DS080- Model evaluation 1 - Regression models

By: eng. Esraa Madhi

Why we have to Evaluate our model?

1. **Performance Validation:** Evaluation helps verify whether the model performs as expected on new, unseen data. It's crucial to ensure that the model accurately predicts or classifies data it wasn't trained on.
2. **Overfitting Detection:** Overfitting occurs when a model learns the details and noise in the training data to an extent that it negatively impacts the performance of the model on new data. Evaluation metrics can help detect this by comparing performance on training data versus unseen data.
3. **Model Comparison:** When developing machine learning models, it's common to create multiple variations. Evaluation metrics allow us to compare these models objectively to select the best one for deployment.
4. **Hyperparameter Tuning:** Evaluation metrics guide the tuning of model hyperparameters. This is often done through processes like grid search or

randomized search where different configurations are evaluated to find the most effective settings.

What is the difference between "parameters" and "hyperparameters" In machine learning?

The terms "parameters" and "hyperparameters" refer to different types of values used in the configuration of machine learning models.

- **Parameters**
  Parameters are the variables that the model learns during training. In supervised learning, these parameters are adjusted through the optimization process (such as gradient descent), aiming to minimize the loss function. **The values of the parameters are derived from the training data** and are not set manually by the modeler.
  - **Examples:**
    - **Coefficients in linear regression:** In regression models, coefficients for each feature are learned during model training, representing the impact of each feature on the prediction.

- **Hyperparameters**
  Hyperparameters, on the other hand, are the settings or configurations that govern the training process itself. **These are not learned from the data but are set prior to training and remain constant during the training process.** Hyperparameters often need to be tuned manually, as they can significantly affect the performance of the model. This tuning is usually done using strategies like grid search, random search, or Bayesian optimization.
  - **Examples:**
    - **Learning rate in gradient descent:** This is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated.

How could we evaluate our model?

To evaluate a machine learning model effectively, it depends on:

1. The specific type of model (e.g., regression, classification)
2. The objectives of the analysis.

Here's a guide on how to evaluate models. Assume you have collected the following data and you will build a linear regression model to predict the test score variable:

Hours Studied (x)	Test Score (y)
1	50
2	51
3	52
4	55
5	56

1. Splitting Data

Firstly, data is typically divided into at least two sets:

- **Training set**: Used to train the model.
- **Testing set**: Used to evaluate the model. This set is never shown to the model during training and helps simulate how the model would perform on new, unseen data.

For more robust evaluation, a third split, called the **validation set**, is often used especially when tuning model parameters.

Hours Studied (x)	Test Score (y)
1	50
2	51
3	52
4	55
5	56

The yellow rows are **Training set** and the green rows are **Testing set**

2. Evaluation Metrics

Evaluation metrics are quantitative measures used to assess the performance of a model or algorithm in statistics, machine learning, and data science. These metrics are crucial for determining the effectiveness of a model during training and testing phases.

After training the model, we will obtain predicted values for both the training and testing sets. We can then assess the model's performance using various evaluation metrics:

Hours Studied (x)	**Test Score (y)**	**Predicted Score** $(\hat{y})$
1	50	49.6
2	51	51.2
3	52	52.8
4	55	54.4
5	56	56.0

i. Mean Absolute Error (MAE):

The average of the **absolute differences** between the predicted values and the actual values. **It gives an idea of how wrong the predictions were.**

To calculate the Mean Absolute Error (MAE), we use the formula:
$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$

Where:

- (y_i) is the actual value,
- $(\hat{y}_i)$ is the predicted value,
- (n) is the number of observations.

Hours Studied (x)	Test Score (y)	Predicted Score $(\hat{y})$	**Absolute Error** $(y - \hat{y})$
1	50	49.6	$|50 - 49.6|$ = 0.4
2	51	51.2	$|51 - 51.2|$ = 0.2
3	52	52.8	$|52 - 52.8|$ = 0.8
4	55	54.4	$|55 - 54.4|$ = 0.6
5	56	56.0	$|56 - 56.0|$ = 0.0

Calculate the MAE for the training set:
$\text{MAE}_{\text{train}} = \frac{1}{3}(0.4 + 0.2 + 0.8) = \frac{1.4}{3} \approx 0.467$

Calculate the MAE for the test set:
$\text{MAE}_{\text{test}} = \frac{1}{2}(0.6 + 0.0) = \frac{0.6}{2} = 0.3$

Thus, the Mean Absolute Error (MAE) for the predictions is 0.3 **This indicates that, on average, the predictions deviate from the actual test scores by 0.3 points.**

Advantages of MAE:

- The MAE you get is in the same unit as the output variable.
- It is most Robust to outliers.

