SVM Regression

Aditi Chaudhari and Abigail Solomon

SVM Regression

Generally, in Regression technique,patterns in the sample data are identified by understanding the numbers-their values and correlations, then it reproduces the predictions of continuous outcomes.Now in our case, we will use the SVM regression to predict the price of diamond based on attributes of the item as indicators.

Load necessary libraries

```
library(e1071)
library(ggplot2)
library(caret, warn.conflicts = FALSE)
```

```
## Loading required package: lattice
```

Import the Data set

Source of the Data Set

Gem stone price prediction data set: ’diamond’dataset (https://www.kaggle.com/datasets/colearninglounge/gemstone-price-prediction)

Read data

We use the read.csv() function to read a csv. Using the dim() structure function, We will see that the data set contain prices and other attributes of the diamond, and it has 26967 observations of 11 variables along with their names.

```
#Read data
df <- read.csv("diamonds.csv")
dim(df)
```

```
## [1] 26967    11
```

```
names(df)
```

```
##  [1] "X"       "carat"   "cut"     "color"   "clarity" "depth"   "table"  
##  [8] "x"       "y"       "z"       "price"
```

##Data Preprocessing Data processing is converting train raw data sets into meaningful sets that are usable.We will be examining the specific types and steps of data cleaning, changing some variables to factors and later the scaling before analyzing the data.

Data cleaning

In order to make our data set for machine learning more meaningful, we may need to fix or remove missing or unwanted, data instances which may not help to solve the problem, from the data set. The sapply() function gives the number of missing values in each column, in this case the depth has 697 NA's.The unique functions retrieves that there are no repetitions to delete.

```
sapply(df, function(x) sum(is.na(x)==TRUE))
```

```
##       X   carat     cut   color clarity   depth   table       x       y       z 
##       0       0       0       0       0     697       0       0       0       0 
##   price 
##       0
```

```
df <- unique(df)
dim(df)
```

```
## [1] 26967    11
```

Data Sampling

Large data sets consume a lot of memory and took ages to run, in our case we are going to take only the first 20,000 observations.We will remove the 697 NA's and the first column, and the rows after the 20,000th. and get the new dim of the new data set

```
df$X <- NULL
df <- na.omit(df)
dim(df)
```

```
## [1] 26270    10
```

```
df <- df[-c(20001:26270), ]
str(df)
```

```
## 'data.frame':    20000 obs. of  10 variables:
##  $ carat  : num  0.3 0.33 0.9 0.42 0.31 1.02 1.01 0.5 1.21 0.35 ...
##  $ cut    : chr  "Ideal" "Premium" "Very Good" "Ideal" ...
##  $ color  : chr  "E" "G" "E" "F" ...
##  $ clarity: chr  "SI1" "IF" "VVS2" "VS1" ...
##  $ depth  : num  62.1 60.8 62.2 61.6 60.4 61.5 63.7 61.5 63.8 60.5 ...
##  $ table  : num  58 58 60 56 59 56 60 62 64 57 ...
##  $ x      : num  4.27 4.42 6.04 4.82 4.35 6.46 6.35 5.09 6.72 4.52 ...
##  $ y      : num  4.29 4.46 6.12 4.8 4.43 6.49 6.3 5.06 6.63 4.6 ...
##  $ z      : num  2.66 2.7 3.78 2.96 2.65 3.99 4.03 3.12 4.26 2.76 ...
##  $ price  : int  499 984 6289 1082 779 9502 4836 1415 5407 706 ...
##  - attr(*, "na.action")= 'omit' Named int [1:697] 27 87 118 149 164 186 259 314 345 368 ...
##   ..- attr(*, "names")= chr [1:697] "27" "87" "118" "149" ...
```

Data Split

We are going to split our data set into training, validation and tests sets. 60% of our data will be attributed to the train data set, 20% will be attributed to the validation data, and the rest 20% will be attributed to the test data. We will then find the dimensions of the data frame using the dim() function.

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

```
## [1] 20000    10
```

Converting to factors

```
df$cut <- factor(df$cut)
df$color <- as.factor(df$color)
df$clarity <- factor(df$clarity)
```

Data Exploration

Data exploration helps us to gain insight into the raw train data and findings of R built-in functions.We will print the first and last six rows using the head() and tail() functions respectively.The summary () function applied on the Amount vector, calculates summary statistics for each of them, it prints the Minimum value, the 1st quartile's value (25th percentile), the median value, the 3rd quartile's value (75th percentile) and the maximum value.

The first six observations

```
head(train)
```

	carat	cut	color	clarity	depth	table	x	y	z
	<dbl>	<chr>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	0.30	Ideal	E	SI1	62.1	58	4.27	4.29	2.66
2	0.33	Premium	G	IF	60.8	58	4.42	4.46	2.70

	carat	cut	color	clarity	depth	table	x	y	z
	<dbl>	<chr>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
3	0.90	Very Good	E	VVS2	62.2	60	6.04	6.12	3.78
4	0.42	Ideal	F	VS1	61.6	56	4.82	4.80	2.96
5	0.31	Ideal	F	VVS1	60.4	59	4.35	4.43	2.65
6	1.02	Ideal	D	VS2	61.5	56	6.46	6.49	3.99

6 rows | 1-10 of 11 columns

The last six observations

```
tail(train)
```

	carat	cut	color	clarity	depth	table	x	y	z
	<dbl>	<chr>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
20516	0.40	Premium	E	VVS2	61.4	56	4.81	4.74	2.93
20517	1.57	Ideal	H	SI1	60.5	57	7.60	7.57	4.59
20518	0.30	Ideal	E	VS2	61.6	55	4.31	4.33	2.66
20522	0.26	Ideal	E	VVS2	62.9	58	4.02	4.06	2.54
20523	2.04	Very Good	I	VS2	62.5	58	8.09	8.22	5.10
20527	0.70	Very Good	F	SI1	59.8	60	5.75	5.83	3.46

6 rows | 1-10 of 11 columns

Summary of the train data set

```
summary(train)
```

```
##      carat            cut                color              clarity
##  Min.   :0.2000   Length:12000       Length:12000       Length:12000
##  1st Qu.:0.4000   Class :character   Class :character   Class :character
##  Median :0.7000   Mode  :character   Mode  :character   Mode  :character
##  Mean   :0.7986
##  3rd Qu.:1.0400
##  Max.   :4.5000
##      depth           table             x                y
##  Min.   :50.80   Min.   :50.00   Min.   : 0.00   Min.   : 0.000
##  1st Qu.:61.00   1st Qu.:56.00   1st Qu.: 4.70   1st Qu.: 4.710
##  Median :61.80   Median :57.00   Median : 5.70   Median : 5.710
##  Mean   :61.74   Mean   :57.44   Mean   : 5.73   Mean   : 5.731
##  3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 6.54   3rd Qu.: 6.540
##  Max.   :71.30   Max.   :79.00   Max.   :10.23   Max.   :10.160
##        z             price
##  Min.   :0.000   Min.   :  326
##  1st Qu.:2.900   1st Qu.:  945
##  Median :3.520   Median : 2390
##  Mean   :3.537   Mean   : 3957
##  3rd Qu.:4.040   3rd Qu.: 5363
##  Max.   :6.720   Max.   :18818
```

