PS1: Regression

DS 5220: Supervised Machine Learning and Learning Theory
Omer Seyfeddin Koc

February 17, 2023

1 Quadratic Regression

You are given the following set of five points: $(-2, 0)$, $(-1, 0)$, $(0, 1)$, $(1, 0)$ and $(2, 0)$, where the first element of each pair is the predictor variable x, and the second element is the response variable y.

1. Find the parabola (a function of the form $y = ax^2 + bx + c = 0$) that best fits these data points using the normal equations.

   Solution : To find the parabola that best fits the given data points using the normal equations, we need to solve for the coefficients a, b, and c in the equation

   There are 3 different equations to solve for this quadratic functions ($n = 5$):

$$\sum_{i=1}^{n} x_i^2 a + \sum_{i=1}^{n} x_i b + nc = \sum_{i=1}^{n} y_i$$
$$\sum_{i=1}^{n} x_i^3 a + \sum_{i=1}^{n} x_i^2 b + \sum_{i=1}^{n} x_i c = \sum_{i=1}^{n} x_i y_i$$
$$\sum_{i=1}^{n} x_i^4 a + \sum_{i=1}^{n} x_i^3 b + \sum_{i=1}^{n} x_i^2 c = \sum_{i=1}^{n} x_i^2 y_i$$

   Values for 5 different points are calculated in the table below.

Points	x	y	x^2	x^3	x^4	xy	x^2y
(-2,0)	-2	0	4	-8	16	0	0
(-1,0)	-1	0	1	-1	1	0	0
(0,1)	0	1	0	0	0	0	0
(1,0)	1	0	1	1	1	0	0
(2,0)	2	0	4	8	16	0	0
SUM	$\sum x = 0$	$\sum y = 1$	$\sum x^2 = 10$	$\sum x^3 = 0$	$\sum x^4 = 34$	$\sum xy = 0$	$\sum x^2y = 0$

(1)

   Solve for a, b, and c by isolating each of these variables.

$$\begin{cases} (10)a + (0)b + (5)c = 1 \\ (0)a + (10)b + (0)c = 0 \\ (34)a + (0)b + (10)c = 0 \end{cases} \rightarrow \begin{cases} 10a + 5c = 1 \\ 10b = 0 \\ 34a + 10c = 0 \end{cases} \rightarrow \begin{cases} 10a + 5c = 1 \\ b = 0 \\ 34a + 10c = 0 \end{cases}$$

$$\begin{cases} 10a + 5c = 1 \\ 34a + 10c = 0 \end{cases} \rightarrow \begin{cases} 34a = -10c \\ 17a = -5c \\ a = -\frac{5}{17}c \end{cases} \rightarrow \begin{cases} 10(-\frac{5}{17}c) + 5c = 1 \\ (\frac{35}{17}c) = 1 \end{cases} \rightarrow \begin{cases} c = \frac{17}{35} \\ a = -\frac{1}{7}) \end{cases}$$

a, b, and c values are:

$$a = -\frac{1}{7}$$

$$b = 0$$

$$c = \frac{17}{35}$$

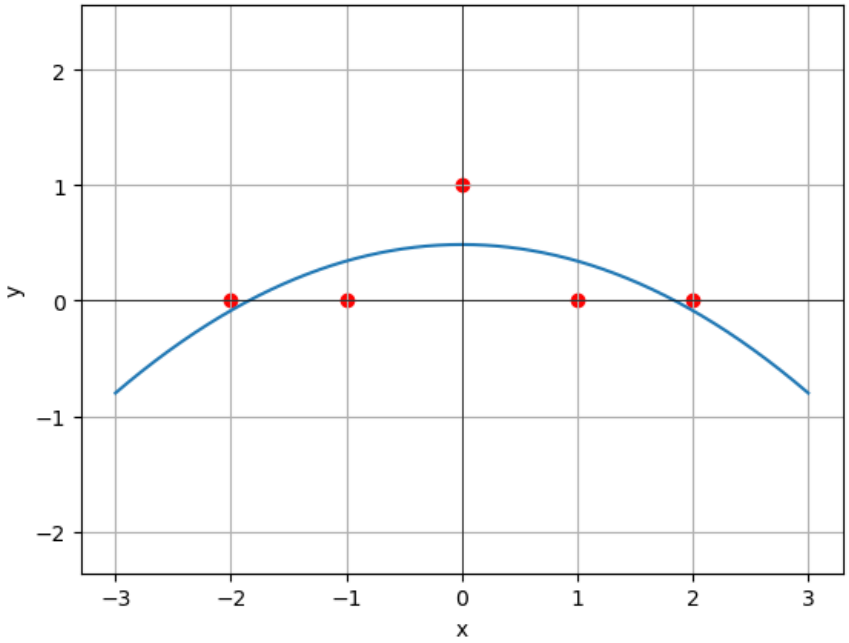
Insert these values into our quadratic equation:

