MDL

# Assignment2

## Task 1:

In the case of supervised machine learning, the method LinearRegression().fit() is used to train a linear regression model on a given dataset.
Linear regression is a machine learning
algorithm that can be used to predict values over a continuous range using a constant slope

There are 2 types of Regression:
1.Simple Regression:

$$y = w_1 x + b$$

where,

- $y$ = Predicted value/Target Value
- $x$ = Input
- $w_1$ = Gradient/slope/Weight
- $b$ = Bias

(only one feature)

2.Multivariable Regression:

$$y = b + \sum_{i=1}^{n} w_i x_i$$

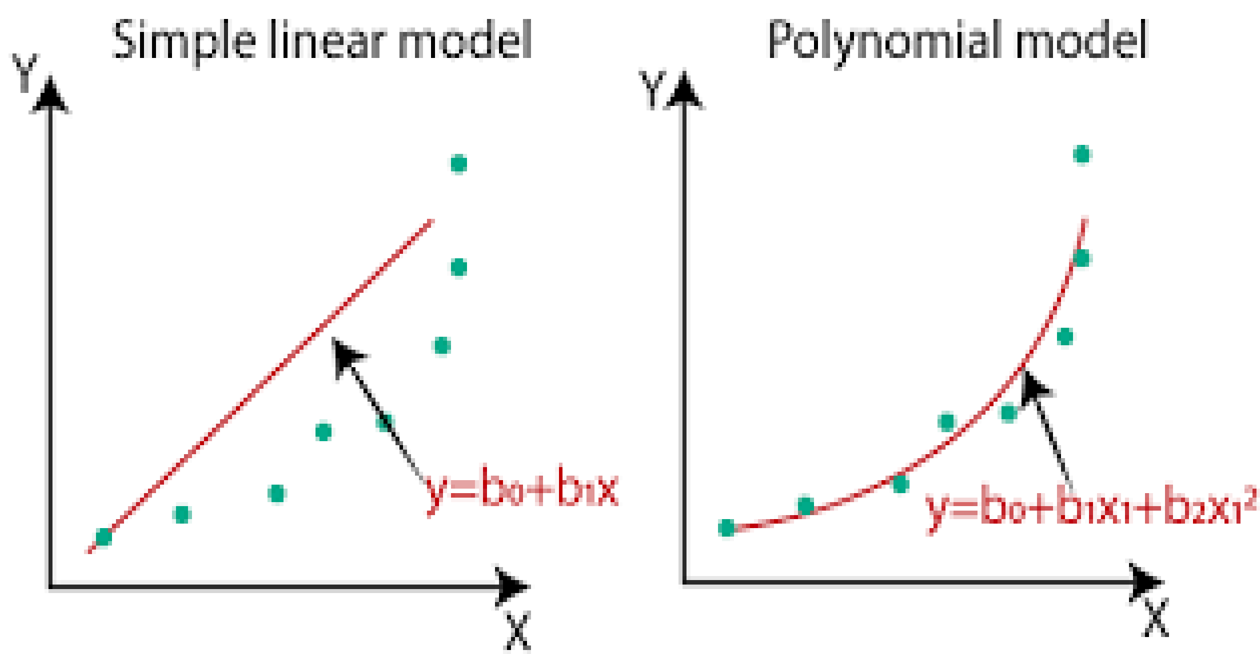
fit() takes in the data of independent variables and their corresponding dependent variable and ***gives the best fit coefficients*** for the model equation.It also gives slope.

Once a model is trained it can be used to predict the target values(**y**) for a given test set.
It uses MSE to find the cost/error in the model

$$1/n \sum (y' - y_i)^2$$

MSE where y' is predicted value yi is actual value

fit() tries to minimize the MSE to improve accuracy of the model by using **Ordinary Least Squares(OLS)** method.
But **Gradient Descent** can also be used in fit().



## Task 2: Gradient Descent

When there is only one independent variable and one dependent variable we have:

$$y = mx + b$$

The MSE(cost function) will be

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(y_i - (mx_i + b))^2$$

Where yi are the actual values from the data set,
(mxi+b) are the predicted values.
Thus we have to make the model best-fit “**learn**” m,b values.
So we use Partial Derivatives,

