190/200

Reza Mahroo EE7600: Machine Learning Project 1

The classification has been implemented in this project for “load breast cancer” and “load digits” data sets and regression for the “fetch_california_housing” data set. The project has been completed by analyzing and training each data set separately using KNN classifier/regressor and Decision tree classifier/regressor algorithms.

Note: some discussions such as overfitting/underfitting and the most favorable neighbors/depth are skipped for similar cases.

This is a good job. I noticed that you have used the maxmin scaler. That is OK. However, you have scaled the data before splitting. That means you have implicitly used the test set in the training. As I have said that is a cardinal sin in ML. The sets must be normalized separately.

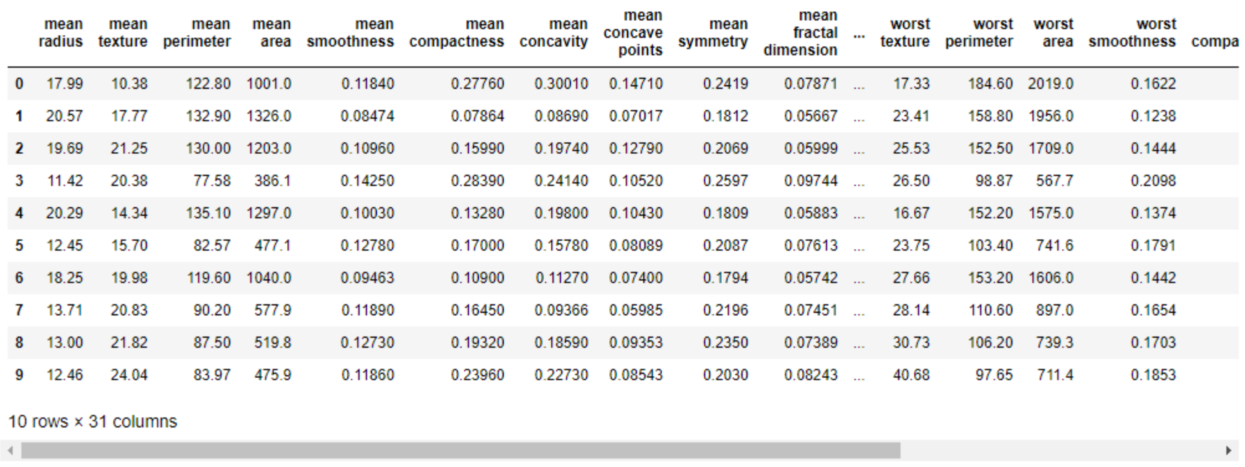
Dataset exploration:

1. load breast cancer information:

- Number of Instances: 569
- Number of features: 30, all numeric
- Target Names: ['malignant', 'benign']
- Feature Names: ['mean radius', 'mean texture', 'mean perimeter', 'mean area', 'mean smoothness', 'mean compactness', 'mean concavity', 'mean concave points', 'mean symmetry', 'mean fractal dimension', 'radius error', 'texture error','perimeter error', 'area error', 'smoothness error', 'compactness error', 'concavity error', 'concave points error','symmetry error', 'fractal dimension error', 'worst radius', 'worst texture', 'worst perimeter', 'worst area', 'worst smoothness', 'worst compactness', 'worst concavity', 'worst concave points','worst symmetry', 'worst fractal dimension']
- Missing Feature Values: None
- Class Distribution: 212 malignant - 357 benign

Visualize some of the data:

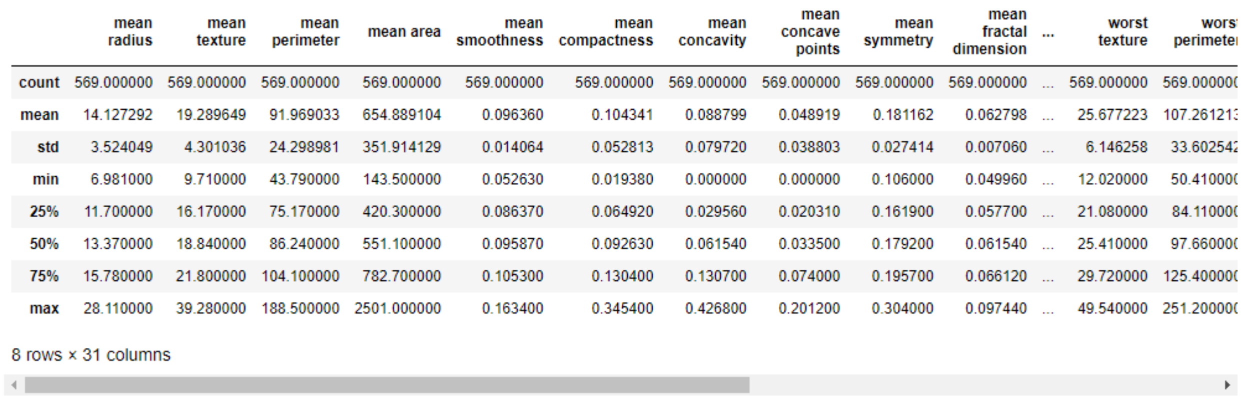
First ten instances of the dataset:



	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	compa
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1374	
5	12.45	15.70	82.57	477.1	0.12780	0.17000	0.15780	0.08089	0.2087	0.07613	...	23.75	103.40	741.6	0.1791	
6	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.11270	0.07400	0.1794	0.05742	...	27.66	153.20	1606.0	0.1442	
7	13.71	20.83	90.20	577.9	0.11890	0.16450	0.09366	0.05985	0.2196	0.07451	...	28.14	110.60	897.0	0.1654	
8	13.00	21.82	87.50	519.8	0.12730	0.19320	0.18590	0.09353	0.2350	0.07389	...	30.73	106.20	739.3	0.1703	
9	12.46	24.04	83.97	475.9	0.11860	0.23960	0.22730	0.08543	0.2030	0.08243	...	40.68	97.65	711.4	0.1853	

10 rows × 31 columns

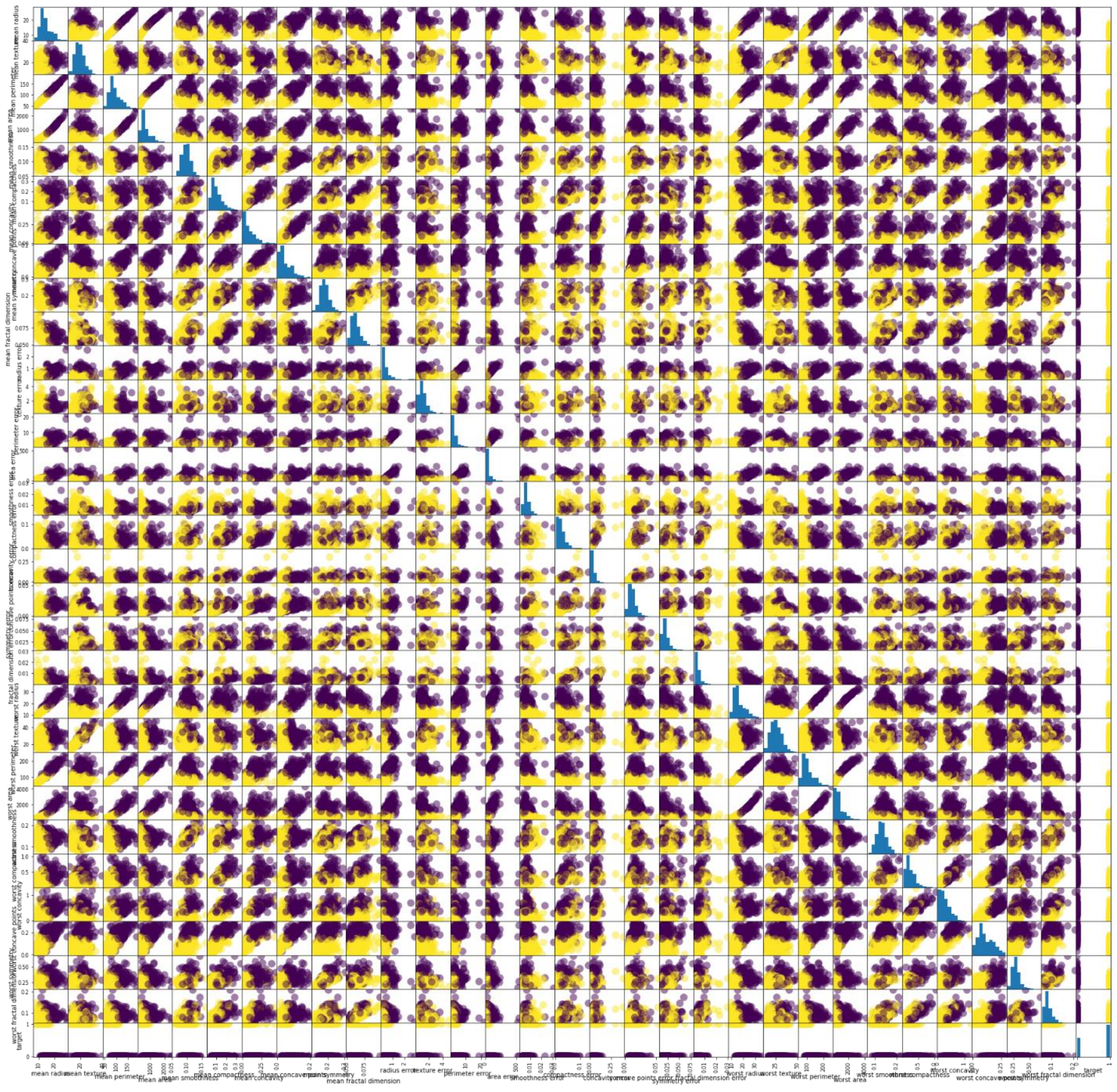
The statistical description of the dataset:



	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	wors perimete
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...	569.000000	569.00000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	0.062798	...	25.677223	107.26121
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	0.007060	...	6.146258	33.602542
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.049960	...	12.020000	50.41000
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	0.057700	...	21.080000	84.11000
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	0.061540	...	25.410000	97.66000
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	0.066120	...	29.720000	125.40000
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.097440	...	49.540000	251.20000

8 rows × 31 columns

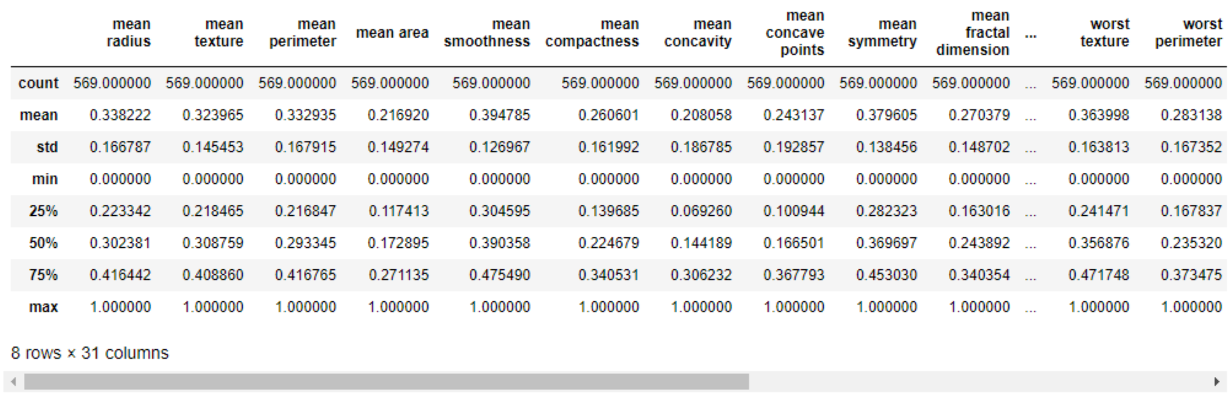
The overview image of the correlation between each pair of the features is printed out here.



To scrutinize the correlation between each pair of features, we can draw the above pictures on a large scale or calculate the Pearson correlation coefficient. The correlation coefficient (Pearson) between each set of features is calculated in the Jupiter code file. Due to its large scale, I could not fit it in here.

