Part 1 regression

The dataset used can be found here: https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset (https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset)

```
library(e1071)

bikeData <- read.csv("hour.csv", header=TRUE)

bikeData$season <- as.factor(bikeData$season)
bikeData$yr <- as.factor(bikeData$yr)
bikeData$weathersit <- as.factor(bikeData$weathersit)
bikeData$weekday <- as.factor(bikeData$weekday)
bikeData$workingday <- as.factor(bikeData$workingday)
bikeData$mnth <- as.factor(bikeData$mnth)
bikeData$hr <- as.factor(bikeData$hr)

#Remove bad columns
bikeData <- bikeData[,-16]
bikeData <- bikeData[,-15]
bikeData <- bikeData[,-2]
bikeData <- bikeData[,-1]

sapply(bikeData, function(x) sum(is.na(x)))
```

```
##     season         yr       mnth         hr    holiday    weekday workingday
##          0          0          0          0          0          0          0
## weathersit       temp      atemp        hum  windspeed        cnt
##          0          0          0          0          0          0
```

```
bikeData <- bikeData[(complete.cases(bikeData)),]
sum(is.na(bikeData))
```

```
## [1] 0
```

```
set.seed(12345)

spec <- c(train=.6, test=.2, validate=.2)
i <- sample(cut(1:nrow(bikeData), nrow(bikeData)*cumsum(c(0,spec)), labels=names(spec)))
train <- bikeData[i=="train",]
test <- bikeData[i=="test",]
vald <- bikeData[i=="validate",]


tempCor <- cor(train$temp, train$cnt)
print(paste("Correlation between temp and cnt: ", tempCor))
```

```
## [1] "Correlation between temp and cnt:  0.402093970252169"
```

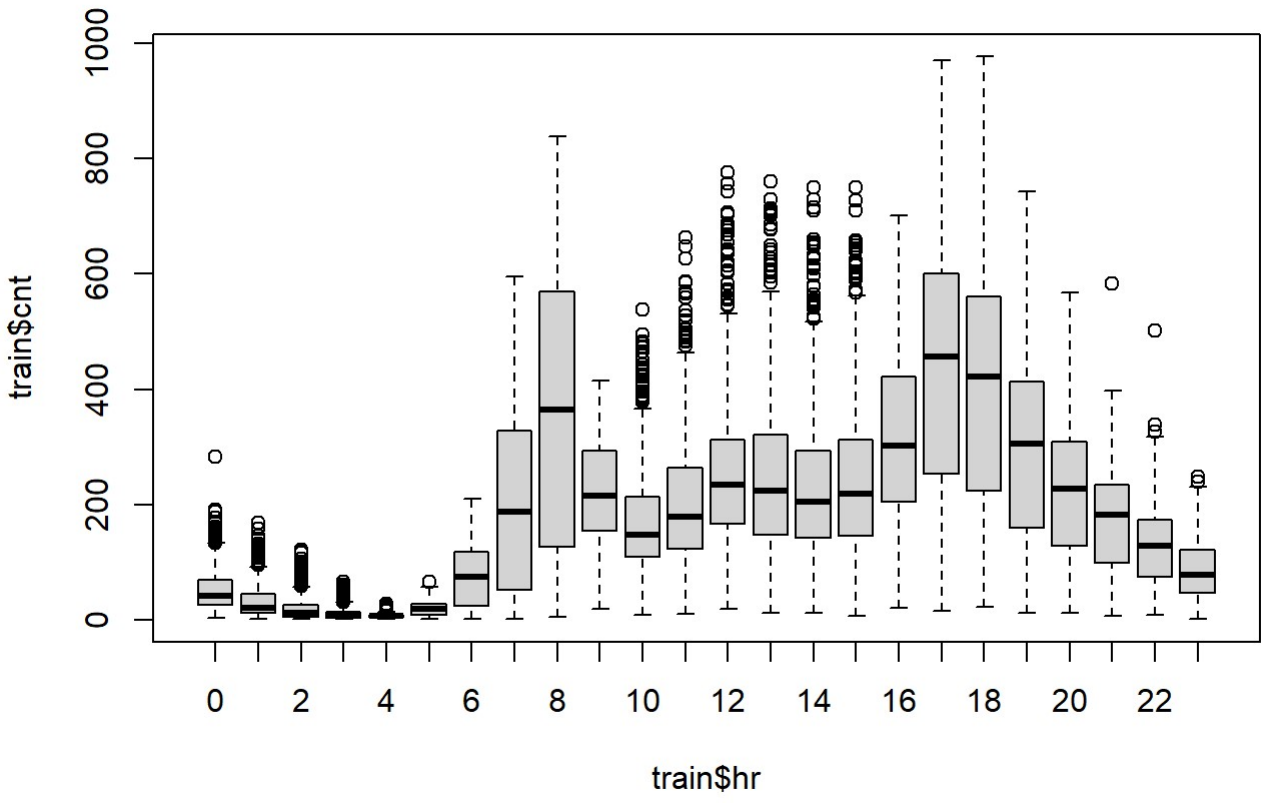
```
atempCor <- cor(train$atemp, train$cnt)
print(paste("Correlation between atemp and cnt: ", atempCor))
```

```
## [1] "Correlation between atemp and cnt:  0.397495698398799"
```

