136_Liz_Project_Step5_SVM Playground

Hyunkyung Kim

December 4, 2018

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

```
#install.packages("glmnet")
#install.packages("mlbench")
#install.packages("Boruta")
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(tidyverse)

## -- Attaching packages ----------------------------------------------------------------------------
------------------------------ tidyverse 1.2.1 --

## v tibble  1.4.2     v purrr   0.2.5
## v tidyr   0.8.1     v dplyr   0.7.7
## v readr   1.1.1     v stringr 1.3.1
## v tibble  1.4.2     v forcats 0.3.0

## -- Conflicts ----------------------------------------------------------------------------------------
------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()

library(psych)

##
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha

library(glmnet)

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following object is masked from 'package:tidyr':
##
##     expand

## Loading required package: foreach
```

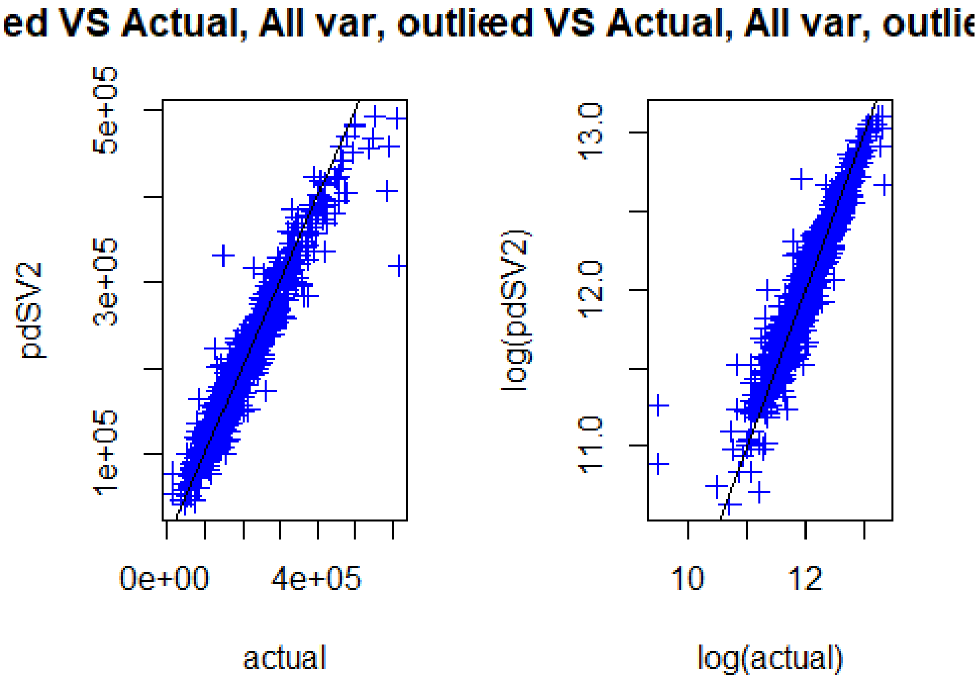
```
##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
##     accumulate, when

## Loaded glmnet 2.0-16

library(mlbench)
library(Boruta)

## Loading required package: ranger

library(MASS) # stepwise regression

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

library(leaps) # all subsets regression
library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ranger':
##
##     importance

## The following object is masked from 'package:psych':
##
##     outlier

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin

library(MLmetrics)

##
## Attaching package: 'MLmetrics'

## The following object is masked from 'package:psych':
##
##     AUC
## The following objects are masked from 'package:caret':
##
##     MAE, RMSE

## The following object is masked from 'package:base':
##
##     Recall

library(e1071)
```