Normalizing the data set changes the statistical indexes like mean/max and Q1/Q2/Q3; however, it does not affect the correlation between the features. In this regard, we expect that the normalization of the data set positively affects the K-nearest-neighbor algorithm, which works based on the distance of the testing data with the K number of its neighbors and does not make an assumption about the distribution of the data set when wants to train a data set.

The statistical description of the normalized dataset:



	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...	569.000000	569.000000
mean	0.338222	0.323965	0.332935	0.216920	0.394785	0.260601	0.208058	0.243137	0.379605	0.270379	...	0.363998	0.283138
std	0.166787	0.145453	0.167915	0.149274	0.126967	0.161992	0.186785	0.192857	0.138456	0.148702	...	0.163813	0.167352
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000
25%	0.223342	0.218465	0.216847	0.117413	0.304595	0.139685	0.069260	0.100944	0.282323	0.163016	...	0.241471	0.167837
50%	0.302381	0.308759	0.293345	0.172895	0.390358	0.224679	0.144189	0.166501	0.369697	0.243892	...	0.356876	0.235320
75%	0.416442	0.408860	0.416765	0.271135	0.475490	0.340531	0.306232	0.367793	0.453030	0.340354	...	0.471748	0.373475
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.000000

8 rows × 31 columns

Train and test:

• K-nearest-neighbor classifier:

The data set is split so that 80 percent of them are used for training the system and 20 percent for testing the algorithm. The classification error vs. the number of neighbors (k) for both training and test sets, and for two distance metrics ($p = 1$ and $p = 2$) are plotted under four below conditions:

1) $p = 1$, and the data set is not normalized

2) $p = 1$, and the data set is normalized

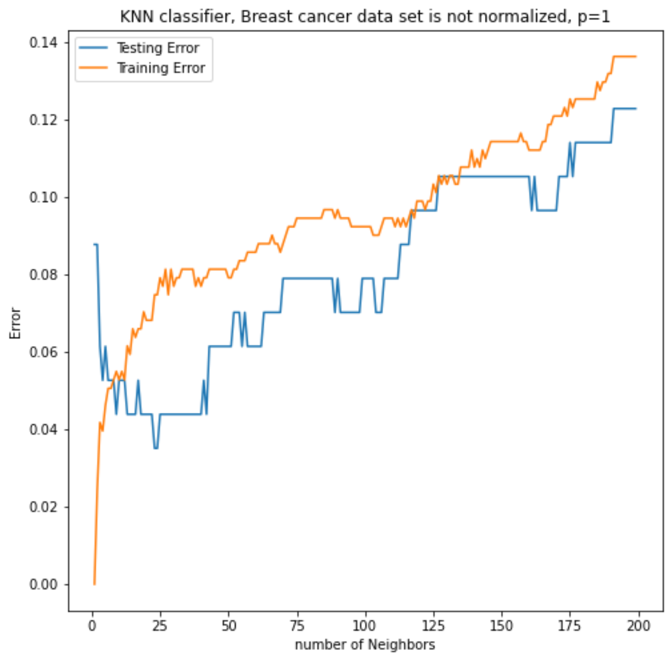
3) $p = 2$, and the data set is not normalized

4) $p = 2$, and the data set is normalized

Then the best possible numbers of neighbors are suggested for each condition, and the results are discussed.

1) $p = 1$, and the data set is not normalized: In this case, the test set minimum errors calculated are 3.5-4.5 percent, and it happens when the number of neighbors is [19,25], and the training error is almost 7.4-8.0 percent.

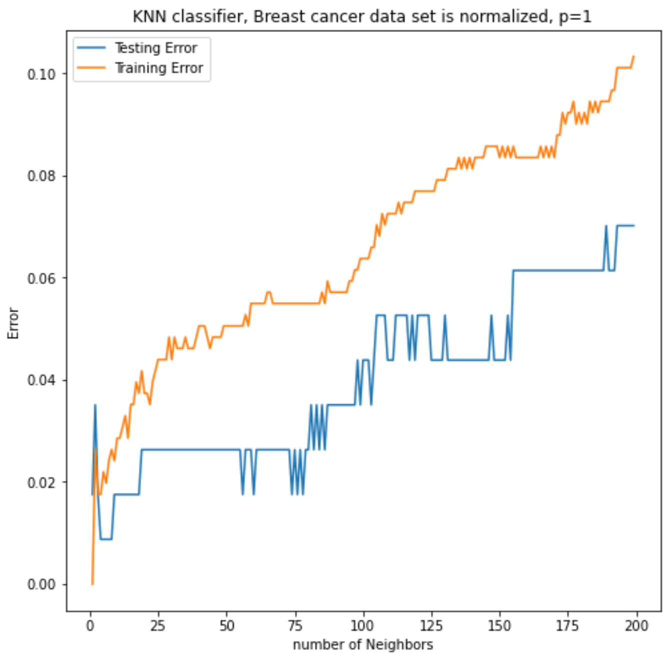
As shown in the figure below, the training error is small when the number of neighbors is small and the testing error is large. For the small number of neighbors, the algorithm undergoes overfitting because it is trained to perfectly predict the training data set; however, it cannot accurately predict the new data (test data set). As we increase the number of neighbors, both the training and testing errors keep increasing due to the underfitting of the algorithm.



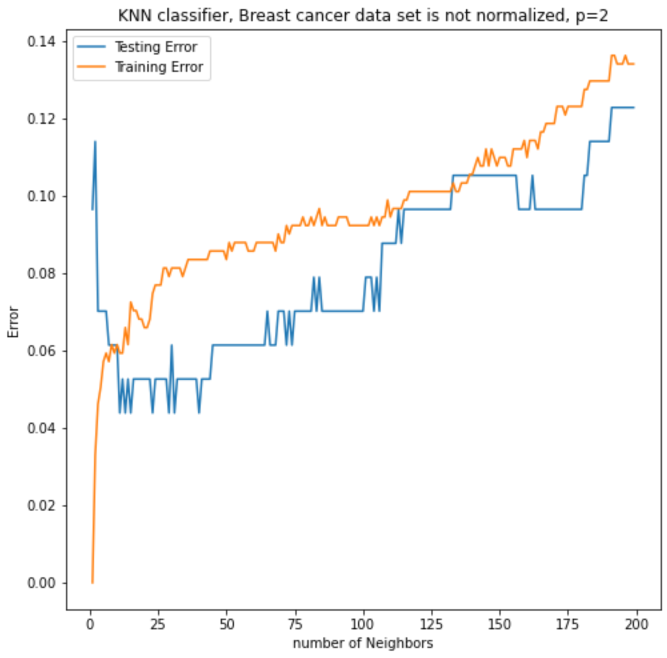
✓

2) $p = 1$, and the data set is normalized: In this case, the test set minimum error calculated is 0.87-1.7 percent, and it happens when the number of neighbors is [4-18], and the training error is [1.7- 2.6], percent. The most favorable k is 4, with the best training and testing errors; However, it might be better to go with the k=5 (an odd number) with a 1.9 training set error to avoid overfitting. The same discussion as the previous case can be applied to this case regarding the overfitting and underfitting. However, the main difference between these two cases is data normalization, which has significantly changed the number of favorable nearest neighbors. Moreover, needing fewer neighbors in this algorithm will significantly affect the solution time of the algorithm when the dimension of the dataset is large scale. The fewer nearest number, the less solution time.

✓



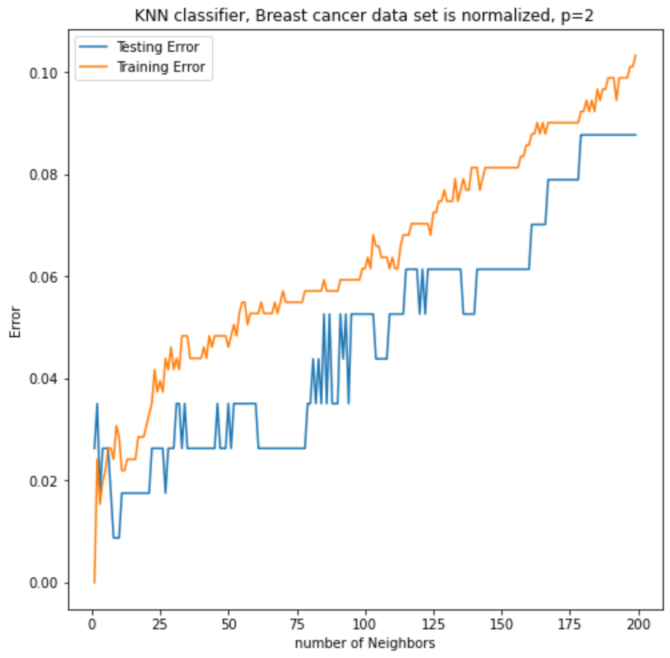
3) $p = 2$, and the data set is not normalized: In this case, the test set minimum error calculated is [4.3-4.9] percent, and it happens when the number of neighbors is [11-21], and the training error is [5.9-8.3], percent. The most favorable k is 11, with the best training and testing errors. The same discussion as the previous case can be applied to this case regarding the overfitting and underfitting. Compared to case one, in which the data set for both is not normalized, but their distance metrics p is different, we obtained different results here. The best test error in case one was better; however, the training error and the number of favorable k, 7.4 and 23, make up for it.



✓

4) $p = 2$, and the data set is normalized: In this case, the test set minimum error calculated is [0.87-1.7] percent, and it happens when the number of neighbors is [8-20], and the training error is [2.4-2.8], percent. The most favorable k is 8, with the best training and testing errors. We obtained different results compared to case two, where both data sets were normalized, but their distance metrics p were different. The best testing errors are the same; however, the training error and the number of favorable k make case two a better case. Moreover, compared to case three, the normalization of the data set has significantly improved the algorithm.

✓



KNN just stores all the training data and corresponding labels, and no distances are calculated at this point. All the calculation work is done during the "predict" phase.

For every value of p, the distance between two data points is measured differently. Thus, each of these values is useful in certain conditions. We can use any arbitrary value of p and can see which gives us the best results.

• Decision Tree Classifier:

The data set is split so that 80 percent of them are used for training the system and 20 percent for testing the algorithm. The classification error vs. tree depth for both training and test sets and for "gini" and "entropy" splitting criterion is plotted under four below conditions:

1) "gini" splitting criterion and the data set is not normalized

2) "gini" splitting criterion and the data set is normalized

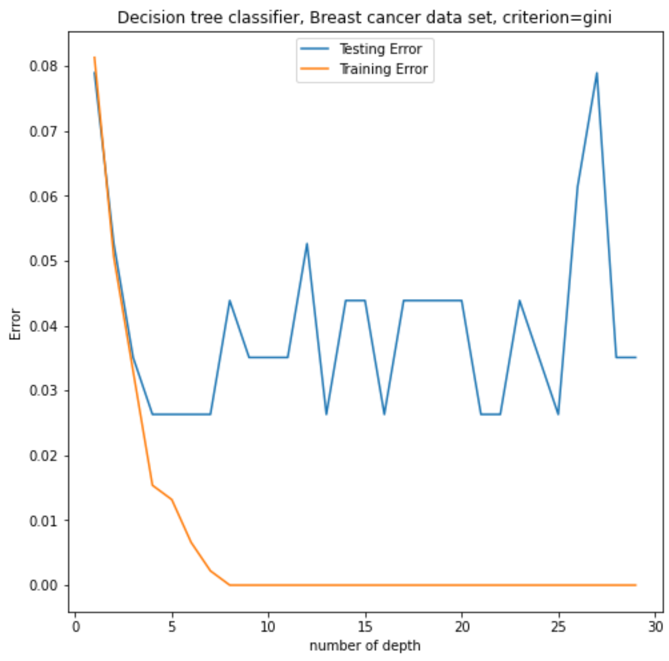
3) “entropy” splitting criterion and the data set is not normalized

4) “entropy” splitting criterion and the data set is normalized

Then, the results are discussed.

