CS412 - HOMEWORK 2 – AHMET ALPEREN GÜNGÖR – 28847

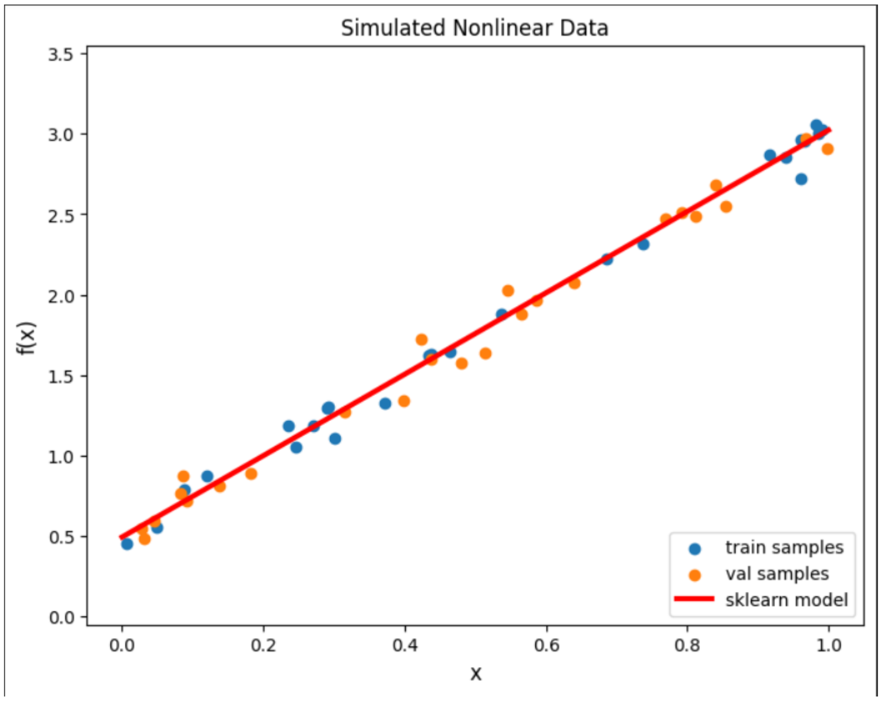
https://colab.research.google.com/drive/17qpBokiNfzM5GlIfGozp7qlc0J_Wrki-?usp=sharing

PART 1)

- First, dataset 1 is created, which has linear relationship. Then the training and validation sets are constructed with same sizes. (25,1 to 25,1)

PART 1.A)

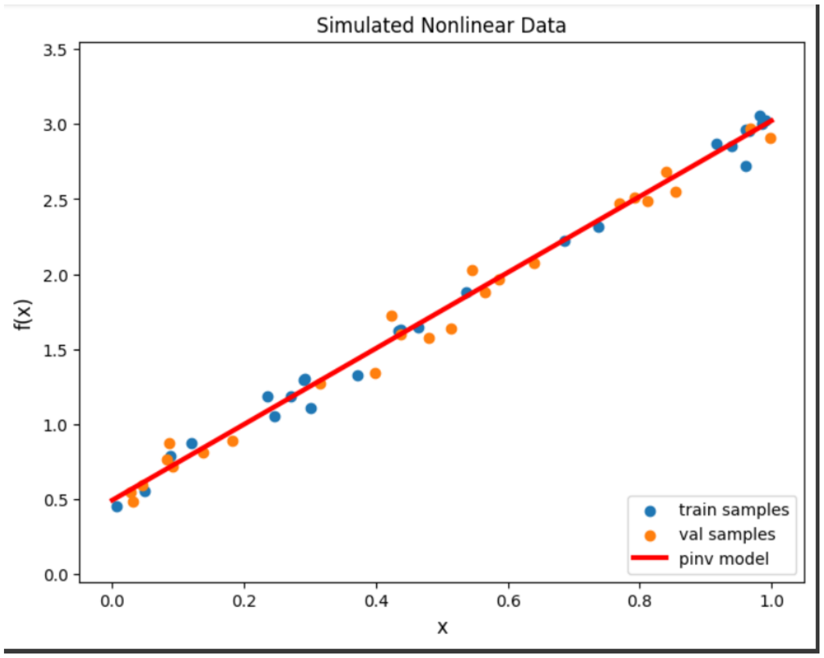
- The linear model lr_model is initialized and fitted to the train data. Intercept of the model is 0.49447669 and coefficient of x1 is 2.52838262.
- Predictions are made using validation data, which resulted with a mean squared error of 0.00795462682779033.
- Scatter plot of train & validation sets are drawn, then regression line is drawn into plot, which can be seen below.



