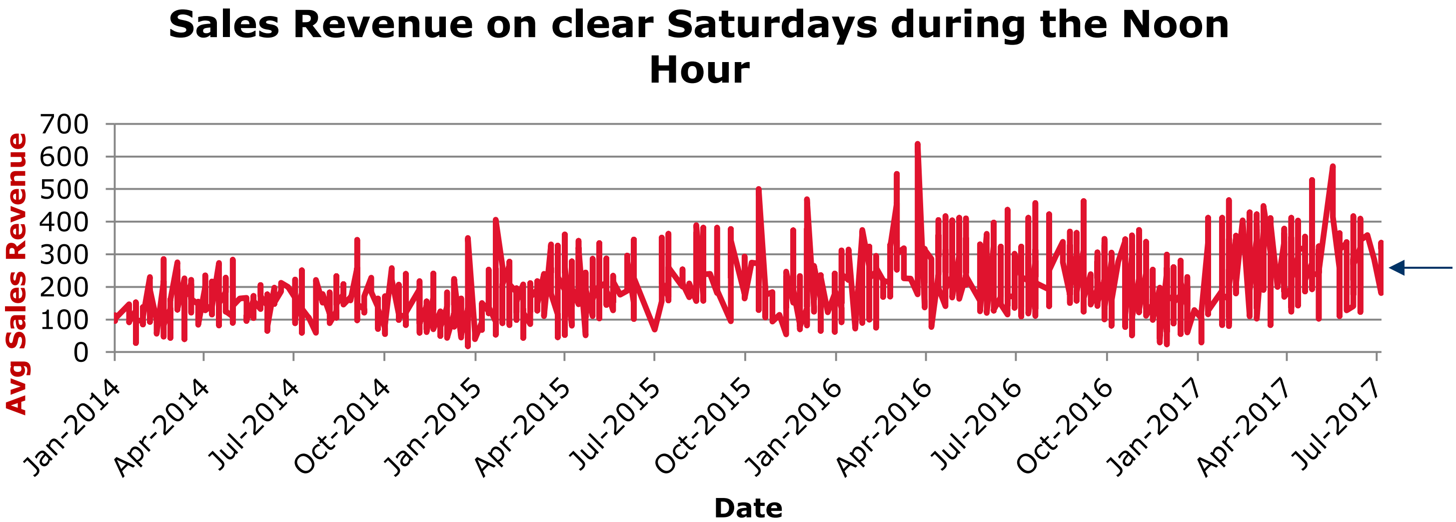
.

PREDICTION
SALES OUTLOOK

Since the best linear model outperformed the replicated partition, the linear model was used to make the prediction using the following input:

- Date: 7/15/2017 (Saturday)
- Hour: 12
- Weather: clear-day
- Temperature: 86.0
- Predicted Hourly SalesRevenue: **233.49**

To check the reasonableness of this prediction, SalesRevenue was graphed over time using only data points that match the conditions above (Hour 12, clear-day, Saturday)



Though this check is not scientific in nature, it does show that a SalesRevenue number of 233.49 is within reasonable range of expectation.

TOP PREDICTORS

STAND-ALONE AND WITHIN THE MODEL

Top predictors are displayed for which variables predict best on their own and which variables were most important to tuning the model.

Stand-Alone Predictors (Linear)

Rank	Formula	MSE
1	SalesRevenue ~ Daypart	7,562.9
2	SalesRevenue ~ I(Hour^2) + Hour	8,316.7
3	SalesRevenue ~ HourlyWeather	9,182.0
4	SalesRevenue ~ AvgHourlyTemp	9,240.2
5	SalesRevenue ~ Date_AsDate	9,453.8
6	SalesRevenue ~ Qtr_AsChar	9,539.0
7	SalesRevenue ~ I(Fiscal_Qtr^2) + Fiscal_Qtr	9,539.1
8	SalesRevenue ~ Hour	9,661.0
9	SalesRevenue ~ DOW_AsChar	9,685.4
10	SalesRevenue ~ Fiscal_Qtr	9,695.9
11	SalesRevenue ~ Fiscal_dayofWk	9,713.5

Tuning Model Predictors (Linear)