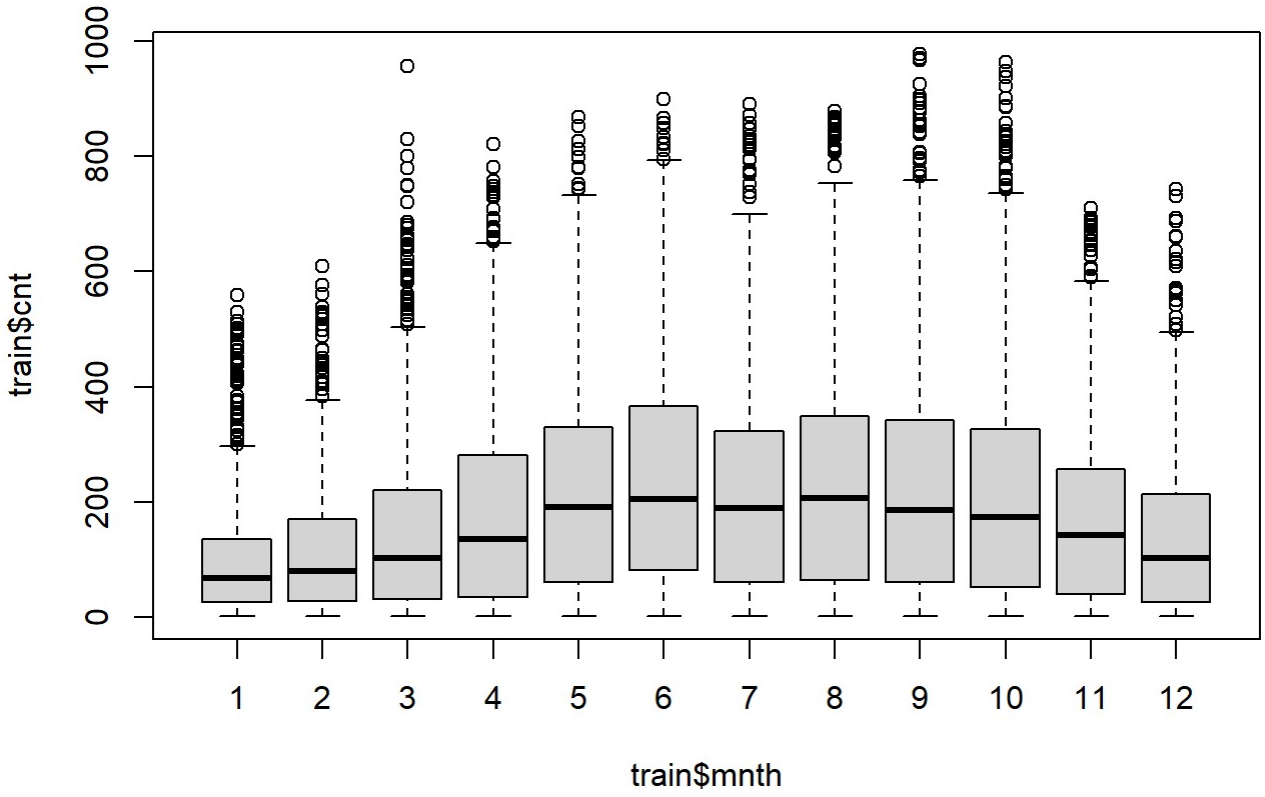
```
humCor <- cor(train$hum, train$cnt)
print(paste("Correlation between humidity and cnt: ", humCor))
```

```
## [1] "Correlation between humidity and cnt:  -0.324625012015373"
```

```
boxplot(train$cnt ~ train$hr)
```




```
boxplot(train$cnt ~ train$mnth)
```




```
svm1 <- svm(cnt~., data=train, kernel="linear", cost=10, scale=TRUE)
summary(svm1)
```

```
##
## Call:
## svm(formula = cnt ~ ., data = train, kernel = "linear", cost = 10,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  10
##       gamma:  0.01851852
##     epsilon:  0.1
##
##
## Number of Support Vectors:  8080
```

```
pred1 <- predict(svm1, newdata=test)
cor_svm1 <- cor(pred1, test$cnt)
mse_svm1 <- mean((pred1 - test$cnt)^2)

print(paste("Correlation: ", cor_svm1))
```

```
## [1] "Correlation:  0.823787898562287"
```

```
print(paste("MSE: ", mse_svm1))
```

```
## [1] "MSE:  10918.6308665753"
```