Import Cleaned Data

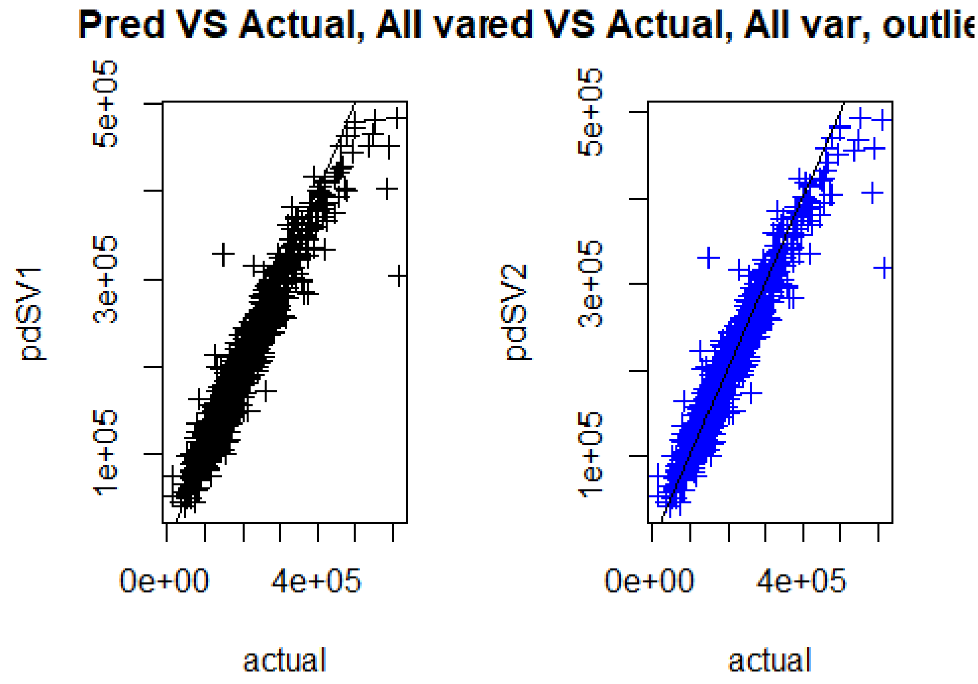
```
H_Clean<-read.csv( file = "C:\\Users\\Hyunkyung
Kim\\Desktop\\CKME999\\136\\dataset\\all\\H_clean.csv")

Train<-H_Clean[!is.na(H_Clean$SalePrice),-1] #Remove ID
Test<-H_Clean[is.na(H_Clean$SalePrice),-1]
actual<-read.csv( file = "C:\\Users\\Hyunkyung
Kim\\Desktop\\CKME999\\136\\dataset\\all\\AMES_test.csv")[1461:2919,2] # Test Price
#Remove Utilities - only one exception and is causing issues. Also will result in huge
variance

tc<-trainControl(method="cv", number=10)

Train11<-Train[,names(Train)!="Utilities"]

f2<-SalePrice ~ OverallQual + GrLivArea + Neighborhood + BsmtFinSF1 +
    RoofMatl + MSSubClass + BsmtExposure + KitchenQual + Condition2 +
    SaleCondition + LotArea + YearBuilt + OverallCond + MasVnrArea +
    PoolQC + BedroomAbvGr + GarageCars + MasVnrType + TotalBsmtSF +
    BldgType + Functional + ExterQual + BsmtCond + Condition1 +
    Exterior1st + MoSold + GarageCond + ScreenPorch + LandContour +
    LowQualFinSF + LotConfig + LotFrontage + TotRmsAbvGrd + KitchenAbvGr +
    WoodDeckSF + Street + GarageArea + LotShape + BsmtQual +
    Fireplaces + FireplaceQu + PoolArea + RoofStyle + BsmtFinSF2 +
     ExterCond # Utilities Removed
f3<- SalePrice ~ LotFrontage + LotArea + Street + LotShape + LandContour +
    + LotConfig + Neighborhood + Condition1 + Condition2 +
    BldgType + HouseStyle + OverallQual + OverallCond + YearBuilt +
    RoofMatl + Exterior1st + MasVnrType + MasVnrArea + ExterQual +
    ExterCond + Foundation + BsmtQual + BsmtCond + BsmtExposure +
    BsmtFinSF1 + BsmtFinSF2 + BsmtUnfSF + X1stFlrSF + X2ndFlrSF +
    HalfBath + BedroomAbvGr + KitchenAbvGr + KitchenQual + TotRmsAbvGrd +
    Functional + Fireplaces + FireplaceQu + GarageType + GarageCars +
    GarageArea + WoodDeckSF + X3SsnPorch + ScreenPorch + PoolQC +
    Fence + MiscFeature + MoSold + SaleCondition    # Utilities Removed - because tuning
was done without it - error otherwise.

f4<-SalePrice
~MSSubClass+MSZoning+LotFrontage+LotArea+Alley+LotShape+LandContour+LandSlope+Neighborhoo
d+
BldgType+HouseStyle+OverallQual+OverallCond+YearBuilt+YearRemodAdd+RoofStyle+Exterior1st+
Exterior2nd+
MasVnrType+MasVnrArea+ExterQual+Foundation+BsmtQual+BsmtCond+BsmtExposure+BsmtFinType1+Bs
mtFinSF1+
BsmtFinType2+BsmtUnfSF+TotalBsmtSF+HeatingQC+CentralAir+X1stFlrSF+X2ndFlrSF+GrLivArea+Bsm
tFullBath+
FullBath+HalfBath+BedroomAbvGr+KitchenAbvGr+KitchenQual+TotRmsAbvGrd+Fireplaces+Fireplace
Qu+GarageType+
GarageYrBlt+GarageFinish+GarageCars+GarageArea+GarageQual+GarageCond+PavedDrive+WoodDeckS
F+OpenPorchSF
```

