Common Regression Metrics

---

1. MAE (Mean Absolute Error)

- **What it means**: "How wrong is my model on **average**?"
- **Calculation**: Average of the absolute differences between predicted and actual values.
- **Example**: If your model predicts house prices:
  - Actual prices: [$200k, $300k, $250k]
  - Predicted prices: [$210k, $290k, $240k]
  - Errors: |200-210| = $10k, |300-290| = $10k, |250-240| = $10k
  - **MAE = (10 + 10 + 10)/3 = $10k**
- **Why use it**: Easy to understand (**same units as your data**). **Doesn't punish large errors harshly**.

How to Interpret:

- Lower MAE means better performance.
- It tells you, on average, how many units off your predictions are.

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

Where:

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

Disadvantage:

- The graph is not differentiable at 0.

---

2. MSE (Mean Squared Error)

- **What it means**: "How big are the mistakes, with extra focus on large errors?"
- **Calculation**: Average of the **squared differences** between predicted and actual values.
- **Same example**:
  - Errors²: $(10)^2 = 100$, $(10)^2 = 100$, $(10)^2 = 100$
  - **MSE = (100 + 100 + 100)/3 = 100**
- **Why use it**: **Punishes large errors more** (e.g., a $40k error contributes 1600 to MSE vs. $40k in MAE).
- **Downside**: Units are squared (e.g., "$²"), which is hard to interpret.

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

3. RMSE (Root Mean Squared Error)

- **What it means**: "Same as MSE, but in understandable units."
- **Calculation**: Just take the square root of MSE.
- **Same example**:
  - **RMSE = √100 = $10k**
- **Why use it**: Fixes MSE’s unit problem. Still punishes large errors more than MAE.

[*USED IN DEEP LEARNING*]()

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Example:

- If MSE is 9.67, then RMSE ≈ √9.67 ≈ 3.11 units.

4. R^2 Score (R-Squared)/Coeff of Determination/ Goodness of Fit

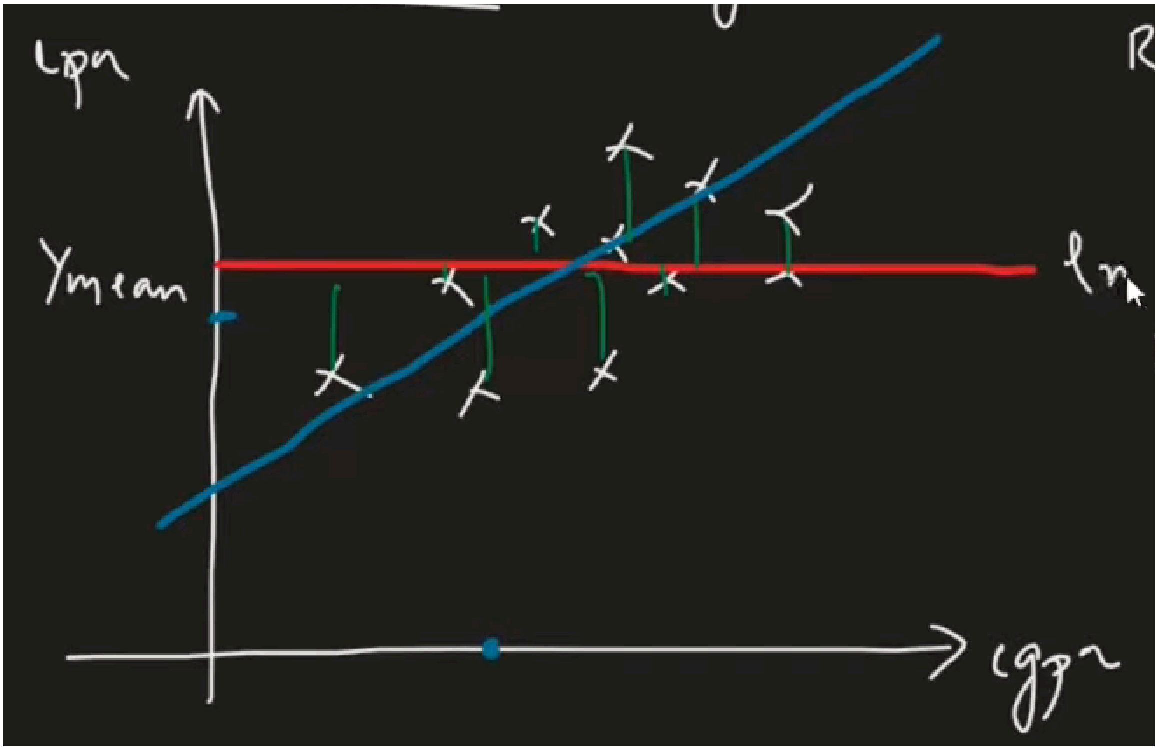
- **What it means**: "**How much better is my model than just guessing the average?**"
- **Range**: **0 to 1** (1 = perfect model, 0 = no better than the mean).
- **Example**:
  - If your model's **R^2 = 0.8**, it explains 80% of the variation in the data.
    - **We cannot explain what is the reason for remaining 20% variance.**
- **Why use it**: Great for comparing models. **A higher R^2 is better!**

