Author: Sai Gonuguntla

# SVM Regression

source: https://www.kaggle.com/datasets/camnugent/california-housing-prices (https://www.kaggle.com/datasets/camnugent/california-housing-prices) load packages and data.

```
library(e1071)
setwd("/Users/saiharshithagonuguntla/downloads")
house <- read.csv("housing.csv", header=TRUE)
df <- house[,c(8,9)]
attach(df)
```

## 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",]
```

## explore the training data statistically and graphically

as median income increases, median house vale increases. The histogram shows that as the price of houses increase the frequency of them decreased.

```
head(df,n=3, data = train)
```

| | median_income<br>&lt;dbl&gt; | median_house_value<br>&lt;dbl&gt; |
|---|---|---|
| 1 | 8.3252 | 452600 |
| 2 | 8.3014 | 358500 |
| 3 | 7.2574 | 352100 |

3 rows

```
tail(df,n=5, data = train)
```

| | median_income | median_house_value |
|---|---|---|

| | <dbl> | <dbl> |
|---|---:|---:|
| 20636 | 1.5603 | 78100 |
| 20637 | 2.5568 | 77100 |
| 20638 | 1.7000 | 92300 |
| 20639 | 1.8672 | 84700 |
| 20640 | 2.3886 | 89400 |

5 rows

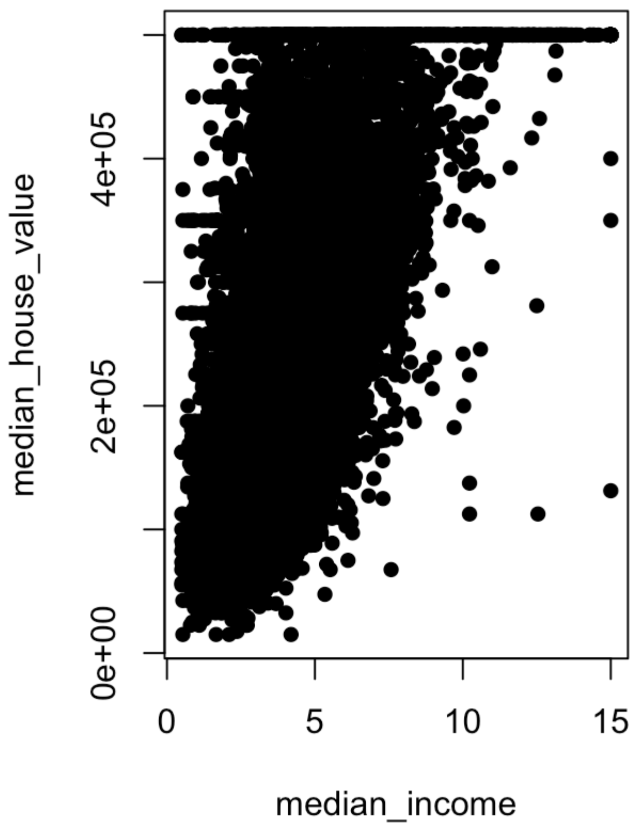
```
mean(median_house_value, data = train)
```

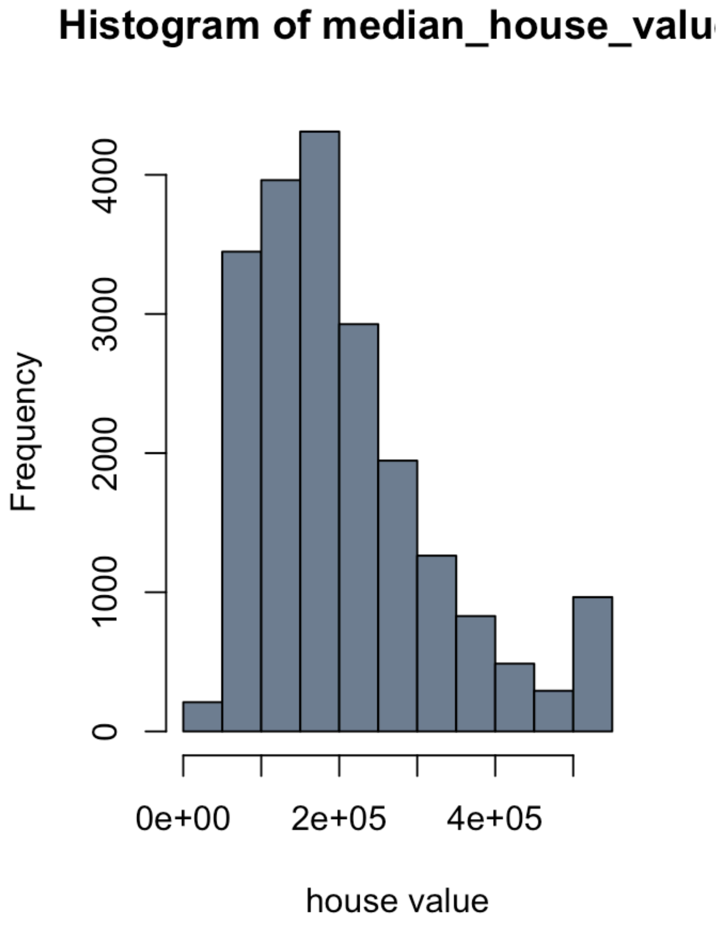
```
## [1] 206855.8
```