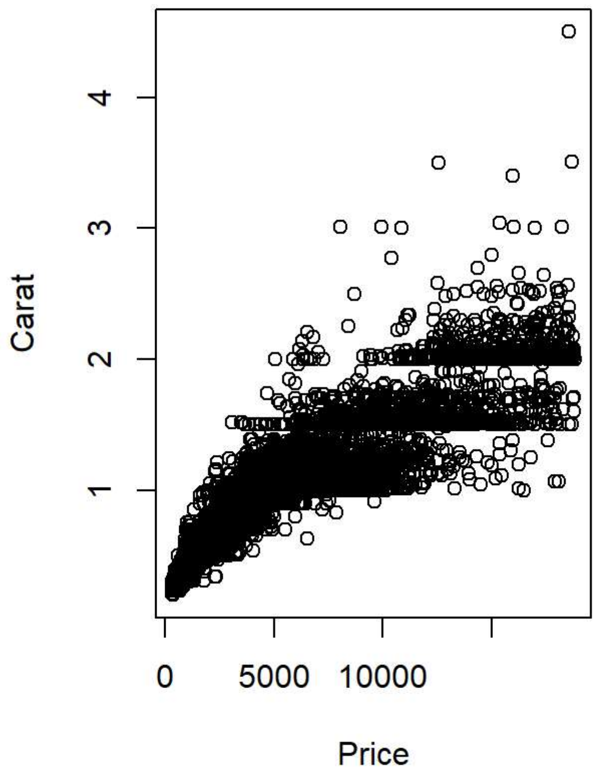
Visual Data Exploration

Data visualization present train data contents in graphical or picture format, enables us to grasp and understand analytics in an easier manner and be able to communicate what has been learned about the data to others, it is also optically entertaining.

Plot

The plot graph shows that there is a linear relationship between price and carat, both increase linearly, of course there are some out liers.

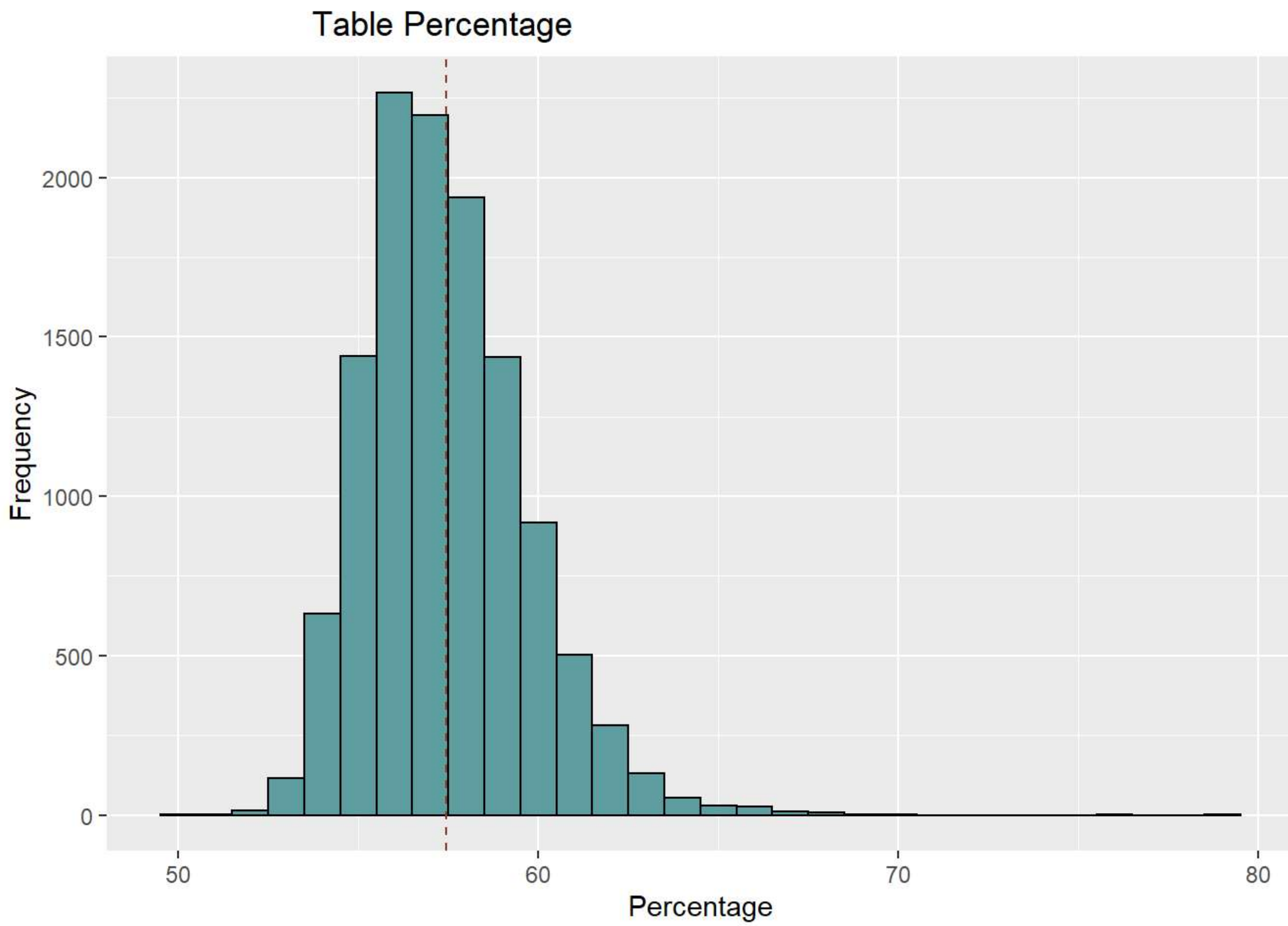
```
par(mfrow=c(1,2))
plot(train$price, train$carat,
     xlab="Price", ylab="Carat")
```



Histogram

The Histogram graph displays the frequency of the x values, in our case it the percentage from the table function. the geom_vline can indicate our mean dashed line.

