SVM Regression

By Spencer Gray & Michael Stinnett

2022-10-23

Data Source: Austin, TX House Listings (https://www.kaggle.com/datasets/ericpierce/austinhousingprices)

Load Packages

```
library(e1071)
```

Load Data

```
df <- read.csv("austinHousingData.csv")
```

Clean Data

Ran linear regression once on the data prior to see what columns are important predictors.

```
df <- df[, c(4, 10, 12, 14, 16, 18, 19, 20, 23, 28, 31, 35, 39, 40, 42, 44, 45, 46)]
df[sapply(df, is.character)] <- lapply(df[sapply(df, is.character)],
                                       as.factor)
df$zipcode <- as.factor(df$zipcode)
```

Divide into train/test/validate

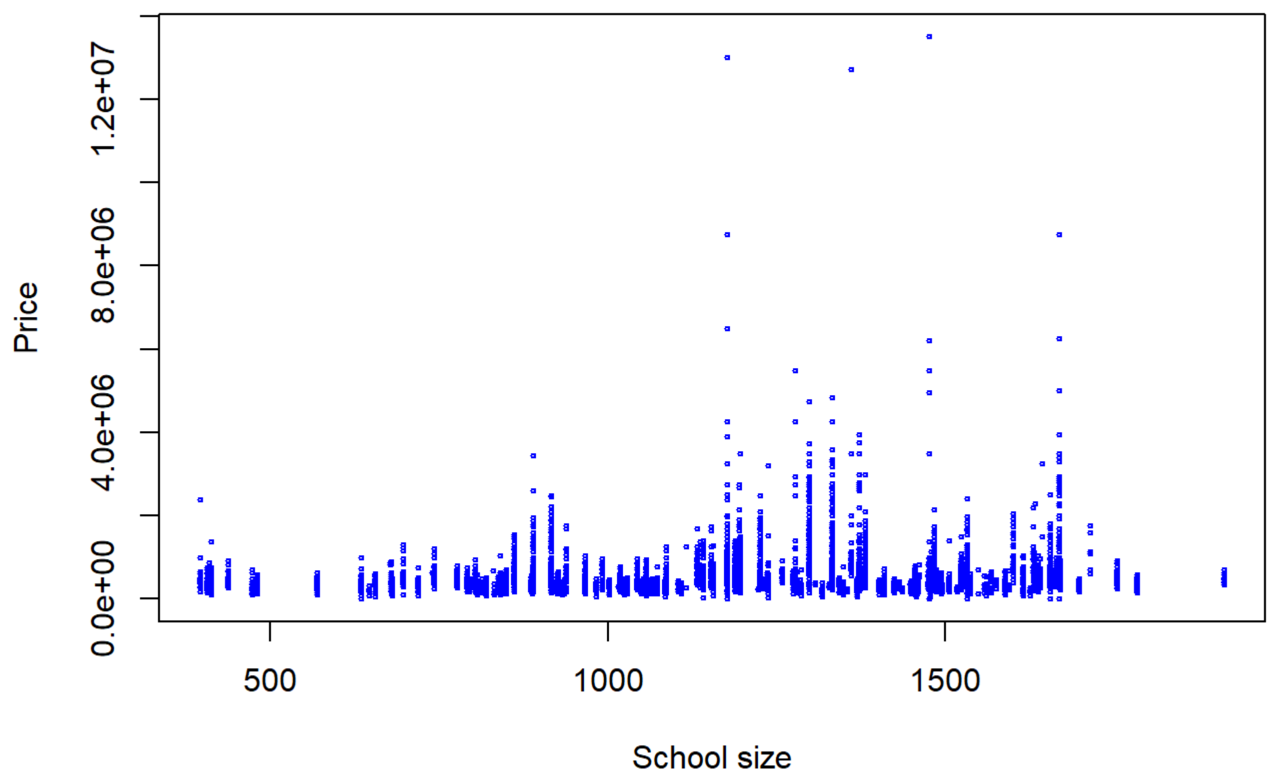
```
set.seed(1234)
spec <- c(train=.6, test=.2, validate=.2)
i <- sample(cut(1:nrow(df),
                nrow(df)*cumsum(c(0,spec)), labels=names(spec)))
train <- df[i=="train",]
test <- df[i=="test",]
vald <- df[i=="validate",]
```