The linear kernel has reasonable correlation to the results but if we tune the cost value we could improve it.

```
tune_svm1 <- tune(svm, cnt~., data=vald, kernel="linear",
                  ranges=list(cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100)))

summary(tune_svm1)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   100
##
## - best performance: 11390.68
##
## - Detailed performance results:
##    cost    error dispersion
## 1 1e-03 22668.45   3013.964
## 2 1e-02 15009.88   2429.181
## 3 1e-01 11719.47   1740.505
## 4 1e+00 11393.62   1573.334
## 5 5e+00 11392.30   1558.273
## 6 1e+01 11392.00   1558.573
## 7 1e+02 11390.68   1558.787
```

```
pred_tune1 <- predict(tune_svm1$best.model, newdata = test)
cor_svm1_tune <- cor(pred_tune1, test$cnt)
mse_svm1_tune <- mean((pred_tune1 - test$cnt)^2)

print(paste("Correlation: ", cor_svm1_tune))
```

```
## [1] "Correlation:  0.822371996745596"
```

```
print(paste("MSE: ", mse_svm1_tune))
```

```
## [1] "MSE:  10861.8505375762"
```

After tuning the linear kernel the correlation is slightly worse but the MSE has improved.

```
svm2 <- svm(cnt~., data=train, kernel="polynomial", cost=10, scale=TRUE)
summary(svm2)
```

```
##
## Call:
## svm(formula = cnt ~ ., data = train, kernel = "polynomial", cost = 10,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  polynomial
##        cost:  10
##      degree:  3
##       gamma:  0.01851852
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  8031
```

```
pred2 <- predict(svm2, newdata=test)
cor_svm2 <- cor(pred2, test$cnt)
mse_svm2 <- mean((pred2 - test$cnt)^2)

print(paste("Correlation: ", cor_svm2))
```

```
## [1] "Correlation:  0.829751752380455"
```

```
print(paste("MSE: ", mse_svm2))
```

```
## [1] "MSE:  13061.7169752144"
```