Disadvantages of MAE:

- MAE is not differentiable at points where the error $(y - \hat{y}) = 0$ (i.e., where the predicted value equals the actual value). This leads to a discontinuity in the gradient, which can cause issues in the gradient descent optimization, as the gradient at these points is undefined.

ii. Mean Squared Error (MSE): The **average of the squares** of the differences between the predicted values and the actual values.

What actually the MSE represents? It represents the squared distance between actual and predicted values. we perform squared to avoid the cancellation of negative terms and it is the benefit of MSE.

To calculate the Mean Squared Error (MSE), we use the formula:
$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

Where:

- (y_i) is the actual value,
- $(\hat{y}_i)$ is the predicted value,
- (n) is the number of observations.

Hours Studied (x)	Test Score (y)	Predicted Score $(\hat{y})$	**Square Error** $(y-\hat{y})^2$
1	50	49.6	((50 - 49.6)^2 = 0.16)
2	51	51.2	((51 - 51.2)^2 = 0.04)
3	52	52.8	((52 - 52.8)^2 = 0.64)
4	55	54.4	((55 - 54.4)^2 = 0.36)
5	56	56.0	((56 - 56.0)^2 = 0.00)

Calculate the MSE for the training set:

$$\text{MSE}_{\text{train}} = \frac{1}{3}(0.16 + 0.04 + 0.64) = \frac{0.84}{3} \approx 0.28$$

Calculate the MSE for the test set:

$$\text{MSE}_{\text{test}} = \frac{1}{2}(0.36 + 0.00) = \frac{0.36}{2} = 0.18$$

Advantages of MSE:

- MSE is a smooth and differentiable function everywhere. This property is crucial because gradient descent requires the computation of derivatives to update the model parameters. The gradient (derivatives) of MSE with respect to the parameters are linear functions of the prediction error, leading to smooth and stable gradient updates.

Disadvantages of MSE

- The value you get after calculating MSE is a squared unit of output. for example, the output variable is in meter(m) then after calculating MSE the output we get is in meter squared.
- If you have outliers in the dataset then **it penalizes** the outliers most and the calculated MSE is bigger. So, in short, It **is not Robust to outliers** which were an advantage in MAE.

iii. Root Mean Squared Error (RMSE):

The **square root** of MSE. It is particularly useful because it gives error terms in the same units as the response variable.

To calculate the Root Mean Squared Error (RMSE), we first compute the MSE:
$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

Then, RMSE is calculated as:
$\text{RMSE} = \sqrt{\text{MSE}}$

Where:

- (y_i) is the actual value,
- $(\hat{y}_i)$ is the predicted value,
- (n) is the number of observations.

Hours Studied (x)	Test Score (y)	Predicted Score $(\hat{y})$	**Square Error** $(y - \hat{y})^2$
1	50	49.6	((50 - 49.6)^2 = 0.16)
2	51	51.2	((51 - 51.2)^2 = 0.04)
3	52	52.8	((52 - 52.8)^2 = 0.64)
4	55	54.4	((55 - 54.4)^2 = 0.36)
5	56	56.0	((56 - 56.0)^2 = 0.00)

Calculate the MSE and then the RMSE for the training set:
$\text{MSEtrain} = \frac{1}{3}(0.16 + 0.04 + 0.64) = 0.28$
$\text{RMSEtrain} = \sqrt{0.28} \approx 0.529$

Calculate the MSE and then the RMSE for the test set:
$\text{MSEtest} = \frac{1}{2}(0.36 + 0.00) = 0.18$
$\text{RMSEtest} = \sqrt{0.18} \approx 0.424$

Advantages of RMSE

- The output value you get is in the same unit as the required output variable which makes interpretation of loss easy.