Data Exploration

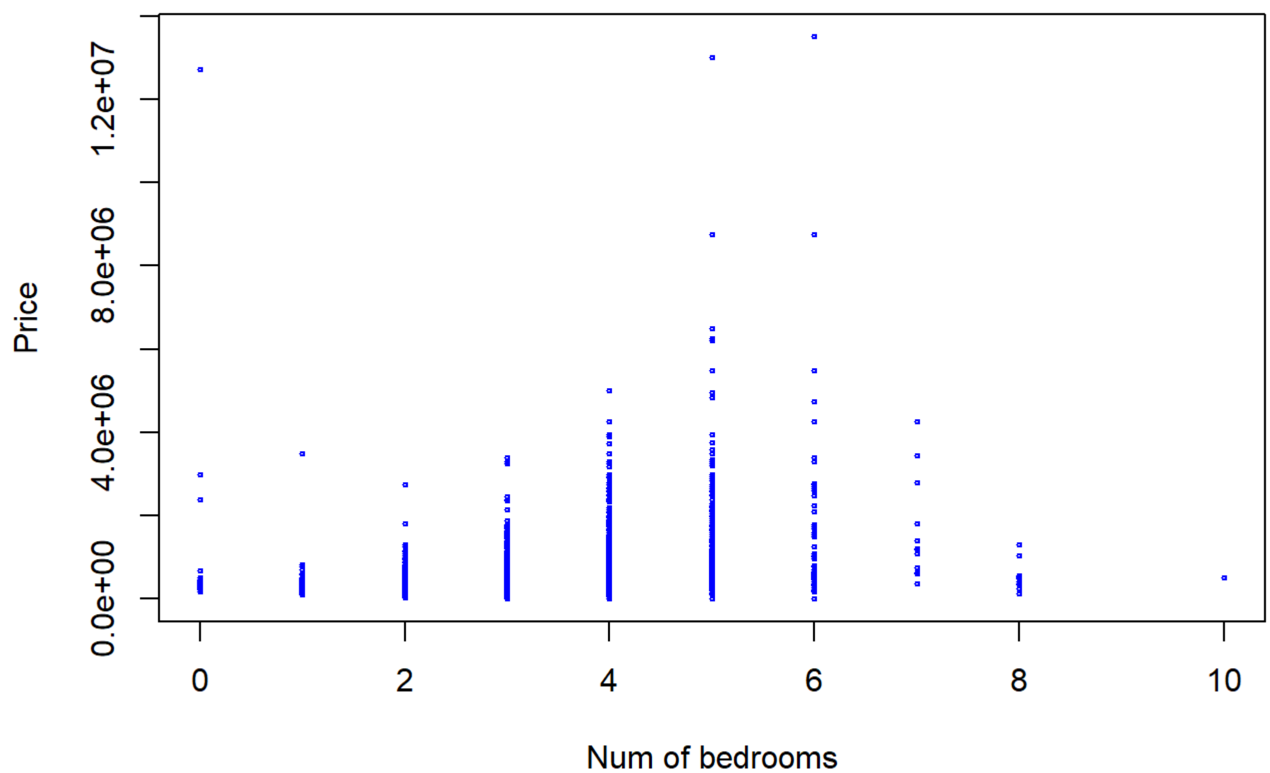
```
summary(train)
```

```
##     zipcode     hasAssociation hasGarage      hasSpa
##  78748  : 710   False:4358     False:4085   False:8380
##  78745  : 630   True :4744     True :5017   True : 722
##  78749  : 452
##  78704  : 431
##  78739  : 380
##  78737  : 366
##  (Other):6133
##               homeType      yearBuilt     latestPrice       numPriceChanges
##  Single Family     :8523   Min.   :1906   Min.   :    5500   Min.   : 1.000
##  Condo             : 305   1st Qu.:1974   1st Qu.:  305000   1st Qu.: 1.000
##  Townhouse         :  99   Median :1993   Median :  400000   Median : 2.000
##  Multiple Occupancy:  64   Mean   :1988   Mean   :  517977   Mean   : 3.051
##  Vacant Land       :  48   3rd Qu.:2006   3rd Qu.:  575000   3rd Qu.: 4.000
##  Apartment         :  23   Max.   :2020   Max.   :13500000   Max.   :23.000
##  (Other)           :  40
##  latest_saleyear numOfParkingFeatures numOfWaterfrontFeatures livingAreaSqFt
##  Min.   :2018    Min.   :0.000        Min.   :0.000000        Min.   :   306
##  1st Qu.:2018    1st Qu.:1.000        1st Qu.:0.000000        1st Qu.:  1485
##  Median :2019    Median :2.000        Median :0.000000        Median :  1974
##  Mean   :2019    Mean   :1.717        Mean   :0.003186        Mean   :  2217
##  3rd Qu.:2020    3rd Qu.:2.000        3rd Qu.:0.000000        3rd Qu.:  2674
##  Max.   :2021    Max.   :6.000        Max.   :2.000000        Max.   :109292
##
##  numOfHighSchools avgSchoolDistance avgSchoolSize  numOfBathrooms
##  Min.   :0.0000   Min.   :0.200     Min.   : 396   Min.   : 0.000
##  1st Qu.:1.0000   1st Qu.:1.100     1st Qu.: 983   1st Qu.: 2.000
##  Median :1.0000   Median :1.533     Median :1281   Median : 3.000
##  Mean   :0.9793   Mean   :1.836     Mean   :1237   Mean   : 2.682
##  3rd Qu.:1.0000   3rd Qu.:2.267     3rd Qu.:1494   3rd Qu.: 3.000
##  Max.   :2.0000   Max.   :9.000     Max.   :1913   Max.   :13.000
##
##  numOfBedrooms     numOfStories
##  Min.   : 0.000   Min.   :1.000
##  1st Qu.: 3.000   1st Qu.:1.000
##  Median : 3.000   Median :1.000
##  Mean   : 3.437   Mean   :1.463
##  3rd Qu.: 4.000   3rd Qu.:2.000
##  Max.   :10.000   Max.   :4.000
##
```