PART 1.B)

- Using concatenate function, train and validation data is extended into matrix of 25x2 by adding a column of 1 for the intercept in order to better represent linear model with an intercept and a coefficient for x1.
- Pseudo-inverse of the matrices are created by linalg.pinv function, then train matrix is multiplied by y_train to find regression coefficients w, which has the values 0.49447669 and 2.52838262, where first one is the intercept and second one is the coefficient of x1, same result as part 1a.

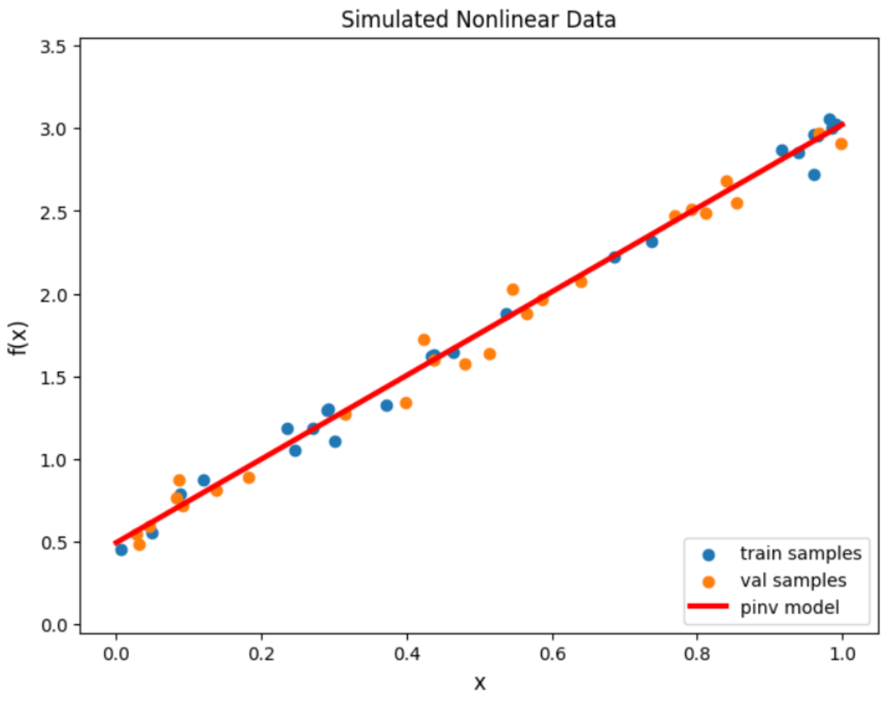
- When mean squared error of this manual model is calculated, it has the same score as part 1a, which is 0.007954626827790339
- Scatter plot of train & validation sets are drawn, then regression line is drawn into plot, which can be seen below.



PART 1.C)

- Random regression coefficients are random normally initialized using random.randn (I preferred this one over zeros, since it reduces the risk of all coefficients being stuck in the same local minima – even though local minima is not an issue for this dataset since it has a linear relation)
- Then for each iteration, steps of gradient descent (compute the values for the current coefficients, compute the difference, compute the cost, update accordingly) are done, which resulted with the following mse rates:
    - **MSE error at step 1: 1.6590**
    - **MSE error at step 100: 0.2168**
    - **MSE error at step 200: 0.0412**
    - **MSE error at step 300: 0.0114**
    - **MSE error at step 400: 0.0063**
    - **MSE error at step 500: 0.0055**
    - **MSE error at step 600: 0.0053**
    - **MSE error at step 700: 0.0053**
    - **MSE error at step 800: 0.0053**
    - **MSE error at step 900: 0.0053**
    - **MSE error at step 1000: 0.0053**
- After step 500, mse converged into 0.0053. The intercept after 1000 iterations is 0.49456397, and the coefficient of x1 is 2.52823264

- **As it can be seen from the values above, gradient descent algorithm found nearly identical values for coefficients and intercept, with very close mean squared error value. For a linear model like that it is expected since there are no local minima to disturb the algorithm and the depth is not too high to cause gradient inactive.**
- Scatter plot of train & validation sets are drawn, then regression line is drawn into plot, which can be seen below.