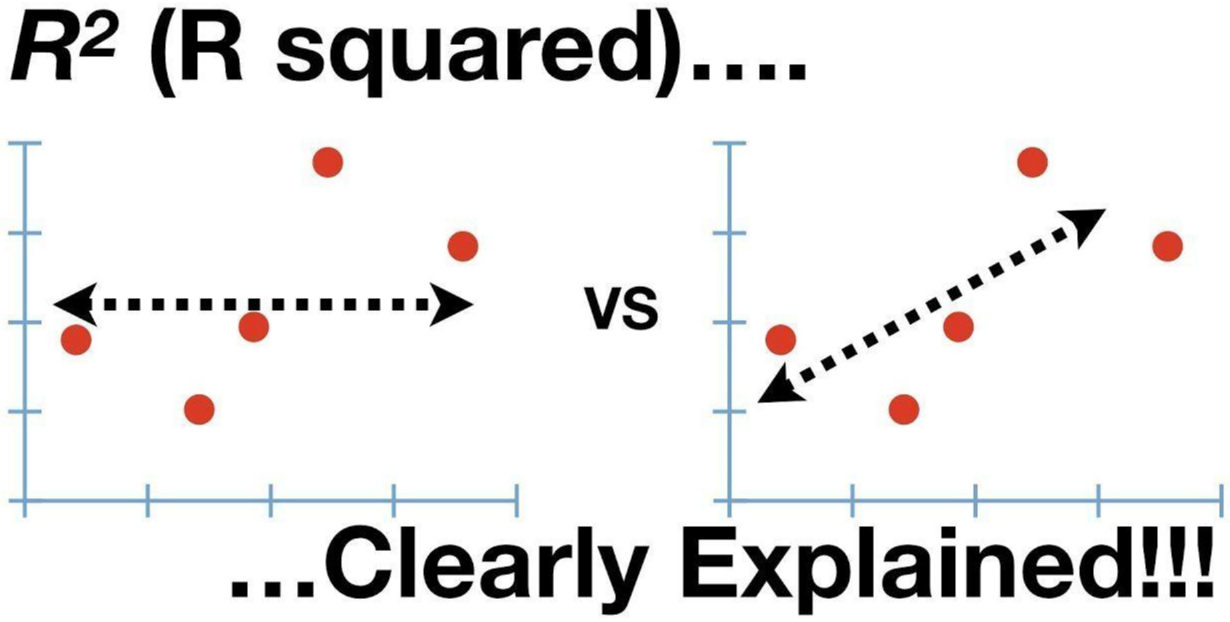
Disadvantages of RMSE

- It is not that robust to outliers as compared to MAE

iv. R-squared (Coefficient of Determination):

R2 score is a metric that tells the performance of your model, not the loss in an absolute sense that how many wells did your model perform.

So, with help of R squared **we have a baseline model to compare a model** which none of the other metrics provides. So basically R2 squared calculates how must regression line is better than a mean line.



R2 score is between zero and one, So we can conclude that as our regression line moves towards perfection, R2 score move towards one. And the model performance improves.

Thus, if R-squared of approximately 0.956 **suggests that about 95.6% of the variation in the test scores can be explained by the hours studied**, according to the model. This indicates a high level of model effectiveness in this context.

So we can define it as a Measurement of the proportion of the variance in the dependent variable that is predictable from the independent variables. It provides an indication of goodness of fit.

To calculate R-squared, we use the formula:

$$R^2 = 1 - \frac{\text{Sum of Squares of Residuals (SSR)}}{\text{Total Sum of Squares (SST)}}$$

Where:

- SSR (Sum of Squares of Residuals): $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- SST (Total Sum of Squares): $\sum_{i=1}^{n}(y_i - \bar{y})^2$
- $(y_i)$is the actual value,
- $(\hat{y}_i)$ is the predicted value,
- $(\bar{y})$is the mean of actual values,
- $(n)$is the number of observations.