$$y = (-\frac{1}{7})x^2 + (0)x + (\frac{17}{35}) = 0$$

$$y = -\frac{1}{7}x^2 + \frac{17}{35}$$

2. Draw a plot that shows the data points and the best fitting parabola.

   Solution : The following graph is obtained by plotting the relevant functions and points in python.



2 Regularized Linear Regression

Recall the problem of linear regression and the derivation of its various learning algorithms. We are given a data set $\{(\mathbf{x}_i + y_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^d$ (i.e., each data point has d features) and $y_1 \in \mathbb{R}$. The hypothesis class we consider in linear regression is, for $w \in \mathbf{R}^{d+1}$:

$$h(\mathbf{x}) = w_0 + w_1 x_1 + \ldots + w_d x_d = w_0 + \sum_{j=1}^{d} w_j x_j$$

In ridge regression, we add a regularization term to the ordinary least squares error function, such that the cost function to minimize is:

$$J(\mathbf{w}) = \sum_{i=1}^{n} \left(h\left(\mathbf{x}_i\right) - y_i\right)^2 + \lambda \sum_{j=0}^{d} w_j^2$$

1. Prove that the vector $\mathbf{w}^*$ that minimizes $J(\mathbf{w})$ is:

$$\mathbf{w}^* = \left(X^T X + \lambda I\right)^{-1} X^T \mathbf{y}$$

where X is the $n \times (d+1)$ design matrix, whose ith row is $\mathbf{x}_i$, and $\mathbf{y} = (y_1 + \ldots, y_n)^T$.

Solution : Ridge regression is the L2 regularized version of linear regression. To find the vector $\mathbf{w}^*$ that minimizes $J(w)$, we can take the derivative of $J(w)$ with respect to w, set it equal to zero, and solve for w.

$$J(\mathbf{w}) = \sum_{i=1}^{n} \left(h\left(\mathbf{x}_i\right) - y_i\right)^2 + \lambda \sum_{j=0}^{d} w_j^2 = \|X\mathbf{w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$$

$$J(\mathbf{w}) = \|\mathbf{y} - X\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$$

Since for a vector v, $\|v\|_2^2 = v^T v$, so

$$\|y - X\mathbf{w}\|_2^2 = (y - X\mathbf{w}^*)^T (y - X\mathbf{w}^*) = y^T y - y^T X\mathbf{w}^* - (\mathbf{w}^*)^T X^T y + (\mathbf{w}^*)^T X^T X\mathbf{w}^*$$

$$\|\mathbf{w}\|_2^2 = \mathbf{w}^T \mathbf{w}$$

Thus

$$J(\mathbf{w}) = y^T y - y^T X\mathbf{w}^* - (\mathbf{w}^*)^T X^T y + (\mathbf{w}^*)^T X^T X\mathbf{w}^* + \lambda (\mathbf{w}^*)^T \mathbf{w}^*$$

Now, as we want to minimize J by setting $\frac{\partial J}{\partial \mathbf{w}}$

$$\frac{\partial J}{\partial \mathbf{w}^*} = \frac{\partial}{\partial \mathbf{w}^*} \left[y^T y - y^T X\mathbf{w}^* - (\mathbf{w}^*)^T X^T y + (\mathbf{w}^*)^T X^T X\mathbf{w}^* + \lambda (\mathbf{w}^*)^T \mathbf{w}^*\right]$$