Radial SVM - best tuned without big anomalies. 524, 1299

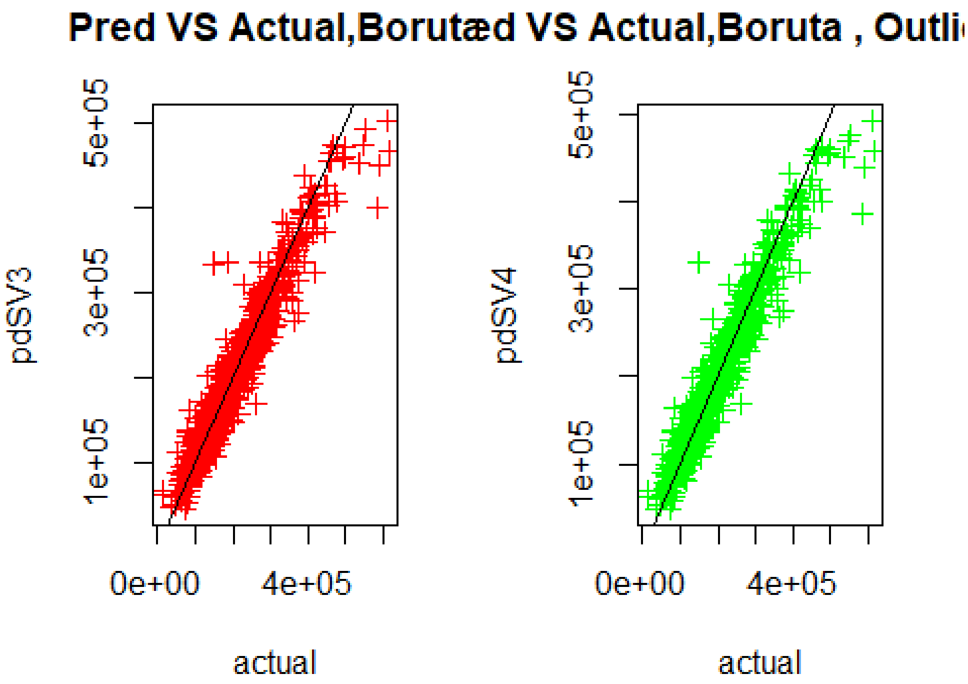
```
SV1<-svm(SalePrice~.,data=Train11, cost=2.25, gamma=0.0035)
SV2<-svm(SalePrice~.,data=Train11[-c(524,1299),], cost=2.25, gamma=0.0035)
#SV2<-svm(f2,data=Train,cost=4)
SV3<-svm(f4,data=Train11[-c(524,1299),],gamma= 0.007,  cost=1.5, espilon=0.06)
SV4<-svm(f4, data=Train11[,], cost=1.5, gamma=0.007, epsilon=0.06)




pdSV1<-predict(SV1,newdata=Test[,-80])
pdSV2<-predict(SV2,newdata=Test[,-80])
pdSV3<-predict(SV3,newdata=Test[,-80])
pdSV4<-predict(SV4,newdata=Test[,-80])




summary(SV1)

## 
## Call:
## svm(formula = SalePrice ~ ., data = Train11, cost = 2.25, gamma = 0.0035)
## 
## 
## Parameters:
##    SVM-Type:  eps-regression 
##  SVM-Kernel:  radial 
##        cost:  2.25 
##       gamma:  0.0035 
##     epsilon:  0.1 
## 
## 
## Number of Support Vectors:  799

par(mfrow=c(1,2))
plot(actual,pdSV2,pch=3, col='blue',main='Pred VS Actual, All var, outlier Rmv')
abline(a=0,b=1)
plot(log(actual),log(pdSV2),pch=3, col='blue',main='Pred VS Actual, All var, outlier
Rmv')
abline(a=0,b=1)
```



```r
plot(actual,pdSV1,pch=3, main='Pred VS Actual, All var')
abline(a=0,b=1)
plot(actual,pdSV2,pch=3, col='blue',main='Pred VS Actual, All var, outlier Rmv')
abline(a=0,b=1)
```



```r
plot(actual,pdSV3,pch=3, col='red',main='Pred VS Actual,Boruta')
abline(a=0,b=1)
plot(actual,pdSV4,pch=3, col='green',main='Pred VS Actual,Boruta , Outlier Rmv')
abline(a=0,b=1)
```



```
print("Test RMSLE & RMSE")

## [1] "Test RMSLE & RMSE"

RMSE(log(pdSV1),log(actual))

## [1] 0.123633

RMSE(log(pdSV2),log(actual))

## [1] 0.1239889

RMSE(log(pdSV3),log(actual))

## [1] 0.1280671

RMSE(log(pdSV4),log(actual))

## [1] 0.1259846

RMSLE(pdSV1,actual)

## [1] 0.1236311

RMSLE(pdSV2,actual)

## [1] 0.1239871

RMSLE(pdSV3,actual)

## [1] 0.1280652

RMSLE(pdSV4,actual)

## [1] 0.1259828
```

```
#RMSE(pdSV1,actual)
#RMSE(pdSV2,actual)
#RMSE(pdSV3,actual)
#RMSE(pdSV4,actual)
```

[1] “Test RMSLE & RMSE” - epsilon not defined [1] 0.1236311 [1] 0.1239871 [1] 0.1280652 [1] 0.127133

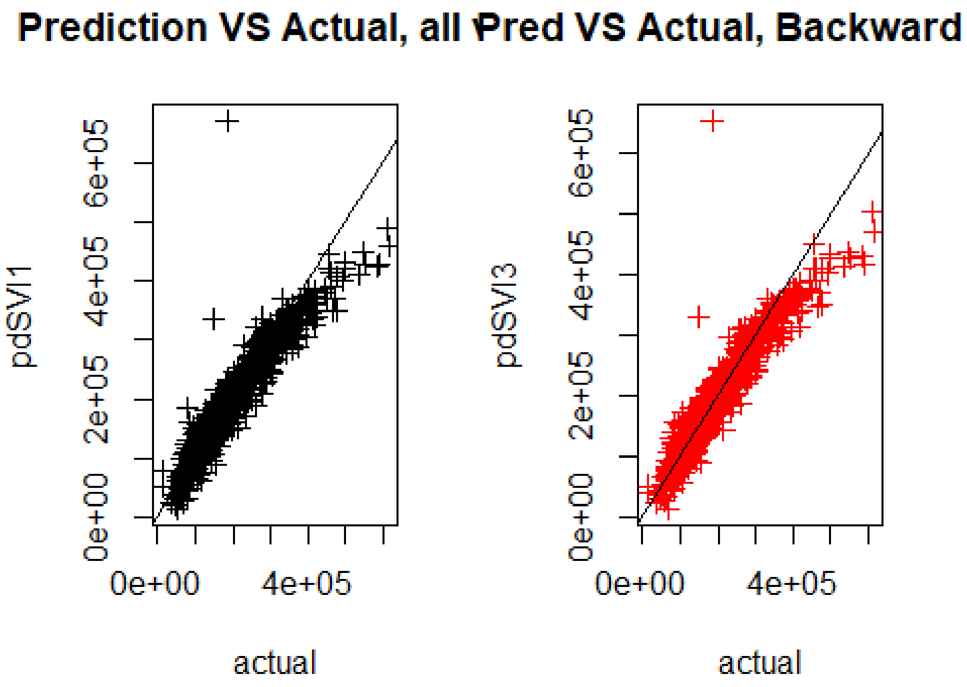
[1] “Test RMSLE & RMSE” - epsilon 0.06 [1] 0.1239504 [1] 0.1241208 [1] 0.1280652 [1] 0.1259828