Given the data:

Hours Studied (x)	Test Score (y)	Predicted Score($\hat{y}$)	Squared Residual $((y_i - \hat{y}_i)^2)$	Squared Deviation $((y_i - \bar{y})^2)$
1	50	49.6	0.16	1
2	51	51.2	0.04	0
3	52	52.8	0.64	1
4	55	54.4	0.36	0.25
5	56	56.0	0.0	0.25

To calculate the R-squared for training set:

- Mean of training set $(\bar{y}_{train} = \frac{50+51+52}{3} = 51)$
- SSR (Training): (0.16 + 0.04 + 0.64 = 0.84)
- SST (Training): (1 + 0 + 1 = 2)
- Now, R-squared for the training set: $R^2_{train} = 1 - \frac{SSR_{train}}{SST_{train}} = 1 - \frac{0.84}{2} \approx 0.42$

To calculate the R-squared for testing set:

- Mean of actual values $\bar{y}_{test} = \frac{55+56}{2} = 55.5)$
- SSR (Test): (0.36 + 0.0 = 0.36)
- SST (Test): (0.25 + 0.25 = 0.5)
- Now, R-squared for the test set: $R^2_{test} = 1 - \frac{SSR_{test}}{SST_{test}} = 1 - \frac{0.36}{0.5} \approx 0.72$

Advantages of R-squared:

- **Scale-independent**: It doesn't inherently consider the scale of the dataset, which makes it useful for comparing models across different scales of data.
- **Non-negative**: R-squared ranges from 0 to 1, where 0 means no correlation and 1 indicates perfect correlation, making it intuitive in terms of explaining the effectiveness of the model's

In certain situations, it may be necessary to utilize a variety of evaluation metrics. If you are new to this field, experimenting with each of these metrics can be very beneficial. This approach will enhance your understanding of each metric and help you determine the most appropriate ones to use in different scenarios.

3. Cross-Validation

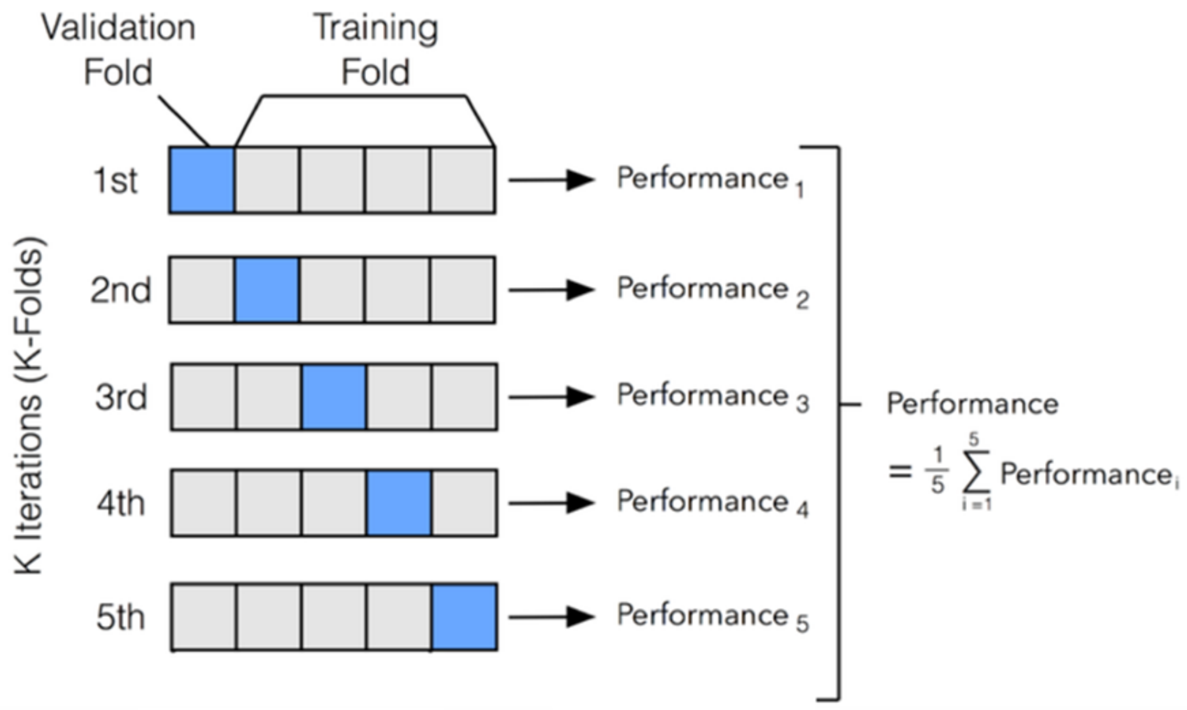
a. **For Effective Use of Data:**

Particularly in situations where the amount of data is limited, using k-fold cross-validation can be a more efficient use of data compared to the traditional train/test split. Every data point is used for both training and validation, and thus, the model training is likely to be more comprehensive.