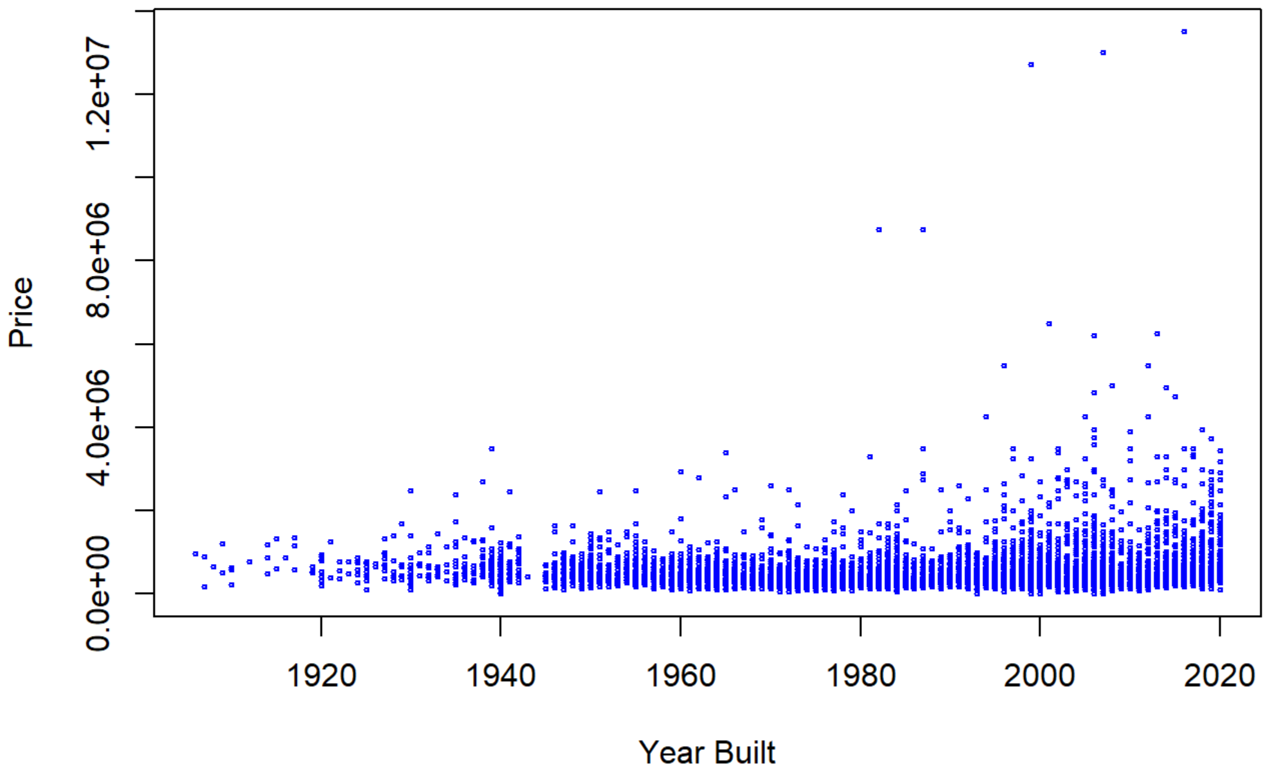
```
plot(train$avgSchoolSize, train$latestPrice, cex = 0.35, col="blue", xlab="School size", ylab="P
rice")
```