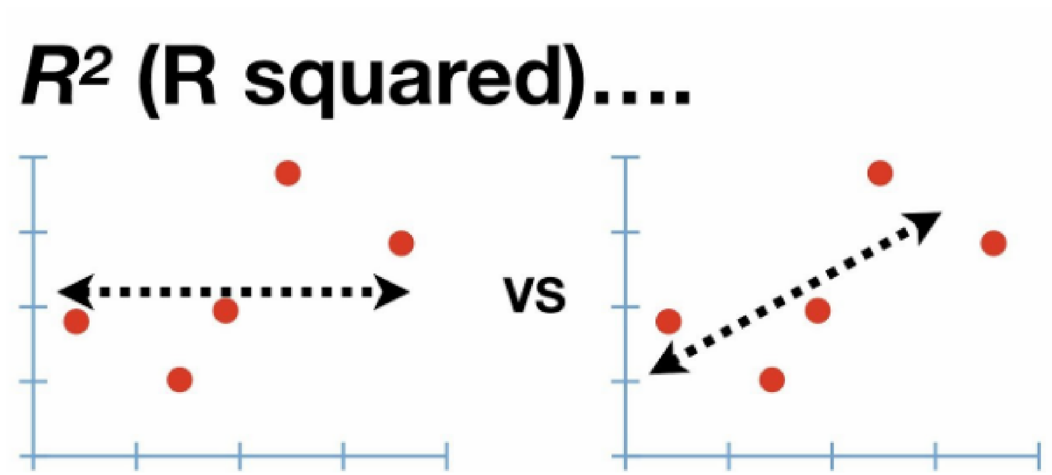
Formula:

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

Where:

- $\bar{y}$ is the **mean** of the actual values.





Interpretation:

- **R^2 = 1** indicates that the model explains 100% of the variance, meaning perfect fit.
- **R^2 = 0** means that the model explains none of the variance, and the predictions are no better than just predicting the mean value for all instances.
- **Negative R^2** can occur if the model is worse than just predicting the mean.

5. Adjusted R^2 Score

- **What it means**: "R^2, but it punishes useless predictors."
- **Why it exists**: If you add random variables to your model, R^2 will *always* increase (even if they're useless). Adjusted R^2 fixes this.

How to Interpret:

- **It penalizes adding predictors that do not improve the model.**
- Useful in multiple regression where adding too many features can artificially inflate R^2.

- **Example**:
  - When comparing two models, if the model with more variables has a similar R^2 but a lower Adjusted R^2, it might be overfitting.
  - Original model (3 predictors): **R^2 = 0.75**
  - New model (5 predictors, 2 are useless): Adjusted R^2 might drop to 0.72.

Formula:

$$R^2_{\text{adj}} = 1 - \left(\frac{(1 - R^2)(n - 1)}{n - p - 1}\right)$$

Where:

- n is the number of data points (samples).
- p is the number of predictors (features) in the model.

- **Use it**: When comparing models with different numbers of predictors.

Interpretation:

- **Adjusted R^2** can be **lower than R^2** if adding new predictors reduces the model's explanatory power.
- **Higher values** of adjusted R^2 indicate a better fit, considering both the goodness of fit and the number of predictors.
- **Negative values** are possible, especially if the model is poor.

When to Use Which?

- Use **MAE/RMSE** to understand prediction errors in real-world units.
- Use **R^2/Adjusted R^2** to compare model performance.
- **RMSE > MAE** if large errors are critical (e.g., in healthcare).

Summary of Common Regression Metrics

Metric	What It Measures	Key Point
MAE	Average absolute error between predictions and actual values.	Average error in same units as data.
MSE	Average of squared differences (errors) between predictions and actual values.	Penalizes large errors more.
RMSE	Square root of MSE; error in same units as data.	Easier to interpret, same units as data.
R^2 Score	Proportion of variance in the dependent variable explained by the model.	Ranges from 0 to 1; higher is better.
Adjusted R^2 Score	R^2 adjusted for the number of predictors in the model.	Penalizes unnecessary variables.