- In a simple words: (when you want to train the model in the whole data but in the same time you want to get an accuracy for the model on un seen data → calculate the average accuracy from multiple iterations of the model trained on different subsets. Use this average as evidence of the model's effectiveness. After validation, train the model on the full dataset.)

https://youtu.be/hoNpvry0370

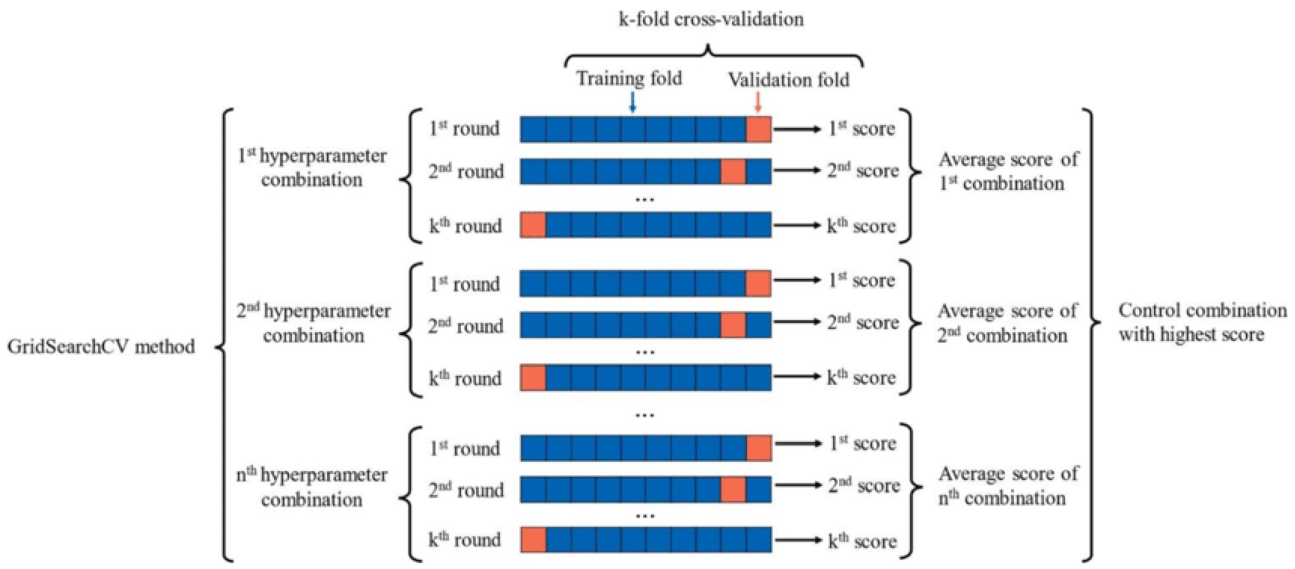
- **How the final performance is calculated ?**