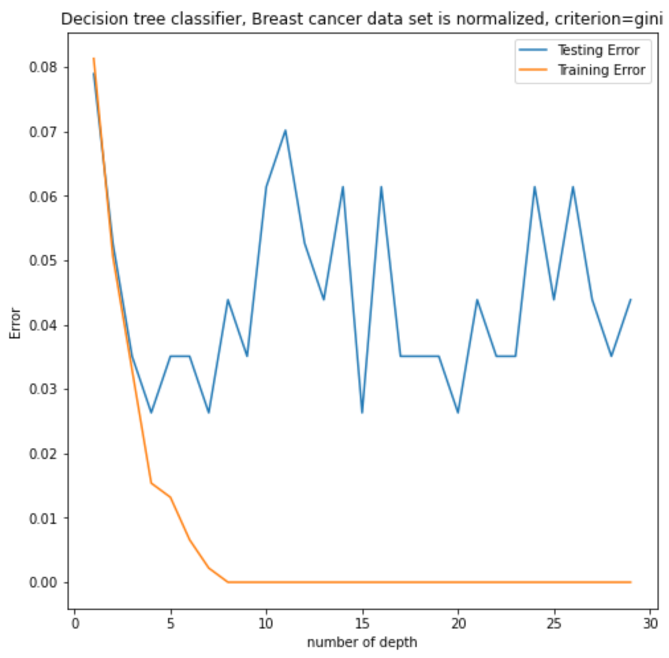
1) “gini” splitting criterion and the data set is not normalized: In this case, the test set minimum error is 2.6 percent, and it happens when the max-depth is [4-7], and the training error is [1.5, 1.3, 0.6, 0.2, 0], percent. For max-depth less than 4, the algorithm is in the underfitting condition, in which both training and testing errors are high. On the other hand, when max-depth is more than 6, the algorithm is prone to overfitting, where only the training error is a small value or zero.

Generally, the underfitting condition results from the max-depth being too small, and the overfitting condition results from the depth exceeding normal. The most favorable depth is when the test and training errors are in their knee area (before the training error reaches zero), which in this case is [4-6]. This argument is true for the next three instances regarding decision three.



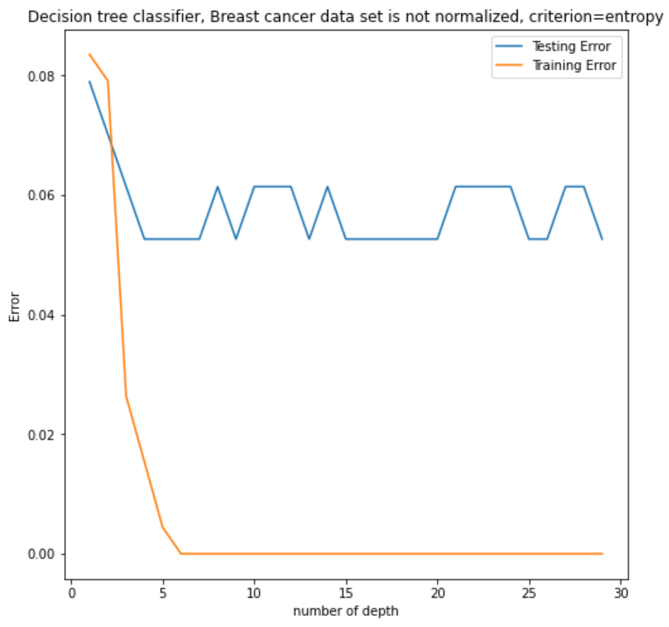
✓

2) “gini” splitting criterion and the data set is normalized: In this case, the test set minimum error is 2.6-3.2 percent, and it happens when the max-depth is [4-7], and the training error is [1.5, 1.3, 0.2], percent. In comparison to the previous case, the normalization had no significant effect on the performance of the algorithm.



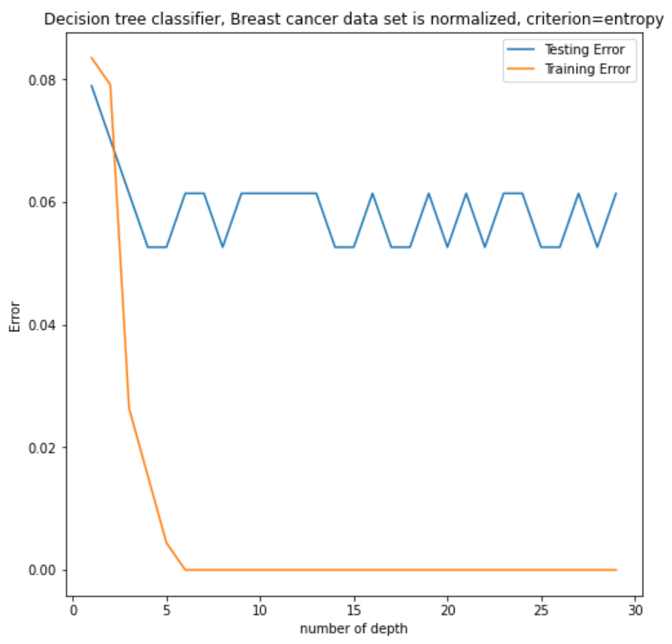
✓

3) "entropy" splitting criterion and the data set is not normalized: In this case, the test set minimum error is 5.2 percent, and it happens when the max-depth is [4, 5, …], and the training error is [1.5, 0.4, 0, …], percent. The most favorable depth might be 4 or 5 for this case in which the test error is the least, and the training error is a small non-zero value. Compared to case 1, the entropy criteria provide inferior results for the not normalized data set.



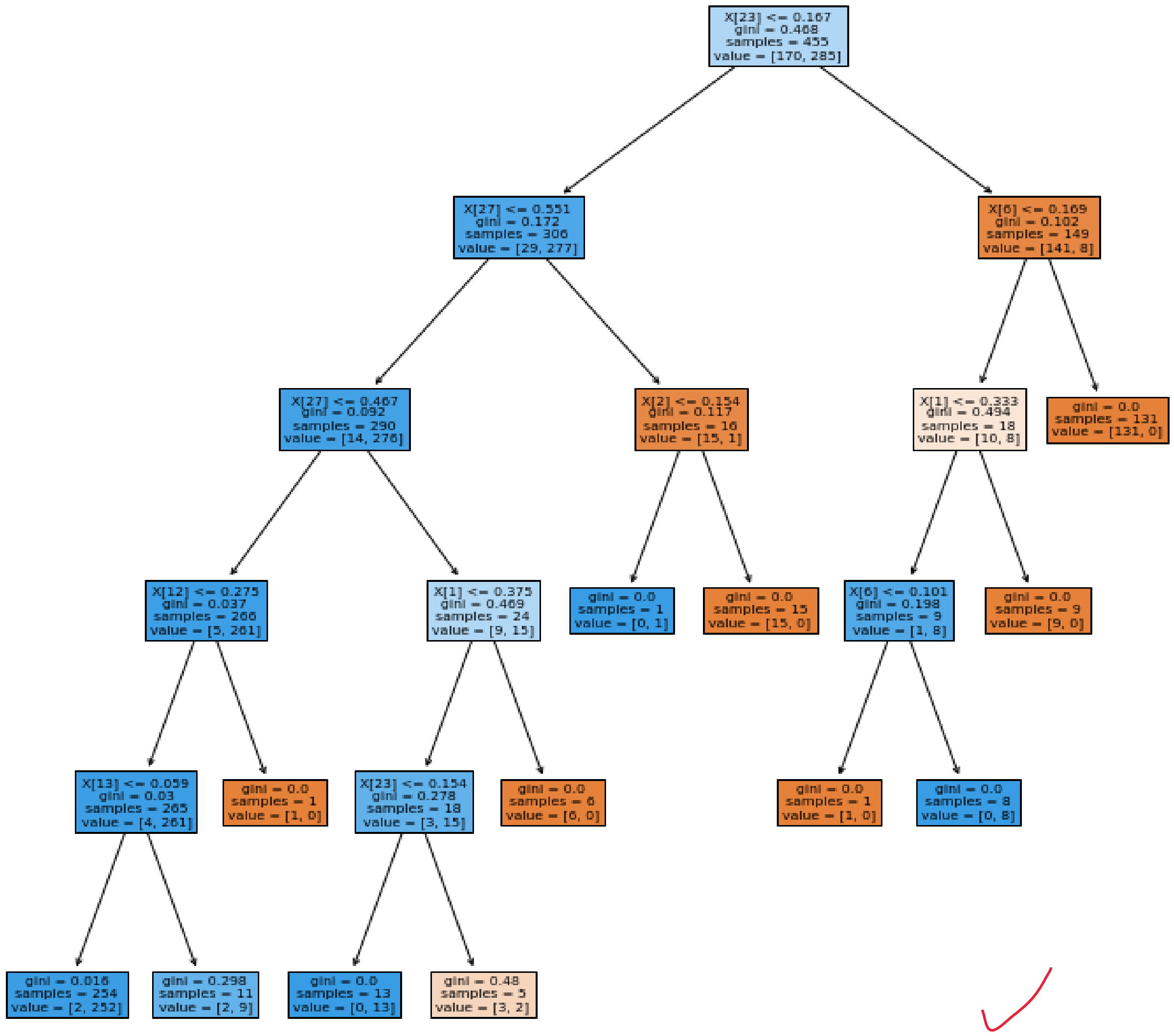
✓

4) "entropy" splitting criterion and the data set is normalized: In this case, the test set minimum error is 5.2-6 percent, and it happens when the max-depth is [4, 8, …], and the training error is [1.5, 0.4, 0, …], percent. The most favorable depth might be 4 or 5 for this case in which the test error is the least, and the training error is a small non-zero value. Compared to case 2, the entropy criteria provide inferior results for the normalized data set.



The entropy criterion also might be a little slower to compute because it requires computing a logarithmic function. Many researchers point out that in most cases, splitting criteria will not make much difference in the tree performance. Each criterion is superior in some cases and inferior in others.

For k=5, the regression tree is plotted as follows:



✓

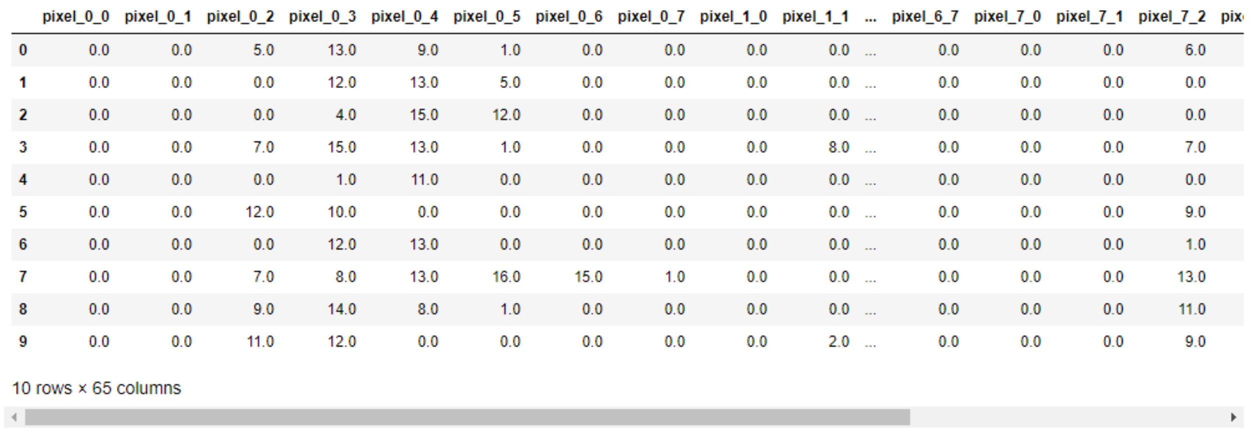
Dataset exploration:

2. load digits information:

- Number of Instances: 1797
- Number of features: 64, all numeric
- Target Names: [0,1,2,3,4,5,6,7,8,9]
- Feature Names: ['pixel_0_0', 'pixel_0_1',..., 'pixel_7_7']
- Missing Feature Values: None