Linear SVM Using Radial Tuned SVM parameters

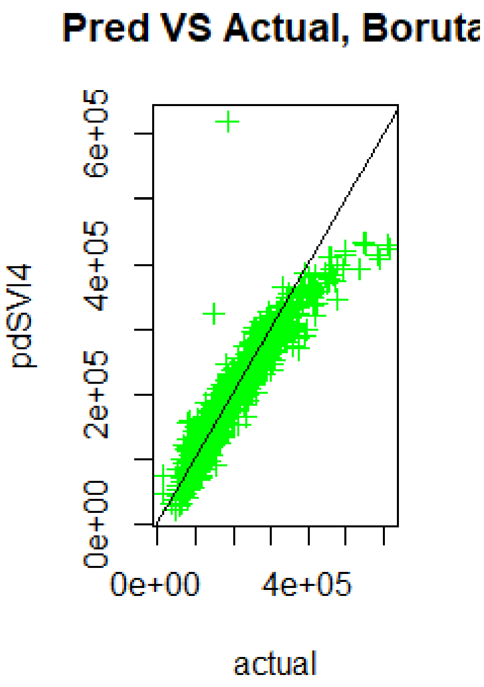
```
SVl1<-svm(SalePrice~.,data=Train11, cost=2.25, epsilon=0.06,kernel='linear')
SVl2<-svm(f2,data=Train11,epsilon=0.06, cost=2.25,kernel='linear')
SVl3<-svm(f3,data=Train11,cost=2.25, epsilon=0.06,kernel='linear')
SVl4<-svm(f4, data=Train11,cost=1.5,  epsilon=0.06, kernel='linear')


pdSVl1<-predict(SVl1,newdata=Test[,-80])
pdSVl2<-predict(SVl2,newdata=Test[,-80])
pdSVl3<-predict(SVl3,newdata=Test[,-80])
pdSVl4<-predict(SVl4,newdata=Test[,-80])

pdSVl01<-predict(SVl1,newdata=Train[,-80])
pdSVl02<-predict(SVl2,newdata=Train[,-80])
pdSVl03<-predict(SVl3,newdata=Train[,-80])
pdSVl04<-predict(SVl4,newdata=Train[,-80])


summary(SVl1)
```

```
##
## Call:
## svm(formula = SalePrice ~ ., data = Train11, cost = 2.25, epsilon = 0.06,
##     kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  2.25
##       gamma:  0.004608295
##     epsilon:  0.06
##
##
## Number of Support Vectors:  1127
```

```
par(mfrow=c(1,2))
plot(actual,pdSVl1,pch=3, main='Prediction VS Actual, all var')
abline(a=0,b=1)
#plot(actual,pdSVl2,pch=3, col='blue',main='Pred VS Actual, FowardSelect')
#abline(a=0,b=1)
plot(actual,pdSVl3,pch=3, col='red',main='Pred VS Actual, BackwardElim')
abline(a=0,b=1)
```



```
plot(actual,pdSVl4,pch=3, col='green',main='Pred VS Actual, Boruta')
abline(a=0,b=1)
```



```
print("Test RMSLE " )

## [1] "Test RMSLE "

RMSLE(pdSVl1,actual)

## [1] 0.1538468
```

```
RMSLE(pdSVl2,actual)

## [1] 0.1420435

RMSLE(pdSVl3,actual)

## [1] 0.1457756

RMSLE(pdSVl4,actual)

## [1] 0.1485868
```