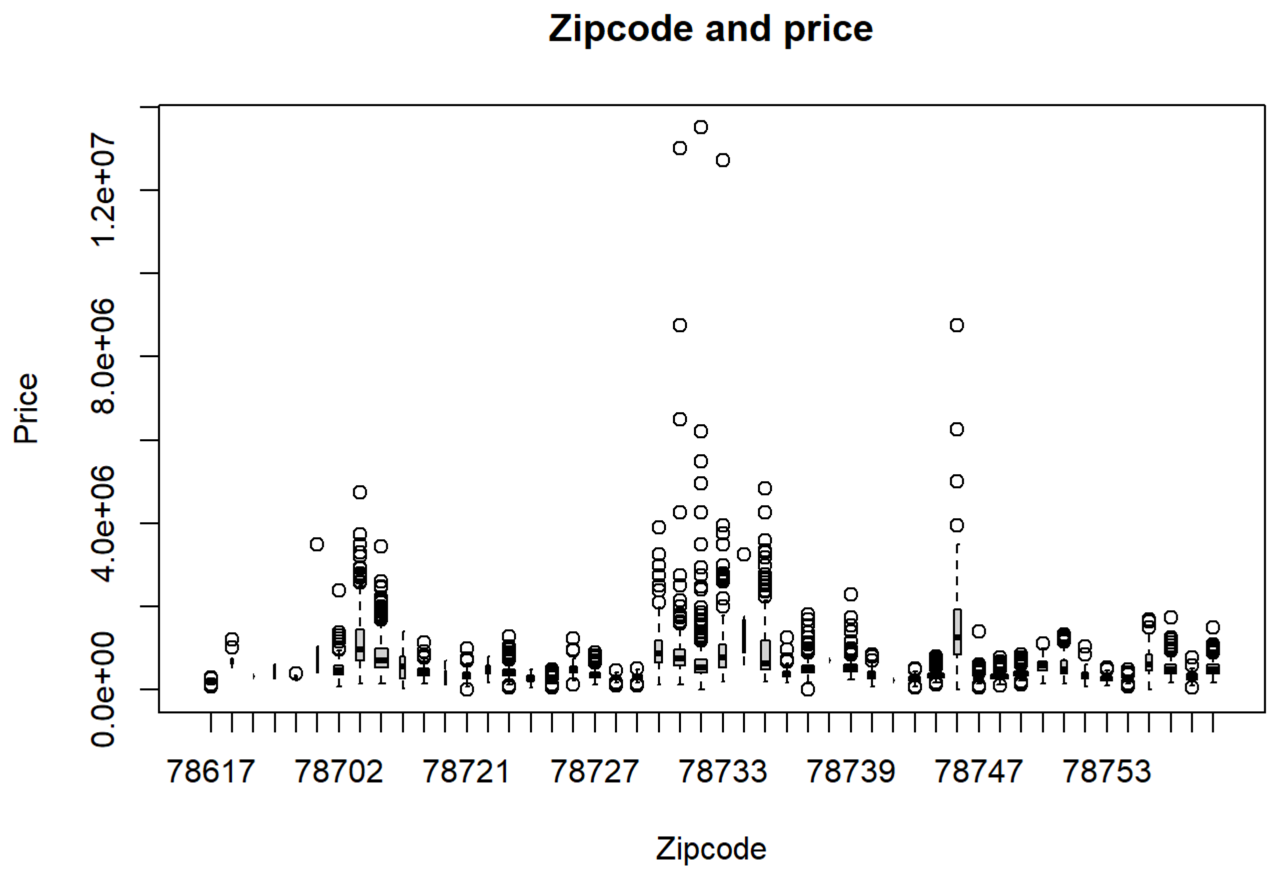

```
plot(train$numOfBedrooms, train$latestPrice, cex = 0.35, col="blue", xlab="Num of bedrooms", ylab="Price")
```




```
plot(train$yearBuilt, train$latestPrice, cex = 0.35, col="blue", xlab="Year Built", ylab="Price"
)
```




```
boxplot(train$latestPrice~train$zipcode, varwidth=TRUE, notch=FALSE,
main="Zipcode and price", xlab="Zipcode", ylab="Price")
```