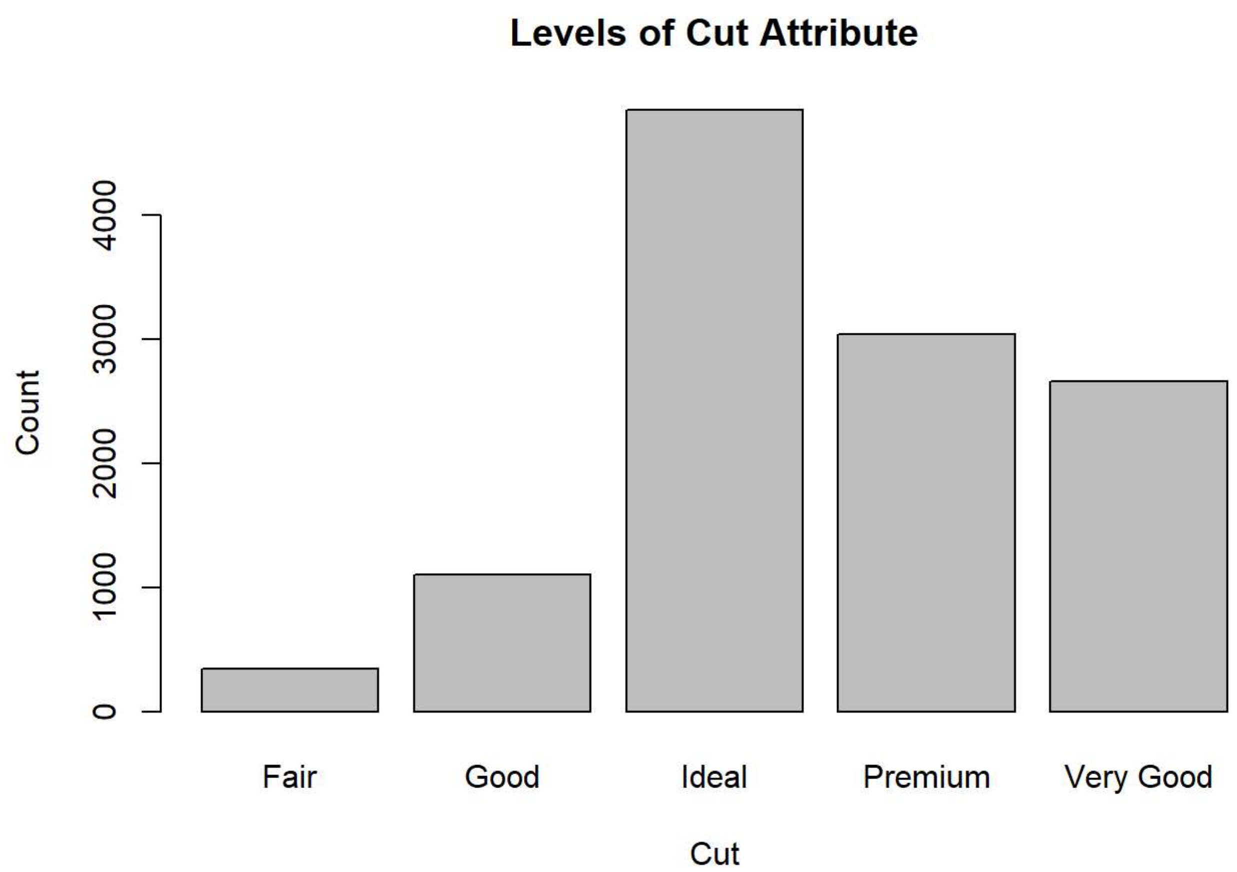
```
ggplot(train, aes(x=table)) +
  geom_histogram(binwidth = 1, color = "black", fill = "cadetblue") +
  geom_vline(aes(xintercept=mean(table)), color = "coral4", linetype = "dashed") +
  labs(title = "                      Table Percentage", x = "Percentage", y = "Frequency")
```



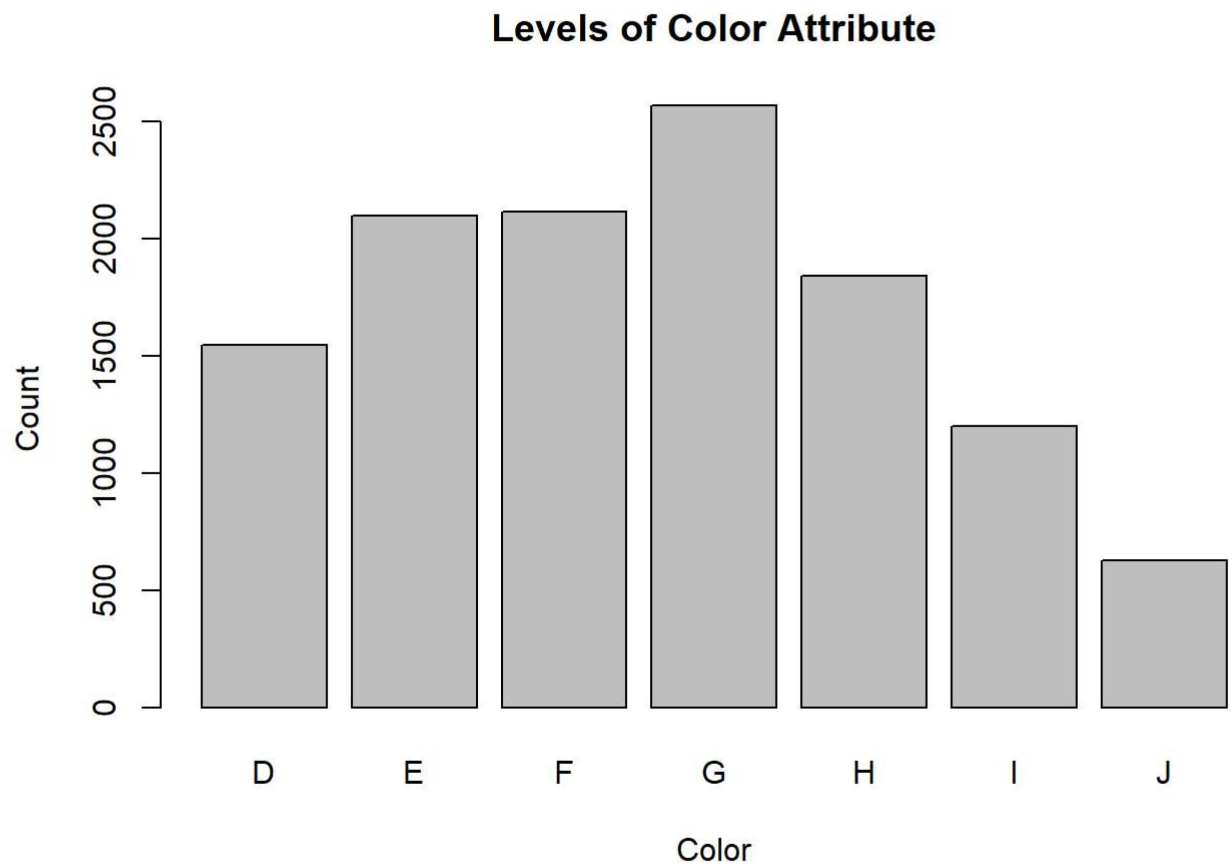
Barplot

Let us see the levels of the cut,color and clarity in bar plot. The ideal cut,SI1 and VS2 clarity have high frequency.

