Name: Tan Ufuk Çelik, ID: 28285, Course: CS412

# Report

--
1-) Prepare a PDF report with the following and **include a link to your Colab Notebook at the top. making sure the link public (-20pts if missing or not public)!**
--

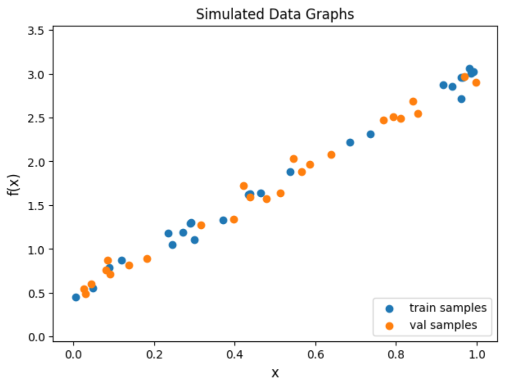
https://colab.research.google.com/drive/1Qw_ToIuXtzXolJJwD9FZ9RIPqEPqyX4z?usp=sharing

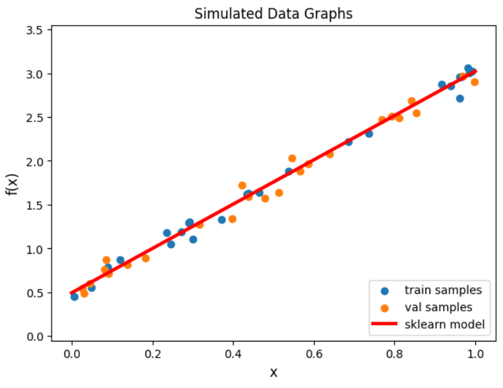
--
2-) Your report should contain sections in the same order as you're seeing in this notebook and labeled as **Part 1.a Results etc). **In each part, you should include the regression coefficients you have found in that part and all the plots and MSE errors.****
--

1.a Results: Here we simply train our model, make predictions, and then use the MSE method to determine how accurate this prediction is.
MSE: 0.00795462682779033

MSE is a measure of error that measures how far a model's predictions are from the true values. Therefore, it can be said that the lower it is, the better the performance of the model.
For example, the MSE value here is quite low (0.00795462682779033).
This shows that the predictions of the model match the real values quite well and indicates that the performance of the model is quite high.
However, it may not always be right to make a comment only on this value.
For the MSE value to be meaningful, other performance measures of the model must also be evaluated. In addition, factors such as the size, structure and purpose of the data set should be taken into account.

When we graph the data we have created here, we observe that the data in the (x,y) state is increasing gradually and the increase is linearly increasing proportionally and regularly. This gives us the meaning that we can predict that this graph will exist in a linear way in the future before we see the future data. In this way, we predict future data using the linear regression method.





1-b Results:

In fact, we find the same thing in all parts of task 1 using different methods.
Now we will use a manual pseudo inverse solution.

MSE of manual model: 0.06362852709262726

I don't put the graphs again because we got the same graph as 1.a.

Note to myself about part 1.b: The x_extended_val matrix contains the feature matrix that the linear regression model will use, while the w matrix represents the coefficients matrix that the linear regression model will predict.
When these two matrices are multiplied, a vector containing the predicted outputs of the linear regression model is created.

1.c Results:

In this part, we use the gradient descent algorithm.
Thanks to the algorithm we use in every step of this algorithm, the value of MSE decreases numerically, which means that our margin of error improves.

I do not put the graphics here because the same graphic is formed as a graphic.

```
SE error at step 1: Train: 3.1845, Val: 2.5459
MSE error at step 100: Train: 0.0615, Val: 0.0586
MSE error at step 200: Train: 0.0148, Val: 0.0161
MSE error at step 300: Train: 0.0069, Val: 0.0092
MSE error at step 400: Train: 0.0056, Val: 0.0081
MSE error at step 500: Train: 0.0053, Val: 0.0079
MSE error at step 600: Train: 0.0053, Val: 0.0079
MSE error at step 700: Train: 0.0053, Val: 0.0079
MSE error at step 800: Train: 0.0053, Val: 0.0080
MSE error at step 900: Train: 0.0053, Val: 0.0080
MSE error at step 1000: Train: 0.0053, Val: 0.0080
```

In the last step, Best MSE for x_train: 0.005275155082594279

2.a Results:

In this part, we created our first nonlinear data.
When we shape them, these are the outputs:

x_train: (25, 1)
x_val: (25, 1)
y_train: (25, 1)
y_val: (25, 1)

Using the "include_bias=False parameter, PolynomialFeatures()
We are preventing the function from generating the additional constant term property.

#Not: şu "include_bias=false" neden false bunu bir TA hocaya sor.