Try linear regression

```
lm1 <- lm(latestPrice~., data=train)
pred <- predict(lm1, newdata=test)
cor_lm1 <- cor(pred, test$latestPrice)
mse_lm1 <- mean((pred-test$latestPrice)^2)
summary(lm1)
```

```
##
## Call:
## lm(formula = latestPrice ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -5636169  -104434    -9590    80954 11898721
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    -6.371e+07  9.766e+06  -6.524 7.23e-11 ***
## zipcode78619                   -1.901e+05  1.275e+05  -1.491 0.135885
## zipcode78652                    1.341e+05  3.475e+05   0.386 0.699545
## zipcode78653                   -6.890e+05  2.504e+05  -2.751 0.005949 **
## zipcode78660                    4.115e+04  8.648e+04   0.476 0.634257
## zipcode78701                    1.015e+06  1.624e+05   6.249 4.31e-10 ***
## zipcode78702                    3.154e+05  5.639e+04   5.593 2.30e-08 ***
## zipcode78703                    7.891e+05  5.500e+04  14.349  < 2e-16 ***
## zipcode78704                    5.582e+05  5.188e+04  10.760  < 2e-16 ***
## zipcode78705                    4.490e+05  7.237e+04   6.204 5.76e-10 ***
## zipcode78717                    1.325e+04  5.127e+04   0.258 0.796145
## zipcode78719                   -3.438e+05  2.066e+05  -1.664 0.096213 .
## zipcode78721                    1.707e+05  6.000e+04   2.845 0.004454 **
## zipcode78722                    3.676e+05  6.743e+04   5.451 5.13e-08 ***
## zipcode78723                    1.839e+05  5.402e+04   3.404 0.000668 ***
## zipcode78724                   -3.243e+04  5.894e+04  -0.550 0.582196
## zipcode78725                   -7.680e+03  6.010e+04  -0.128 0.898327
## zipcode78726                    5.822e+04  5.799e+04   1.004 0.315403
## zipcode78727                    9.670e+04  5.252e+04   1.841 0.065605 .
## zipcode78728                    1.343e+04  5.564e+04   0.241 0.809212
## zipcode78729                    2.509e+04  5.410e+04   0.464 0.642851
## zipcode78730                    3.260e+05  5.661e+04   5.758 8.78e-09 ***
## zipcode78731                    4.874e+05  5.200e+04   9.373  < 2e-16 ***
## zipcode78732                    1.842e+05  5.106e+04   3.608 0.000310 ***
## zipcode78733                    4.397e+05  6.398e+04   6.873 6.71e-12 ***
## zipcode78734                    7.763e+05  1.324e+05   5.864 4.68e-09 ***
## zipcode78735                    2.974e+05  5.573e+04   5.336 9.72e-08 ***
## zipcode78736                    4.507e+04  6.049e+04   0.745 0.456299
## zipcode78737                   -4.903e+04  5.587e+04  -0.878 0.380158
## zipcode78738                    2.141e+05  3.476e+05   0.616 0.537864
## zipcode78739                    1.206e+05  5.011e+04   2.407 0.016125 *
## zipcode78741                    1.484e+05  5.813e+04   2.553 0.010697 *
## zipcode78742                   -1.360e+05  3.483e+05  -0.391 0.696094
## zipcode78744                   -1.005e+04  5.188e+04  -0.194 0.846332
## zipcode78745                    1.421e+05  5.150e+04   2.758 0.005819 **