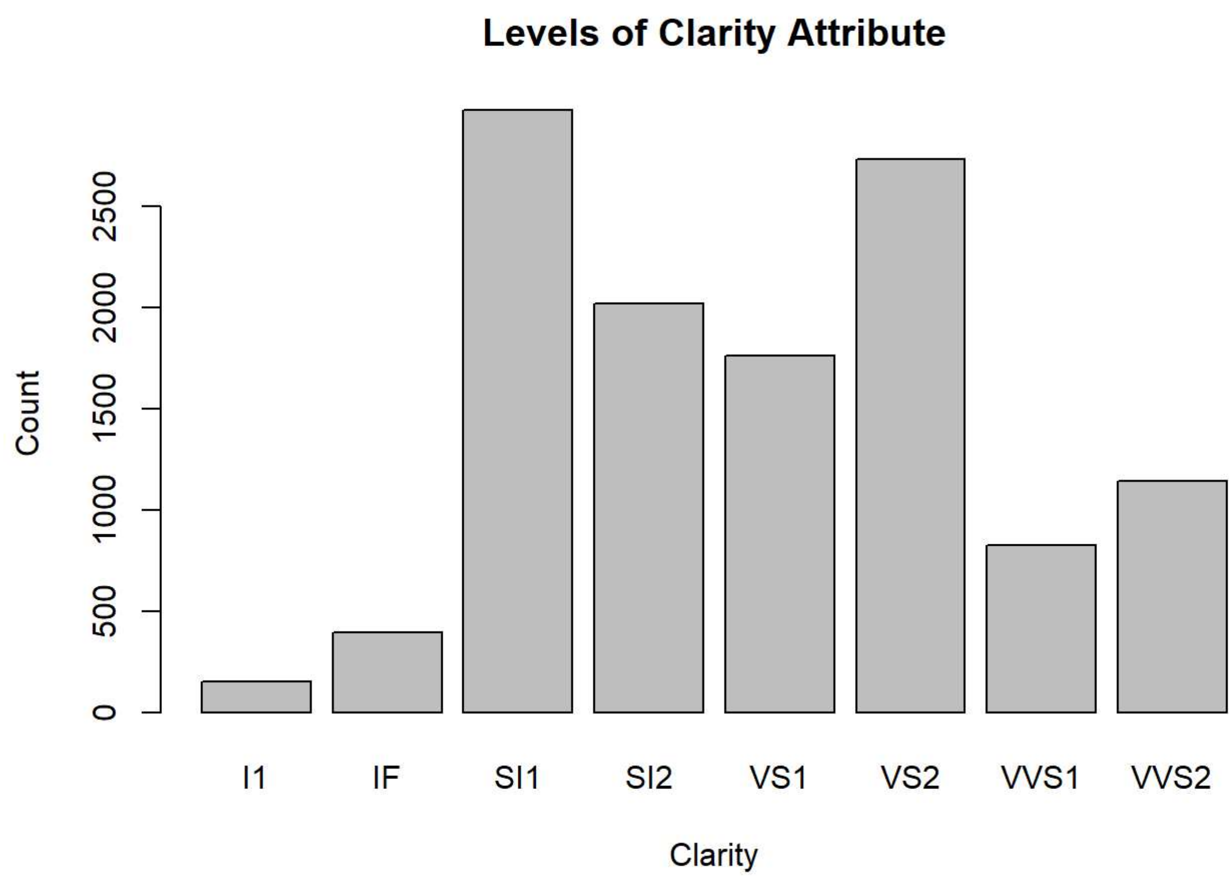
```
barplot(table(train$cut),
        main="Levels of Cut Attribute", xlab="Cut", ylab="Count")
```




```
barplot(table(train$color),
        main="Levels of Color Attribute", xlab="Color", ylab="Count")
```




```
barplot(table(train$clarity),
        main="Levels of Clarity Attribute", xlab="Clarity", ylab="Count")
```



Data Modeling

We are going to build the linear regression, linear SVM, the Polynomial SVM,and the Radial Kernels SVM and see the accuracy of each model.

Linear Regression model

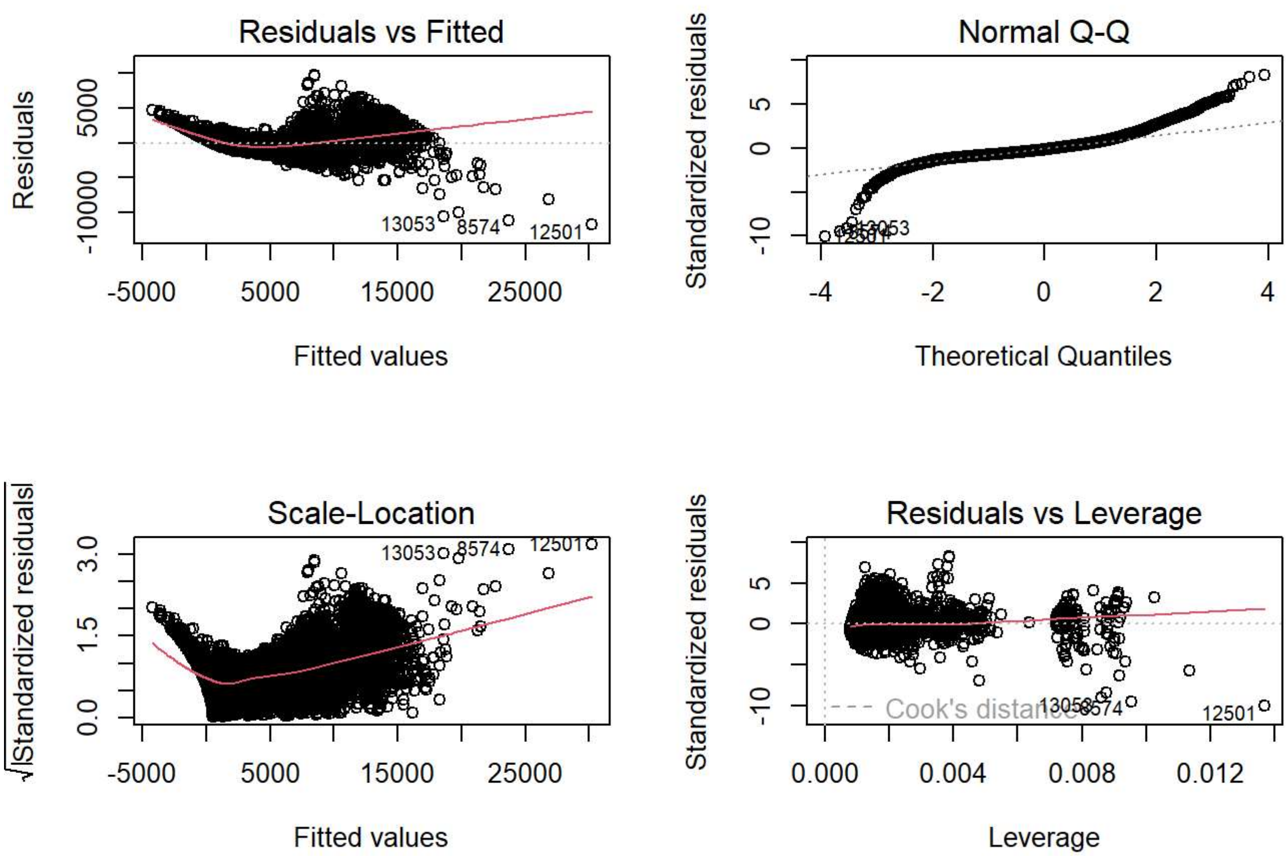
Summary of the model:

```
lm1 <- lm(price~carat+cut+color+clarity, data=train)
summary(lm1)
```

```
##
## Call:
## lm(formula = price ~ carat + cut + color + clarity, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11672.9   -695.6   -198.0    479.8   9653.6
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -7792.90     115.96 -67.206  < 2e-16 ***
## carat         8959.87      25.63 349.594  < 2e-16 ***
## cutGood        676.10      72.58   9.315  < 2e-16 ***
## cutIdeal      1100.63      66.28  16.606  < 2e-16 ***
## cutPremium     954.58      66.97  14.254  < 2e-16 ***
## cutVery Good   923.85      67.67  13.652  < 2e-16 ***
## colorE        -221.38      39.23  -5.643 1.71e-08 ***
## colorF        -325.36      39.32  -8.274  < 2e-16 ***
## colorG        -531.26      38.26 -13.885  < 2e-16 ***
## colorH        -997.04      40.92 -24.363  < 2e-16 ***
## colorI       -1507.04      46.10 -32.690  < 2e-16 ***
## colorJ       -2322.61      56.63 -41.017  < 2e-16 ***
## clarityIF     5742.35     114.22  50.272  < 2e-16 ***
## claritySI1    3908.61      98.28  39.770  < 2e-16 ***
## claritySI2    2889.42      98.66  29.286  < 2e-16 ***
## clarityVS1    4895.61     100.35  48.787  < 2e-16 ***
## clarityVS2    4558.24      98.78  46.143  < 2e-16 ***
## clarityVVS1   5402.11     105.73  51.095  < 2e-16 ***
## clarityVVS2   5301.42     102.96  51.490  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1168 on 11981 degrees of freedom
## Multiple R-squared:  0.9166, Adjusted R-squared:  0.9164
## F-statistic:  7312 on 18 and 11981 DF,  p-value: < 2.2e-16
```

Plot Residuals

```
par(mfrow = c(2,2))
plot(lm1)
```



Evaluate on the test set

The model has correlation of 95%, which is nice, and mse value 1247793

```
pred_lm <- predict(lm1, newdata=test)
cor_lm <- cor(pred_lm, test$price)
mse_lm <- mean((pred_lm-test$price)^2)
rmse_lm <- sqrt(mse_lm)
print(paste('correlation:', cor_lm))
```

```
## [1] "correlation: 0.958328252770474"
```

```
print(paste('mse:', mse_lm))
```

```
## [1] "mse: 1247793.67596431"
```

```
print(paste('rmse:', rmse_lm))
```

```
## [1] "rmse: 1117.04685486523"
```

Linear SVM model

Let us build an SVM1 model on the train set using cost=10 and kernel="linear".

