Prediction of Diffusion Coefficient using Machine Learning with Physical Formula Integration

[github]

Model Selection

1. Decision Tree Regressor
2. Random Forest Regressor

The models are trained and evaluated on three different target variables:

1. D
2. D*ρ
3. D*ρ*/√T

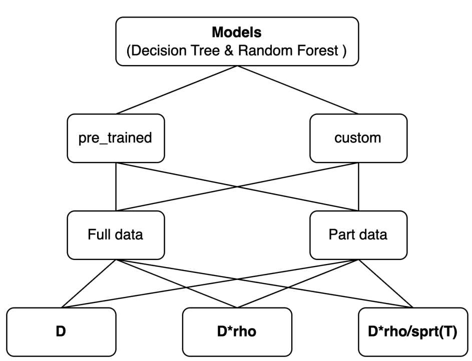
Data Splitting

- Training Set: $\rho^* \geq 0.1$
- Testing Set: $\rho^* < 0.1$

Performance Metrics

1. Mean Squared Error (**MSE**): Measures the average squared difference between actual and predicted values.
2. Mean Absolute Error (**MAE**): Measures the average absolute difference between actual and predicted values.
3. Relative Error **(%):** Measures the percentage deviation from actual values.

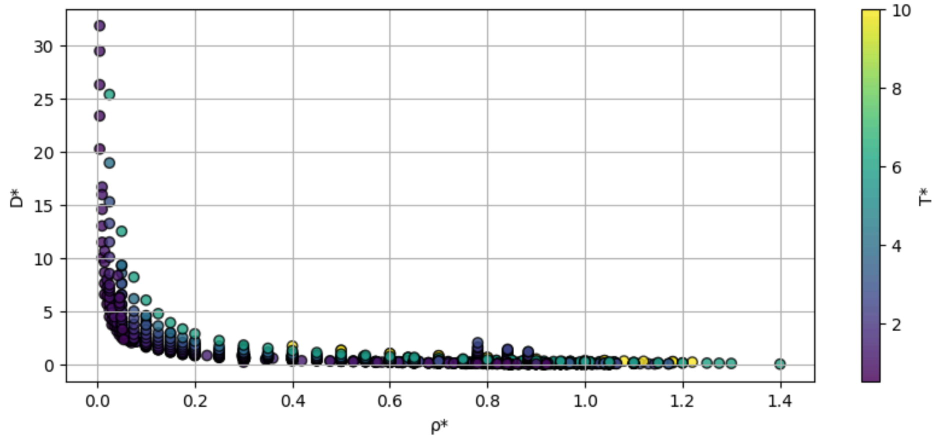
Experimental Design



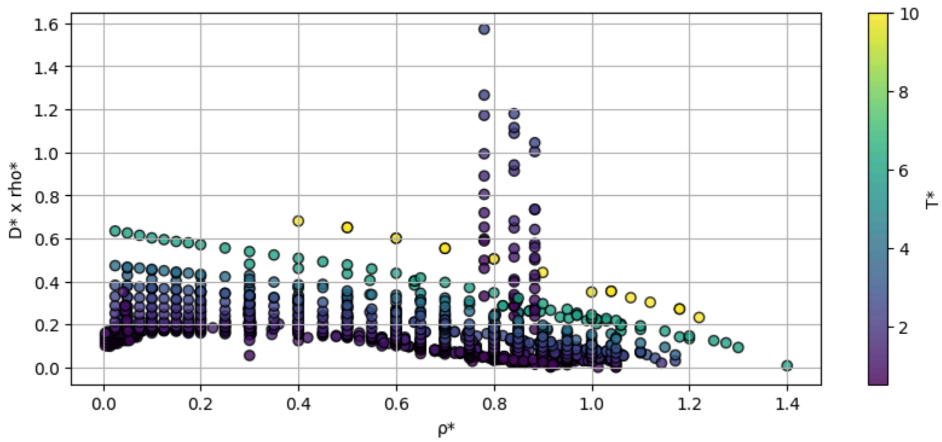
Dataset

1. Origin dataset

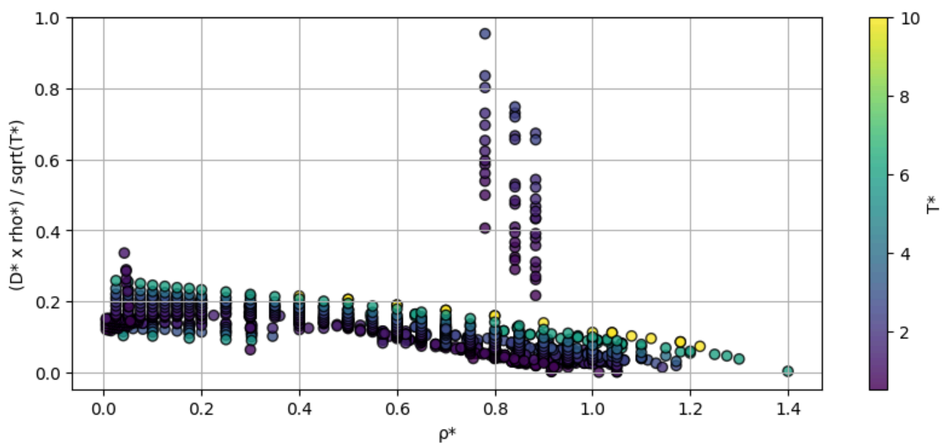
- D and ρ*



- D*ρ and ρ*



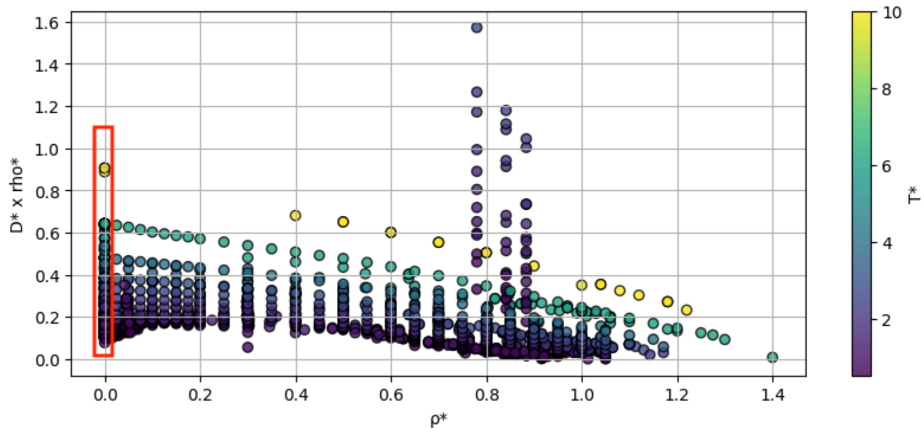
- D*ρ*/√T and ρ*



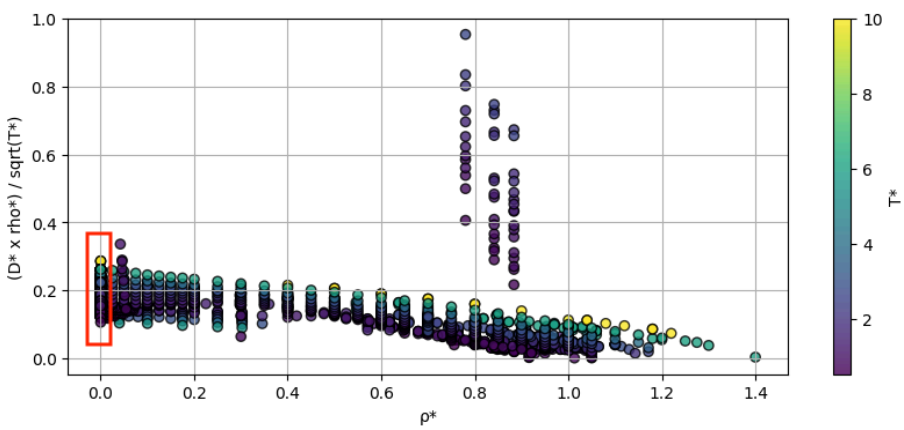
2. Augmented dataset

- New dataset added to train dataset (ρ* ≥ 0.1) with ρ* is zero to help model enhance pattern where ρ* is zero
- Original has **1251** samples with **566** unique values of T.
- New data created from **566** unique values of T with ρ* is zero → dataset with **566** samples.
- Merge original dataset and new data → augmented dataset with **1817** samples.