## zipcode78746                    9.408e+05  6.364e+04  14.783  < 2e-16 ***
## zipcode78747                   -8.868e+03  5.217e+04  -0.170 0.865033
## zipcode78748                    6.768e+04  4.853e+04   1.395 0.163181
## zipcode78749                    1.196e+05  4.971e+04   2.405 0.016180 *
## zipcode78750                    2.158e+05  5.340e+04   4.042 5.35e-05 ***
## zipcode78751                    3.766e+05  5.942e+04   6.339 2.43e-10 ***
## zipcode78752                    2.513e+05  6.108e+04   4.115 3.91e-05 ***
```

```
## zipcode78753                  5.821e+04  5.353e+04   1.087 0.276906    
## zipcode78754                 -1.016e+05  6.089e+04  -1.669 0.095203 .  
## zipcode78756                  4.336e+05  6.068e+04   7.145 9.70e-13 ***
## zipcode78757                  3.257e+05  5.206e+04   6.256 4.12e-10 ***
## zipcode78758                  1.145e+05  5.380e+04   2.128 0.033399 *  
## zipcode78759                  2.131e+05  5.115e+04   4.166 3.13e-05 ***
## hasAssociationTrue           -6.745e+04  1.116e+04  -6.045 1.56e-09 ***
## hasGarageTrue                -3.346e+04  1.172e+04  -2.855 0.004312 ** 
## hasSpaTrue                    6.897e+04  1.397e+04   4.938 8.03e-07 ***
## homeTypeCondo                -2.491e+05  7.518e+04  -3.313 0.000925 ***
## homeTypeMobile / Manufactured -3.267e+05  1.498e+05  -2.181 0.029205 *  
## homeTypeMultiFamily          -4.433e+05  1.507e+05  -2.941 0.003280 ** 
## homeTypeMultiple Occupancy    -2.543e+05  8.563e+04  -2.970 0.002990 ** 
## homeTypeOther                 3.949e+05  2.120e+05   1.863 0.062449 .  
## homeTypeResidential          -8.419e+04  1.030e+05  -0.818 0.413538    
## homeTypeSingle Family        -1.091e+05  7.288e+04  -1.497 0.134321    
## homeTypeTownhouse            -1.894e+05  8.066e+04  -2.348 0.018881 *  
## homeTypeVacant Land           5.263e+05  8.844e+04   5.951 2.77e-09 ***
## yearBuilt                     8.237e+02  2.525e+02   3.262 0.001110 ** 
## numPriceChanges              -1.241e+04  1.482e+03  -8.369  < 2e-16 ***
## latest_saleyear               3.072e+04  4.842e+03   6.345 2.33e-10 ***
## numOfParkingFeatures          2.286e+04  7.200e+03   3.175 0.001502 ** 
## numOfWaterfrontFeatures       4.971e+05  5.512e+04   9.019  < 2e-16 ***
## livingAreaSqFt                5.625e+01  2.854e+00  19.710  < 2e-16 ***
## numOfHighSchools              5.200e+04  2.712e+04   1.917 0.055208 .  
## avgSchoolDistance             3.089e+04  5.153e+03   5.994 2.12e-09 ***
## avgSchoolSize                -3.440e+01  2.302e+01  -1.495 0.135012    
## numOfBathrooms                2.316e+05  5.934e+03  39.035  < 2e-16 ***
## numOfBedrooms                -2.164e+04  6.191e+03  -3.495 0.000476 ***
## numOfStories                 -1.353e+05  8.970e+03 -15.079  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 344300 on 9030 degrees of freedom
## Multiple R-squared:  0.5078, Adjusted R-squared:  0.5039 
## F-statistic: 131.2 on 71 and 9030 DF,  p-value: < 2.2e-16
```