```
median(median_house_value, data = train)
```

```
## [1] 179700
```

```
par(mfrow=c(1,2))
plot(median_income, median_house_value,pch=16)
hist(median_house_value, col="slategray",xlab="house value")
```





## Try linear regression

cor is .698 and mse is 1.909

```
lm1 <- lm(median_income~., data=train)
pred <- predict(lm1, newdata=test)
cor_lm1 <- cor(pred, test$median_income)
mse_lm1 <- mean((pred-test$median_income)^2)
print(paste("cor= ", cor_lm1, "mse= ", mse_lm1))
```

```
## [1] "cor=  0.698040321199839 mse=  1.90871375494309"
```

## Try a linear kernel

10849 support vectors. cor is.698 and mse is 1.912. Linear regression did a tiny bit better.

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

```
##
## Call:
## svm(formula = median_income ~ ., data = train, kernel = "linear",
##     cost = 10, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  10
##       gamma:  1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  10849
```

```
pred <- predict(svm1, newdata=test)
cor_svm1 <- cor(pred, test$median_income)
mse_svm1 <- mean((pred - test$median_income)^2)
print(paste("cor_svm1: ", cor_svm1,"mse_svm1: ", mse_svm1))
```

```
## [1] "cor_svm1:  0.698040321199108 mse_svm1:  1.91226899100042"
```

## Tune

best was cost 1, best performance 1.977

```
tune_svm1 <- tune(svm, median_income~., 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
##     1
##
## - best performance: 1.97688
##
## - Detailed performance results:
##    cost    error dispersion
## 1 1e-03 2.098075  0.3424052
## 2 1e-02 1.982449  0.3094545
## 3 1e-01 1.977542  0.3052510
## 4 1e+00 1.976880  0.3049593
## 5 5e+00 1.976882  0.3049673
## 6 1e+01 1.976898  0.3049748
## 7 1e+02 1.976903  0.3049703
```

## Evaluate on best linear svm

“cor_svm1_tune: 0.698040321199833 mse_svm1_tune: 1.9159001821779”

```
pred1 <- predict(tune_svm1$best.model, newdata=test)
cor_svm1_tune <- cor(pred1, test$median_income)
mse_svm1_tune <- mean((pred1 - test$median_income)^2)
print(paste("cor_svm1_tune: ", cor_svm1_tune,"mse_svm1_tune: ", mse_svm1_tune))
```

```
## [1] "cor_svm1_tune:  0.698040321199833 mse_svm1_tune:  1.9159001821779"
```

## Try a polynomial kernel

11107 support vectors. “cor_sv2: 0.614509891698021 mse_svm2: 2.35991139410056”. cor is lower than linear svm and mse is higher than linear svm so linear svm performed better than polynomial

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

```
##
## Call:
## svm(formula = median_income ~ ., data = train, kernel = "polynomial",
##     cost = 10, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  polynomial
##        cost:  10
##      degree:  3
##       gamma:  1
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  11107
```

```
pred2 <- predict(svm2, newdata=test)
cor_svm2 <- cor(pred2, test$median_income)
mse_svm2 <- mean((pred2 - test$median_income)^2)
print(paste("cor_svm2: ", cor_svm2,"mse_svm2: ", mse_svm2))
```

```
## [1] "cor_svm2:  0.614509891698021 mse_svm2:  2.35991139410056"
```

## Tune

cost .001 had the best performance of 2.455