- D*ρ and ρ*



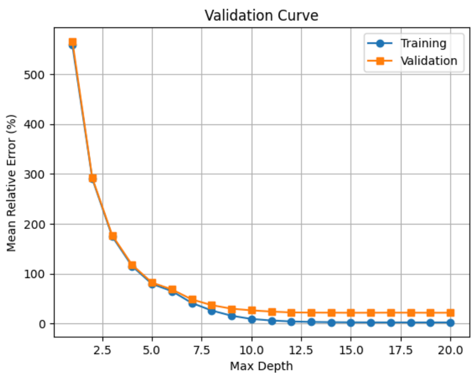
- D*ρ*/√T and ρ*



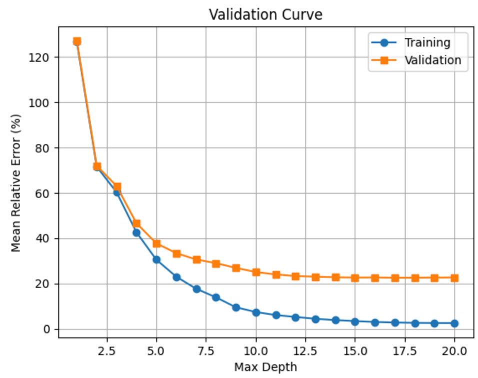
Balance trade-off

Purpose: Finding a best max depth to build custom decision model
Steps: Finding max depth all targets on 2 datasets (original and augmented)
Method: Validation curve of scikit learn library

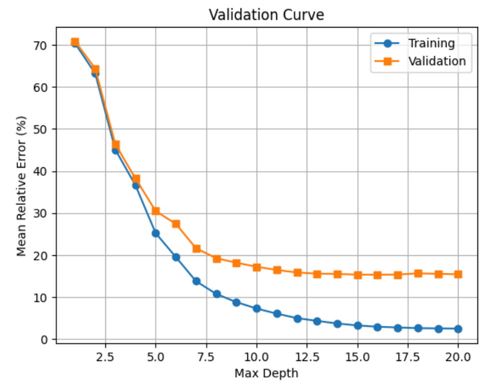
Original



D

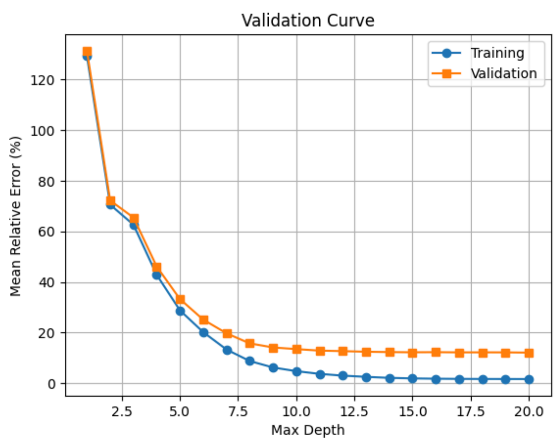


*D*ρ*

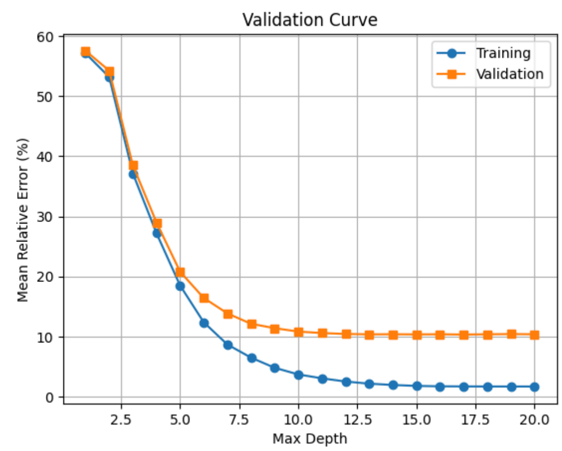


*D*ρ*/√T*

Augmented



*D*ρ*



*D*ρ*/√T*

Max depth range	Training relative error	Validation relative error	Model behavior	Interpretation
1-4	high	high	underfitting	model is too simple to capture data pattern
5-10	sharply decreasing	sharply decreasing	well-fitting, balanced	model captures patterns well
11-20	very low	slightly decreasing	overfitting	Model memorizes training data and starts to lose generalization