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

```
##
## Call:
## svm(formula = price ~ ., data = train, kernel = "linear", cost = 10,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  10
##       gamma:  0.04166667
##     epsilon:  0.1
##
##
## Number of Support Vectors:  5559
```

Evaluate the linear svm (SVM1)

Now we have a model, we can predict the value of the new data set, we got a correlation of 95%, which is the same as the linear regression, the svm1's mse has increased which is not good.

```
pred_svm1 <- predict(svm1, newdata=test)
cor_svm1 <- cor(pred_svm1, test$price)
mse_svm1 <- mean((pred_svm1 - test$price)^2)
rmse_svm1 <- sqrt(mse_svm1)
print(paste('correlation:', cor_svm1))
```

```
## [1] "correlation: 0.957854953827 84"
```

```
print(paste('mse:', mse_svm1))
```

```
## [1] "mse: 1263911.24648289"
```

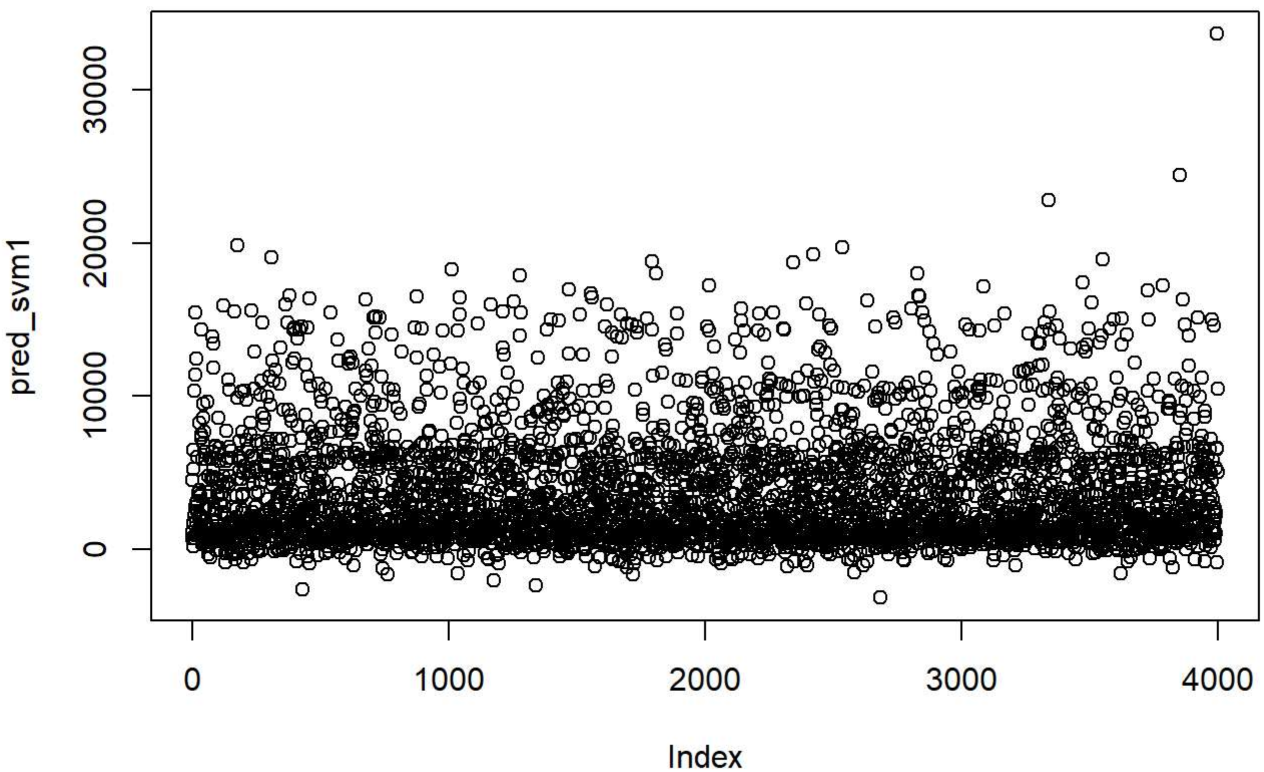
```
print(paste('rmse:', rmse_svm1))
```

```
## [1] "rmse: 1124.23807375613"
```