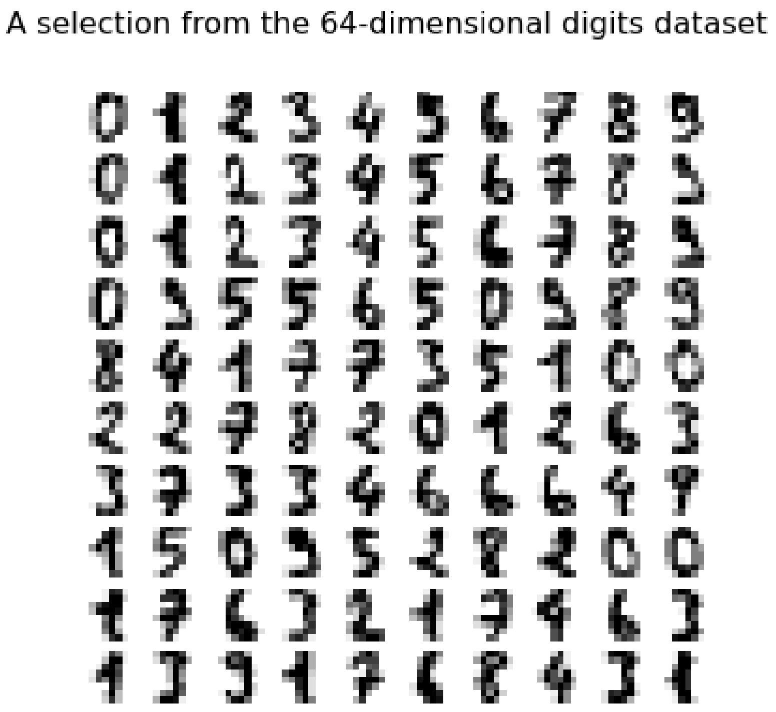
Visualize some of the data:

First ten samples of the dataset:

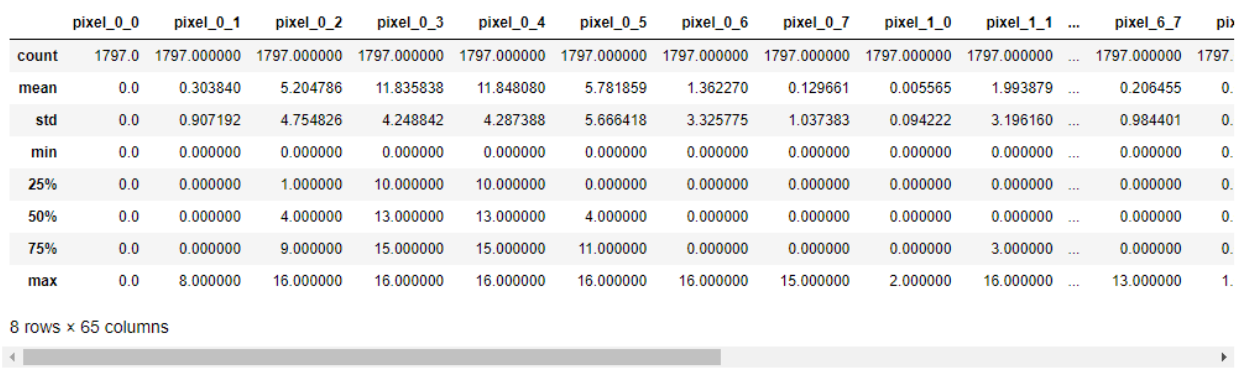


	pixel_0_0	pixel_0_1	pixel_0_2	pixel_0_3	pixel_0_4	pixel_0_5	pixel_0_6	pixel_0_7	pixel_1_0	pixel_1_1	...	pixel_6_7	pixel_7_0	pixel_7_1	pixel_7_2	pix
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	6.0	
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
3	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0	...	0.0	0.0	0.0	7.0	
4	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
5	0.0	0.0	12.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	9.0	
6	0.0	0.0	0.0	12.0	13.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	
7	0.0	0.0	7.0	8.0	13.0	16.0	15.0	1.0	0.0	0.0	...	0.0	0.0	0.0	13.0	
8	0.0	0.0	9.0	14.0	8.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	11.0	
9	0.0	0.0	11.0	12.0	0.0	0.0	0.0	0.0	0.0	2.0	...	0.0	0.0	0.0	9.0	

10 rows × 65 columns



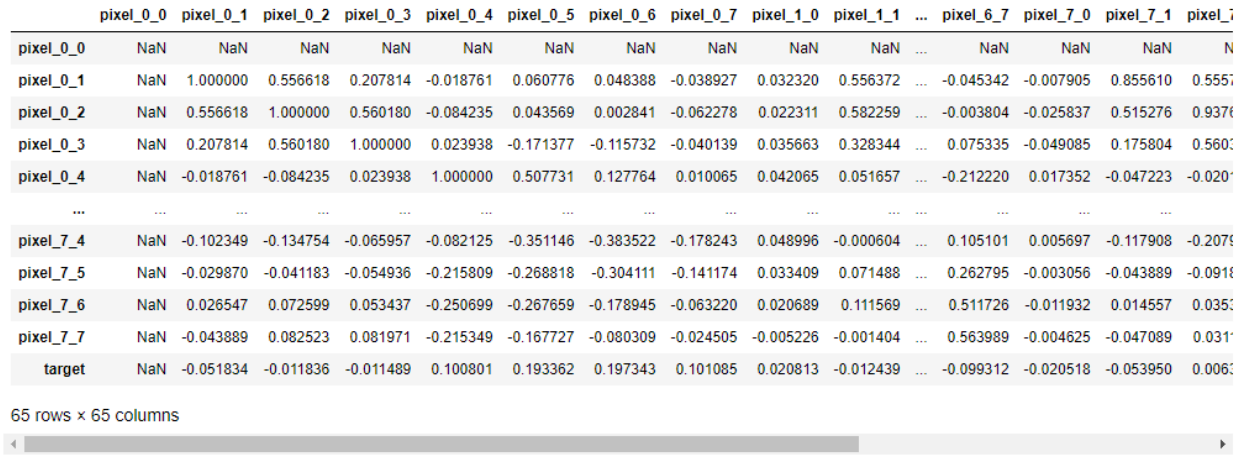
The statistical description of the dataset:



	pixel_0_0	pixel_0_1	pixel_0_2	pixel_0_3	pixel_0_4	pixel_0_5	pixel_0_6	pixel_0_7	pixel_1_0	pixel_1_1	...	pixel_6_7	pi
count	1797.0	1797.000000	1797.000000	1797.000000	1797.000000	1797.000000	1797.000000	1797.000000	1797.000000	1797.000000	...	1797.000000	1797.
mean	0.0	0.303840	5.204786	11.835838	11.848080	5.781859	1.362270	0.129661	0.005565	1.993879	...	0.206455	0.
std	0.0	0.907192	4.754826	4.248842	4.287388	5.666418	3.325775	1.037383	0.094222	3.196160	...	0.984401	0.
min	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.
25%	0.0	0.000000	1.000000	10.000000	10.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.
50%	0.0	0.000000	4.000000	13.000000	13.000000	4.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.
75%	0.0	0.000000	9.000000	15.000000	15.000000	11.000000	0.000000	0.000000	0.000000	3.000000	...	0.000000	0.
max	0.0	8.000000	16.000000	16.000000	16.000000	16.000000	16.000000	15.000000	2.000000	16.000000	...	13.000000	1.

8 rows × 65 columns

The correlation coefficient (Pearson) between each set of features is calculated here. Since the very first pixel has never taken any value, its correlation with other pixels is not defined.



	pixel_0_0	pixel_0_1	pixel_0_2	pixel_0_3	pixel_0_4	pixel_0_5	pixel_0_6	pixel_0_7	pixel_1_0	pixel_1_1	...	pixel_6_7	pixel_7_0	pixel_7_1	pixel_
pixel_0_0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	N
pixel_0_1	NaN	1.000000	0.556618	0.207814	-0.018761	0.060776	0.048388	-0.038927	0.032320	0.556372	...	-0.045342	-0.007905	0.855610	0.555
pixel_0_2	NaN	0.556618	1.000000	0.560180	-0.084235	0.043569	0.002841	-0.062278	0.022311	0.582259	...	-0.003804	-0.025837	0.515276	0.937
pixel_0_3	NaN	0.207814	0.560180	1.000000	0.023938	-0.171377	-0.115732	-0.040139	0.035663	0.328344	...	0.075335	-0.049085	0.175804	0.560
pixel_0_4	NaN	-0.018761	-0.084235	0.023938	1.000000	0.507731	0.127764	0.010065	0.042065	0.051657	...	-0.212220	0.017352	-0.047223	-0.020
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
pixel_7_4	NaN	-0.102349	-0.134754	-0.065957	-0.082125	-0.351146	-0.383522	-0.178243	0.048996	-0.000604	...	0.105101	0.005697	-0.117908	-0.207
pixel_7_5	NaN	-0.029870	-0.041183	-0.054936	-0.215809	-0.268818	-0.304111	-0.141174	0.033409	0.071488	...	0.262795	-0.003056	-0.043889	-0.091
pixel_7_6	NaN	0.026547	0.072599	0.053437	-0.250699	-0.267659	-0.178945	-0.063220	0.020689	0.111569	...	0.511726	-0.011932	0.014557	0.035
pixel_7_7	NaN	-0.043889	0.082523	0.081971	-0.215349	-0.167727	-0.080309	-0.024505	-0.005226	-0.001404	...	0.563989	-0.004625	-0.047089	0.031
target	NaN	-0.051834	-0.011836	-0.011489	0.100801	0.193362	0.197343	0.101085	0.020813	-0.012439	...	-0.099312	-0.020518	-0.053950	0.006

65 rows × 65 columns

Train and test:

• K-nearest-neighbor classifier:

The data set is split so that 80 percent of them are used for training the system and 20 percent for testing the algorithm. The classification error vs. the number of neighbors (k) for both training and test sets, and for two distance metrics ($p = 1$ and $p = 2$) are plotted under four below conditions:

1) $p = 1$, and the data set is not normalized

2) $p = 1$, and the data set is normalized

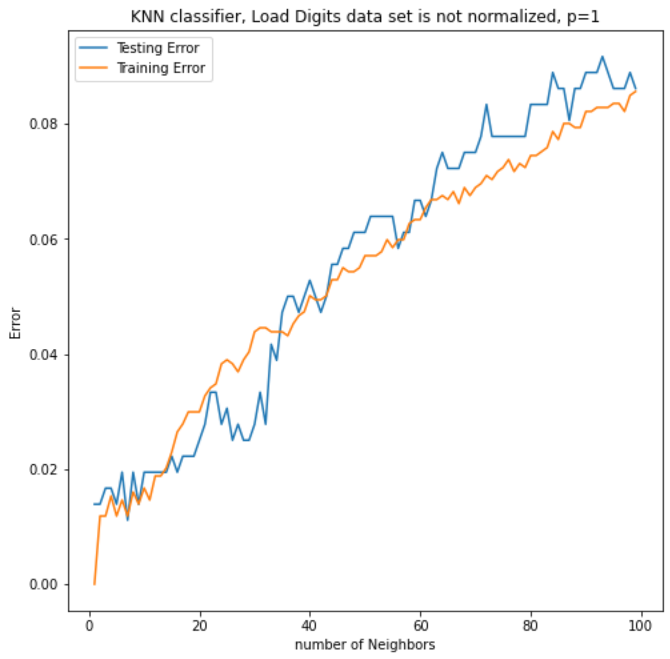
3) $p = 2$, and the data set is not normalized

4) $p = 2$, and the data set is normalized

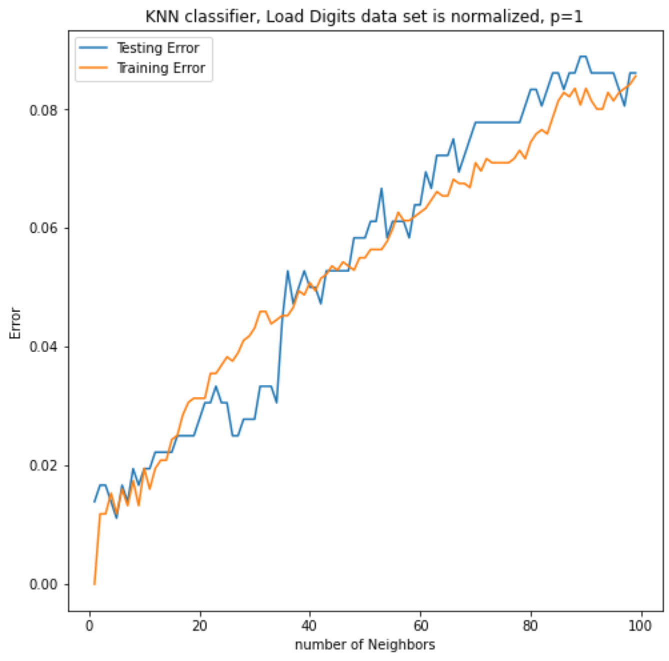
Then the best possible number of neighbors is obtained for each condition, and the results are discussed.

