Report introML 20-20

Lorenzo Beltrame

25 October 2021

1 Task 1

1.1 Subtask 1

The first thing I did was to plot each component of the X-train space into a graph.

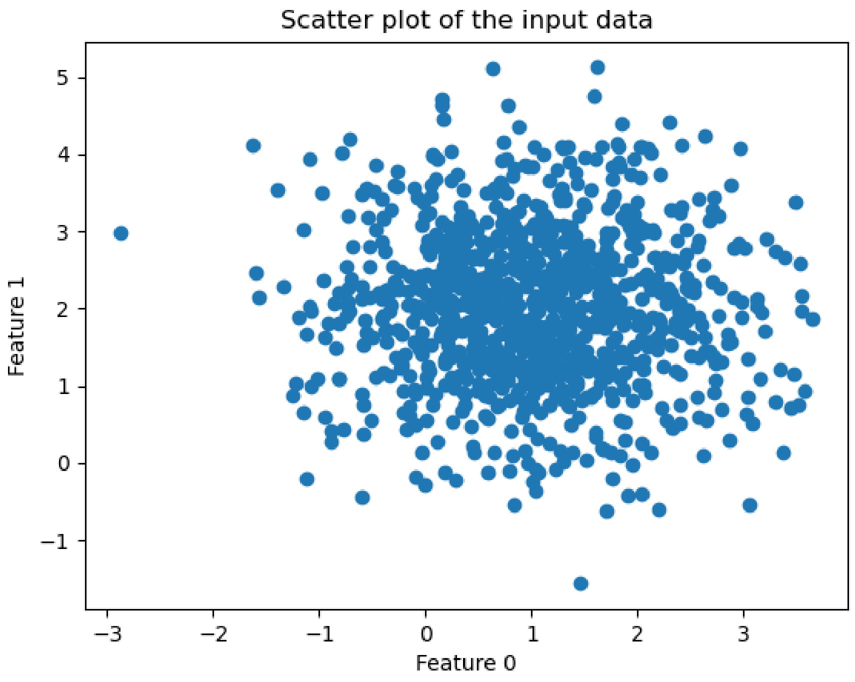


Figura 1: Scatter plot of the given features for the training dataset

The data present a globular distribution, therefore the data I generated are the distance from the central point of the globe.

In particular I calculated the mean value for both distribution and I assumed it to be the centre of my distribution. Later I computed the "radius" ρ by using the following:

$$\rho = \sqrt{(x - \bar{x})^2 + (y - \bar{y})^2} \tag{1}$$

Where x and y are the respective features and $\bar{x}$ and $\bar{y}$ the respective means rounded to an integer.

This alone already led to a really nice mse score on the test set (0.008).

I wanted to improve it, therefore I added some hyperbolic functions (tanh and cosh), the exponential function and the tangent. Since those equations are trivial, I won't report them here.

The result on the test data set was slightly better (0.007) but was totally unnecessary. I just did it because I found it interesting.

If we want to confront the result obtained with nonlinear features and the result obtained with linear features, the best way is trough a graph:

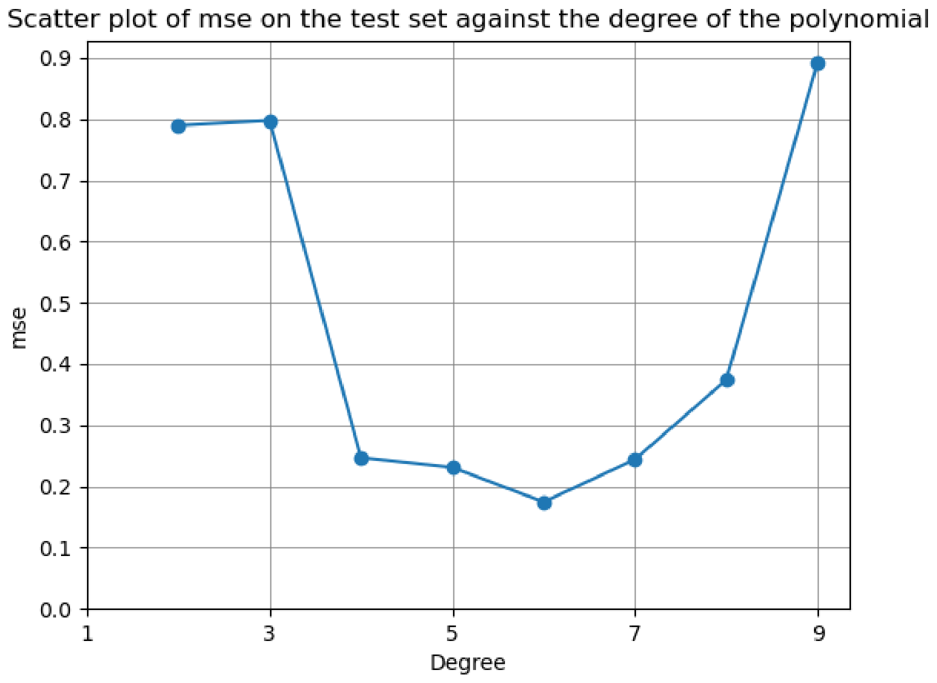


Figura 2: Plot of the mse calculated on the test dataset in function of the degree of the polynomial used.

The best polynomial degree resulted to be 6.

Although the results obtained trough a polynomial feature generation are satisfactory for a polynomial feature generation with degree ranging from 4 to 8, the non linear feature gives overall a better predictor on the test set.

It is possible to appreciate the overfitting for polynomial of order higher that 9.

1.2 Subtask 2

The sub task is similar to the previous one. I adopted the same approach since the distribution of the features was globular (The graph was extermly similar to the previous one, therfore I decided not to report it here. In the code it will be generated anyway).

On this basis, I performed the same feature generation with the exception of not including the radius (If I included the radius I obtained an accuracy of 0.949).

To perform the learning algorithm I used a Support Vector Machine (SVM) with a linear kernel. The accuracy score is indeed really good, since I obtained an accuracy of: 0.952.

To understand if it is a good classifier we plot the confusion matrix:

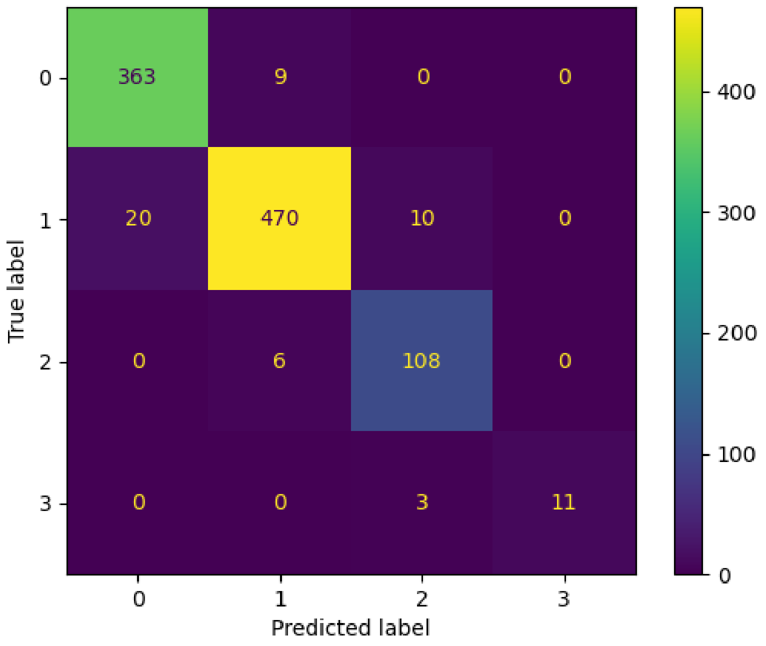


Figura 3: Confusion matrix for our classifier.

NOTE: there are extremely few points that belongs to class 4, therefore in the split I used to plot this confusion matrix, none of them belongs to the test set. As a consequence, the results are not presented on this matrix.