Try a linear kernel

```
svm1 <- svm(latestPrice~., data=train, kernel="linear", cost=5, scale=TRUE)
summary(svm1)
```

```
##
## Call:
## svm(formula = latestPrice ~ ., data = train, kernel = "linear", cost = 5,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  5
##       gamma:  0.01388889
##     epsilon:  0.1
##
##
## Number of Support Vectors:  5175
```

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

Tune linear kernel

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

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 61168842229
##
## - Detailed performance results:
##    cost       error  dispersion
## 1 1e-03 93068039652 66009942949
## 2 1e-02 71207190232 60756996002
## 3 1e-01 62786398441 59096095387
## 4 1e+00 61362578762 58455659820
## 5 5e+00 61180448038 58342963028
## 6 1e+01 61168842229 58342120860
```

```
summary(tune_svm1$best.model)
```

```
##
## Call:
## best.tune(method = svm, train.x = latestPrice ~ ., data = vald, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 5, 10)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  10
##       gamma:  0.01388889
##     epsilon:  0.1
##
##
## Number of Support Vectors:  1940
```

```
pred <- predict(tune_svm1$best.model, newdata=test)
tuned_cor_svm1 <- cor(pred, test$latestPrice)
tuned_mse_svm1 <- mean((pred - test$latestPrice)^2)
```

Try a polynomial kernel

```
svm2 <- svm(latestPrice~., data=train, kernel="polynomial", cost=5, scale=TRUE)
summary(svm2)
```

```
##
## Call:
## svm(formula = latestPrice ~ ., data = train, kernel = "polynomial",
##     cost = 5, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  polynomial
##        cost:  5
##      degree:  3
##       gamma:  0.01388889
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  5380
```

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

Tune polynomial kernel

```
tune_svm2 <- tune(svm, latestPrice~., data=vald, kernel="polynomial",
                  ranges=list(cost=c(0.001, 0.01, 0.1, 1, 5, 10)))
summary(tune_svm2)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 134551886847
##
## - Detailed performance results:
##    cost        error   dispersion
## 1 1e-03 148821509731  91604459659
## 2 1e-02 147178673630  93064042399
## 3 1e-01 134551886847  90662558836
## 4 1e+00 140552561394 147496630492
## 5 5e+00 176871396206 241202528620
## 6 1e+01 222143196458 375965681664
```