1) $p = 1$, and the data set is not normalized: In this case, the test set minimum error calculated is 1.1-1.8 percent, and it happens when the number of neighbors is [7-12], and the training error is [1.18-1.7] percent.

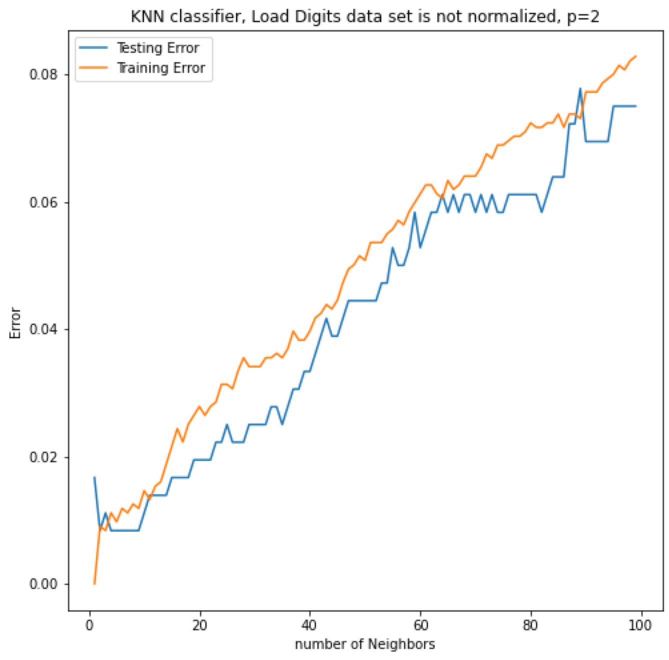
As shown in the figure below, the training error is small when the number of neighbors is small, and the testing error is larger. The algorithm is prone to overfitting in this condition because of small training and large testing errors. As we increase the number of neighbors, both the training and testing errors keep increasing due to the underfitting of the algorithm.



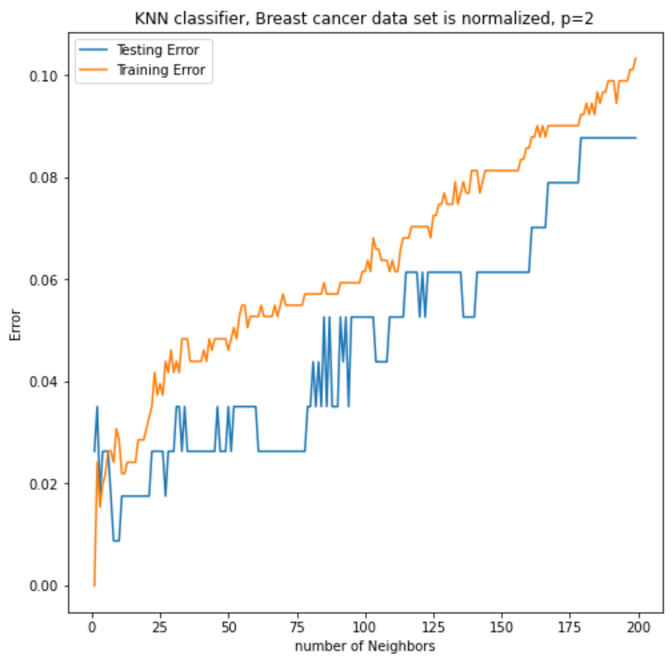
2) $p = 1$, and the data set is normalized: In this case, the test set minimum error calculated is [1.1-1.8] percent, and it happens when the number of neighbors is [3-12], and the training error is [1.2-1.7], percent. The most favorable k is 3, with the best training and testing errors; However, it might be better to go with the k=5 (an odd number) with a 1.9 training set error to avoid overfitting. The same discussion as the previous case can be applied to this case regarding the overfitting and underfitting. However, the main difference between these two cases is data normalization, which has changed the number of favorable nearest neighbors. Needing fewer neighbors in this algorithm will significantly affect the solution time of the algorithm when the dimension of the dataset is large. The fewer nearest number, the less solution time. Normalization seems to have no major effect on the minimum error in this case.



3) $\boldsymbol{p = 2}$**, and the data set is not normalized:** In this case, the test set minimum error calculated is 0.83 percent, and it happens when the number of neighbors is [2-9], and the training error is [0.9-1.25], percent. The most favorable k might be 10 in this case, with the best training and testing errors. The same discussion as the previous case can be applied to this case regarding the overfitting and underfitting. Compared to case one, in which the data set for both is not normalized, but their distance metrics p is different, we obtained different results here. The best test error in this case is better, and we can train the algorithm with a smaller number of neighbors.



4) $p = 2$, and the data set is normalized: In this case, the test set minimum error calculated is [0.83-1.9] percent, and it happens when the number of neighbors is [4- 9], and the training error is [0.9-1.25], percent. The most favorable k might be 7, with the best training and testing errors. We obtained different results compared to case two, where both data sets were normalized, but their distance metrics p were different. The best testing errors are the same; however, the training error and the number of favorable k make case two a better case. Moreover, compared to case three, the normalization of the data set has significantly improved the algorithm.



- ## Decision Tree Clasifier:

The data set is split so that 80 percent of them are used for training the system and 20 percent for testing the algorithm. The classification error vs. tree depth for both training and test sets and for “gini” and “entropy” splitting criterion is plotted under four below conditions:

1) “gini” splitting criterion and the data set is not normalized

2) “gini” splitting criterion and the data set is normalized

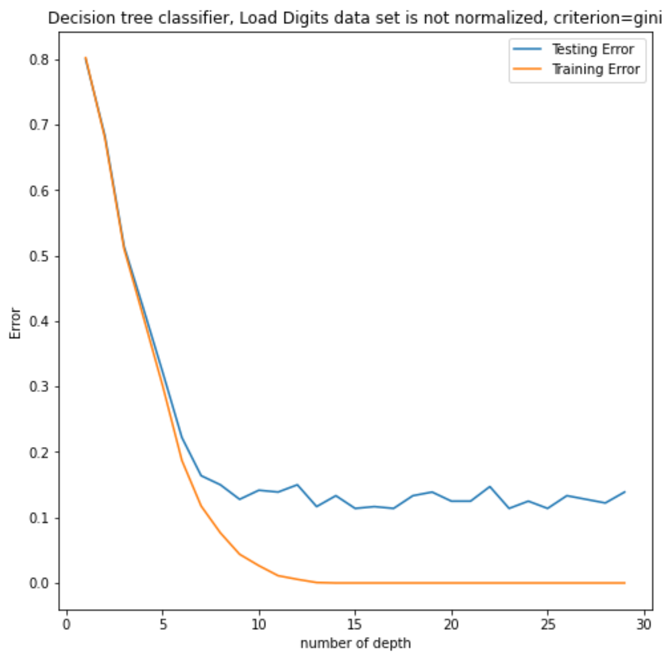
3) “entropy” splitting criterion and the data set is not normalized

4) “entropy” splitting criterion and the data set is normalized

Then, the results are discussed.

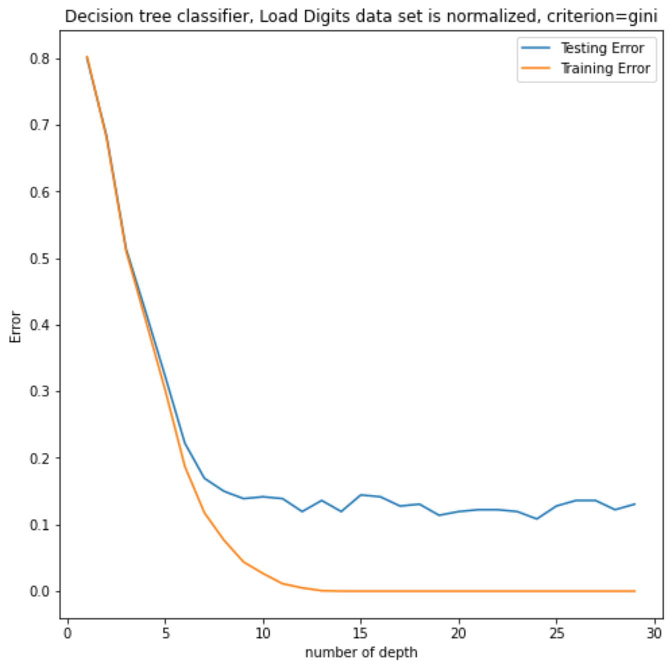
1) “gini” splitting criterion and the data set is not normalized: In this case, the test set minimum error is 11.38-14.5 percent, and it happens when the max-depth is [8-10], and the training error is less than 5, percent.

For max-depth less than 4, the algorithm is in the underfitting condition, in which both training and testing errors are high. On the other hand, when max-depth is more than 10, the algorithm undergoes overfitting, where only the training error is a small value or zero. Generally, the underfitting condition results from the max-depth being too small, and the overfitting condition results from the depth exceeding normal. The most favorable depth is when the error is in the knee area.



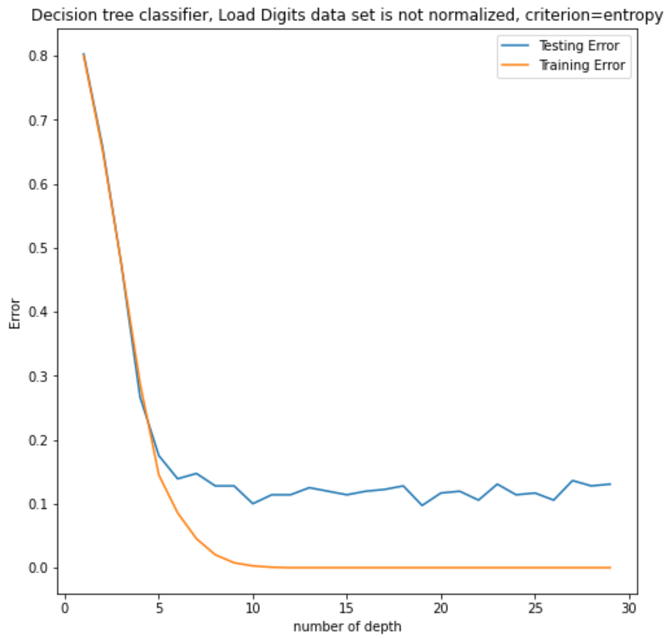
✓

2) "gini" splitting criterion and the data set is normalized: In this case, the test set minimum error is 11.38-14 percent, and it happens when the max-depth is [8-10], and the training error is less than 5, percent.



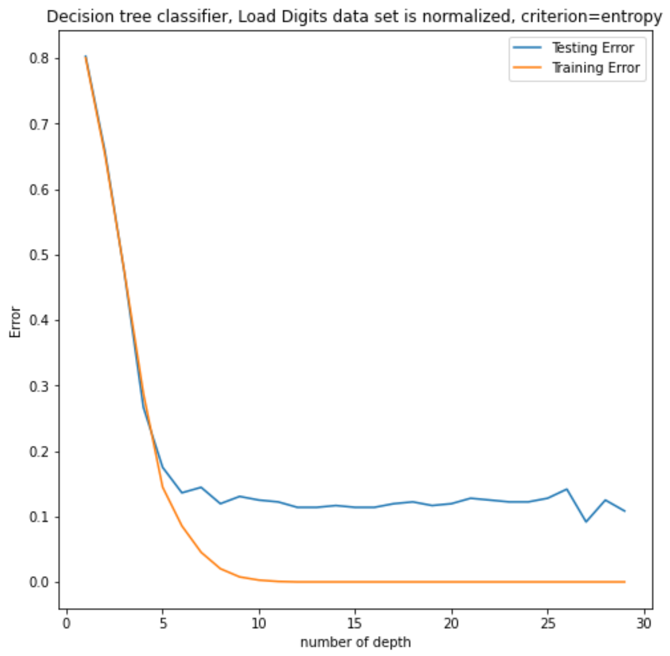
✓

3) "entropy" splitting criterion and the data set is not normalized: In this case, the test set minimum error is [10-14] percent, and it happens when the max-depth is [7-10], and the training error is almost 5, percent.