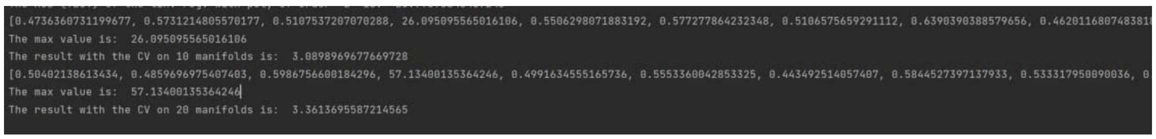
We can see that some classes are definitely composed of few elements, but the classification is still good, since most of the instances lie on the principal diagonal. Nevertheless, we do not have enough data to claim that our algorithm works well with the instances with labels ”3” or ”4”.

2 Task 2

2.1 Important note on the seed

In my programme I fixed the seed of the train-test split to be seed=20. This seed separate nicely the data into the train and the test, avoiding that the instances with high values of the "MedHouseValue" (the target) are split unevenly. This seed was found manually.

To solve a normal regression problem, optimising the seed is a mistake. On this dataset, considering that I am not allowed to do more preprocessing to my data, I chose to pick a "nice" seed in the didactic spirit of the assignment. I want to highlight that this is not the correct approach to a ML assigment. In fact, I observed that when I run the cross validation and the data are not split evenly, one or more manifolds present extremely high MSE values. I present an example screenshot from the execution of my code with a seed=42:



```
[0.4736360731199677, 0.5731214805570177, 0.5107537207070288, 26.095095565016106, 0.5506298071883192, 0.577277864232348, 0.5106575659291112, 0.6390390388579656, 0.462011680748381
The max value is:  26.095095565016106
The result with the CV on 10 manifolds is:  3.0898969677669728
[0.50402138613434, 0.4859696975407403, 0.5986756600184296, 57.13400135364246, 0.4991634555165736, 0.5553360042853325, 0.443492514057407, 0.5844527397137933, 0.533317950090036, 0
The max value is:  57.13400135364246
The result with the CV on 20 manifolds is:  3.3613695587214565
```

Figura 4: screenshot from the execution of my code with a seed different from seed=42. I printed, in order, all the MSE obtained during CV, the biggest MSE among them and their average.

As a reference for my claims I present the distribution of the various features of the dataset:

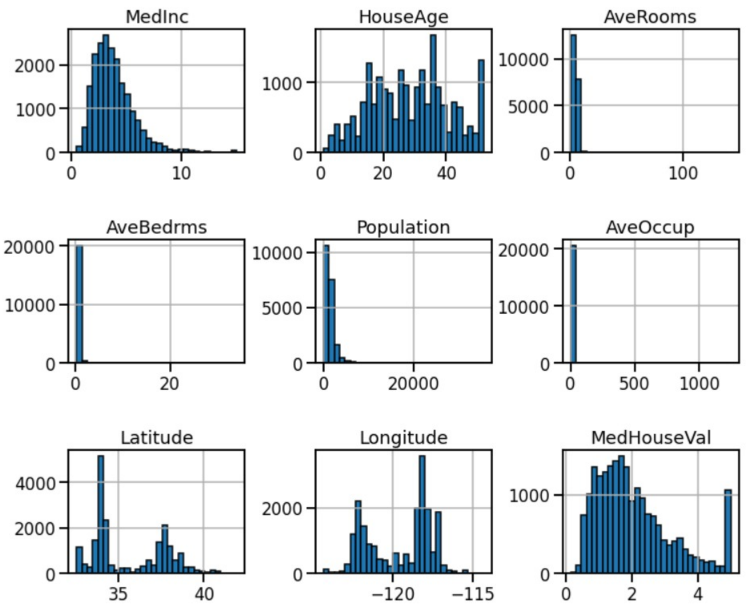


Figura 5: Distribution of the various features and the target (MedHouseVal).

2.2 Subtask 1

I performed a least square regression on the dataset "California Housing" with a train test split of [0.3/0.7]. The data were scaled with MinMaxScaler before being given to the learning algorithm.

The resulting MSE were indeed good and really similar for the test and the train dataset:

- The MSE (TRAIN) of the lin. reg. is: 0.53
- The MSE (TEST) of the lin. reg. is: 0.52

The MSE are satisfactory and since we obtain the same MSE on both sets I am not overfitting my model.