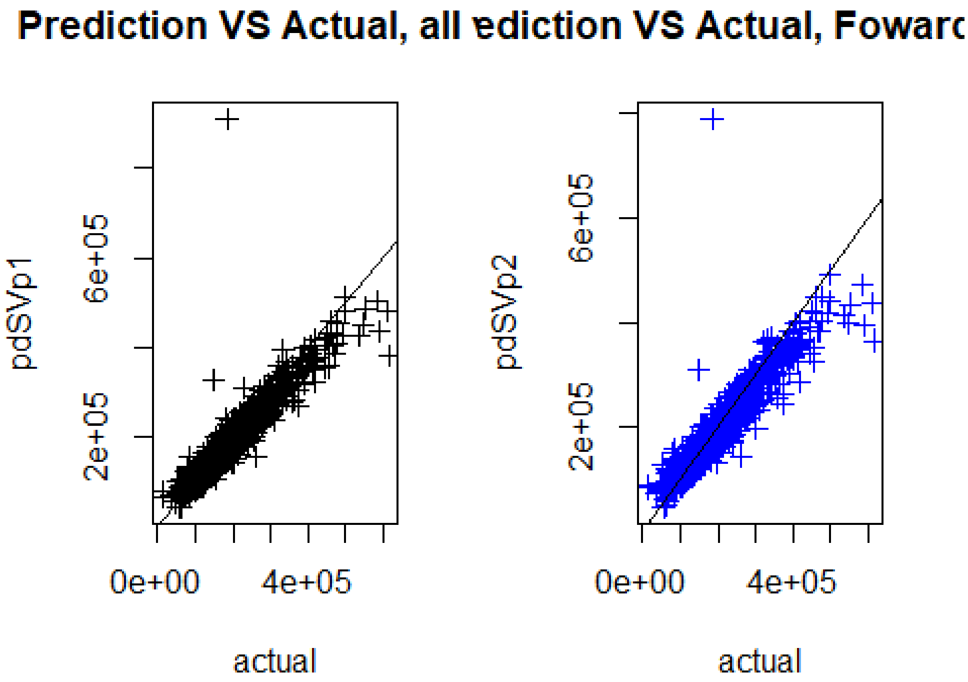
Train[,names(Train)!=“Utilities”] #### POLYNOMIAL SVM

Degree =2

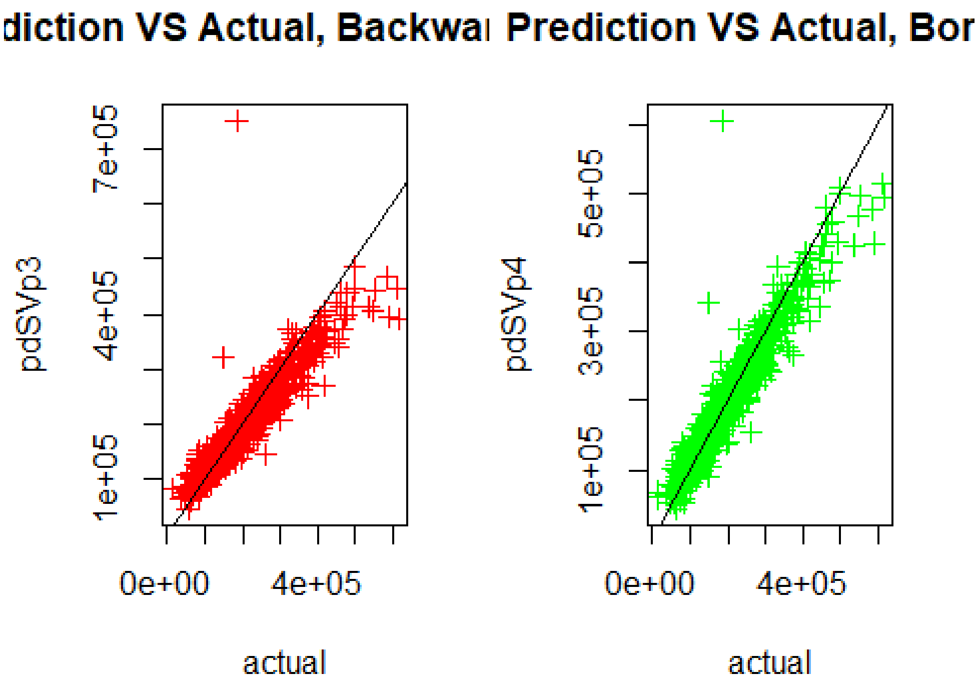
```
SVp1<-svm(SalePrice~.,data=Train11, cost=2.25, gamma=  0.0035, degree=2,
kernel='polynomial')
SVp2<-svm(f2,data=Train11, cost=2.25,degree=2, gamma=  0.0035,kernel='polynomial')
SVp3<-svm(f3,data=Train11,cost=2.25 ,degree=2,gamma=  0.0035, kernel='polynomial')
SVp4<-svm(f4,data=Train11,cost=1.5, degree=2,gamma=  0.007, kernel='polynomial')


pdSVp1<-predict(SVp1,newdata=Test[,-80])
pdSVp2<-predict(SVp2,newdata=Test[,-80])
pdSVp3<-predict(SVp3,newdata=Test[,-80])
pdSVp4<-predict(SVp4,newdata=Test[,-80])


summary(SVp1)

## 
## Call:
## svm(formula = SalePrice ~ ., data = Train11, cost = 2.25, gamma = 0.0035, 
##     degree = 2, kernel = "polynomial")
## 
## 
## Parameters:
##    SVM-Type:  eps-regression 
##  SVM-Kernel:  polynomial 
##        cost:  2.25 
##      degree:  2 
##       gamma:  0.0035 
##      coef.0:  0 
##     epsilon:  0.1 
## 
## 
## Number of Support Vectors:  826

summary(SVp2)

## 
## Call:
## svm(formula = f2, data = Train11, cost = 2.25, degree = 2, gamma = 0.0035, 
##     kernel = "polynomial")
## 
## 
```

```
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  polynomial
##        cost:  2.25
##      degree:  2
##       gamma:  0.0035
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  920
```

```
par(mfrow=c(1,2))
plot(actual,pdSVp1,pch=3, main='Prediction VS Actual, all var')
abline(a=0,b=1)
plot(actual,pdSVp2,pch=3, col='blue',main='Prediction VS Actual, FowardSelect')
abline(a=0,b=1)
```



```
plot(actual,pdSVp3,pch=3, col='red',main='Prediction VS Actual, Backwards Elim')
abline(a=0,b=1)
plot(actual,pdSVp4,pch=3, col='green',main='Prediction VS Actual, Boruta')
abline(a=0,b=1)
```



```
print("Test RMSLE")

## [1] "Test RMSLE"

RMSLE(pdSVp1,actual)

## [1] 0.1394217

RMSLE(pdSVp2,actual)

## [1] 0.1605652

RMSLE(pdSVp3,actual)

## [1] 0.1528928

RMSLE(pdSVp4,actual)

## [1] 0.1375273
```

Have issue with one point that is low price but prediced high for all. This is a very big difference. Would like to investigate this.