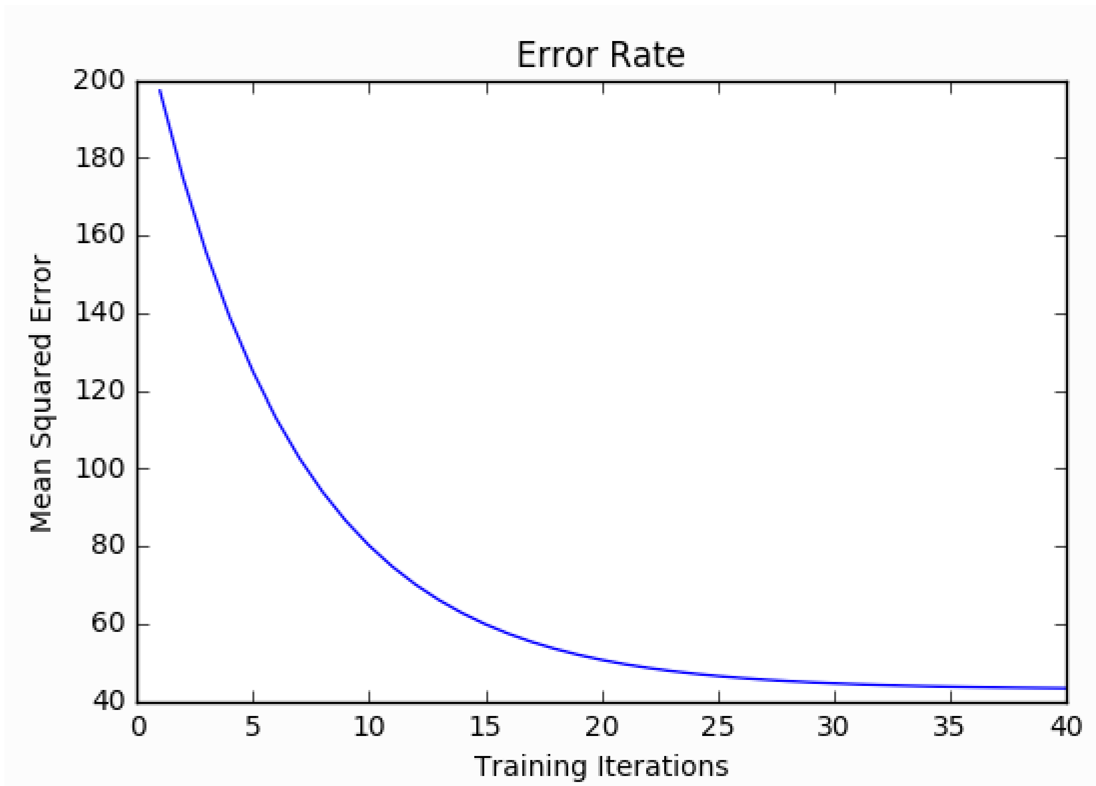
$$f(m,b) = \frac{1}{N}\sum_{i=1}^{n}(y_i - (mx_i + b))^2$$

Now by applying chain rule to find **df/dm** and **df/db** we get

$$f'(m,b) = \begin{bmatrix} \frac{df}{dm} \\ \frac{df}{db} \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\sum -x_i \cdot 2(y_i - (mx_i + b)) \\ \frac{1}{N}\sum -1 \cdot 2(y_i - (mx_i + b)) \end{bmatrix}$$
$$= \begin{bmatrix} \frac{1}{N}\sum -2x_i(y_i - (mx_i + b)) \\ \frac{1}{N}\sum -2(y_i - (mx_i + b)) \end{bmatrix}$$

Now we iterate through the data points and calculate the average partial derivatives and
the slope of the function we get at **our current position is gradient ascent** thus we take the *negative of gradient and move in the opposite direction to reduce the cost/error.*

We also use the **learning rate** – the size of steps taken every time we calculate gradient.
*We do this until we reach the minimum threshold or we fail to reduce cost in subsequent iterations.*



## Task 3:(2.3.2)

| | Bias | Variance |
|---|---|---|
| 1 | 0.269831 | 0.005708 |
| 2 | 0.086700 | 0.001127 |
| 3 | 0.033488 | 0.000463 |
| 4 | 0.023947 | 0.000523 |
| 5 | 0.023811 | 0.000578 |
| 6 | 0.023972 | 0.000672 |
| 7 | 0.024629 | 0.001041 |
| 8 | 0.024492 | 0.001374 |
| 9 | 0.026168 | 0.002083 |
| 10 | 0.026308 | 0.003341 |
| 11 | 0.027670 | 0.004764 |
| 12 | 0.030992 | 0.026002 |
| 13 | 0.030812 | 0.021660 |
| 14 | 0.037333 | 0.039962 |
| 15 | 0.055917 | 0.249697 |