Formula	Improved	MSE
SalesRevenue ~ Daypart	n/a	7,562.5
+ Date_AsDate	(291.4)	7,271.1
+ AvgHourlyTemp	(227.3)	7,043.7
+ Qtr_AsChar	(76.9)	6,966.8
+ DOW_AsChar	(76.5)	6,890.3

Stand-Alone Predictors (RPart)

Rank	Formula	MSE
1	SalesRevenue ~ Hour	7,275.4
2	SalesRevenue ~ Daypart	7,640.7
3	SalesRevenue ~ AvgHourlyTemp	9,276.8
4	SalesRevenue ~ HourlyWeather	9,300.3
5	SalesRevenue ~ Date_AsDate	9,412.2
6	SalesRevenue ~ Qtr_AsChar	9,568.3
7	SalesRevenue ~ Fiscal_Qtr	9,603.7
8	SalesRevenue ~ Fiscal_dayofWk	9,717.3
9	SalesRevenue ~ DOW_AsChar	9,717.3

REFLECTION
ADJUSTMENTS AND SURPRISES

After a few iterations of modelling, a few variables were adjusted from a straight linear fit to a polynomial fit do to the shape of their correlation with **SalesRevenue**:

1. **Hour**: This variable was adjusted to be a 2nd order polynomial due to the curve created by most shoppers going to the store around midday. This polynomial achieved a better MSE score than its linear counterpart.
2. **Fiscal_Qtr**: This variable was adjusted to be 2nd order polynomial due to the curve created by the heaviest shopping being done over the spring and summer. This polynomial achieved a better MSE score than its linear counterpart.

A few unexpected finding that could be further analyzed.

1. The top predictor was **Daypart**, which is just a generalized function of **Hour**. Perhaps if the curve for Hour were optimized, it would outperform Daypart as a predictor of what part of the day sales will peak.
2. Day of Week (**DOW_AsChar**) performed curiously poor as a predictor of SalesRevenue. With an even distribution and a clear pattern of better sales near the weekend, perhaps this variable could be adjusted to become more effective.

REFLECTION
BUILDING A BETTER PROCESS

After observing the R-squared value at the model that was created (0.291), it became clear that there was still much room for improvement of this model. Future adjustments to potentially train the model to predict the data better could include:

1. Improve the process for comparing models
   - Simply comparing average MSE scores within the tuning algorithm potentially added explanatory values that improved the MSE in a statistically insignificant manner. Using a more robust comparison, perhaps a t-test of MSEs of the base model vs the comparison model might yield different results.
2. Explore more types of non-linear variables
   - The variables **Hour** and **Fiscal_Qtr** were adjusted to be 2nd order polynomials, which resulted in better predictions based on those variables. More types of non-linear variables could be systematically tested including higher-order polynomials, logarithmic, and exponential functions.
3. Abandon the greedy algorithm
   - The greedy algorithm was designed to most quickly add all of the most predictive explanatory variables one-by-one. It is possible that certain variables in combination with each other could produce a better model, but the one-by-one approach greedy algorithm never gives them a chance to be tested together. Developing algorithm that allows more combinations of explanatory variables could improve the model.