✓

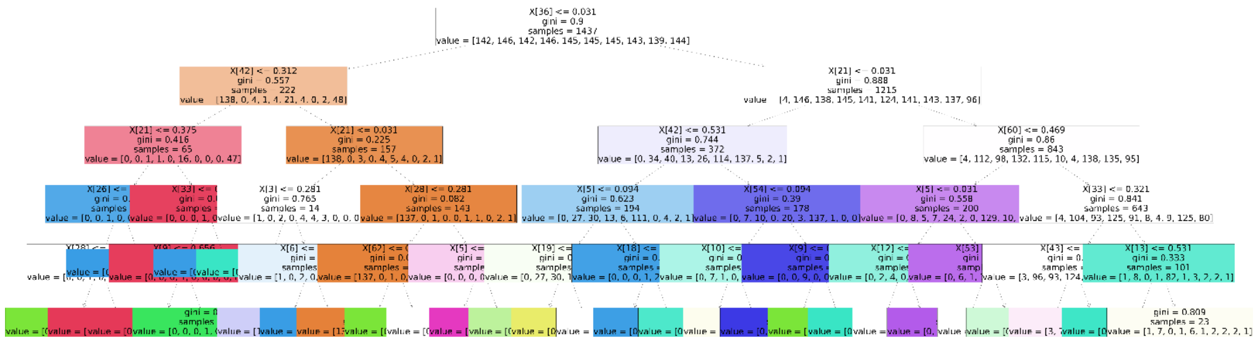
4) "entropy" splitting criterion and the data set is normalized: In this case, the test set minimum error is 9.4 percent, and it happens when the max-depth is [3, 4, 7, …], and the training error is [1.5, 0.4, 0, …], percent.



✓

The entropy criterion also might be a little slower to compute because it requires computing a logarithmic function. Many researchers point out that in most cases, splitting criteria will not make much difference in the tree performance. Each criterion is superior in some cases and inferior in others.

For k=5, the regression tree is plotted as follows:



X[36] <= 0.031
gini = 0.9
samples = 1437
value = [142, 146, 142, 146, 145, 145, 145, 143, 139, 144]
X[42] <= 0.312
gini = 0.557
samples = 222
value = [138, 0, 4, 1, 4, 21, 4, 0, 2, 48]
X[21] <= 0.031
gini = 0.888
samples = 1215
value = [4, 146, 138, 145, 141, 124, 141, 143, 137, 96]

Dataset exploration:

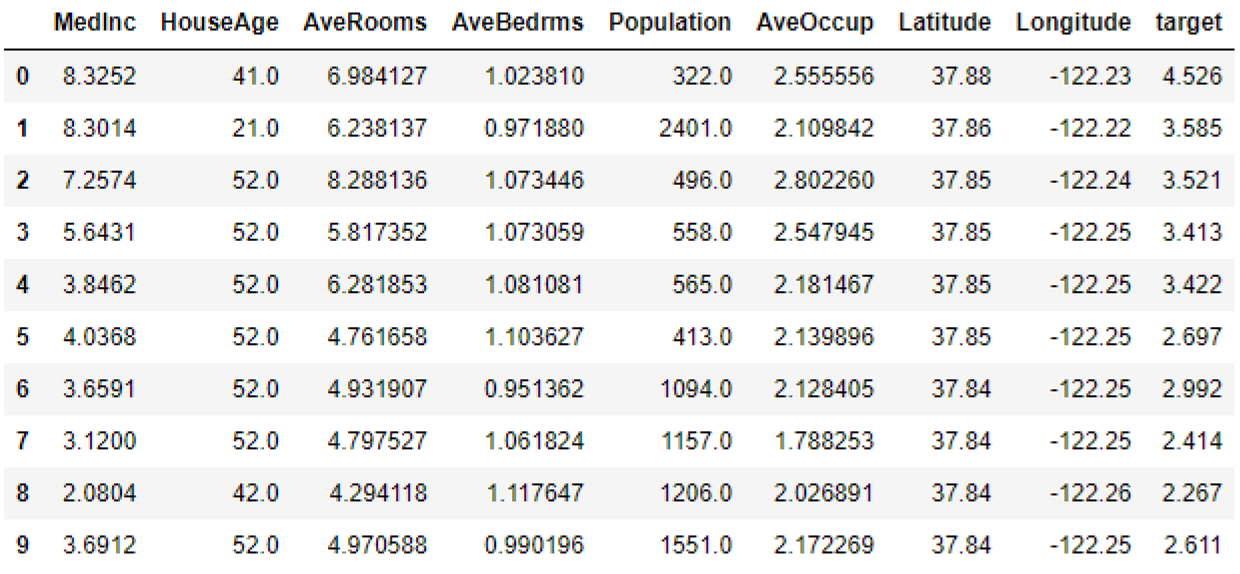
3. load digits information:

- Number of Instances: 20640
- Number of features: 8, all numeric

- Target Names: [‘MedHouseVal’]
- Feature Names:['MedInc', 'HouseAge','AveRooms', 'AveBedrms','Population', 'AveOccup', 'Latitude', 'Longitude']
- Missing Feature Values: None

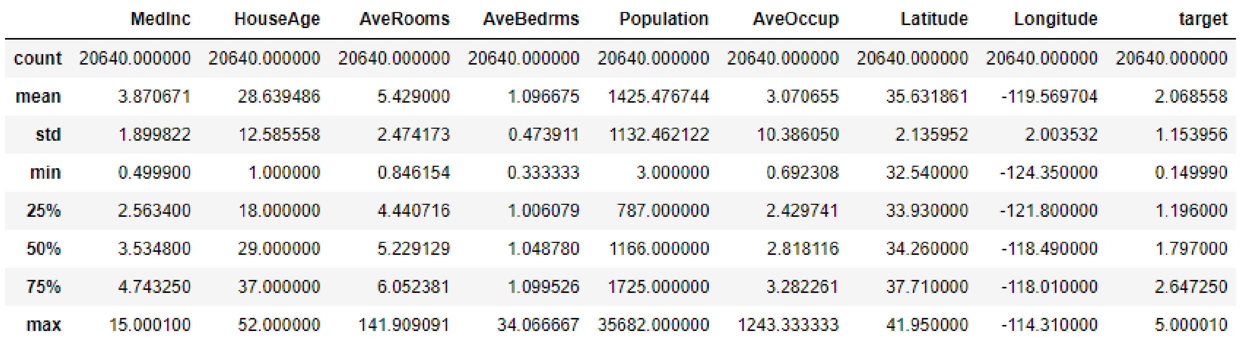
Visualize some of the data:

First ten samples of the dataset:



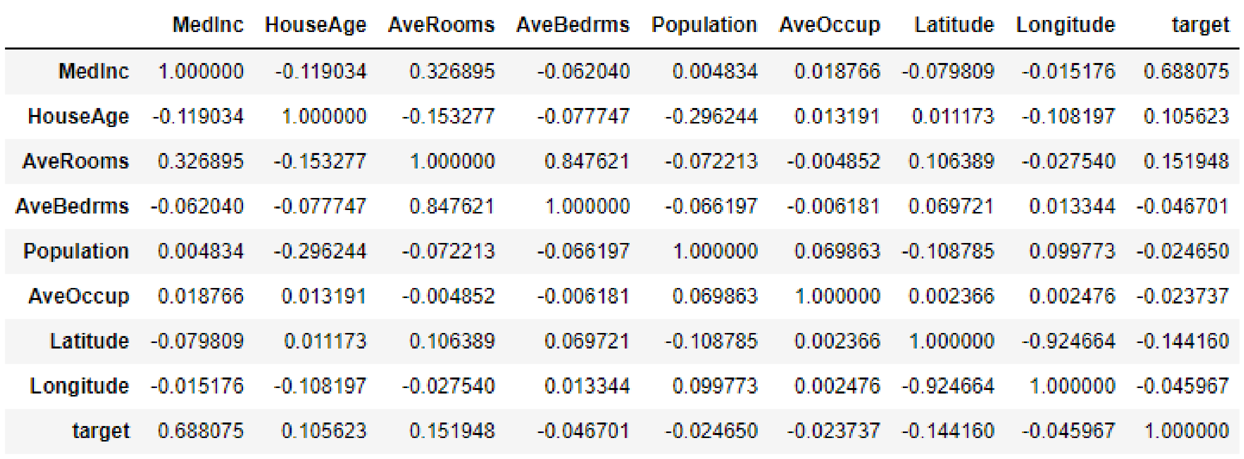
	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	target
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422
5	4.0368	52.0	4.761658	1.103627	413.0	2.139896	37.85	-122.25	2.697
6	3.6591	52.0	4.931907	0.951362	1094.0	2.128405	37.84	-122.25	2.992
7	3.1200	52.0	4.797527	1.061824	1157.0	1.788253	37.84	-122.25	2.414
8	2.0804	42.0	4.294118	1.117647	1206.0	2.026891	37.84	-122.26	2.267
9	3.6912	52.0	4.970588	0.990196	1551.0	2.172269	37.84	-122.25	2.611

The statistical description of the dataset: not-scaled



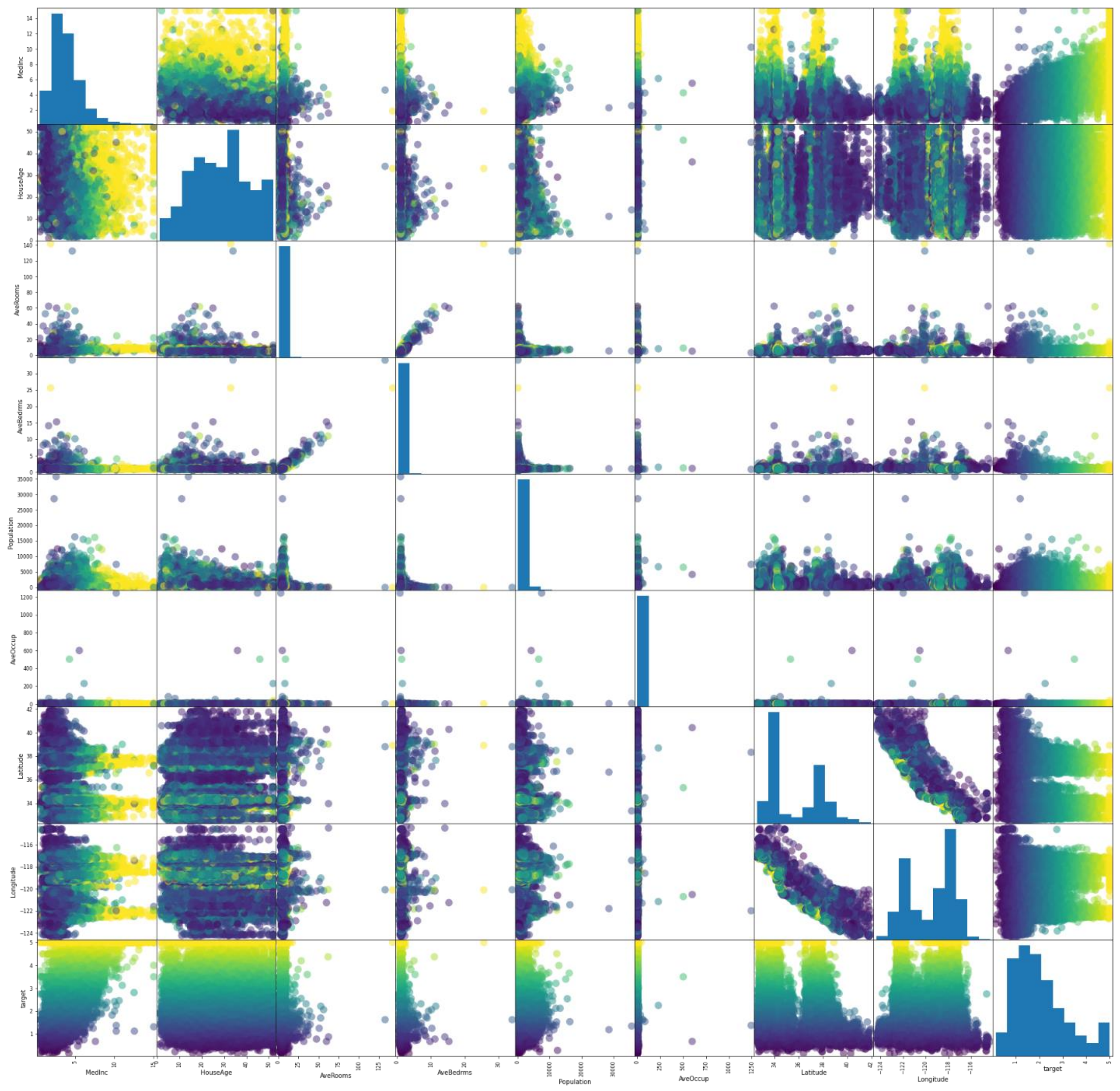
	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	target
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	3.870671	28.639486	5.429000	1.096675	1425.476744	3.070655	35.631861	-119.569704	2.068558
std	1.899822	12.585558	2.474173	0.473911	1132.462122	10.386050	2.135952	2.003532	1.153956
min	0.499900	1.000000	0.846154	0.333333	3.000000	0.692308	32.540000	-124.350000	0.149990
25%	2.563400	18.000000	4.440716	1.006079	787.000000	2.429741	33.930000	-121.800000	1.196000
50%	3.534800	29.000000	5.229129	1.048780	1166.000000	2.818116	34.260000	-118.490000	1.797000
75%	4.743250	37.000000	6.052381	1.099526	1725.000000	3.282261	37.710000	-118.010000	2.647250
max	15.000100	52.000000	141.909091	34.066667	35682.000000	1243.333333	41.950000	-114.310000	5.000010