Python Code

We Need 2 things:

1. **Y-Actual**
2. **Y-Predicted**

```
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
```

```
y_pred= lr.predict(x_test)
```

- This gives **predicted values of y**.
- **Actual values of y**→ `y_test`

MAE (Mean Absolute Error)

```
MAE = mean_absolute_error(y_test, y_pred)
MAE

Output: 0.2884710931878175 #LPA
```

MSE (Mean Squared Error)

```
mean_squared_error(y_test, y_pred)

Output: 0.12129235313495527 #LPA^2
```

RMSE (Root Mean Squared Error)

```
RMSE = np.sqrt(mean_squared_error(y_test, y_pred))
RMSE

Output: 0.34827051717731616 #LPA
```

R^2 Score (R-Squared)

```
r2= r2_score(y_test, y_pred)
r2

Output: 0.780730147510384
```

Adjusted R^2 Score

- First, we need n

```
x_test.shape

Output: (40, 1)
```

- ***n=40***

Put values in the formula:

$$R^2_{\text{adj}} = 1 - \left(\frac{(1 - R^2)(n - 1)}{n - p - 1} \right)$$

Where:

- n is the number of data points (samples).
- p is the number of predictors (features) in the model.

```
1 - ((1-r2)*(40-1)/(40-1-1))

Output: 0.7749598882343415
```

Let’s add a random column in the df and check R2 and Adj R2

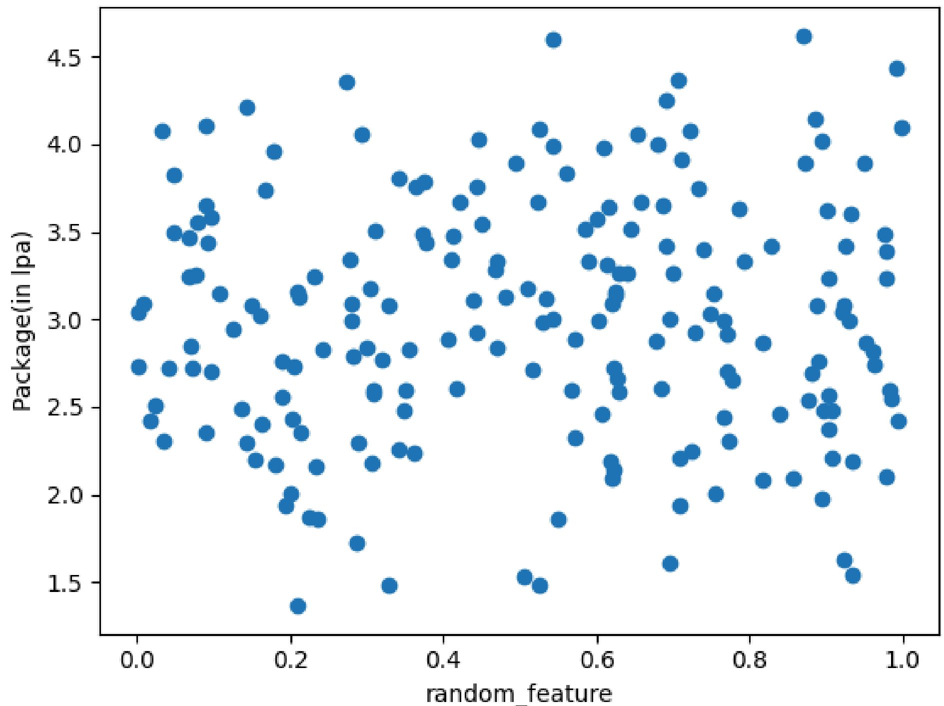
```
new_df1 = df.copy()
new_df1['random_feature'] = np.random.random(200)

new_df1.head()
```

	cgpa	package	random_feature
0	6.89	3.26	0.699989
1	5.12	1.98	0.894708
2	7.82	3.25	0.077334
3	7.42	3.67	0.523681
4	6.94	3.57	0.600437

random_feature vs package

```
plt.scatter(new_df1['random_feature'],new_df1['package'])
plt.xlabel('random_feature')
plt.ylabel('Package(in lpa)')
```




```
x1_train, x1_test, y1_train, y1_test = train_test_split(new_df1[['cgpa','random_fea
ture']],new_df1['package'], test_size=0.2, random_state=2)

lr2= LinearRegression()

lr2.fit(x1_train,y1_train)

y1_pred= lr2.predict(x1_test)
```

Now calculate r2 and adj r2

```
print("R2 score",r2_score(y_test,y_pred))
r2 = r2_score(y_test,y_pred)

Output: R2 score 0.780730147510384
```

```
r2adj= 1 - ((1-r2)*(40-1)/(40-1-2))

print("R2 adjusted",r2adj)

Output: R2 adjusted 0.7688777230514858
```

- **R2 adjusted has decreased as the column `random_feature` does not have any impact on the package.**