```
tune_svm2 <- tune(svm, median_income~., 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
##  0.001
##
## - best performance: 2.45487
##
## - Detailed performance results:
##    cost    error dispersion
## 1 1e-03 2.454870  0.2229416
## 2 1e-02 2.459750  0.2216008
## 3 1e-01 2.461457  0.2216105
## 4 1e+00 2.461696  0.2211937
## 5 5e+00 2.461605  0.2213224
## 6 1e+01 2.461547  0.2213026
## 7 1e+02 2.459267  0.2232217
```

## Evaluate on best polynomial svm

"cor_svm2_tune: 0.617682760350229 mse_svm2_tune: 2.33220302287721". The tuned mse is slightly lower and the tuned cor is slightly higher

```
pred2 <- predict(tune_svm2$best.model, newdata=test)
cor_svm2_tune <- cor(pred2, test$median_income)
mse_svm2_tune <- mean((pred2 - test$median_income)^2)
print(paste("cor_svm2_tune: ", cor_svm2_tune,"mse_svm2_tune: ", mse_svm2_tune))
```

```
## [1] "cor_svm2_tune:  0.617682760350229 mse_svm2_tune:  2.33220302287721"
```

## Try a radial kernel

Support Vectors: 10814, "cor_svm3: 0.707634685886697 mse_svm3: 1.86228952606457". Radial kernel performed the best out of all kernels as it had the best mse and cor

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

```
##
## Call:
## svm(formula = median_income ~ ., 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:  10814
```

```
pred3 <- predict(svm3, newdata=test)
cor_svm3 <- cor(pred3, test$median_income)
mse_svm3 <- mean((pred3 - test$median_income)^2)
print(paste("cor_svm3: ", cor_svm3,"mse_svm3: ", mse_svm3))
```

```
## [1] "cor_svm3:  0.707634685886697 mse_svm3:  1.86228952606457"
```

## Tune hyperperameters

the best parameters are c 10 and gamma 1. Which had a performance of 1.966.

```
set.seed(1234)
tune.out <- tune(svm, median_income~., 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     1
##
## - best performance: 1.965964
##
## - Detailed performance results:
##      cost gamma    error dispersion
## 1  1e-01   0.5 1.997647  0.2901644
## 2  1e+00   0.5 1.972786  0.2723209
## 3  1e+01   0.5 1.967868  0.2667559
## 4  1e+02   0.5 1.966042  0.2629943
## 5  1e+03   0.5 1.966488  0.2607549
## 6  1e-01   1.0 1.987384  0.2858519
## 7  1e+00   1.0 1.967954  0.2692280
## 8  1e+01   1.0 1.965964  0.2638120
## 9  1e+02   1.0 1.967834  0.2632166
## 10 1e+03   1.0 1.969031  0.2667994
## 11 1e-01   2.0 1.984230  0.2834309
## 12 1e+00   2.0 1.966680  0.2685330
## 13 1e+01   2.0 1.967955  0.2655378
## 14 1e+02   2.0 1.968518  0.2650636
## 15 1e+03   2.0 1.970056  0.2666831
## 16 1e-01   3.0 1.979805  0.2794170
## 17 1e+00   3.0 1.966909  0.2686699
## 18 1e+01   3.0 1.970584  0.2656872
## 19 1e+02   3.0 1.971303  0.2659081
## 20 1e+03   3.0 1.967184  0.2582832
## 21 1e-01   4.0 1.979197  0.2797594
## 22 1e+00   4.0 1.970198  0.2674059
## 23 1e+01   4.0 1.971158  0.2659707
## 24 1e+02   4.0 1.968070  0.2609235
## 25 1e+03   4.0 1.966507  0.2563595
```

###Evaluate on best radial svm “cor_svm4_tune: 0.70581336554732 mse_svm4_tune: 1.87781319461082”

```
pred4 <- predict(tune.out$best.model, newdata=test)
cor_svm4_tune <- cor(pred4, test$median_income)
mse_svm4_tune <- mean((pred4 - test$median_income)^2)
print(paste("cor_svm4_tune: ", cor_svm4_tune,"mse_svm4_tune: ", mse_svm4_tune))
```

```
## [1] "cor_svm4_tune:  0.70581336554732 mse_svm4_tune:  1.87781319461082"
```

## Radial SVM performed the best with a corr of 0.706 and mse of 1.878 followed by linear SVM with a corr of 0.698 and mse of 1.916. The SVM that performed the worst is polynomial with the corr of 0.618 and mse of 2.322. Radial SVM performed the best likely because the dataset used may not be linearly separable. Linear SVM didnt perform tthe best likely because it works the best when there are a lot of features but in this situation only 1 feature is used.