The Prediction of Median House Prices for California Districts

Şeyma Günönü

Supervised Learning: Regression Task

Regression task/problem is one of the task/problem of supervised learning models. In this task, we will interest the regression problem about the prediction of median house prices for California districts. In this regression task; Y: house prices (Our target variable is house prices which is numeric and continuous variable.), X1: total rooms, X2: population, X3: households, … (These are our features. In regression analysis, we called them independent or explanatory variables.).

Packages

Before we start to describing our dataset, we need to install some packages.

```
 #install.packages("readr") # This package provide us a read to use dataset.
 #install.packages("ggplot2") # This package has visualization tools. So, we can
#use for visualize our dataset.
 #install.packages("car") # car package is related with checking assumption of
#linear regression models.
library(readr)
library(ggplot2)
library(car)
```

Dataset

We use a data set about median house prices for California districts derived from the 1990 census from Kaggle. Our data set is a csv file, so we use readr package or import data set from environment window to add the data set.

```
housing <- read_csv("housing.csv")
```

Using str() function, we look at our data set as a data frame.

```
str(housing)
```

```
spc_tbl_ [20,640 x 10] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ longitude         : num [1:20640] -122 -122 -122 -122 -122 ...
 $ latitude          : num [1:20640] 37.9 37.9 37.9 37.9 37.9 ...
 $ housing_median_age: num [1:20640] 41 21 52 52 52 52 52 52 42 52 ...
 $ total_rooms       : num [1:20640] 880 7099 1467 1274 1627 ...
 $ total_bedrooms    : num [1:20640] 129 1106 190 235 280 ...
 $ population        : num [1:20640] 322 2401 496 558 565 ...
 $ households        : num [1:20640] 126 1138 177 219 259 ...
 $ median_income     : num [1:20640] 8.33 8.3 7.26 5.64 3.85 ...
 $ median_house_value: num [1:20640] 452600 358500 352100 341300 342200 ...
 $ ocean_proximity   : chr [1:20640] "NEAR BAY" "NEAR BAY" "NEAR BAY" "NEAR BAY" ...
 - attr(*, "spec")=
  .. cols(
  ..   longitude = col_double(),
  ..   latitude = col_double(),
  ..   housing_median_age = col_double(),
  ..   total_rooms = col_double(),
  ..   total_bedrooms = col_double(),
  ..   population = col_double(),
  ..   households = col_double(),
  ..   median_income = col_double(),
  ..   median_house_value = col_double(),
  ..   ocean_proximity = col_character()
  .. )
 - attr(*, "problems")=<externalptr>
```

Our variables are;

1. longitude: A measure of how far west a house is; a higher value is farther west

2. latitude: A measure of how far north a house is; a higher value is farther north
3. housingMedianAge: Median age of a house within a block
4. totalRooms: Total number of rooms within a block
5. totalBedrooms: Total number of bedrooms within a block
6. population: Total number of people residing within a block
7. households: Total number of households
8. medianIncome: Median income for households within a block of houses (measured in tens of thousands of US Dollars)
9. medianHouseValue: Median house value for households within a block (measured in US Dollars)
10. oceanProximity: Location of the house ocean/sea

We can say that, we have only one categorical variable which is the distance of the location of the houses to the sea. And, the others are numeric and continuous variables. Our target variable is median_house_value, and it takes around ranges, so we can confirm that again this is a regression task.

```
dim(housing)
```

```
[1] 20640    10
```

Using `dim()` function, we can tell that dimension of data set. As you can see our data set have 20640 observation and 10 variables. Before training our model we have to exclude the all 'NA' variables in data set.

```
housing <- na.exclude(housing)
```

Training

We are trying to train a linear regression model on the California houses prices data set to predict of median house prices which is our target variable.

1 - Splitting the Data set

In this part, we have to split the data set to predict target variable using a set of features. First, we should split the data set as a two subset which are train and test set by using `sample()` function. Before this, we must set a seed to get same observations. We create a sample, then assign this sample as a index object. This sample starts with a sequence from 1 to number of row of housing object, and we should split 80% of the data set as a train set and 20% as a test set. While we assign our index object to the train, we assign our -index object to the test.

```r
set.seed(123)
index <- sample(1 : nrow(housing), round(nrow(housing) * 0.80))
train <- housing[index, ]
test <- housing[-index, ]
```

2 - Training a linear regression model

This step, our aim is to predict the target variable. We can use `lm()` function to train the model. First, we take our train set and add our target variable which is median_house_value to the function, and then add all the features with a '.'. After that, we assign a new object called lrm_model.

```r
lrm_model <- lm(median_house_value ~., data = train)
lrm_model
```

```
Call:
lm(formula = median_house_value ~ ., data = train)

Coefficients:
               (Intercept)                  longitude
                -2.292e+06                 -2.708e+04
                  latitude         housing_median_age
                -2.576e+04                  1.057e+03
              total_rooms             total_bedrooms
                -6.195e+00                  9.151e+01
               population                 households
                -3.734e+01                  5.891e+01
            median_income      ocean_proximityINLAND
                 3.923e+04                 -3.781e+04
    ocean_proximityISLAND    ocean_proximityNEAR BAY
                 1.709e+05                 -3.503e+03
ocean_proximityNEAR OCEAN
                 4.828e+03
```

