**LR - MODEL EVALUATION(WITHOUT FIXED RANDOM STATE)**

MSE: 2137968371.3047886   
MAE:18963.498963019814   
R2 Score:0.6584559987753926

Awesome — you've reached the **model evaluation stage**! Let's go through your results and explain what each metric means 👇

**✅ Your Model Evaluation Results:**

| **Metric** | **Value** | **What It Means** |
| --- | --- | --- |
| **MSE (Mean Squared Error)** | 2,137,968,371 | Average of squared prediction errors (big errors penalized more heavily) |
| **MAE (Mean Absolute Error)** | 18,963.49 | Average of absolute prediction errors — on average, your model is off by ~₹19K |
| **R² Score (Coefficient of Determination)** | 0.658 | Your model explains **65.8%** of the variance in house prices |

**🔍 Let's Interpret It Clearly:**

**🔹 1. MAE – ₹18,963 error**

* On average, your model is off by **~₹19,000**
* For a dataset where house prices might range from ₹50,000 to ₹500,000+, this is **decent**

**🔹 2. MSE – Very High**

* MSE is in the **billions**, which looks scary — but that’s normal for squared errors when dealing with large numbers (like sale prices).
* It’s mostly useful for comparison **between different models** — not for human interpretation.

**🔹 3. R² Score – 0.658**

* This means your model explains **~65.8% of the variance** in sale prices.
* That’s a **solid baseline**, especially for a first attempt with simple preprocessing and Linear Regression.
* **1.0 is perfect**, **0.0 is useless**, and **negative means worse than a dummy model**

**📈 So… Is this good?**

**Yes — it’s a solid start!**  
But there's room to improve, especially if your goal is a higher R².

**🔧 What You Can Try Next to Improve:**

| **Idea** | **How It Helps** |
| --- | --- |
| **Feature Scaling** (StandardScaler) | Helps models like Ridge, Lasso |
| **Remove Outliers** | Reduces MSE and improves fit |
| **Try Other Models** | Use RandomForestRegressor, XGBoost, etc. |
| **Cross-validation** | More reliable performance check |
| **Polynomial Features** | Capture non-linear patterns |

**🎯 TL;DR**

✅ You're off to a great start  
📊 Your model has reasonable error and a decent R²  
🚀 Try ensemble models next for improvement

Want help trying RandomForest or tuning features?

[P.T.O]

**LR - MODEL EVALUATION(FIXED RANDOM STATE)**

MSE: 876261511.2537556MAE:18612.204114825836R2 Score:0.8857595508216202

Whoa! 🚀 That’s a **huge improvement** just by changing the train\_test\_split!

Let’s compare:

| **Metric** | **Without Random State** |  |  | **With Random State (assumed)** |
| --- | --- | --- | --- | --- |
| **MSE** | 2,137,968,371 |  |  | **876,261,511** ✅ (↓ better) |
| **MAE** | 18,963.49 |  |  | **18,612.20** ✅ (↓ better) |
| **R² Score** | 0.658 |  |  | **0.885** ✅ (↑ better) |

**🔍 Why did this happen?**

When you don’t set random\_state, train\_test\_split randomly shuffles your data **differently each time**, so:

* You may accidentally send **outliers or rare patterns** to the test set.
* Or maybe you got a "luckier" train-test split that better represents the real data.

**🧠 So should you always use random\_state?**

Yes — **for reproducibility**.

Use:

xtrain, xtest, ytrain, ytest = train\_test\_split(X, Y, test\_size=0.2, random\_state=42)

This ensures:

* You get the **same split every time**
* Results are **consistent**, and
* You can **compare models fairly**

**✅ TL;DR:**

* 📈 Your new R² is **very good (0.88+)**
* 🎯 Keep using random\_state=42 (or any fixed number)
* 💡 Now you have a strong baseline model to compare against future improvements!

Let me know if you’d like to try RandomForestRegressor or visualize which features are most important!