Polynomial with degree 3 (default)

```
SVp31<-svm(SalePrice~.,data=Train11, cost=2.25, gamma=  0.0035, kernel='polynomial')
SVp32<-svm(f2,data=Train11, cost=2.25, gamma=  0.0035,kernel='polynomial')
SVp33<-svm(f3,data=Train11,cost=2.25 ,gamma=  0.0035, kernel='polynomial')
SVp34<-svm(f4,data=Train11,cost=1.5, gamma=  0.007, kernel='polynomial')


pdSVp31<-predict(SVp31,newdata=Test[,-80])
pdSVp32<-predict(SVp32,newdata=Test[,-80])
pdSVp33<-predict(SVp33,newdata=Test[,-80])
```

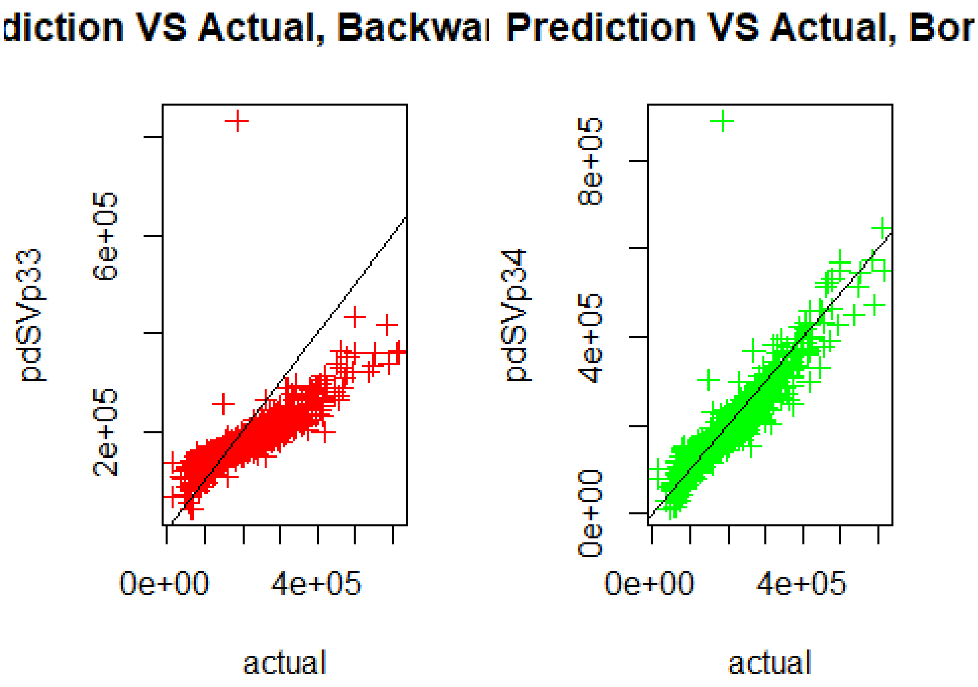
```r
pdSVp34<-predict(SVp34,newdata=Test[,-80])

#pdSVl1101<-predict(SVl111,newdata=Train[,-80])
#pdSVl1102<-predict(SVl112,newdata=Train[,-80])
#pdSVl1103<-predict(SVl113,newdata=Train[,-80])
#pdSVl1104<-predict(SVl114,newdata=Train[,-80])


summary(SVp31)
```

```
##
## Call:
## svm(formula = SalePrice ~ ., data = Train11, cost = 2.25, gamma = 0.0035,
##     kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  polynomial
##        cost:  2.25
##      degree:  3
##       gamma:  0.0035
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  968
```