After running the code, we get the model formula, the estimated values of the model parameters which are betas, and finally the trained data. We can directly see which variables are categorical and which are numeric, but we use `summary()` function to see more details.

```r
summary(lrm_model)
```

```
Call:
lm(formula = median_house_value ~ ., data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-555127  -42863  -10715   28683  756919

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)               -2.292e+06  9.858e+04 -23.254  < 2e-16 ***
longitude                 -2.708e+04  1.143e+03 -23.696  < 2e-16 ***
latitude                  -2.576e+04  1.127e+03 -22.849  < 2e-16 ***
housing_median_age         1.057e+03  4.940e+01  21.407  < 2e-16 ***
total_rooms               -6.195e+00  8.724e-01  -7.102 1.28e-12 ***
total_bedrooms             9.151e+01  7.551e+00  12.119  < 2e-16 ***
population                -3.734e+01  1.181e+00 -31.606  < 2e-16 ***
households                 5.891e+01  8.214e+00   7.172 7.71e-13 ***
median_income              3.923e+04  3.761e+02 104.308  < 2e-16 ***
ocean_proximityINLAND     -3.781e+04  1.955e+03 -19.339  < 2e-16 ***
ocean_proximityISLAND      1.709e+05  3.450e+04   4.953 7.38e-07 ***
ocean_proximityNEAR BAY   -3.503e+03  2.141e+03  -1.636    0.102
ocean_proximityNEAR OCEAN  4.828e+03  1.757e+03   2.748    0.006 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 68910 on 16333 degrees of freedom
Multiple R-squared:  0.6459,    Adjusted R-squared:  0.6457
F-statistic:  2483 on 12 and 16333 DF,  p-value: < 2.2e-16
```

The output shows us the model formula, residuals which are quartiles and min-max values, and coefficients. We can see the results and make inferences using these significance tests for each parameter. Especially, we can interpret the multiple R^2 and adjusted R^2 to see the model performance on the train data. Finally, we can see F-statistics and p-value which are shows us the significance of the model. If we want to make inferences using only multiple R^2, and adjusted R^2, we can say that it explains the model around 64% which is not good but not that bad.

3 - Measuring model performance

We have trained data that we don't know anything about model performance, so we need to check model performance. We should calculate the predicted value of target varible on test set. We should not forget to exclude the actual target variable. After predicting the values instead of looking all of the values, we can check the first six values using `head()` function.

```
predicted_house_prices <- predict(lrm_model, test[, -9])
head(predicted_house_prices)
```

```
      1        2        3        4        5        6
378962.7 255538.2 257098.7 188356.3 160631.5 222061.5
```

Obtaining the predicted values of target variables, we can calculate the error and check the first six values like we checked the predicted values.

```
error <- test$median_house_value - predicted_house_prices
head(error)
```

```
        1          2          3          4          5          6
-26862.66  -14138.23  -15298.66  -88656.28  -55731.52 -112361.46
```

After finding the error, we can use these metrics which are mean squared error, root mean squared error, and median absolute error to find the which metrics will be fit our model performance.

MSE is a metric that shows how much the estimated value differs from the actual number. On the other hand, RMSE is used when the MSE value is incomparably large and thus allows us to make more comfortable inferences. Shortly, RMSE is better in model performance when dealing with large error values.

On this data set, we will use root mean squared error (RMSE).

```
rmse_model <- sqrt(mean(error ^ 2))
rmse_model
```

```
[1] 67676.82
```

4 - Checking any problem related to over-fitting and under-fitting

Using RMSE, we can compare the model performance on train and test set. As a result, we can find out, if there is a problem with over-fitting or under-fitting.

```
rmse_train <- sqrt(mean((lrm_model$residuals) ^ 2))
rmse_test <- rmse_model
rmse_train - rmse_test
```

```
[1] 1207.155
```

The output that we get is positive and for the performance of the model, we can say that the train set is better than the test set. Since our model is better in the train set, it may be a cause under-fitting problem.

Adding New Observation

We are creating a new data frame called new_observation, and using our model to predict this observation.

```
new_observation <- data.frame(longitude = -117.90,
                              latitude = 33.92,
                              housing_median_age = 39,
                              total_rooms = 1728,
                              total_bedrooms = 293,
                              population = 547,
                              households = 219,
                              median_income = 2.2015,
                              ocean_proximity = "INLAND")
```

```
predicted_house_prices2 <- predict(lrm_model, new_observation)
predicted_house_prices2
```

```
       1
124678.8
```