# SVM Regression

Haniyyah Hamid, Jered Hightower

10/18/2022

Data set used: https://www.kaggle.com/datasets/dhirajnirne/california-housing-data

## Loading packages

```
library(e1071)
library(MASS)
```

## Importing data

```
df <- read.csv("housing.csv")
str(df)
```

```
## 'data.frame':    20640 obs. of  10 variables:
##  $ longitude         : num  -122 -122 -122 -122 -122 ...
##  $ latitude          : num  37.9 37.9 37.9 37.9 37.9 ...
##  $ housing_median_age: num  41 21 52 52 52 52 52 52 42 52 ...
##  $ total_rooms       : num  880 7099 1467 1274 1627 ...
##  $ total_bedrooms    : num  129 1106 190 235 280 ...
##  $ population        : num  322 2401 496 558 565 ...
##  $ households        : num  126 1138 177 219 259 ...
##  $ median_income     : num  8.33 8.3 7.26 5.64 3.85 ...
##  $ median_house_value: num  452600 358500 352100 341300 342200 ...
##  $ ocean_proximity   : chr  "NEAR BAY" "NEAR BAY" "NEAR BAY" "NEAR BAY" ...
```

## Clean up data

We will remove unnecessary columns. We will reduce the number of rows as well to be 10k to make tuning faster.

```
df <- df[,c(3, 7, 8, 9)]
df <- head(df, - 10000)
#df$income <- factor(df$income)
str(df)
```

```
## 'data.frame':    10640 obs. of  4 variables:
##  $ housing_median_age: num  41 21 52 52 52 52 52 52 42 52 ...
##  $ households        : num  126 1138 177 219 259 ...
##  $ median_income     : num  8.33 8.3 7.26 5.64 3.85 ...
##  $ median_house_value: num  452600 358500 352100 341300 342200 ...
```