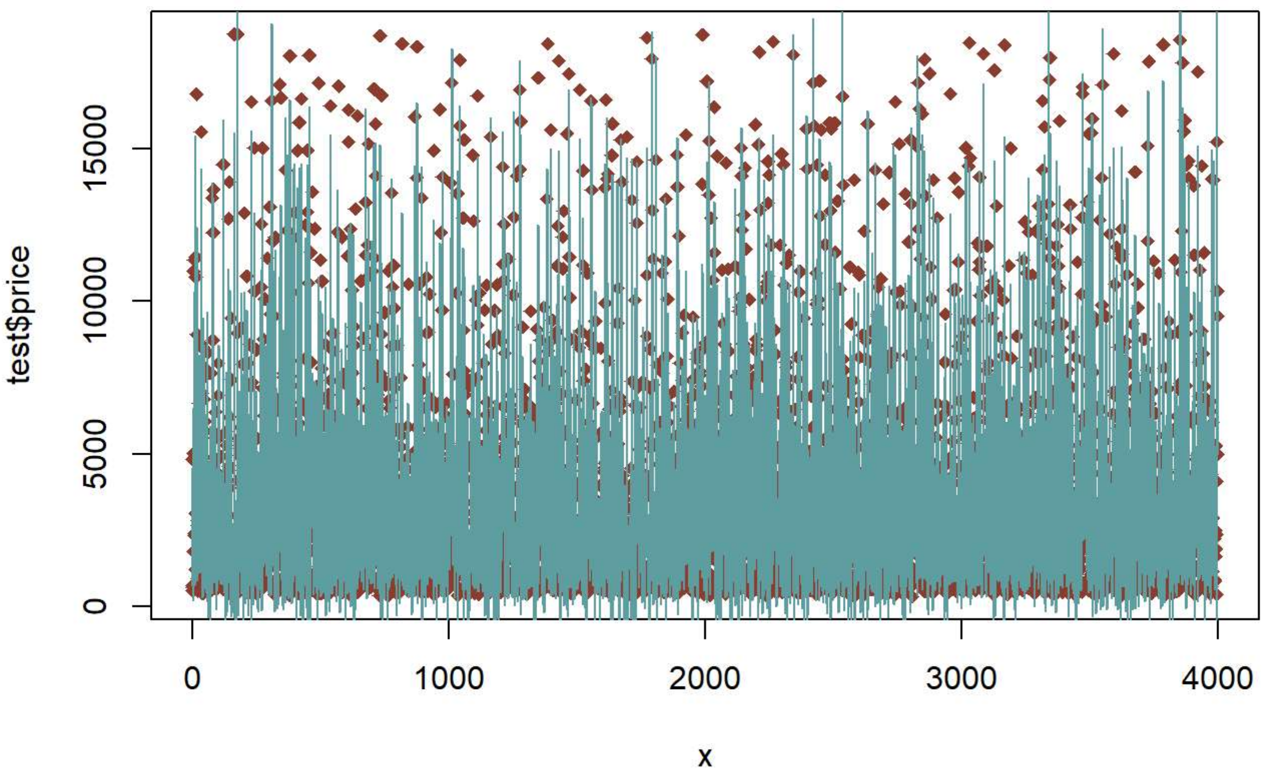
Linear svm Plot

Plot the Support Vectors

```
plot(pred_svm1)
```



```
x = 1:length(test$price)
plot(x,test$price, pch=18, col="coral4")
lines(x,pred_svm1, lwd="1", col="cadetblue")
```



Linear svm Tuning

The cost parameter determines how much slack variables will be allowed.Experiment with various cost values to get the best model.The hyperparameters are tuned on the validation set to not over fit data and not against good principles by letting the algorithm see test data.Larger C have larger margins, smaller C, move the model toward lower bias, higher variance. The summary of tune_svm1 tells us the best cost is 0.1.The next syntax will use the best model value to make predictions on the test data. We got a correlation of 95%, which is the same as the linear regression, the tune_svm1’s mse has increased more than svm1 even though it is tuned.

```r
set.seed(1234)
tune_svm1 <- tune(svm, price~., 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
##   0.1
##
## - best performance: 1518496
##
## - Detailed performance results:
##    cost   error dispersion
## 1 1e-03 2955754   453703.0
## 2 1e-02 1701965   407578.9
## 3 1e-01 1518496   475618.5
## 4 1e+00 1543023   584202.5
## 5 5e+00 1603236   731473.1
## 6 1e+01 1610517   752714.2
## 7 1e+02 1598275   721775.2
```

```r
pred <- predict(tune_svm1$best.model, newdata=test)
cor_svm1_tune <- cor(pred, test$price)
mse_svm1_tune <- mean((pred - test$price)^2)
print(paste('correlation:', cor_svm1_tune))
```

```
## [1] "correlation: 0.956060102251215"
```

```r
print(paste('mse:', mse_svm1_tune))
```

```
## [1] "mse: 1324882.55038628"
```

Polynomial SVM model

Let us build an SVM2 model on the train set using cost=10 and kernel="Polynomial".

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

```
##
## Call:
## svm(formula = price ~ ., data = train, kernel = "polynomial", cost = 10,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  polynomial
##        cost:  10
##      degree:  3
##       gamma:  0.04166667
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  3348
```

Evaluate the polynomial svm (SVM2)

Let us predict the value of the new data set with the model,SVM2. We got a correlation of 98%, greater than the linear regression, svm1, and tune_svm1, the svm1's mse value has has also lowered by half, which is promising.

```
pred_svm2 <- predict(svm2, newdata=test)
cor_svm2 <- cor(pred_svm2, test$price)
mse_svm2 <- mean((pred_svm2 - test$price)^2)
rmse_svm2 <- sqrt(mse_svm2)
print(paste('correlation:', cor_svm2))
```

```
## [1] "correlation: 0.980608147189513"
```

```
print(paste('mse:', mse_svm2))
```

```
## [1] "mse: 585194.350411438"
```

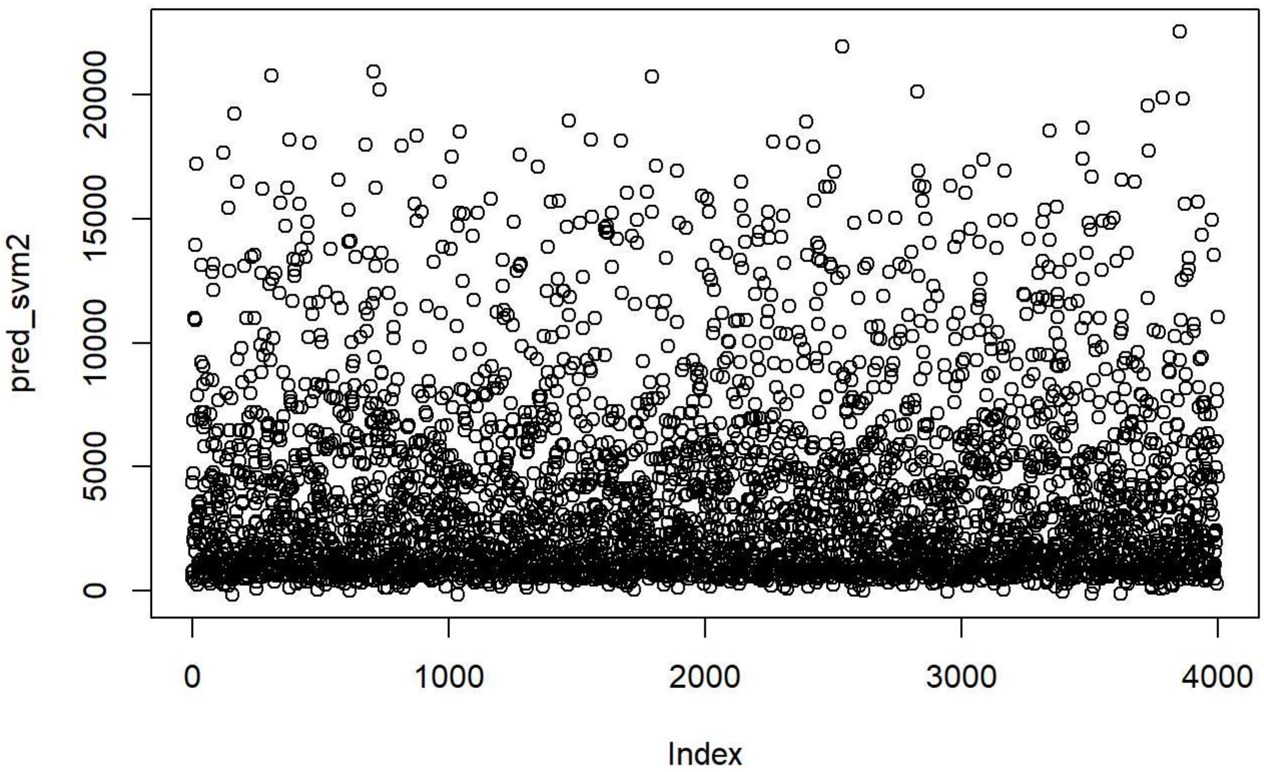
```
print(paste('rmse:', rmse_svm2))
```

```
## [1] "rmse: 764.979967326882"
```