```
summary(tune_svm2$best.model)
```

```
##
## Call:
## best.tune(method = svm, train.x = latestPrice ~ ., data = vald, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 5, 10)), kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  polynomial
##        cost:  0.1
##      degree:  3
##       gamma:  0.01388889
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  2492
```

```
pred <- predict(tune_svm2$best.model, newdata=test)
tuned_cor_svm2 <- cor(pred, test$latestPrice)
tuned_mse_svm2 <- mean((pred - test$latestPrice)^2)
```

Try a radial kernel

```
svm3 <- svm(latestPrice~., data=train, kernel="radial", cost=5, gamma=1, scale=TRUE)
summary(svm3)
```

```
##
## Call:
## svm(formula = latestPrice ~ ., data = train, kernel = "radial", cost = 5,
##     gamma = 1, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  5
##       gamma:  1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  6541
```

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

Tune radial Kernel

```
set.seed(1234)
tune_svm3 <- tune(svm, latestPrice~., data=vald, kernel="radial",
                 ranges=list(cost=c(0.1,1,10),
                             gamma=c(0.5,1,2,3,4)))
summary(tune_svm3)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10   0.5
##
## - best performance: 94925420080
##
## - Detailed performance results:
##    cost gamma        error  dispersion
## 1   0.1   0.5 131210891860 84626521439
## 2   1.0   0.5 104877791679 80130361233
## 3  10.0   0.5  94925420080 76256523053
## 4   0.1   1.0 144344757397 85663156939
## 5   1.0   1.0 127670934016 82721581427
## 6  10.0   1.0 122165304220 80549722409
## 7   0.1   2.0 148906372294 85946947867
## 8   1.0   2.0 138381609610 84088478177
## 9  10.0   2.0 135558913699 82199577979
## 10  0.1   3.0 149852382543 86008813476
## 11  1.0   3.0 140732186623 84369192277
## 12 10.0   3.0 138546323335 82586762683
## 13  0.1   4.0 150251564094 86027119918
## 14  1.0   4.0 141716168049 84454427745
## 15 10.0   4.0 139807081007 82708980731
```

```
summary(tune_svm3$best.model)
```

```
##
## Call:
## best.tune(method = svm, train.x = latestPrice ~ ., data = vald, ranges = list(cost = c(0.1,
##     1, 10), gamma = c(0.5, 1, 2, 3, 4)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  0.5
##     epsilon:  0.1
##
##
## Number of Support Vectors:  2208
```

```
pred <- predict(tune_svm3$best.model, newdata=test)
tuned_cor_svm3 <- cor(pred, test$latestPrice)
tuned_mse_svm3 <- mean((pred - test$latestPrice)^2)
```

Results

Linear Regression
Correlation: 0.7613644
MSE: 70568735724

Linear Kernel
Correlation: 0.7835833
MSE: 69222878510

Tuned Linear Kernel
Correlation: 0.7767439
MSE: 71307832537

Polynomial Kernel
Correlation: 0.782148
MSE: 66577688810

Tuned Polynomial Kernel
Correlation: 0.5486643
MSE: 1.40312e+11

Radial Kernel
Correlation: 0.5212503
MSE: 125615340679

Tuned Radial Kernel
Correlation: 0.5769459
MSE: 115627177155

The results seem to show that a linear kernel best models our data.The hyperplanes of our data are shaped linearly. Meaning there is likely a linear relationship between our predictors and the value to be predicted. Based on our data exploration, that seemed to be the case.

Due to the large size of the data, testing a large amount of hyperparameters seemed to overwork my computer. Thus, I was limited in the amount of tuning I could. For both linear and polynomial, the tuned model was worse than my guessed model. However, both linear and polynomial gave slightly better results than linear regression non tuned.