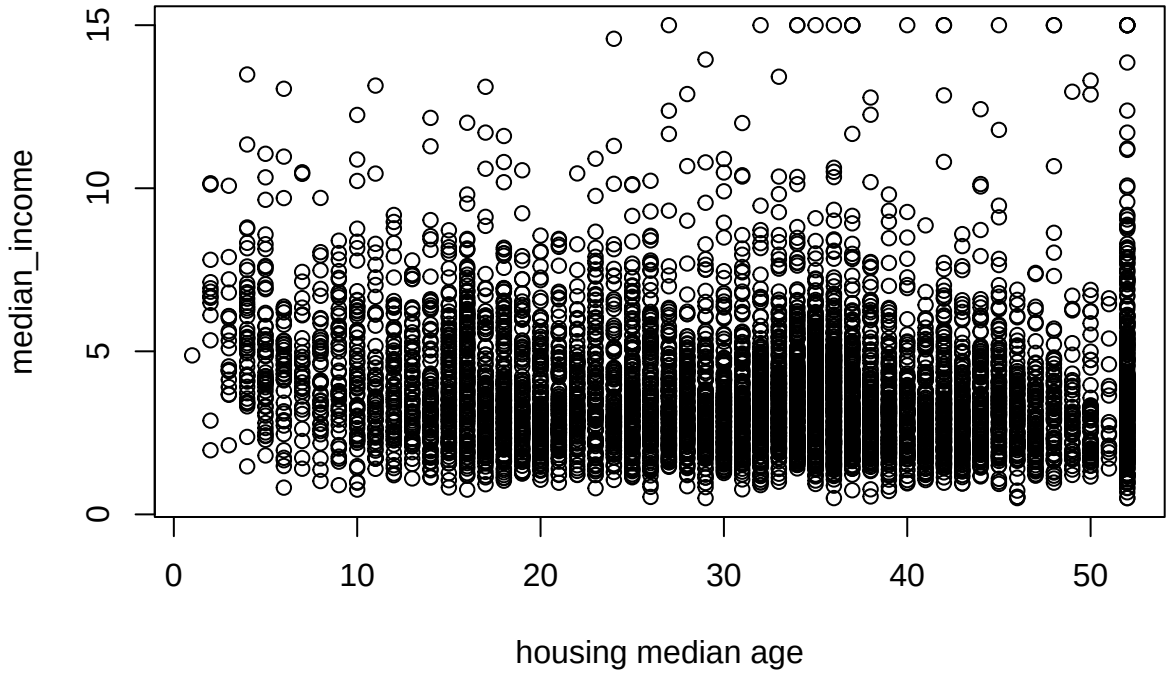
```
#head(df)
```

### Divide into train, test, validate

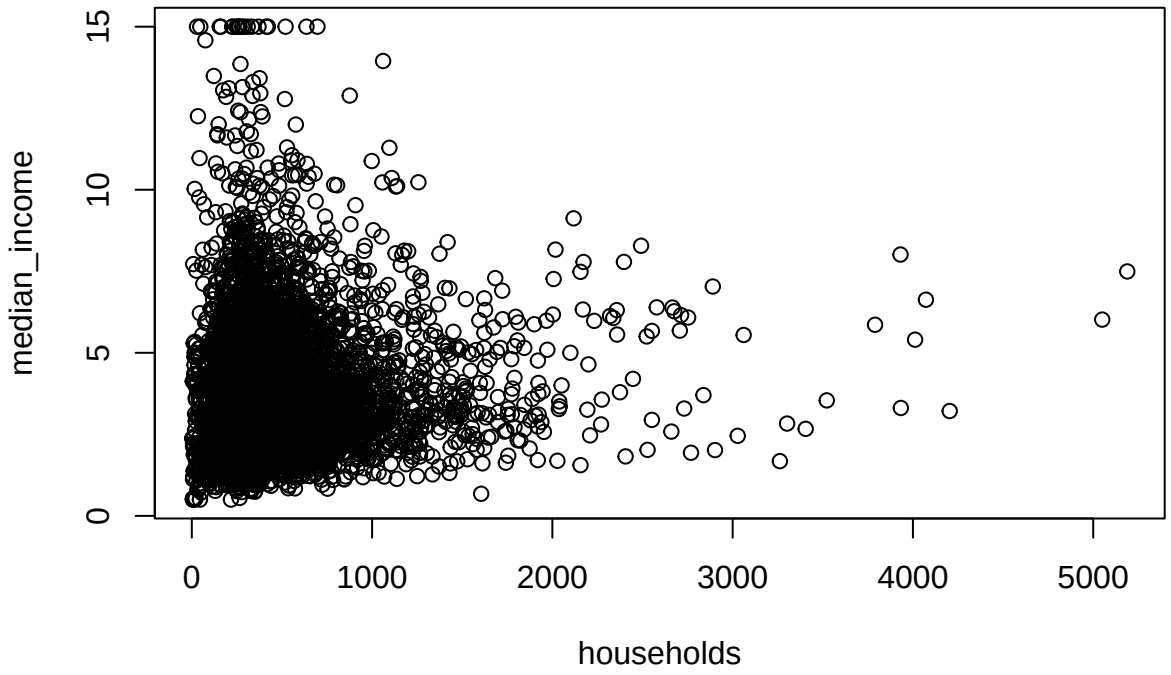
```
set.seed(1234)
spec <-c(train=.6, test=.2, validate=.2)
i <- sample(cut(1:nrow(df),nrow(df)*cumsum(c(0,spec)), labels=names(spec)))
train <- df[i=="train",]
test <- df[i=="test",]
vald <- df[i=="validate",]
```