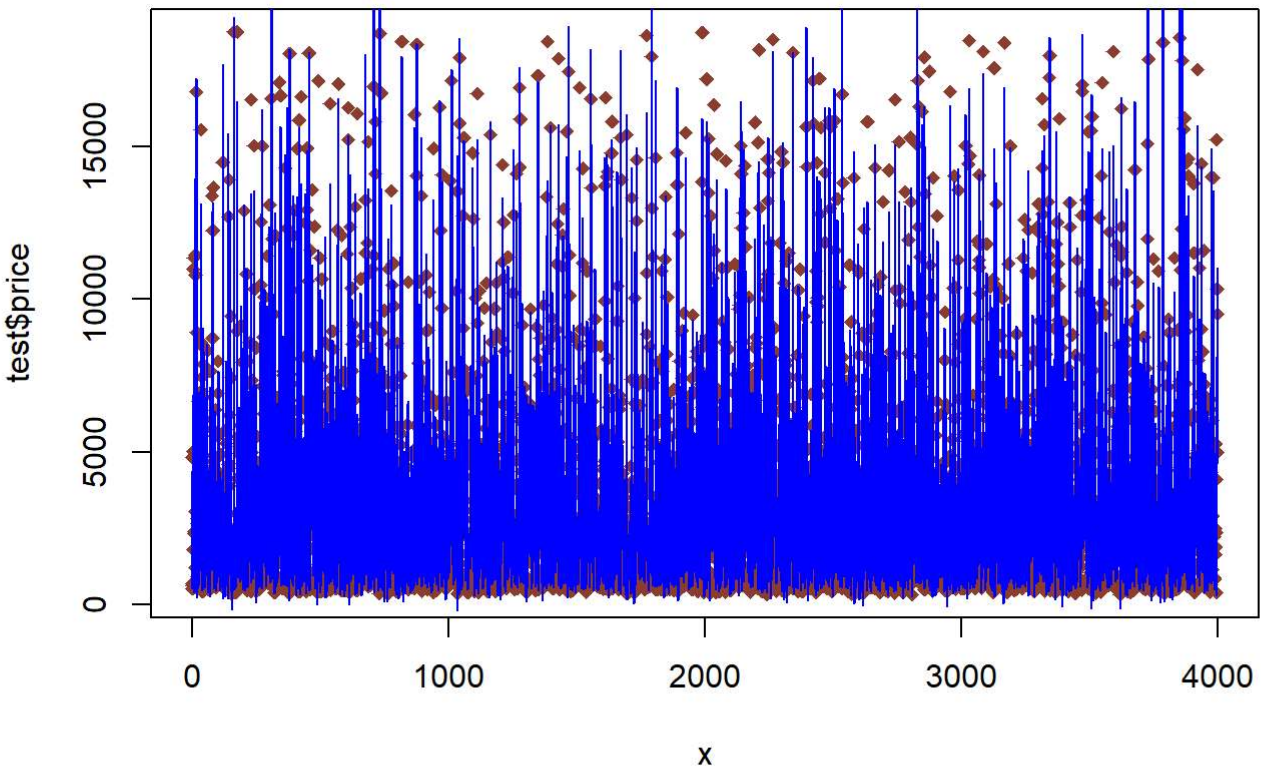
Polynomial svm Plot

Plot the Support Vectors

```
plot(pred_svm2)
```




```
x = 1:length(test$price)
plot(x,test$price, pch=18, col="coral4")
lines(x,pred_svm2, lwd="1", col="blue")
```



Polynomial svm Tuning

The cost parameter determines how much slack variables will be allowed.Experiment with various cost values to get the best model.The hyperparameters are tuned on the validation set to not over fit data and not against good principles by letting the algorithm see test data.Larger C have larger margins, smaller C, move the model toward lower bias, higher variance. The summary of tune_svm2 tells us the best cost is 100.The next syntax will use the best model value to make predictions on the test data. We got a correlation of 98%, the same as svm2,but the mse has slightly increased.

```
set.seed(1234)
tune_svm2 <- tune(svm, price~., data=vald, kernel="polynomial",
            ranges=list(cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
summary(tune_svm2)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   100
##
## - best performance: 690253.5
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-03 15931349.6  1730856.2
## 2 1e-02  8819095.6  2588000.7
## 3 1e-01  5888005.2  3380494.8
## 4 1e+00  1751483.1   386495.0
## 5 5e+00  1010352.7   268020.4
## 6 1e+01   826850.8   246842.7
## 7 1e+02   690253.5   258709.2
```

```
pred <- predict(tune_svm2$best.model, newdata=test)
cor_svm2_tune <- cor(pred, test$price)
mse_svm2_tune <- mean((pred - test$price)^2)
print(paste('correlation:', cor_svm2_tune))
```

```
## [1] "correlation: 0.979385519473522"
```

```
print(paste('mse:', mse_svm2_tune))
```

```
## [1] "mse: 624218.853879549"
```

Radial Kernel SVM model

Let us build SVM3 model on the train set using cost=1 and kernel="radial"

```
svm3 <- svm(price~., data=train, kernel="radial", cost=100, gamma=0.5, scale=TRUE)
summary(svm3)
```

```
##
## Call:
## svm(formula = price ~ ., 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:  4436
```

Evaluate the Kernel svm (SVM3)

Let us predict the value of the new data set with the model,SVM3. We got a correlation of 96%, which is lower than svm2 and tune_svm2, but higher than the linear regression and svm1, similarly the mse value is bigger than mse of svm2 and tune_svm2, but lower than mse of linear regression and svm1.

```
pred_svm3 <- predict(svm3, newdata=test)
cor_svm3 <- cor(pred_svm3, test$price)
mse_svm3 <- mean((pred_svm3 - test$price)^2)
rmse_svm3 <- sqrt(mse_svm3)
print(paste('correlation:', cor_svm3))
```

```
## [1] "correlation: 0.969121499403657"
```

```
print(paste('mse:', mse_svm3))
```

```
## [1] "mse: 928911.279021698"
```

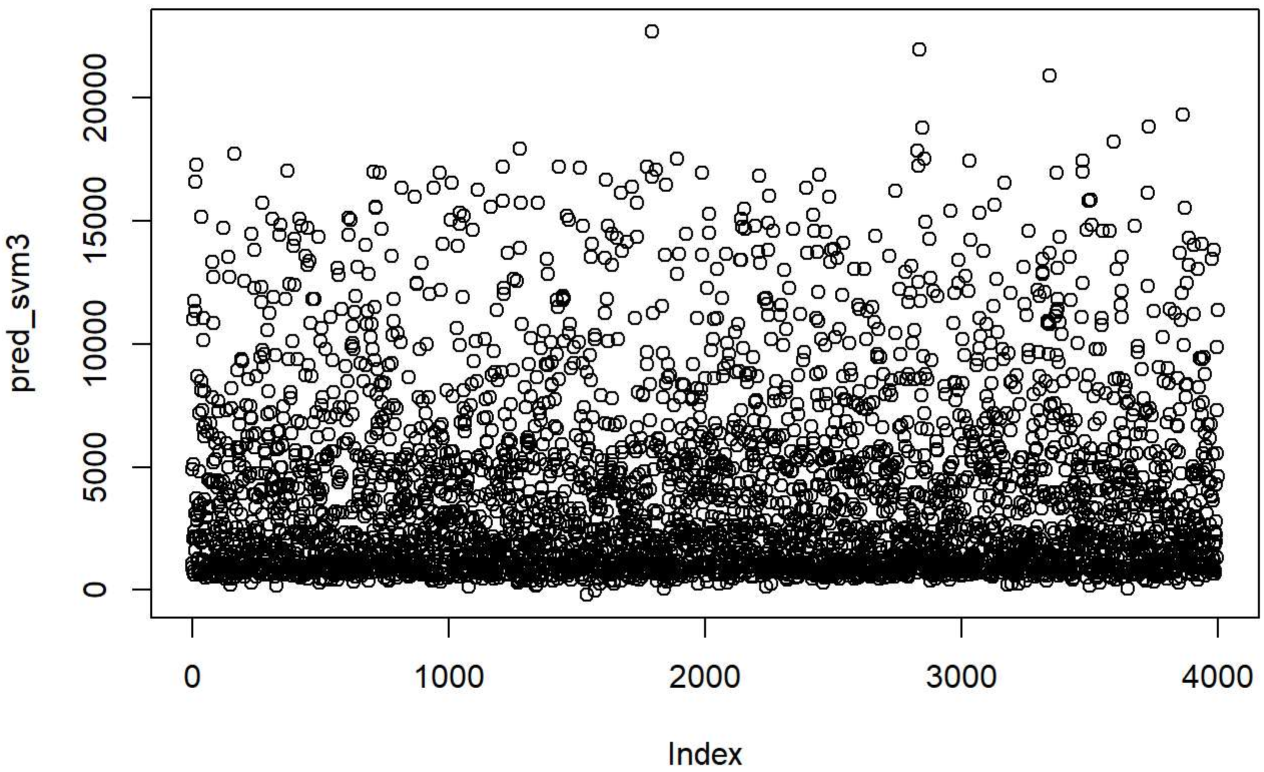
```
print(paste('rmse:', rmse_svm3))
```

```
## [1] "rmse: 963.800435267435"
```