The correlation coefficient (Pearson) between each set of features is calculated here.

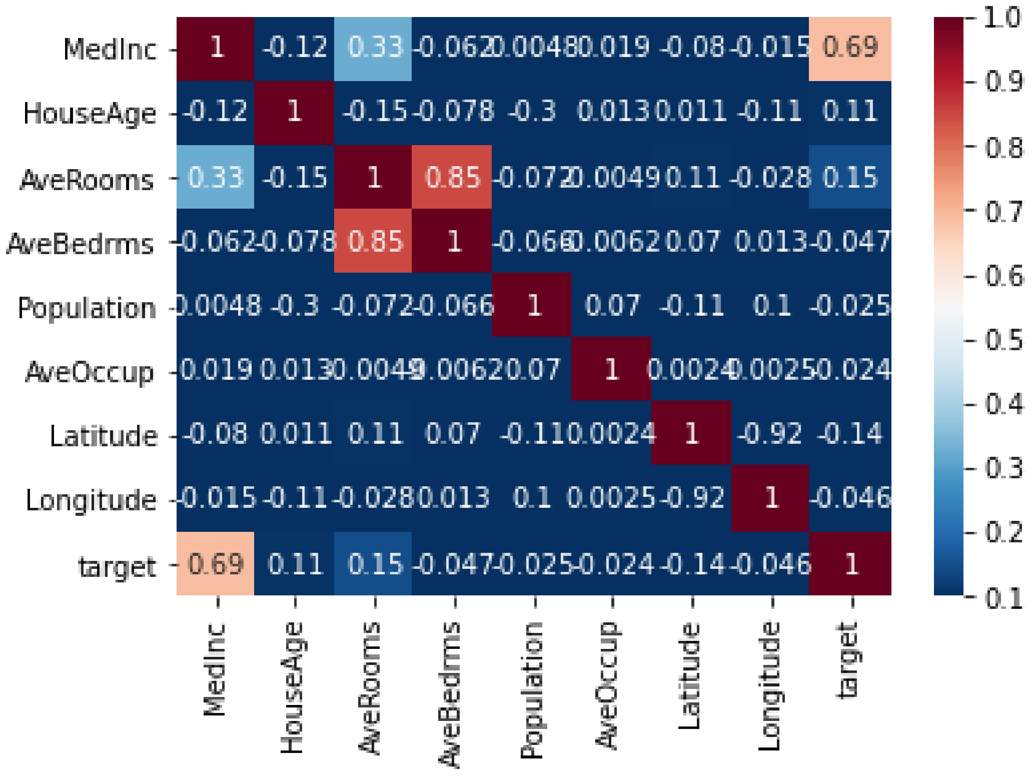


	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	target
MedInc	1.000000	-0.119034	0.326895	-0.062040	0.004834	0.018766	-0.079809	-0.015176	0.688075
HouseAge	-0.119034	1.000000	-0.153277	-0.077747	-0.296244	0.013191	0.011173	-0.108197	0.105623
AveRooms	0.326895	-0.153277	1.000000	0.847621	-0.072213	-0.004852	0.106389	-0.027540	0.151948
AveBedrms	-0.062040	-0.077747	0.847621	1.000000	-0.066197	-0.006181	0.069721	0.013344	-0.046701
Population	0.004834	-0.296244	-0.072213	-0.066197	1.000000	0.069863	-0.108785	0.099773	-0.024650
AveOccup	0.018766	0.013191	-0.004852	-0.006181	0.069863	1.000000	0.002366	0.002476	-0.023737
Latitude	-0.079809	0.011173	0.106389	0.069721	-0.108785	0.002366	1.000000	-0.924664	-0.144160
Longitude	-0.015176	-0.108197	-0.027540	0.013344	0.099773	0.002476	-0.924664	1.000000	-0.045967
target	0.688075	0.105623	0.151948	-0.046701	-0.024650	-0.023737	-0.144160	-0.045967	1.000000

The overview image of the correlation between each pair of the features is printed out here.



Heatmap of the data also shows the correlation of the different features. Only one of those features with high correlation might be enough to train the algorithm. Here we do not apply feature reduction.



Train and test:

• K-nearest-neighbor regressor:

The data set is split so that 80 percent of them are used for training the system and 20 percent for testing the algorithm. The mean-squared error vs. the number of neighbors (k) for both training and test sets, and for two distance metrics ($p = 1$ and $p = 2$) are plotted under four below conditions:

1) $\boldsymbol{p} = 1$, and the data set is not normalized

2) $\boldsymbol{p} = 1$, and the data set is normalized

3) $\boldsymbol{p} = 2$, and the data set is not normalized

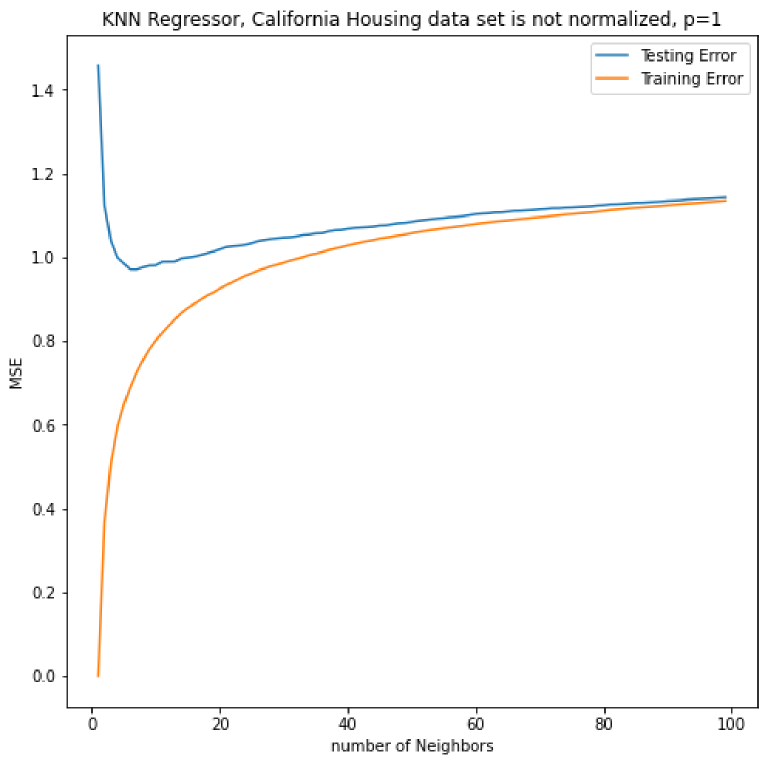
4) $\boldsymbol{p} = 2$, and the data set is normalized

The Mean squared errors tells you how close a regression line is to a set of points.

---

1) $\boldsymbol{p} = 1$, and the data set is not normalized: The most favorable k happens around the knee area of both training and test set MSE, which is [6-9] and the MSE is almost [0.97-1.1] for the test set and [0.7-0.82] for the training set.

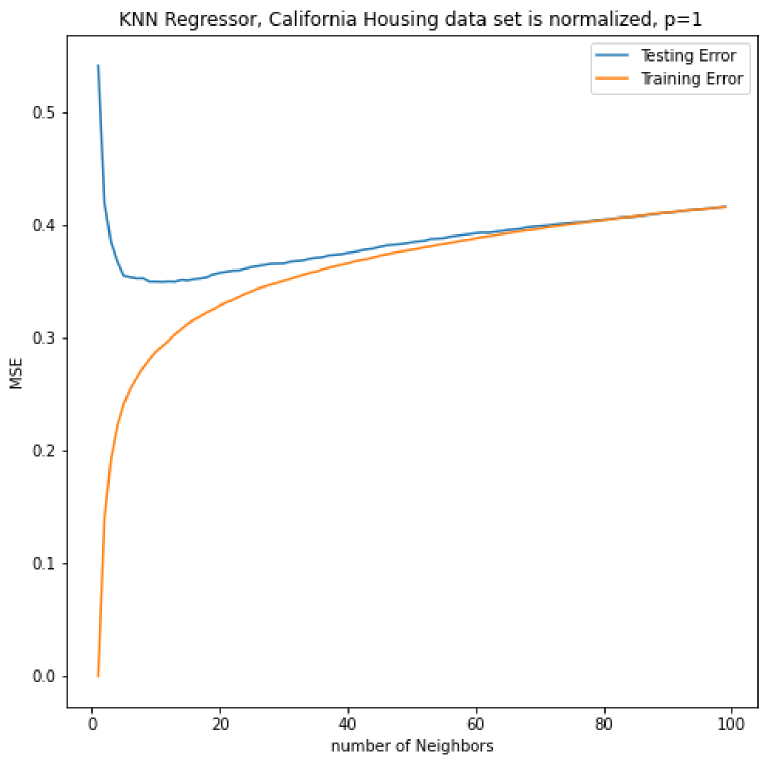
For the small number of neighbors, overfitting is happening, and as it goes up, the MSE increase and underfitting might happen.



2) $p = 1$, and the data set is normalized

The most favorable k happens around the knee area of both training and test set MSE, which is [6-14] and the MSE is almost [0.35-0.36] for the test set and [0.29-0.32] for the training set.

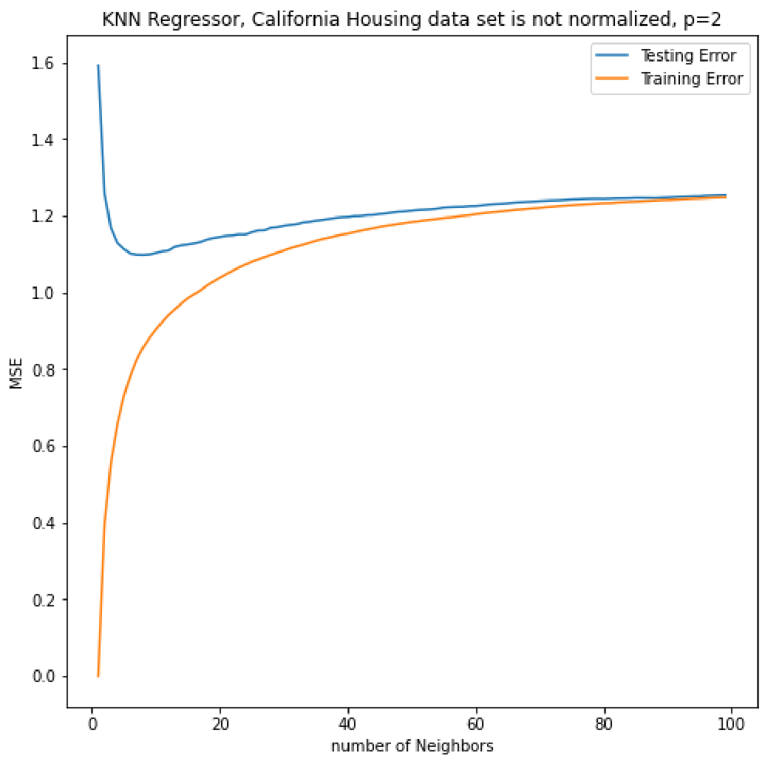
In comparison to previous case the normalization significantly reduced the MSE. Also, the knee area is more flat and we can chose the number of neighbors easier.