Formula for Matrix calculus

$$\frac{\partial \mathbf{a}}{\partial \mathbf{w}} = \mathbf{0}, \quad \frac{\partial \mathbf{w}}{\partial \mathbf{w}} = \mathbf{I}, \quad \frac{\partial \mathbf{A}\mathbf{w}}{\partial \mathbf{w}} = \mathbf{A}, \quad \frac{\partial \mathbf{w}^\mathbf{T}\mathbf{A}}{\partial \mathbf{w}} = \mathbf{A}^\mathbf{T}, \quad \frac{\partial \mathbf{w}^\mathbf{T}\mathbf{A}\mathbf{w}}{\partial \mathbf{w}} = \left(\mathbf{A} + \mathbf{A}^\mathbf{T}\right)\mathbf{w}$$

Since $\mathbf{w}^T \mathbf{w}$ is symmetric matrix, and therefore

$$\frac{\partial \mathbf{w}^\mathbf{T}\mathbf{A}\mathbf{w}}{\partial \mathbf{w}} = 2\mathbf{A}\mathbf{w}$$

So, our equation equals:

$$\frac{\partial J}{\partial \mathbf{w}^*} = \left[-y^T X - y^T X + 2X^T X\mathbf{w}^* + 2\lambda \mathbf{w}^*\right]$$

$$\frac{\partial J}{\partial \mathbf{w}^*} = -2X^T y + 2\left(X^T X + \lambda I\right)\mathbf{w}^*$$

Equate the derivative function to zero and find $\mathbf{w}^*$.

$$-2X^T y + 2\left(X^T X + \lambda I\right)\mathbf{w}^* = 0$$

$$X^T y = \left(X^T X + \lambda I\right)\mathbf{w}^*$$

$$\mathbf{w}^* = \left(X^T X + \lambda I\right)^{-1} X^T y$$

Therefore, we prove that the vector $\mathbf{w}^*$ that minimize $J(\mathbf{w})$.

2. λ is known as a regularization constant; it is a hyperparameter that is chosen by the algorithm designer and fixed during learning. What do you expect to happen to the optimal w^* when $\lambda = 0$? $\lambda \to +\infty$? $\lambda \to -\infty$? Explain your answer.

   Solution : Ridge Regression is obtained by adding a regularization term ($\lambda \sum_{j=0}^{d} w_j^2$) to our cost function in linear regression. With this addition, the learning algorithm both learns the data and tries to keep the model weights as small as possible. The λ hyper-parameter controls how much we regularize the model.

   The λ is actually the parameter that provides the balance between the fit of the model to the data and over-learning, and this parameter must be chosen correctly in order for the Ridge regression to achieve its purpose.

   If $\lambda = 0$, model will result in the traditional least squares coefficients since the penalty term has no impact. This can result in over-fitting, in which the model becomes overly complex and fits the training data exceptionally well. The optimal value of $\mathbf{w}$ in this situation is easily determined by the formula for linear regression.

   If $\lambda \to \infty$, due to the penalty term's dominance, the model is less dependent on the input properties. Except $\mathbf{w}_0$, optimal $\mathbf{w}$ will be very close to zero and the result will be a straight line passing through the mean of the data. Optimal $\mathbf{w}_0$ value will be as in linear regression. The final model will be a flat line that crosses the mean value of the target variable which is not useful for making predictions (under-fitting). Large λ penalizes weight values more.

   If $\lambda \to -\infty$, since our goal is to bring the cost function $J(\mathbf{w})$ to the minimum value, a negative λ will make $J(\mathbf{w})$ negative (if the $\mathbf{w}_j$ value is not zero). However, this is a hypothetical situation, we need to choose the λ value as zero or greater in real applications. In general, the λ is chosen between 0 and 1 in most application.

3. Find the partial derivatives $\frac{\partial J}{\partial w_j}, j \in \{0, \ldots, d\}$. (You can only derive once for an arbitrary index j.)