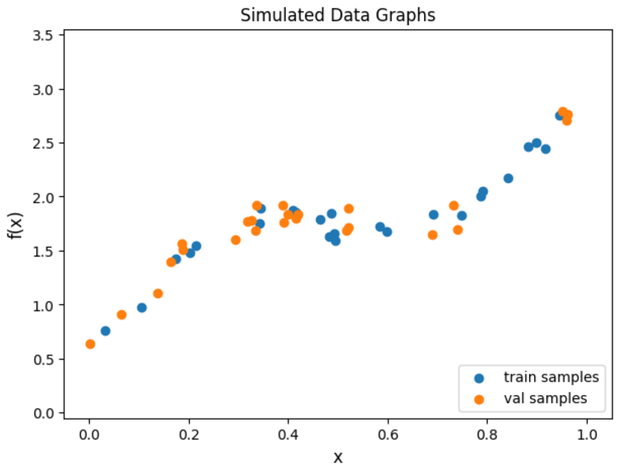
Here, however, if our data does not progress linearly, we can use the polynomial regression estimation method if it proceeds in a polynomial way instead. (I thought this graph might be useful for simple dollar/tl graphs.) As an extra here, the degrees of the polynomial are important because it gives an idea about the curvature of the polynomial. This k value, that is, its degree, is important in order to fit our data better. We've been given ratings of 1,3, 5, 7 to try here.

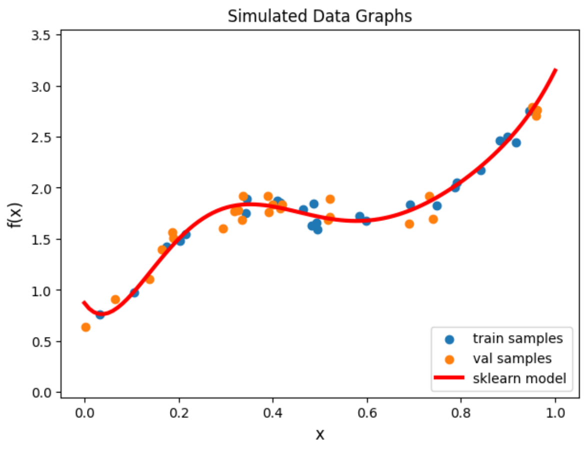
Here is the result as follows:

MSE of sklearn model for k= 1 : 0.06362852709262727
MSE of sklearn model for k= 3 : 0.012059223742868 09
MSE of sklearn model for k= 5 : 0.007475463079281072
MSE of sklearn model for k= 7 : 0.011549227838818095

Best Model k= 5 and MSE: 0.007475463079281072

The result from our graphs is as follows:





2.b Results:

```
# 1. construct the data matrix for train
X_train = np.concatenate([np.ones((len(x_train), 1)), x_train ** 3, x_train ** 2, x_train],
axis=1)

# 2. construct the data matrix for val
X_val = np.concatenate([np.ones((len(x_val), 1)), x_val ** 3, x_val ** 2, x_val], axis=1)
```

Here, after converting our data to a more suitable matrix, we take the pseudoinverse and find the w value. Then we calculate our MSE value.

```
MSE of sklearn model:  0.012059223742868233
```

And then we find the same graph as the graph in 2.a.

General:
In this assignment, we generally tried linear regression and polynomial regression estimation methods and different methods to reach their MSE values. In this way, we saw how the forecast was progressing by seeing the different methods and visualizing the data and the red line for the forecast.

***   In Part 1, you should comment on whether the gradient descent solution is the same (or very close) to solutions obtained for Part1.a and b. If not, add a line of explanation as to why you think it is not.**

The difference between the results of these methods may depend on many factors, such as the size of the dataset, the complexity of the linear regression model, and the parameters used. However, both methods train the linear regression model and both evaluate their performance using an error measure, the mean square error (MSE). The gradient descent method is a common method used in solving linear regression problems, but if the correct hyperparameters are not selected or a model that is compatible with the dataset structure, this method may not give accurate results like the method presented in Chapter 1.

----

*   In Part 2, comment on the effect of the **degree** parameter. What happens when it is chosen too small or too big? What do you think is the optimal **degree** value, and why? Discuss from the perspective of **underfitting**/**overfitting**.

I mentioned this part lightly above, but I'm talking in more detail:

If the degree parameter is chosen too small, a situation may arise where the model will be insufficient, that is, underfitting may occur. In this case, the model may not be able to make accurate predictions because it is not adequately suited to the complexity of the data. On the other hand, if the degree parameter is chosen too large, a situation may arise where the model will be too complex and may experience overfitting. In this case, the model will be very sensitive to the noise or variance of the data. This means that the model will not perform well on real-life data. Therefore, the optimal degree parameter may vary depending on the data set. The most appropriate grade should be chosen according to the complexity of the data. If the complexity of the data is low, the rating can be kept low. But as the complexity of the data increases, the degree can be increased. In general, the grade selection should be done correctly so that the model does not experience overfitting.