### Plotting and statistically exploring the training data

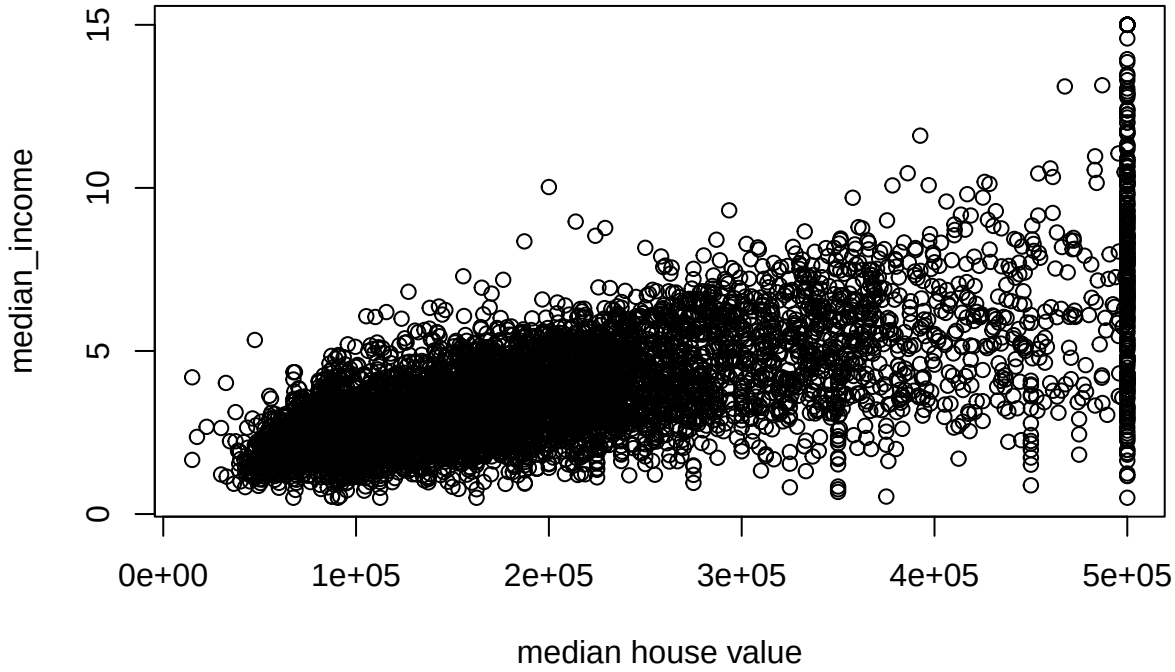
```
plot(train$median_income~train$housing_median_age, xlab="housing median age", ylab="median_income")
```




```
plot(train$median_income~train$households, xlab="households", ylab="median_income")
```




```
plot(train$median_income~train$median_house_value, xlab="median house value", ylab="median_income")
```




```
summary(df)
```

```
##  housing_median_age   households     median_income     median_house_value
##  Min.   : 1.00      Min.   :   2.0   Min.   : 0.4999   Min.   : 14999
##  1st Qu.:22.00      1st Qu.: 272.0   1st Qu.: 2.4666   1st Qu.:121600
##  Median :33.00      Median : 390.0   Median : 3.4111   Median :181050
##  Mean   :31.03      Mean   : 482.8   Mean   : 3.8111   Mean   :207878
##  3rd Qu.:40.00      3rd Qu.: 574.0   3rd Qu.: 4.6736   3rd Qu.:264750
##  Max.   :52.00      Max.   :6082.0   Max.   :15.0001   Max.   :500001
```

Of the three graphs, the relationship between median income and the median house value seemed to be the most linear.The graph of median income vs. households seems to all be clustered in the bottom left corner of the graph. While the graph of the median income vs. housing median age yields a graph that is distributed across the x axis, making it difficult to determine a particular pattern. ## Try linear regression

```
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)
```

## Try a linear kernel

```
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:  0.3333333
##     epsilon:  0.1
##
##
## Number of Support Vectors:  5472
```

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

## Tune

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

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     5
##
## - best performance: 1.934334
##
## - Detailed performance results:
##    cost    error dispersion
## 1 1e-03 2.293132  0.4154893
```

```
## 2 1e-02 1.940273  0.3596556
## 3 1e-01 1.934894  0.3506409
## 4 1e+00 1.934530  0.3489227
## 5 5e+00 1.934334  0.3487512
## 6 1e+01 1.934380  0.3487129
## 7 1e+02 1.934372  0.3489700
```

### Evaluate on best linear SVM

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

### Try a polynomial kernel

```
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:  0.3333333
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  5611
```

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

### Try a radial kernel

```
svm3 <- svm(median_income~., data=train, kernel="radial", cost=10, gamma=1, scale=TRUE)
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:  5375
```

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

### Tune hyperparameters