   Solution : To avoid confusion in the partial derivative, updated the j value in the regularization term sum in the equation to k. The new equation is as follows:

$$J(\mathbf{w}) = \sum_{i=1}^{n} \left(h\left(\mathbf{x}_i\right) - y_i\right)^2 + \lambda \sum_{k=0}^{d} w_k^2$$

   Take the partial derivative of both sides of the equation according to w_j.

$$\frac{\partial}{\partial w_j} J(w) = \frac{\partial}{\partial w_j} \left[\sum_{i=1}^{n} \left(h\left(\mathbf{x}_i\right) - y_i\right)^2 + \lambda \sum_{k=0}^{d} w_k^2 \right]$$

   Distribute the derivative for each sum and written separately.

$$\begin{aligned} \frac{\partial}{\partial w_j} J(w) &= \frac{\partial}{\partial w_j} \left[\sum_{i=1}^{n} \left(h\left(\mathbf{x}_i\right) - y_i\right)^2 \right] + \frac{\partial}{\partial w_j} \left[\lambda \sum_{k=0}^{d} w_k^2 \right] \\ &= \sum_{i=1}^{n} \frac{\partial}{\partial w_j} \left(h\left(\mathbf{x}_i\right) - y_i\right)^2 + \lambda \sum_{k=0}^{d} \frac{\partial}{\partial w_j} (w_k^2) \end{aligned}$$

   The chain rule is applied.

$$\frac{\partial}{\partial w_j} J(w) = \sum_{i=1}^{n} \left(2\left(h\left(\mathbf{x}_i\right) - y_i\right)\right) \left[\frac{\partial}{\partial w_j} \left(h\left(\mathbf{x}_i\right) - y_i\right) \right] + \lambda \sum_{k=0}^{d} 2 w_k$$

   The $h(x)$ function is substituted in the equation and the partial derivative is calculated according to

w_j

$$\begin{aligned}\frac{\partial}{\partial w_j}J(w) &= \sum_{i=1}^{n} 2\left(h\left(\mathbf{x}_i\right)-y_i\right)\left[\frac{\partial}{\partial w_j}\left(w_0+w_1x_1^{(i)}+\ldots w_jx_{ij}+\ldots+w_dx_i-y_i\right)\right]+2\lambda w_j \\ &= \sum_{i=1}^{n} 2\left(h\left(\mathbf{x}_i\right)-y_i\right)\left[(0+\ldots+x_{ij}+\ldots+0-0)\right]+2\lambda w_j \\ &= \left[2\sum_{i=1}^{n}\left(h\left(\mathbf{x}_i\right)-y_i\right)x_{ij}\right]+2\lambda w_j\end{aligned}$$

4. Write out the update rule for gradient descent applied to the error function $J(w)$ above. Compare with the gradient descent update rule for standard linear regression; what is the difference? How does this difference affect gradient descent, assuming $\lambda > 0$? Consider what happens both when w_j is positive and when it is negative.

   Solution : The corresponding update rule for gradient descent applied to the error function J(w) is:

   $$w_j^{t+1} \leftarrow w_j^t - \alpha\frac{\partial}{\partial w_j}J\left(\mathbf{w}^t\right)$$

   Substituted the $J(w)$ function we calculated above into the equation:

   $$w_j^{t+1} \leftarrow w_j^t - \alpha\left\{2\sum_{i=1}^{n}\left[\left(h_{\mathbf{w}^t}\left(x_i\right)-y_i\right)x_{ij}\right]+2\lambda w_j^t\right\}$$

   Compared to the gradient descent rule for ordinary linear regression, the most important difference is that the updated rule has an extra $-\alpha(2\lambda w_j)$ that coming from regularization term. The regularization penalty reduces the weights value, simplifying the model and reducing the likelihood of overfitting. Also, other difference is that linear regression has the coefficient $\frac{\alpha}{n}$ instead of the α coefficient in front of the sum operation. This is actually not a significant difference. Since the α value is constant, we can multiply it with another constant value. Briefly, ridge regression weights (w_j^t) are reduced by $(1-2\alpha\lambda)$ and the remaining operations are standard linear regression operations. This implies that the gradient descent method will consider both the size of the weights and the gradient of the loss function.

   Since w_j value is negative ($w_j < 0$), α and λ are positive, this term will be positive. If the w_j value is positive ($w_j > 0$), the term will be negative. There is an inverse correlation. For this, it is assumed that the α value is greater than zero and small, otherwise a sign changes. Also, ridge regularization reduces the weights but does not eliminate them.

5. In order for the problem to be well-defined, an appropriate regularization constant λ must be chosen for the given problem (data set). How should the designer choose λ ?