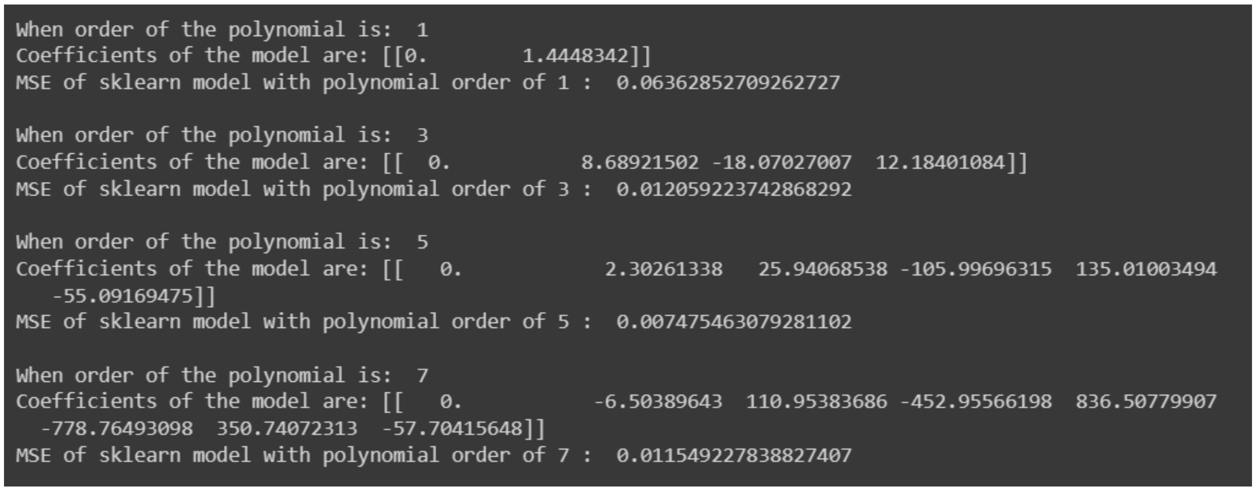
PART 2)

- Datasets with non-linear relationships are loaded from .npy files, then training and validation sets are constructed with the same sizes. (25,1 to 25,1)

PART 2.A)

- By using various degrees (1,3,5 and 7) of polynomials, Linear Regression model is trained.

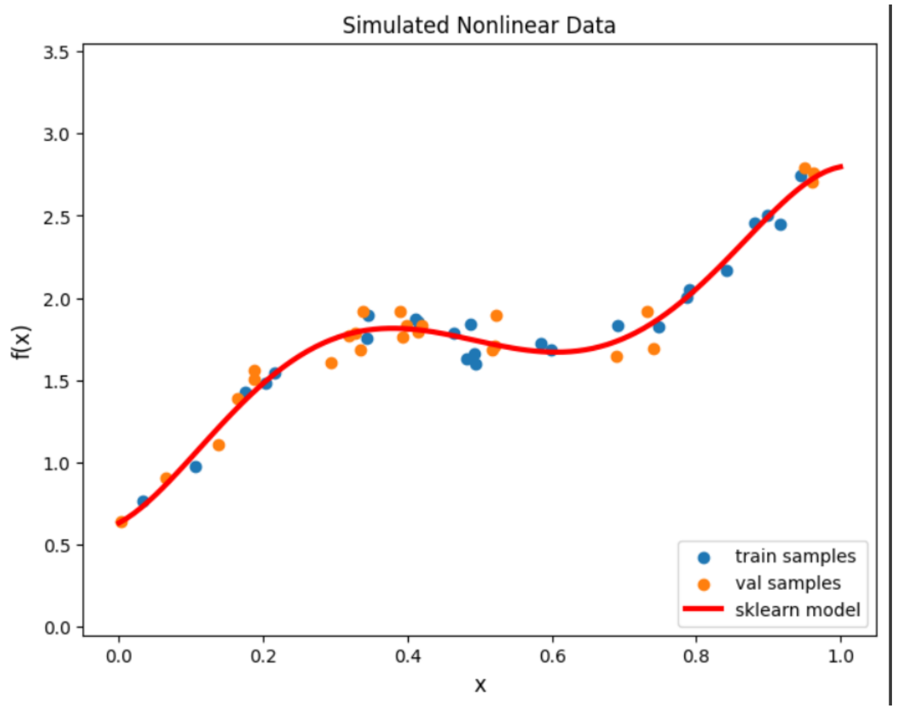
- Coefficients for each degree, as well as their mse on validation set are below.



```
When order of the polynomial is:  1
Coefficients of the model are: [[0.        1.4448342]]
MSE of sklearn model with polynomial order of 1 :  0.06362852709262727

When order of the polynomial is:  3
Coefficients of the model are: [[  0.           8.68921502 -18.07027007  12.18401084]]
MSE of sklearn model with polynomial order of 3 :  0.012059223742868292

When order of the polynomial is:  5
Coefficients of the model are: [[   0.            2.30261338   25.94068538 -105.99696315  135.01003494
   -55.09169475]]
MSE of sklearn model with polynomial order of 5 :  0.007475463079281102

When order of the polynomial is:  7
Coefficients of the model are: [[   0.           -6.50389643  110.95383686 -452.95566198  836.50779907
  -778.76493098  350.74072313  -57.70415648]]
MSE of sklearn model with polynomial order of 7 :  0.011549227838827407
```