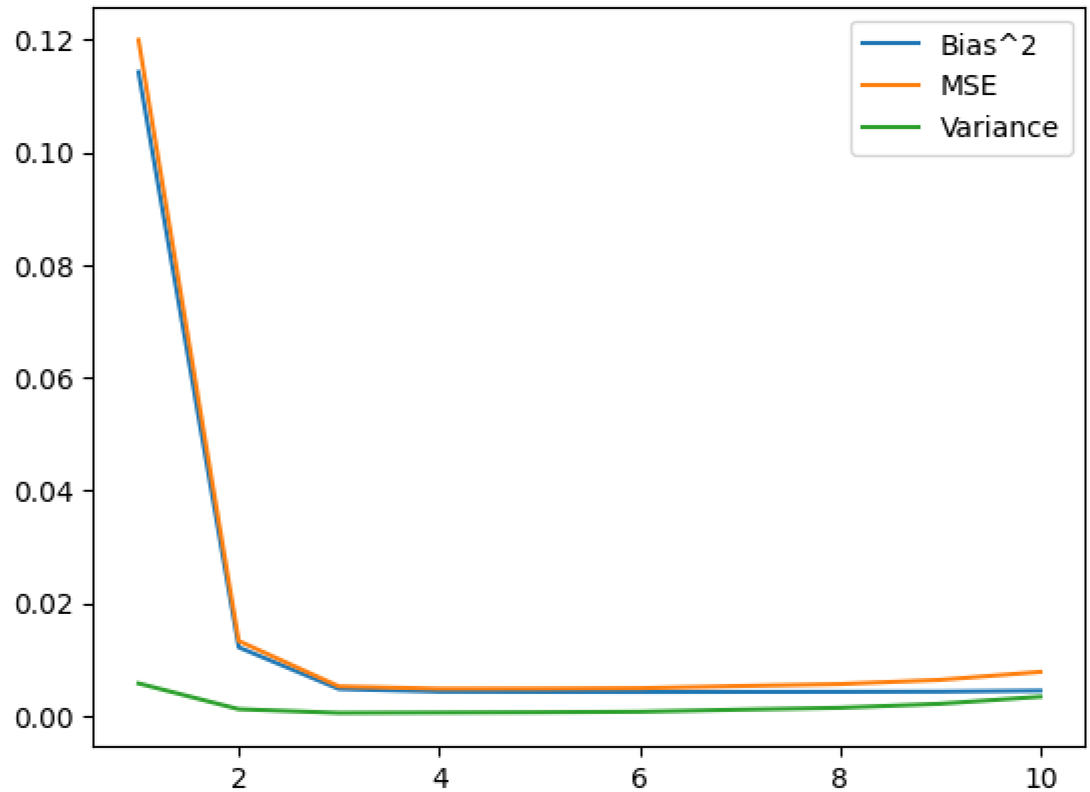
For degrees 1,2 the bias and variance both are high thus the model didn't fit well thus it is **'underfit'**.
But after that as the degree of polynomial increases
the **Bias value decreases**, causing the model to *over learn* which implies **overfitting**
thus the Variance value increases(model can't perform well on test data)

## Task 4:(2.4)

| | irreducibleErr |
|---|---|
| 1 | 2.949030e-17 |
| 2 | -4.987330e-18 |
| 3 | 1.192622e-18 |
| 4 | 2.927346e-18 |
| 5 | -4.987330e-18 |
| 6 | 2.168404e-19 |
| 7 | 8.673617e-19 |
| 8 | -1.084202e-18 |
| 9 | 4.336809e-19 |
| 10 | 4.336809e-18 |
| 11 | 8.673617e-18 |
| 12 | 3.469447e-18 |
| 13 | -3.469447e-18 |
| 14 | 0.000000e+00 |
| 15 | -1.110223e-16 |

The irreducible error is the '***Noise'*** that can't be eliminated by improving the model.
The values of the irreducible error are extremely small of the order of 1e-18.
*The Noise may be due to faults while collecting the data*.
As *irreducibleError* is due to noise, there is barely any change in its values.
The values of irreducible error don't explicitly depend on the model itself.

## Task 5:(2.5)




From the graph we observe that as the degree/complexity of the model increases the **bias decreases and the variance increases**.

We also observe in the above graph that variance decreases till degree 3(as the model was completely *underfit* before) then it starts increasing as the complexity increases.

We observe a point where bias and variance both are minimum possible at (degree=4, with minimum total error).

**Thus (degree=4) is the best fit model according to the given data with minimum total error(MSE).**

| | mse |
|---|---|
| **1** | 0.119942 |
| **2** | 0.013246 |
| **3** | 0.005177 |
| **4** | 0.004762 |
| **5** | 0.004783 |
| **6** | 0.004856 |
| **7** | 0.005244 |
| **8** | 0.005588 |
| **9** | 0.006330 |
| **10** | 0.007754 |
| **11** | 0.009163 |
| **12** | 0.031183 |
| **13** | 0.026501 |
| **14** | 0.047096 |
| **15** | 0.265264 |