   Solution : There are several methods to select the λ value, but the most known and used method is the **cross-validation** method. **Cross-validation** is a statistical resampling method used to evaluate the performance of the machine learning model on data it does not see, as objectively and accurately as possible. This technique; trains the model with training data, while evaluating the performance of the model using the remaining data (validation data). In this way; a more accurate idea of how the model will perform with real-world data.

3 California House Prices Prediction

The file housing.csv contains data on median house prices in California districts, derived from the 1990 census data. The data set contains 20,640 rows, one row per district (house block). Each row contains the following features:

- **longitude:** how far west a house is; a higher value is farther west.
- **latitude:** how far north a house is; a higher value is farther north.
- **housing_median_age:** median age of a house within the block; a lower number is a newer building.
- **total_rooms:** total number of rooms within the block.
- **total_bedrooms:** total number of bedrooms within the block.
- **population:** total number of people residing within the block.
- **households:** total number of households (a group of people residing within a home unit) within the block.
- **median_income:** median income for households within the block (measured in tens of thousands of US dollars).
- **median_house_value:** median house value for households within the block (measured in US dollars).
- **ocean_proximity:** location of the house with respect to the ocean. Can have one of the following values: NEAR BAY, NEAR OCEAN, ¡1H OCEAN, INLAND, ISLAND.

The objective in this data set is to predict the median house value in a given district based on the values of the other features.

1. Explore the data set and display summary statistics of the data. What can you learn from it on the data?

   Solution : Reached the summary and statistical figures of our dataset with the **describe()** and **info()** functions. In general, the first notable features are house value and income values.

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	206855.816909
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	115395.615874
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	119600.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	179700.000000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	264725.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

   The mean house value is $206k while the standard deviation is $115k. Also, the median house value spans from $15k to $500k. This indicates that there are a wide range of pricing, with some highly costly and others very affordable homes. With a mean of $3.87 and a standard deviation of $1.90, the

median annual income varies from $0.5 to $15. This shows that the earnings are also highly diverse, with some households having very low incomes and others having very high incomes.

In order to completely comprehend the characteristics of the other numerical data (longitude, latitude, housing median age, total rooms, total bedrooms, population, and households), additional study is required.

In addition, there are 207 Null values in the total_bedrooms feature. We will clear them in the data cleaning phase. Apart from that, there is no missing data on other features.

2. Compute the correlation between each feature and the target median_house_value. Which features have strong correlation with the target?

   Solution : The correlation coefficient is the coefficient that indicates the direction and magnitude of the relationship between the independent variables. This coefficient takes a value between (-1) and $(+1)$. Positive values indicate direct linear relationship; negative values indicate an inverse linear relationship.

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
longitude	1.000000	-0.924664	-0.108197	0.044568	0.069608	0.099773	0.055310	-0.015176	-0.045967
latitude	-0.924664	1.000000	0.011173	-0.036100	-0.066983	-0.108785	-0.071035	-0.079809	-0.144160
housing_median_age	-0.108197	0.011173	1.000000	-0.361262	-0.320451	-0.296244	-0.302916	-0.119034	0.105623
total_rooms	0.044568	-0.036100	-0.361262	1.000000	0.930380	0.857126	0.918484	0.198050	0.134153
total_bedrooms	0.069608	-0.066983	-0.320451	0.930380	1.000000	0.877747	0.979728	-0.007723	0.049686
population	0.099773	-0.108785	-0.296244	0.857126	0.877747	1.000000	0.907222	0.004834	-0.024650
households	0.055310	-0.071035	-0.302916	0.918484	0.979728	0.907222	1.000000	0.013033	0.065843
median_income	-0.015176	-0.079809	-0.119034	0.198050	-0.007723	0.004834	0.013033	1.000000	0.688075
median_house_value	-0.045967	-0.144160	0.105623	0.134153	0.049686	-0.024650	0.065843	0.688075	1.000000

   The above table shows the correlation coefficients of each feature with each other. There is a high and positive correlation between total_bedrooms/total_rooms and households. The reason for this is that as the number of households increases, the number of rooms needed in the house increases.

   In addition, an inverse and high correlation is seen between latitude and longitude. However, this is meaningless because the values for latitude and longitude have a location info.

   The feature with the highest correlation with the target is the median_income_value. The correlation coefficient was calculated as $+0.68$. This indicates high and positive correlation. All other pairs of features are relatively uncorrelated.