3) $p = 2$, and the data set is not normalized:

The most favorable k happens around the knee area of both training and test set MSE, which is [8-10], and the MSE is almost [1.1-1.12] for the test set and [0.85-1.0] for the training set.

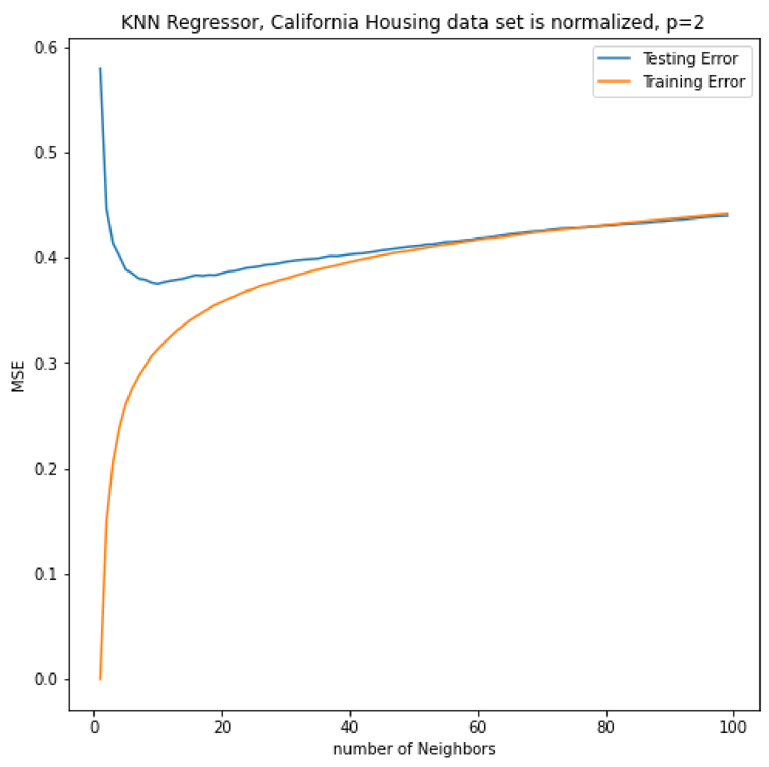
In comparison to case One, the p=1 comes to a slightly better result than p=2.



4) $p = 2$, and the data set is normalized:

The most favorable k happens around the knee area of both training and test set MSE, which is [9-12], and the MSE is almost [0.37-0.38] for the test set and [0.30-0.33] for the training set.

In comparison to case Two, the p=2 comes to a slightly better result than p=1. Also, normalizing data set reduced the MSE compared to case Three.



If the scale of features is very different, then normalization is required. This is because the distance calculation done in KNN uses feature values. When the one feature values are larger than the other, that feature will dominate the distance hence the outcome of the KNN.

In this case, we had different scale values for features; therefore, normalization was effective on the KNN algorithm.

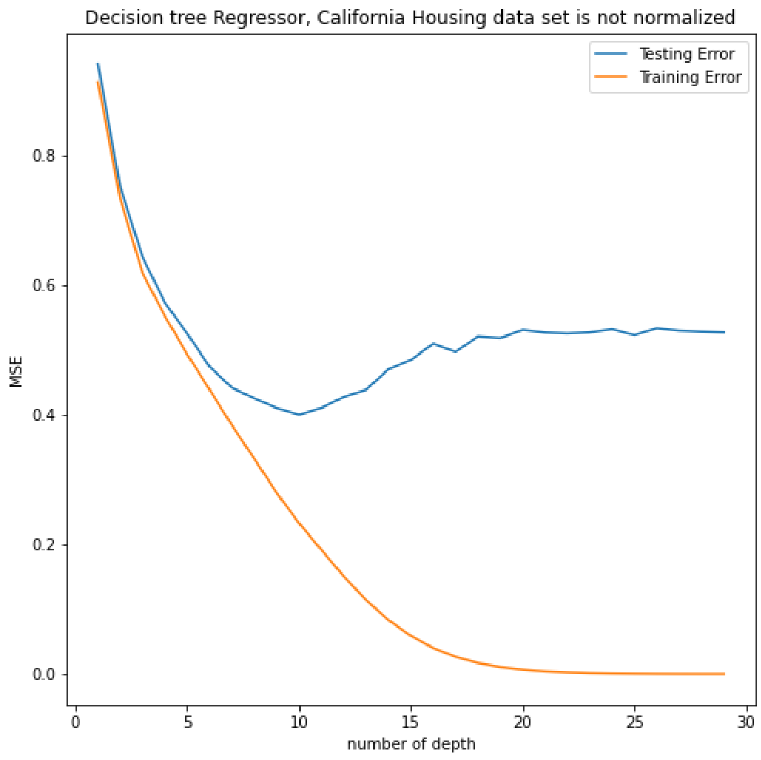
• Decision Tree Classifier:

The data set is split so that 80 percent of them are used for training the system and 20 percent for testing the algorithm. The mean-squared error vs. the max-depth for both training and test sets, are plotted under two below conditions:

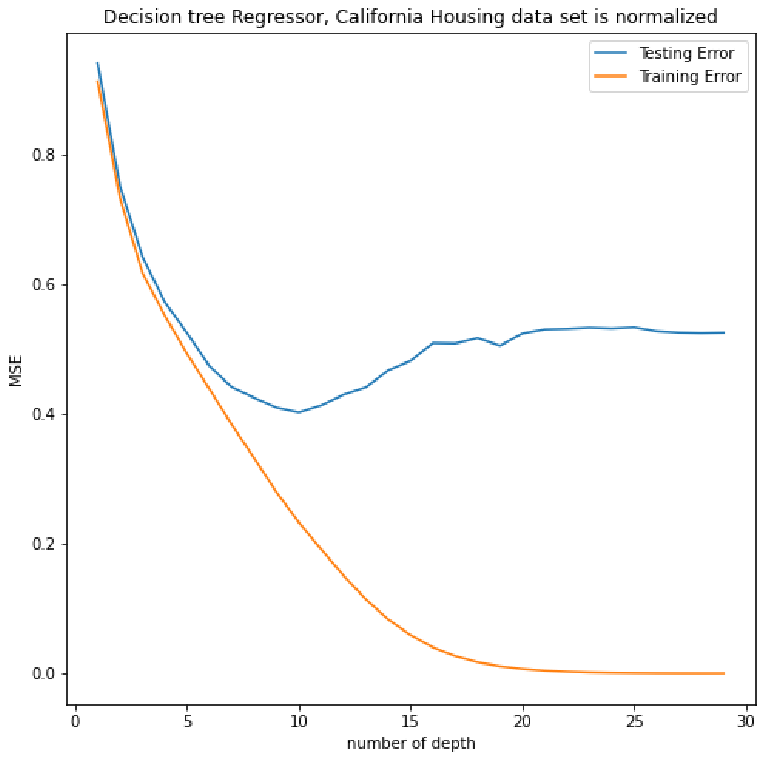
1) not normalized data set

2) normalized data set

1) not normalized data set:

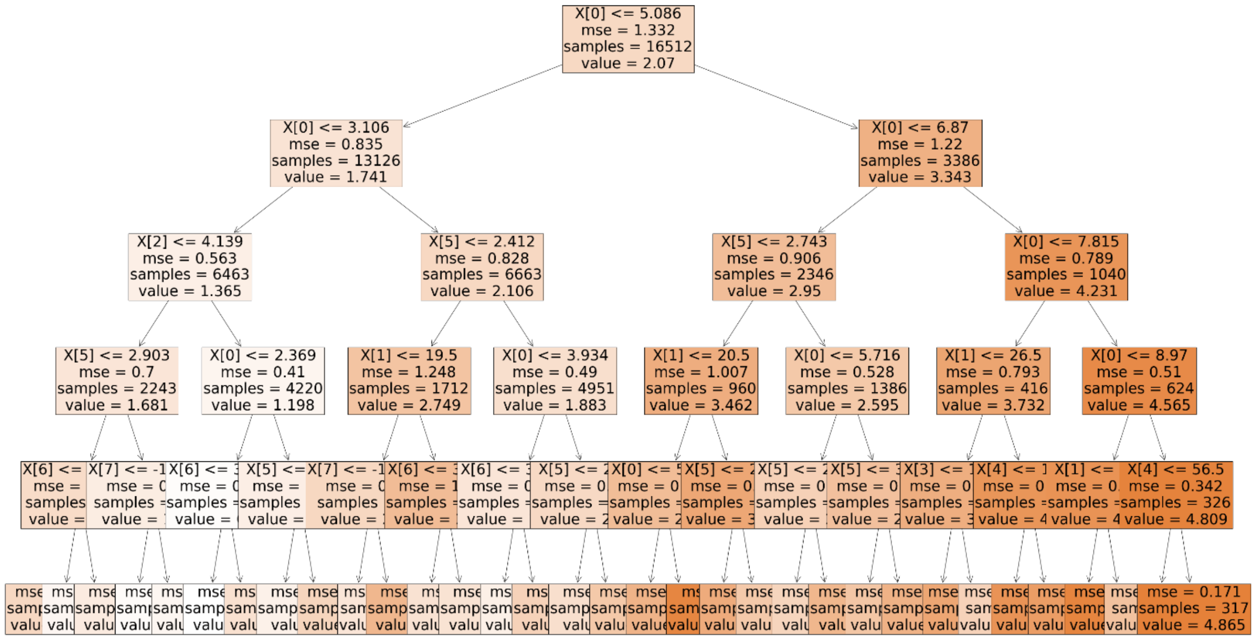


2) normalized data set:



In both cases above, the minimum MSE is 0.4, and it happens when the maximum depth is set to 10, and the training MSE is 0.23. The best choice for maximum depth is before the minimum MSE happens. Moreover, these plots also illustrate that normalizing data pose no major effect on decision tree regressor.

For k=5, the regression tree is plotted as follows:



X[0] <= 5.086
mse = 1.332
samples = 16512
value = 2.07
X[0] <= 3.106
mse = 0.835
samples = 13126
value = 1.741
X[0] <= 6.87
mse = 1.22
samples = 3386
value = 3.343
X[2] <= 4.139
mse = 0.563
samples = 6463
value = 1.365
X[5] <= 2.412
mse = 0.828
samples = 6663
value = 2.106
X[5] <= 2.743
mse = 0.906
samples = 2346
value = 2.95
X[0] <= 7.815
mse = 0.789
samples = 1040
value = 4.231
X[5] <= 2.903
mse = 0.7
samples = 2243
value = 1.681
X[0] <= 2.369
mse = 0.41
samples = 4220
value = 1.198
X[1] <= 19.5
mse = 1.248
samples = 1712
value = 2.749
X[0] <= 3.934
mse = 0.49
samples = 4951
value = 1.883
X[1] <= 20.5
mse = 1.007
samples = 960
value = 3.462
X[0] <= 5.716
mse = 0.528
samples = 1386
value = 2.595
X[1] <= 26.5
mse = 0.793
samples = 416
value = 3.732
X[0] <= 8.97
mse = 0.51
samples = 624
value = 4.565
X[4] <= 56.5
mse = 0.342
samples = 326
value = 4.809
mse = 0.171
samples = 317
value = 4.865