APPENDIX

BEST MODEL SUMMARY

R Console

```
>
>
> generic_model_formula
SalesRevenue ~ Daypart + Date_AsDate + AvgHourlyTemp + Qtr_AsChar +
    DOW_AsChar
> print(generic_model)

Call:
lm(formula = best_model_object$Formula, data = mixdata)

Coefficients:
      (Intercept)  DaypartBreakfast      DaypartDinner  DaypartLate Night       DaypartLunch        Date_AsDate
        -637.5893          -34.8824           -38.9108          -104.2628            71.3289             0.0404
    AvgHourlyTemp       Qtr_AsChar2        Qtr_AsChar3        Qtr_AsChar4        DOW_AsChar2        DOW_AsChar3
           1.2258           -5.1906           -17.8638           -23.4216             4.0771             8.1091
      DOW_AsChar4       DOW_AsChar5        DOW_AsChar6        DOW_AsChar7
          13.0671           19.1704            10.7026           -11.7654

> summary(generic_model)

Call:
lm(formula = best_model_object$Formula, data = mixdata)

Residuals:
   Min      1Q Median      3Q     Max
-889.2  -45.6  -11.1    30.2 3674.7

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)        -6.376e+02  1.087e+01 -58.668  < 2e-16 ***
DaypartBreakfast   -3.488e+01  7.547e-01 -46.220  < 2e-16 ***
DaypartDinner      -3.891e+01  6.706e-01 -58.028  < 2e-16 ***
DaypartLate Night  -1.043e+02  1.808e+00 -57.677  < 2e-16 ***
DaypartLunch        7.133e+01  7.066e-01 100.946  < 2e-16 ***
Date_AsDate         4.040e-02  6.504e-04  62.122  < 2e-16 ***
AvgHourlyTemp       1.226e+00  2.466e-02  49.701  < 2e-16 ***
Qtr_AsChar2        -5.191e+00  8.008e-01  -6.481 9.12e-11 ***
Qtr_AsChar3        -1.786e+01  9.612e-01 -18.584  < 2e-16 ***
Qtr_AsChar4        -2.342e+01  7.144e-01 -32.786  < 2e-16 ***
DOW_AsChar2         4.077e+00  8.574e-01   4.755 1.99e-06 ***
DOW_AsChar3         8.109e+00  8.581e-01   9.450  < 2e-16 ***
DOW_AsChar4         1.307e+01  8.594e-01  15.206  < 2e-16 ***
DOW_AsChar5         1.917e+01  8.534e-01  22.463  < 2e-16 ***
DOW_AsChar6         1.070e+01  8.730e-01  12.259  < 2e-16 ***
DOW_AsChar7        -1.177e+01  9.337e-01 -12.601  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 83 on 125776 degrees of freedom
Multiple R-squared:  0.2911,    Adjusted R-squared:  0.291
F-statistic:  3443 on 15 and 125776 DF,  p-value: < 2.2e-16

>
>
```

APPENDIX
R SCRIPT FILE

```
# --------------------------------------------------------------------------------
#
#    Interview Code
#    Sales Revenue Prediction
#
#    Author: G. Martin
#    Date: 8JAN2017
#
# --------------------------------------------------------------------------------

# ---------- Setup ----------

#load rpart library
library(rpart)

#choosed model
modelChoice = c("Linear","Rpart","GLM")
modelType = modelChoice[1]

#set limit on iterations of tuning algorythm
iter_limit = 100
iter_counter = 0

# ---------- Load and Transform data ----------

# load data
filename="C:/Users/874700/xArchive/20180105/Data.csv"
rawdata <- read.csv(file=filename)

# add Date_AsDate
x = substr(rawdata$DateStringYYYYMMDD,1,4)
x = paste(x, "-", sep = "")
x = paste(x, substr(rawdata$DateStringYYYYMMDD,5,6), sep = "")
x = paste(x, "-", sep = "")
x = paste(x, substr(rawdata$DateStringYYYYMMDD,7,8), sep = "")
rawdata$Date_AsDate = as.Date(x)

# add DOW_AsChar
rawdata$DOW_AsChar = as.character(rawdata$Fiscal_dayofWk)

# add Qtr_AsChar
rawdata$Qtr_AsChar = as.character(rawdata$Fiscal_Qtr)

```

APPENDIX

R SCRIPT FILE (cont.)

```r

# --- Assign k-fold subsamples ---

# randomly mix up the data
mixdata=rawdata[sample(nrow(rawdata),nrow(rawdata)),]

# create folds vector (for 5 folds)
fold_count = 5
fold = c(0:(nrow(rawdata)-1))
fold = (fold %% fold_count) + 1

# assign folds to mixdata dataframe
mixdata$fold <- fold


# ---------- polyStr_function ----------
polyStr = function(VarName, Order){

    #Order must be at least 2
    if(Order < 2){return(VarName)}

    #establish string to build
    retStr = ""

    #loop thru from Order, creating a polynomial string
    for(i in Order:1){
        if(i == Order){
            retStr = paste("I(", VarName, "^", i, ")", sep = "")
        } else if(i ==1) {
            retStr = paste(retStr, " + ", VarName, sep = "")
        } else {
            retStr = paste(retStr, " + I(", VarName, "^", i, ")", sep = "")
        }
    }

    #return string
    return(retStr)
}