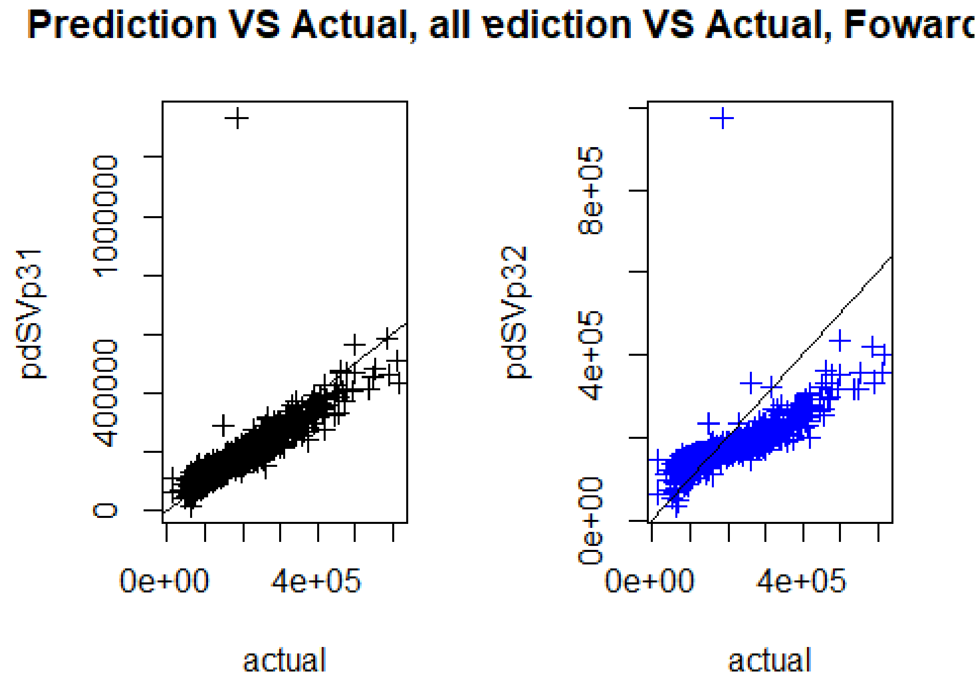
```r
summary(SVp32)
```

```
##
## Call:
## svm(formula = f2, data = Train11, cost = 2.25, gamma = 0.0035,
##     kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  polynomial
##        cost:  2.25
##      degree:  3
##       gamma:  0.0035
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  1249
```

```r
par(mfrow=c(1,2))
plot(actual,pdSVp33,pch=3, col='red',main='Prediction VS Actual, Backwards Elim')
abline(a=0,b=1)
plot(actual,pdSVp34,pch=3, col='green',main='Prediction VS Actual, Boruta')
abline(a=0,b=1)
```



```
plot(actual,pdSVp31,pch=3, main='Prediction VS Actual, all var')
abline(a=0,b=1)
plot(actual,pdSVp32,pch=3, col='blue',main='Prediction VS Actual, FowardSelect')
abline(a=0,b=1)
```



```
print("Test RMSLE")

## [1] "Test RMSLE"
```

```r
RMSLE(pdSVp31,actual)
```

```
## [1] 0.1749054
```

```r
RMSLE(pdSVp32,actual)
```

```
## [1] 0.2733861
```

```r
RMSLE(pdSVp33,actual)
```

```
## [1] 0.2528627
```

```r
RMSLE(pdSVp34,actual)
```

```
## [1] 0.1778839
```

SVM - Sigmoid

```r
SVs1<-svm(SalePrice~.,data=Train11, cost=2.25, gamma=  0.0035, kernel='sigmoid')
SVs2<-svm(f2,data=Train11, cost=2.25, gamma=  0.0035,kernel='sigmoid')
SVs3<-svm(f3,data=Train11,cost=2.25 ,gamma=  0.0035, kernel='sigmoid')
SVs4<-svm(f4,data=Train11,cost=1.5, gamma=  0.007, kernel='sigmoid')


pdSVs1<-predict(SVs1,newdata=Test[,-80])
pdSVs2<-predict(SVs2,newdata=Test[,-80])
pdSVs3<-predict(SVs3,newdata=Test[,-80])
pdSVs4<-predict(SVs4,newdata=Test[,-80])

#pdSVl1101<-predict(SVl111,newdata=Train[,-80])
#pdSVl1102<-predict(SVl112,newdata=Train[,-80])
#pdSVl1103<-predict(SVl113,newdata=Train[,-80])
#pdSVl1104<-predict(SVl114,newdata=Train[,-80])


summary(SVs1)
```

```
## 
## Call:
## svm(formula = SalePrice ~ ., data = Train11, cost = 2.25, gamma = 0.0035, 
##     kernel = "sigmoid")
## 
## 
## Parameters:
##    SVM-Type:  eps-regression 
##  SVM-Kernel:  sigmoid 
##        cost:  2.25 
##       gamma:  0.0035 
##      coef.0:  0 
##     epsilon:  0.1 
## 
## 
## Number of Support Vectors:  1008
```

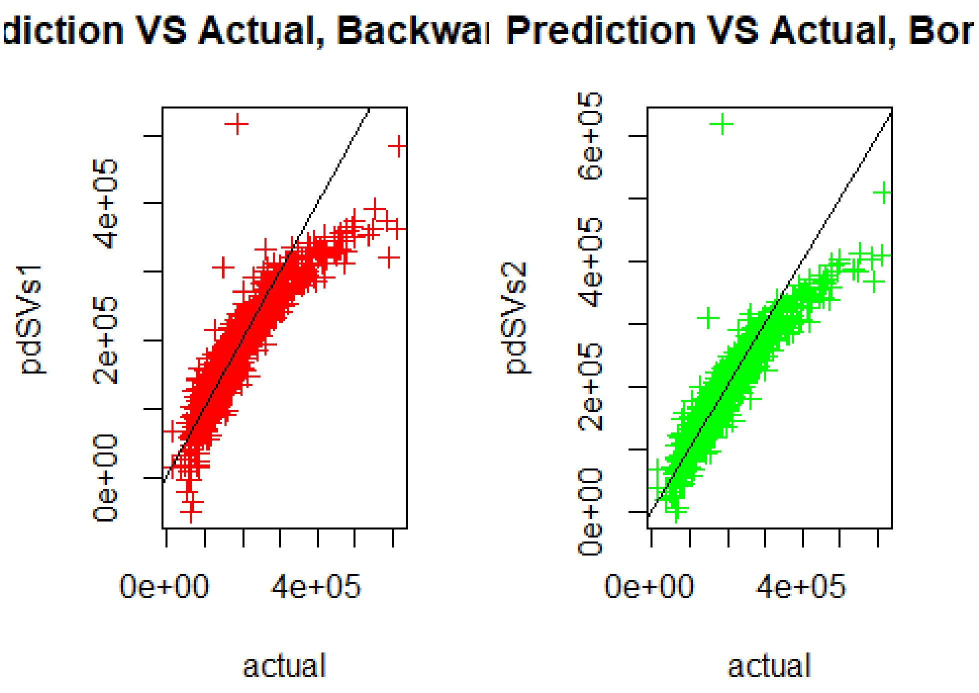
```r
summary(SVs2)
```

```
## 
## Call:
```

