# Evaluation Metrics for Regression Model:

## 1. Mean absolute Error (MAE):→

we know that error is given by actual value - predicted value

so, error = Actual val - Predicted val.

Now actual value is lets say 100. Then predicted values can be both larger than or smaller than 100. If predicted value is 90, then error is +10. But if the predicted value is 120, the error is -20. So error can be both positive or negative. So in this case if we calculate the mean of the error, then it can be both some number or +ve & -ve values can cancel each other to give mean = 0. Hence we calculate the absolute error and then calculate the mean. So, the mean absolute error is given by.

$$MAE = \frac{\sum_{i=1}^{n} |\text{actual values} - \text{Predicted values}|}{n}$$

we have a direct function in sklearn for the MAE.

$$MAE = \frac{\sum_{i=1}^{n} |y - \hat{y}|}{n}$$

we can also calculate the mean absolute error manually using numpy.
simply do np.abs(actual values - predicted values)

## Mean Squared Error (MSE):

Mean Squared error is similar to mean absolute error. Instead of taking the absolute error, we square the error. Rest is same.

$$MSE = \frac{\sum_{i=1}^{n} (\text{actual value} - \text{pred. val})^2}{n}$$

## Root Mean Squared Error (RMSE):

when we take the square root of mean squared error, then we get the Root Mean Squared Error.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\text{actual} - \text{predicted})^2}{n}}$$

Now, let us consider two different scenarios for RMSE

Case 1

| Actual | Predicted |
|---|---|
| 1 | 401 |

Case 2

| Actual | Predicted |
|---|---|
| 10000 | 10401 |

In both these scenarios, the error is common 401.

→ If we calculate the RMSE, it will be same for both the cases

But clearly case 1 is very bad as actual value is 1 but pred. is 401. Case 2 is slightly better as actual is 10000 but pred. is 10401. So, in the case 1 prediction is bad as compared to case 2, but error is showing both as same

Now to overcome this problem, we use the Root Mean Squared Log Error.

# Root Mean Squared Log Error (RMSLE):

Now, before calculating the error, we take the log of actual and predicted values in RMSLE. This brings all the values to a common scale, and then it is possible to accurately measure the errors performance of the model.

$$RMSLE = \sqrt{\frac{\sum_{i=1}^{n} (\log(\text{actual}) - \log(\text{predict}))^2}{n}}$$

But when we take the log of actual or predicted values, these values can be $=0$ as well. It is very important to note that $\log(0)$ is not defined. So this will throw error. So, we do $+1$ then take its log. There is already a function in numpy to do both the operations.

`np.log1p`, `np.log`,

To reverse the log, `np.exp`, `np.expm1`

## RELATIVE ERRORS :

When we want to compare the errors or the performance of two different models, then we calculate the relative errors. We compare every model that we build with the baseline model.

## Relative Squared Errors:

$$\frac{MSE(model)}{MSE(baseline)} = \frac{\sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}}{\sqrt{\frac{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}{n}}}$$

Here, lesser the Relative error value, better is the model performance.

## R-Squared:

We know that lesser the relative error, better the model performance. We now calculate the $R^2$ or R-square value, which is nothing but the difference between Relative error and 1.

$$R^2 = 1 - \frac{MSE(model)}{MSE(baseline)}$$

So greater the $R^2$ value, better the model performance.

$R^2$ can also be considered as an measure of accuracy. But there is a problem with $R^2$ as an accuracy measure. We know that the equation of linear regression is given by,

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots\ldots$$

Now, if we add a new variable or feature to the model, then the $R^2$ value either remains the same or it always increases. To overcome this problem, we use some measures and we adjust the $R^2$.

## Adjusted $R^2$:

$$\text{Adjusted } R^2 = \bar{R}^2 = 1 - (1 - R^2)\left[\frac{n-1}{n-(k+1)}\right]$$

Now the adjusted $R^2$ is far better and its gives an accurate measure of the performance of the models.

Each and every function is present in the sklearn library and we donot need to do anything manually.