```

APPENDIX
R SCRIPT FILE (cont.)

```
# ---------- testing function ----------
test_this_formula = function(test_formula, type){

    #establish MSE vector
    vec_MSE = vector()

    #run thru each test set
    for(i in 1:fold_count){

        #establish data sets
        train_data = subset(mixdata, fold!=i) #get train data
        test_data = subset(mixdata, fold==i) #get test data

        #establish model
        if(type == "Linear"){
            model <- lm(test_formula, data = train_data)
        } else if(type == "Rpart"){
            model <- rpart(test_formula, data = train_data)
        } else if(type == "GLM"){
            model <- glm(test_formula, data = train_data)
        } else {
            #invalid function
            return(0)
        }

        #run prediction
        prediction <- predict(model, newdata = test_data)

        #get MSE
        vec_MSE[i] = mean((test_data$SalesRevenue - prediction)^2)
    }

    #print out formula and MSE
    x = as.character(format(test_formula))
    print(paste(type, "|", x, "|", mean(vec_MSE)))

    #return list object
    ret_list = list(mean(vec_MSE), test_formula, model)
    names(ret_list) = c("MSE","Formula","Model")
    return(ret_list)
}
```

APPENDIX
R SCRIPT FILE (cont.)

```
# ---------- Greedy algorithm ----------

#establish the response variable
rv = "SalesRevenue"

#establish a vector of vnames
vname = vector()
vname[length(vname)+1] = polyStr("Fiscal_Qtr",2)
vname[length(vname)+1] = "Fiscal_Qtr"
vname[length(vname)+1] = "Qtr_AsChar"
vname[length(vname)+1] = "Fiscal_dayofWk"
vname[length(vname)+1] = "DOW_AsChar"
vname[length(vname)+1] = "Daypart"
vname[length(vname)+1] = "HourlyWeather"
vname[length(vname)+1] = "Hour"
vname[length(vname)+1] = polyStr("Hour",2)
vname[length(vname)+1] = "AvgHourlyTemp"
vname[length(vname)+1] = "Date_AsDate"

#create a data frame with these names
ev = data.frame(idx = 1:length(vname), vname)

#establish baseline best model object
best_model_object = list(MSE = 10000000)

```

APPENDIX

R SCRIPT FILE (cont.)

```
# run algorithm
while(nrow(ev)>0){

    #increment iteration counter
    iter_counter = iter_counter+ 1

    #establish base formula
    if(is.null(best_model_object$Formula)){

        #if best model DNE yet
        base_frm_str = "SalesRevenue ~ "
    } else {

        #grab beginning of base model
        base_frm_str = as.character(format(best_model_object$Formula))
        base_frm_str = paste(base_frm_str, " + ")
    }

    #reset added_var_idx
    added_var_idx = 0

    #loop through remaining explanatory variables
    for(i in 1:nrow(ev)){

        #build formula
        new_ev = as.character(ev$vname[i])
        formula_str = paste(base_frm_str, new_ev)
        this_formula = as.formula(formula_str)

        #test this formula and store results
        this_model_object = test_this_formula(this_formula, modelType)

        #compare against the best
        if(this_model_object$MSE < best_model_object$MSE){

            best_model_object = this_model_object
            added_var_idx = ev$idx[i]
        }
    }

```

APPENDIX

R SCRIPT FILE (cont.)

```r

    #check results of this iteration
    if(added_var_idx == 0 || iter_counter == iter_limit){

        #stop running if no improvement by clearing the frame
        ev = data.frame()
    } else {

        #remove the used variable
        ev = subset(ev, ev$idx != added_var_idx)
        print(ev)
    }
}

# ---------- make generic prediction ----------

#copy best model
generic_model = lm(best_model_object$Formula, data = mixdata)

# build generic data frame
Fiscal_Qtr=3
Qtr_AsChar="3"
Date_AsDate=as.Date("2017-07-15")
Fiscal_dayofWk=6
DOW_AsChar="6"
Daypart="Lunch"
HourlyWeather="clear-day"
Hour=12
AvgHourlyTemp=86
generic_data = data.frame(Fiscal_Qtr,Date_AsDate,Fiscal_dayofWk,Daypart,HourlyWeather,Hour,AvgHourlyTemp)

#make prediction
my_pred = predict(generic_model, newdata = generic_data)

#print prediction, model, and summary
print(my_pred)
print(generic_model)
summary(generic_model)
```