3. Which actions do you need to take in order to prepare the data set for the learning algorithm? Identify at least four such actions.

   Solution : There are 6 different data preparation processes that we need to do in order to make the data set suitable for the learning algorithm.

   (a) Data cleaning: This involves removing any invalid or inconsistent data, such as missing values, outliers, or duplicates.

   (b) Feature selection: Choosing the most important and relevant features for the model, and removing any redundant or irrelevant features.

   (c) Feature engineering: Creating new features that may help the model make more accurate predictions.

   (d) Normalization and scaling: Rescaling the features to a similar range, such as between 0 and 1, so that they have similar units and magnitudes.

   (e) Encoding categorical variables: Converting categorical variables to numerical values that can be processed by the model.

   (f) Splitting the data set: Separating the data set into training, validation, and testing sets, in order to train and evaluate the model on different data.

4. Clean the data set using the data preprocessing techniques discussed in class. Show a sample of the data set before and after the cleaning.

   Solution : First, we start by checking for Null values, which is the oldest known cleaning method. We have 207 NULL values in total_bedroom feature. This number is very small compared to the total number of samples in the model and missing values do not affect the model. That's why we drop these values.

   For categorical feature, the ISLAND in ocean_proximity exists only 5 times, although other classes contain over 2500 examples. Since it contains a small number of samples, we deleted the samples containing the ISLAND value from our table in order not to affect the model fitting.

   In order to analyze the ocean_proximity categorical feature, I manipulated the data and converted it to dummy or indicator variables. This process was done with the get_dummies function in the pandas library. Thus, the ocean_proximity property, which has 4 different values, is specified in the table as 4 different properties.

   Before data cleaning, our table consisted of 20640 rows and 10 columns. After clearing the null values and deleting the rows with ISLAND categorical variable, our table decreased to 20428 rows. Also, as we changed the ocean_proximity categorical variable to a numeric variable, the number of columns increased to 13.

5. Extract at least two new features from the data set that have strong correlation with the target feature.

   Solution : Some properties given, such as population, total number of bedrooms or total number of rooms, are not very meaningful on their own, so we will create new and more meaningful properties from these data. Two new features have been created, rooms per household and the rooms per bedrooms.

   Among these two new features we created, the correlation coefficient between the roomsperbedrooms feature and our target value was found to be 0.38, and with avgroom to be 0.15.

6. Run linear regression on the transformed data set. Use 80% of the data set as your training set and 20% as your test set. Compute RMSE and R^2 score both on the training and the test sets.

   Solution : With the train_test_split function, I divided our data into 20% test and 80% train. R^2 and RMSE values calculated for both data groups are as follows:

   Training set RMSE: 67977.93
   Training set R2 score: 0.65
   Test set RMSE: 69511.31
   Test set R2 score: 0.64

7. Does adding regularization improve the results? If yes, which value of λ provides the best RMSE on the test set? If no, what is the reason for it?

   Solution : We used ridge regression to see if the model would improve by adding regularization. For this, I used the sklearn library in python. As an example, we determined our lambda values as 0.001, 0.01, 0.1, 1, 10, 100, 1000, respectively, and recorded them in the alphas list. Then we applied the ridge regression function for each lambda value with the for loop and suppressed the RMSE values.

   As a result, the best RMSE value was calculated as 69505 for λ=100. This value is very close to the RMSE 69511 value we calculated in linear regression, so it was determined that there was no improvement in the model.

   The fact that our λ value is very high indicates that our model is simple. Therefore, our model will not know enough about the training data to make useful predictions. In short, under-fitting will be experienced.

8. Now replace linear regression with DecisionTreeRegressor (in sklearn.tree). What is the RMSE on the training and test this time? Which phenomenon is observed here?

   Solution : When we used Decision Tree Regressor, the RMSE value for our training data was 0. This shows that our model is over-fitting for the training data. Methods such as constraints to model parameters and reducing can be used to solve this problem.

When the model was applied to the test data, the RMSE value was found to be 67249.

In this case, the decision tree model is able to fit the training data well, but does not generalize well to new data.

Decision Tree Regressor - Training set RMSE: 0.0
Decision Tree Regressor - Training set R2 score: 1.0
Decision Tree Regressor - Test set RMSE: 67249
Decision Tree Regressor - Test set R2 score: 0.6589