The correlation for the polynomial kernel is better but the MSE is not quite as good as the linear kernel. However, the run time for this kernel was considerably faster than the last two. This may be because a polynomial line to divide the data is easier to calculate than a linear one.

```
svm3 <- svm(cnt~., data=train, kernel="radial", cost=10, gamma=1, scale=TRUE)
summary(svm3)
```

```
##
## Call:
## svm(formula = cnt ~ ., data = train, kernel = "radial", cost = 10,
##     gamma = 1, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  8944
```

```
pred3 <- predict(svm3, newdata=test)
cor_svm3 <- cor(pred3, test$cnt)
mse_svm3 <- mean((pred3 - test$cnt)^2)

print(paste("Correlation: ", cor_svm3))
```

```
## [1] "Correlation:  0.838611028703094"
```

```
print(paste("MSE: ", mse_svm3))
```

```
## [1] "MSE:  12701.0170876902"
```

The correlation for the radial kernel is better than all of the previous kernels. However the MSE is worse than the linear kernel.

```
set.seed(12345)
tune.out <- tune(svm, cnt~., data=vald, kernel="radial",
                 ranges=list(cost=c(0.1,1,10,100,1000),
                             gamma=c(0.5,1,2,3,4)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10   0.5
##
## - best performance: 9950.1
##
## - Detailed performance results:
##     cost gamma    error dispersion
## 1  1e-01   0.5 23657.16  3794.2530
## 2  1e+00   0.5 11444.75  1414.3853
## 3  1e+01   0.5  9950.10   967.6471
## 4  1e+02   0.5  9950.10   967.6471
## 5  1e+03   0.5  9950.10   967.6471
## 6  1e-01   1.0 33216.37  5093.5793
## 7  1e+00   1.0 23751.01  3576.9095
## 8  1e+01   1.0 21412.52  2722.9528
## 9  1e+02   1.0 21412.52  2722.9528
## 10 1e+03   1.0 21412.52  2722.9528
## 11 1e-01   2.0 35199.04  5301.0388
## 12 1e+00   2.0 32188.41  4793.1881
## 13 1e+01   2.0 31108.28  4109.0076
## 14 1e+02   2.0 31108.28  4109.0076
## 15 1e+03   2.0 31108.28  4109.0076
## 16 1e-01   3.0 35341.41  5313.9921
## 17 1e+00   3.0 33205.31  4918.3647
## 18 1e+01   3.0 32463.41  4335.0312
## 19 1e+02   3.0 32463.41  4335.0312
## 20 1e+03   3.0 32463.41  4335.0312
## 21 1e-01   4.0 35375.07  5316.3778
## 22 1e+00   4.0 33447.47  4940.7497
## 23 1e+01   4.0 32779.96  4387.2694
## 24 1e+02   4.0 32779.96  4387.2694
## 25 1e+03   4.0 32779.96  4387.2694
```

```
svm4 <- svm(cnt~., data=train, kernel="radial", cost=100, gamma=0.5, scale=TRUE)
summary(svm4)
```

```
##
## Call:
## svm(formula = cnt ~ ., data = train, kernel = "radial", cost = 100,
##     gamma = 0.5, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  100
##       gamma:  0.5
##     epsilon:  0.1
##
##
## Number of Support Vectors:  7339
```

```
pred4 <- predict(svm4, newdata=test)
cor_svm4 <- cor(pred4, test$cnt)
mse_svm4 <- mean((pred4 - test$cnt)^2)

print(paste("Correlation: ", cor_svm4))
```

```
## [1] "Correlation:  0.933872986504449"
```

```
print(paste("MSE: ", mse_svm4))
```

```
## [1] "MSE:  4619.89171039394"
```

After tuning the radial kernel we reach a correlation that is substantially better than anything that came before. The only problem is the considerable amount of time it took to run the algorithim, far longer than any of the other kernels.