If I generate the quadratic features of my dataset and perform the same linear regression I observe the following results:

- The MSE (TRAIN) of the lin. reg. with poly of order 2 is: 0.40
- The MSE (TEST) of the lin. reg. with poly of order 2 is: 70.46

The fact that the test dataset presents an higher value of the MSE is expected, since we are overfitting our model. This hypothesis is also corroborated by the fact that adding polynomial features of order two led to a better training MSE score on the train set.

2.3 Subtask 2

I implemented the CV with the strategy "opt one manifold out".

NOTE: in my code I did not set a seed inside the CV since I was supposed to run the CV various times in the next section!

I obtained the following MSE:

- The result with the CV on 10 manifolds is: 0.55

The resulting MSE of the cross validation is only slightly worse then the MSE on the test dataset previously computed. This was expected, since we are training on a smaller subset of the training data in each iteration.

If I run the CV multiple time (100) and I calculate the variance of the results, I see that the results are really consistent since the variance is low:

- The variance of the CV over 100 repetitions is: 0.86
- The mean of the CV over 100 repetitions is: 0.81

If I increase the number of manifold to 20, I obtained the following MSE:

- The result with the CV on 20 manifolds is: 0.55

The resulting MSE of the cross validation with 20 manifolds is really similar to the CV MSE with 10 folds.

If I run the CV multiple time (100) and I calculate the variance of the results, I see that the results are really consistent since the variance is low:

- The variance of the CV over 100 repetitions is: 0.64
- The mean of the CV over 100 repetitions is: 0.74

We can see that we obtain better MSE results if we use k=20 in the CV, since both the mean and the average are lower. This is due to the fact that we are using more data in the training and we obtain more mses trough which compute the final cross validated MSE. The results I reported contain a random component, since I did not fixed a seed for the CV.

2.4 Subtask 3

If we want to select the best degree of the polynomial feature we want to add with respect to the training set, the best degree is 5 with the following MSE (TRAIN): 0.20.

This is due to the fact that we are clearly over fitting our model and increasing the order of the polynomial clearly lower the MSE calculated with respect to the training set.

If I divide my training set in half, I assign to half of it the name "New training" and to half of it the name "New testing". In that case, the best degree results 1 with the following MSE (TEST): 10.80.

The poor result is expected, since we are training of a small training set. The important part is that we are not over fitting our model, since higher-order polynomial feature would have generalised far worse error on the new test set.

Now I don't want to underfit and, at the same time, avoid to overfit the model when selecting the degree of the polynomial. To do so I implement a cross validation. I used both a 10 fold CV and a 20 fold CV.

The results in the programme may vary from the reported ones, since CV in a function without a fixed seed.

- K = 10: the best degree is: 2 with the following MSE (CV): 0.44
- K = 20: the best degree is: 1 with the following MSE (CV): 0.62

To test the generalisation capability of the algorithm I compute the MSE on the test dataset (7/10 of the whole data set). The results, with respect to the order the results were discussed, were:

1. The MSE (TEST) of the lin. reg. with poly of order 5 is: 5.38 e+18
2. The MSE (TEST) of the lin. reg. with poly of order 1 is: 0.52
3. The MSE (TEST) of the lin. reg. with poly of order 2 is: 70.47
4. The MSE (TEST) of the lin. reg. with poly of order 1 is: 0.52

The results, as expected, are not consistent.

The first approach is simply overfitting.

The approaches that yield to an optimal degree are the splitting of the dataset in half (approach 2) and the 20-fold CV (approach 4). This is mainly because they avoid to find a parameter that overfit the function, thus leading to an higher estimation of the error on the test set.

Among the two, I think that the 20-fold CV is more roboust.

The robustness of the CV is highlighted by the fact that, only using data from the train dataset, the algorithm does not select the highest degree polynomial. Estimating the ideal number of fold is non trivial: when we use large number of folds, we use more training data in each training iteration. This lowers the bias but generally increases the variance. On a more general setting and with more training points, I think that the best way to choose the degree of the polynomial features is through the 10-fold CV.