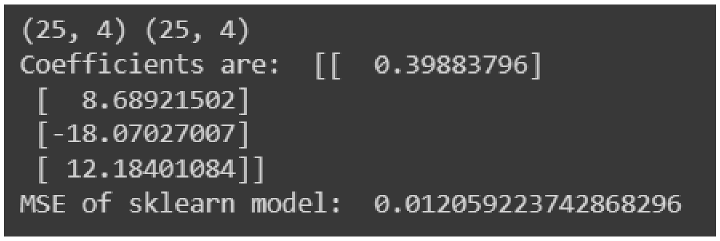
- As it can be seen above, polynomial of degree 5 has the least mse, meaning the best model hyperparameter for the data.

- **There isn't a linear relation between the degree of polynomial and performance of the model. Even though higher orders fit the training data better, too much of a fit may cause overfitting; resulting in worse performance for validation (and test) data. Therefore, compared to degree of 1 or 7, 3 or 5 is a better choice to make this data. And as it can be seen from the results above, degree of 5 gave the best performance.**
- Scatter plot of train & validation sets are drawn, then regression line is drawn into plot, which can be seen below. (polynomial degree of 5 is used for this plot, since it had best outcome)



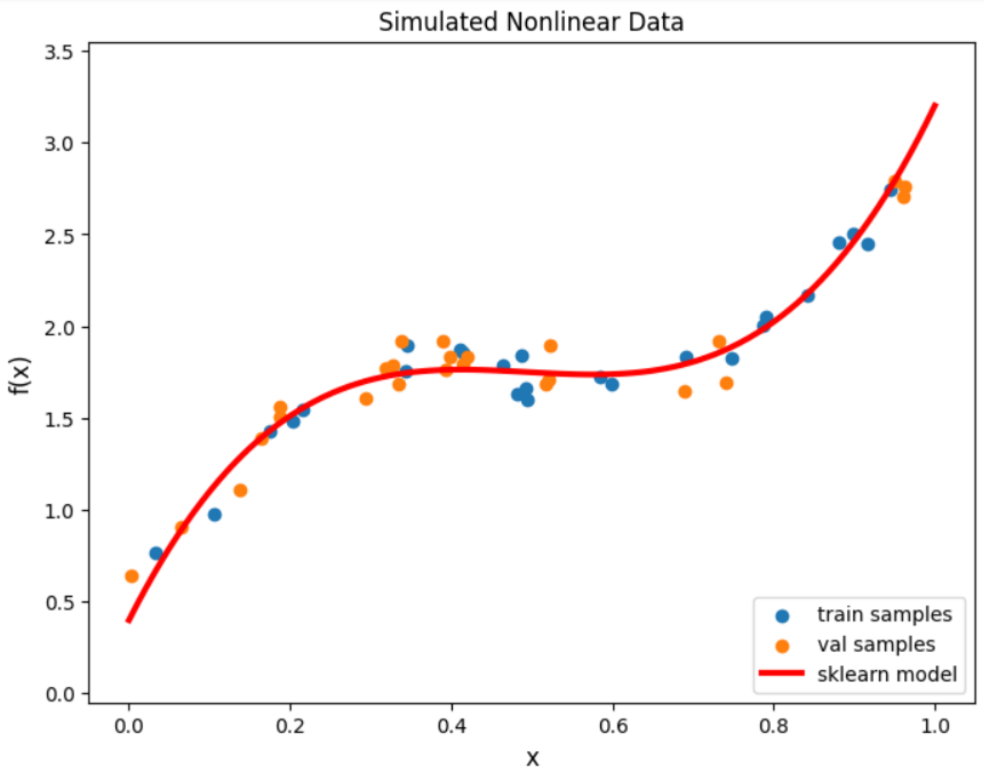
PART 2.B)

- By using polynomial features with degree 3, train and validation matrix are created (has degree 3, intercept is 1). They have shapes (25,4)
- By multiplying inverse of train matrix with their y_train, coefficients (w) are found, and these values are tested on validation data with the following mse value and coefficients:



```
(25, 4) (25, 4)
Coefficients are:  [[  0.39883796]
 [  8.68921502]
 [-18.07027007]
 [ 12.18401084]]
MSE of sklearn model:  0.012059223742868296
```

- Scatter plot of train & validation sets are drawn, then regression line is drawn into plot, which can be seen below.


Simulated Nonlinear Data
f(x)
x
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
train samples
val samples
sklearn model