```
set.seed(1234)
tune.out <- tune(svm, median_income~., data=vald, kernel="radial", ranges=list(cost=c(0.1, 1, 10, 100,

summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1   0.5
##
## - best performance: 1.88658
##
## - Detailed performance results:
##     cost gamma    error dispersion
## 1  1e-01   0.5 2.081694  0.5409536
## 2  1e+00   0.5 1.886580  0.4149945
## 3  1e+01   0.5 1.888237  0.3744141
## 4  1e+02   0.5 2.006697  0.3468583
## 5  1e+03   0.5 2.642510  0.9184082
## 6  1e-01   1.0 2.145600  0.5637796
## 7  1e+00   1.0 1.900661  0.4150708
## 8  1e+01   1.0 1.991709  0.3633376
## 9  1e+02   1.0 2.363641  0.5089473
## 10 1e+03   1.0 4.306751  1.4651162
## 11 1e-01   2.0 2.292982  0.6115231
## 12 1e+00   2.0 1.964157  0.4065734
## 13 1e+01   2.0 2.136911  0.3939455
## 14 1e+02   2.0 2.883719  0.6769403
## 15 1e+03   2.0 4.712409  0.8883540
## 16 1e-01   3.0 2.404436  0.6513568
## 17 1e+00   3.0 2.019773  0.4075856
## 18 1e+01   3.0 2.292819  0.4146198
## 19 1e+02   3.0 3.026054  0.5804927
## 20 1e+03   3.0 6.440924  2.9993571
## 21 1e-01   4.0 2.510757  0.6839675
## 22 1e+00   4.0 2.062425  0.3988601
```

```
## 23 1e+01   4.0 2.376883  0.4352241
## 24 1e+02   4.0 3.332250  0.5747416
## 25 1e+03   4.0 7.273278  2.2675758
```

```
svm4 <- svm(median_income~., data=train, kernel="radial", cost=1, gamma=0.5, scale=TRUE)
summary(svm4)
```

```
##
## Call:
## svm(formula = median_income ~ ., data = train, kernel = "radial",
##     cost = 1, gamma = 0.5, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.5
##     epsilon:  0.1
##
##
## Number of Support Vectors:  5403
```

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

### Comparing statistics of each of the SVM kernels

First, the correlations of each kernel.

```
print(paste("cor_lm1 = ", cor_lm1))
```

```
## [1] "cor_lm1 =  0.740081889222345"
```

```
print(paste("cor_svm1_tune = ", cor_svm1_tune))
```

```
## [1] "cor_svm1_tune =  0.740054481751177"
```

```
print(paste("cor_svm2 = ", cor_svm2))
```

```
## [1] "cor_svm2 =  0.66297783743742"
```

```
print(paste("cor_svm3 = ", cor_svm3))
```

```
## [1] "cor_svm3 =  0.737893942891624"
```

```
print(paste("cor_svm4 = ", cor_svm4))
```

```
## [1] "cor_svm4 =  0.747757223387749"
```

We see the greatest correlation was found when performing logistic regression and the linear kernel of SVM is almost the exact same, meaning the linear decision boundary probably yielded the best correlation of the 3 kernels. The worst was the polynomial SVM kernel.

Now, the mean standard errors of each kernel.

```
print(paste("mse_lm1 = ", mse_lm1))
```

```
## [1] "mse_lm1 =  1.77490141707553"
```

```
print(paste("mse_svm1_tune = ", mse_svm1_tune))
```

```
## [1] "mse_svm1_tune =  1.77169381292695"
```

```
print(paste("mse_svm2 = ", mse_svm2))
```

```
## [1] "mse_svm2 =  2.23768862007334"
```

```
print(paste("mse_svm3 = ", mse_svm3))
```

```
## [1] "mse_svm3 =  1.78560439538668"
```

```
print(paste("mse_svm4 = ", mse_svm4))
```

```
## [1] "mse_svm4 =  1.72927586428152"
```

We see the lowest MSE was found when performing the linear SVM kernel, which also yielded about the same as the logistic regression model. The highest MSE of the kernels was the polynomial SVM kernel.

The linear SVM kernel also yielded the most support vectors as well.

Therefore, we can assume that because of all the given results, the linear SVM kernel performed the best of the 3 kernels.