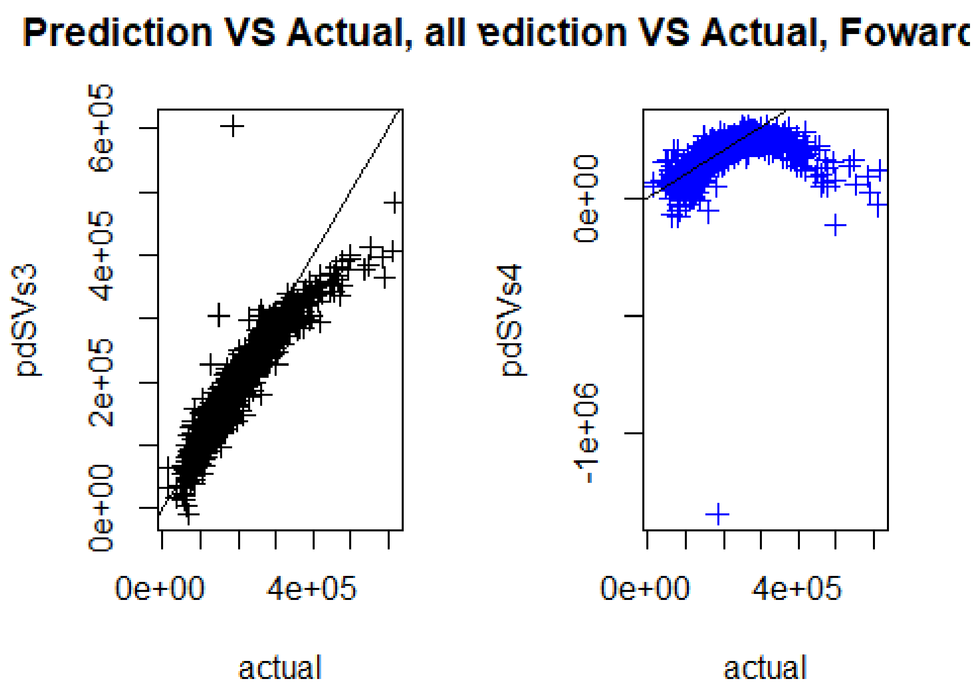
```
## svm(formula = f2, data = Train11, cost = 2.25, gamma = 0.0035,
##     kernel = "sigmoid")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  sigmoid
##        cost:  2.25
##       gamma:  0.0035
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  932
```

SVM - Sigmoid : Disaster. RMSE cannot be calculated.

```
par(mfrow=c(1,2))
plot(actual,pdSVs1,pch=3, col='red',main='Prediction VS Actual, Backwards Elim')
abline(a=0,b=1)
plot(actual,pdSVs2,pch=3, col='green',main='Prediction VS Actual, Boruta')
abline(a=0,b=1)
```



```
plot(actual,pdSVs3,pch=3, main='Prediction VS Actual, all var')
abline(a=0,b=1)
plot(actual,pdSVs4,pch=3, col='blue',main='Prediction VS Actual, FowardSelect')
abline(a=0,b=1)
```



```
print("Test RMSLE")

## [1] "Test RMSLE"

RMSLE(pdSVs1,actual)

## Warning in log(1 + y_pred): NaNs produced

## [1] NaN

RMSLE(pdSVs2,actual)

## Warning in log(1 + y_pred): NaNs produced

## [1] NaN

RMSLE(pdSVs3,actual)

## Warning in log(1 + y_pred): NaNs produced

## [1] NaN

RMSLE(pdSVs4,actual)

## Warning in log(1 + y_pred): NaNs produced

## [1] NaN
```

TUNING FOR LINEAR DO NOT RUN

```
SV_Lin_tune1<-tune.svm(SalePrice~.,data=Train11,cost=c(20,10,1), kernel='linear')

plot(SV_Lin_tune1)

print(SV_Lin_tune1)
sqrt(SV_Lin_tune1$best.performance)
```

```r
SV_Lin_tune2<-tune.svm(SalePrice~.,data=Train11,cost=c(5,0.5), kernel='linear')

Train11<-Train[,names(Train)!='Utilities']
SV_Lin_tune12<-tune.svm(SalePrice~.,data=Train11,cost=seq(1,21,2),gamma=seq(1,21,2))

plot(SV_Lin_tune12)

print(SV_Lin_tune12)
sqrt(SV_Lin_tune12$best.performance)

SV_Lin_tune13<-tune.svm(SalePrice~.,data=Train11,cost=c(2,5,0.5),gamma=seq(0.5,1.5,0.1))
```