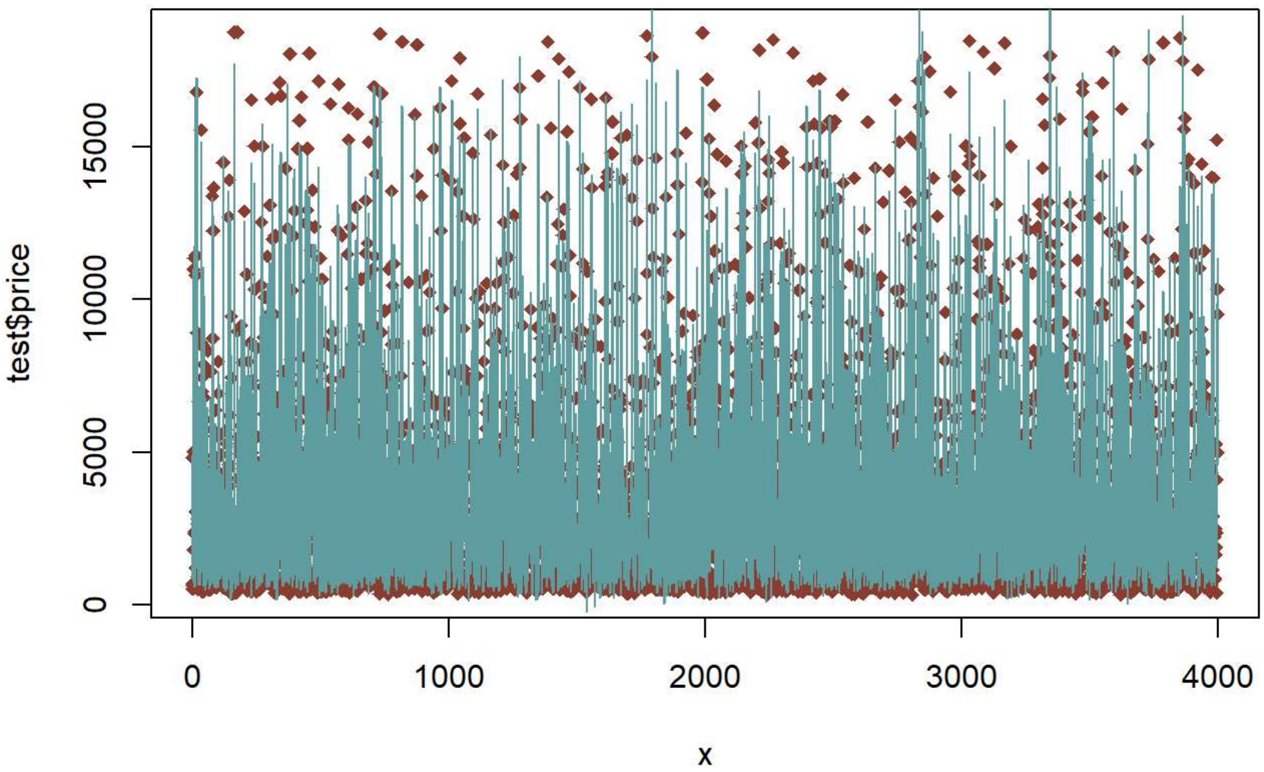
SVM3 Classification Plot

Plot the Support Vectors

```
plot(pred_svm3)
```




```
x = 1:length(test$price)
plot(x,test$price, pch=18, col="coral4")
lines(x,pred_svm3, lwd="1", col="cadetblue")
```



SVM3 Tuning

The gamma hyperparameter is tuned on validation data,larger gamma can over fit and move the model toward high variance, and lower gamma can under fit, leading the model with high bias.The summary of tune_svm3 tells us the best cost is 10 and gamma 0.5.The next syntax will use the best model value to make predictions on the test data. We got a correlation of 96%, which is the same as svm3, the mse is high but not higher than svm1.

```
set.seed(1234)
tune_svm3 <- tune(svm, price~., data=vald, kernel="radial",
                 ranges=list(cost=c(0.1,1,10,100,1000),
                             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: 1508222
##
## - Detailed performance results:
##     cost gamma    error dispersion
## 1  1e-01   0.5  4317965   688936.5
## 2  1e+00   0.5  1528934   311357.9
## 3  1e+01   0.5  1508222   286431.9
## 4  1e+02   0.5  1651674   311060.2
## 5  1e+03   0.5  1706438   332986.5
## 6  1e-01   1.0 10447454  1357461.1
## 7  1e+00   1.0  4023981   712773.2
## 8  1e+01   1.0  3460254   591604.2
## 9  1e+02   1.0  3489903   603310.1
## 10 1e+03   1.0  3489903   603310.1
## 11 1e-01   2.0 14773972  1752714.7
## 12 1e+00   2.0  8811757  1122375.2
## 13 1e+01   2.0  7574706   930926.9
## 14 1e+02   2.0  7576554   933692.4
## 15 1e+03   2.0  7576554   933692.4
## 16 1e-01   3.0 15914903  1834291.3
## 17 1e+00   3.0 10768007  1251623.4
## 18 1e+01   3.0  9506617  1026396.0
## 19 1e+02   3.0  9505987  1027377.5
## 20 1e+03   3.0  9505987  1027377.5
## 21 1e-01   4.0 16454590  1865265.4
## 22 1e+00   4.0 11786009  1308989.6
## 23 1e+01   4.0 10560622  1056566.1
## 24 1e+02   4.0 10560622  1056566.1
## 25 1e+03   4.0 10560622  1056566.1
```

```
pred <- predict(tune_svm3$best.model, newdata=test)
cor_svm3_tune <- cor(pred, test$price)
mse_svm3_tune <- mean((pred - test$price)^2)
print(paste('correlation:', cor_svm3_tune))
```

```
## [1] "correlation: 0.964483472954589"
```

```
print(paste('mse:', mse_svm3_tune))
```

```
## [1] "mse: 1100574.69548411"
```

Which model is the best?

Out of all the models, the best model is svm2,Polynomial SVM model with out tuning, it has 98% correlation, and the mse value is smaller almost by half compared to the mse value of all models built in this svm regression notebook.