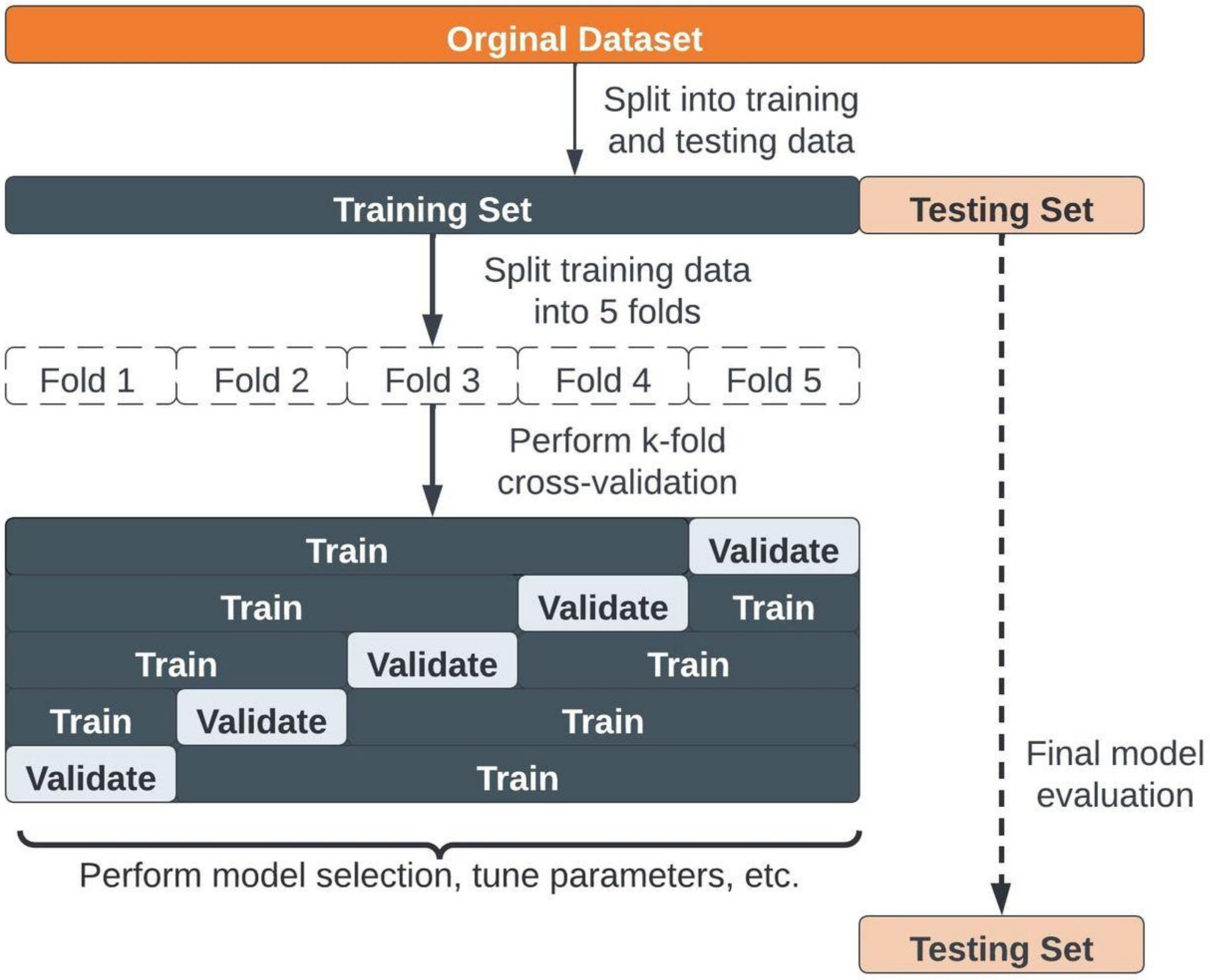
b. **Tuning Hyperparameters**:

It is extensively used for comparing the performances of different hyperparameter settings within a learning algorithm. By observing how the model performance varies across different folds, we can select the hyperparameters that yield the best generalized performance.

The data set is divided into k smaller sets (or folds). The model is trained on k-1 of these folds, with the remaining part used as a test set. This process is repeated k times, with each of the k folds used exactly once as the test set.


k-fold cross-validation
Training fold
Validation fold
GridSearchCV method
1st hyperparameter combination
2nd hyperparameter combination
nth hyperparameter combination
1st round
2nd round
kth round
1st score
2nd score
kth score
Average score of 1st combination
Average score of 2nd combination
Average score of nth combination
Control combination with highest score

https://youtu.be/hEtWQcIL3NI



c. **Reducing Overfitting:**
Using cross-validation can safeguard against overfitting more effectively than using a simple train/test split. This is because it ensures that the model's ability to generalize is not just due to a lucky split of data, but rather a consistent pattern across multiple splits.

Resources:

- https://towardsdatascience.com/ways-to-evaluate-regression-models-77a3ff45ba70

- https://towardsdatascience.com/what-are-the-best-metrics-to-evaluate-your-regression-model-418ca481755b
- https://www.geeksforgeeks.org/regression-metrics/
- https://www.analyticsvidhya.com/blog/2021/10/evaluation-metric-for-regression-models/
- https://towardsdatascience.com/5-reasons-why-you-should-use-cross-validation-in-your-data-science-project-8163311a1e79
- https://towardsdatascience.com/the-importance-of-cross-validation-in-machine-learning-35b728bbce33