Best max depth:

Optimal max_depth: **9 or 10**

A good trade-off between bias and variance

→ Building a Decision with max depth is **10** and min samples split is **2**

Results and Discussion

1. Decision Tree

Alias :

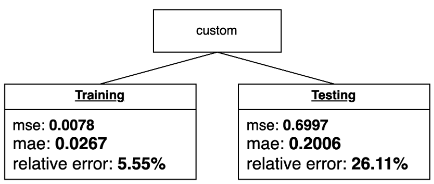
- max_depth = md
- min_samples_split = msp
- mean relative error = mre

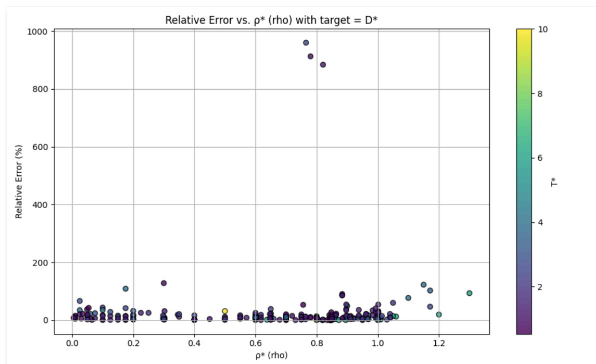
Default params:

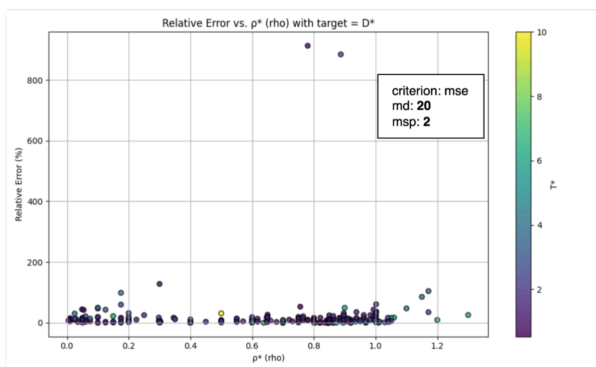
- criterion: mre
- max depth: **10**
- min samples split: **2**

Full data

- ***D***





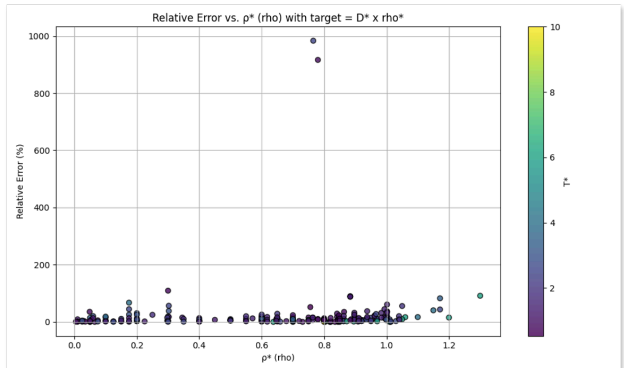


I realize that a custom model is worse than a pre-trained model in the training phase but is better in the testing phase. It proves that a pre-trained model is overfitting than a custom model. Show through a deeper tree (max-depth = 20).
→ Prefer to use a custom model .

- ***D*ρ****

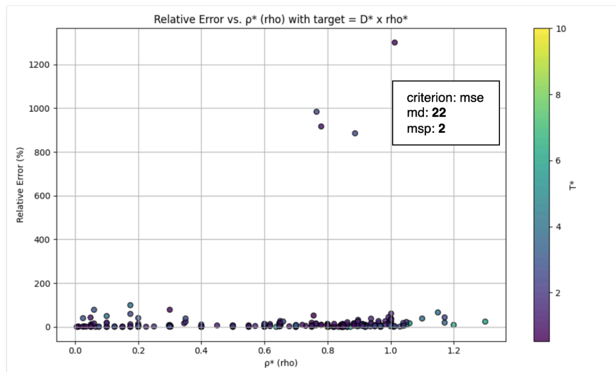
custom

Training	Testing
mse: **0.0003**	mse: **0.0113**
mae: **0.0087**	mae: **0.0269**
relative error: **5.80%**	relative error: **20.28%**



pre_trained

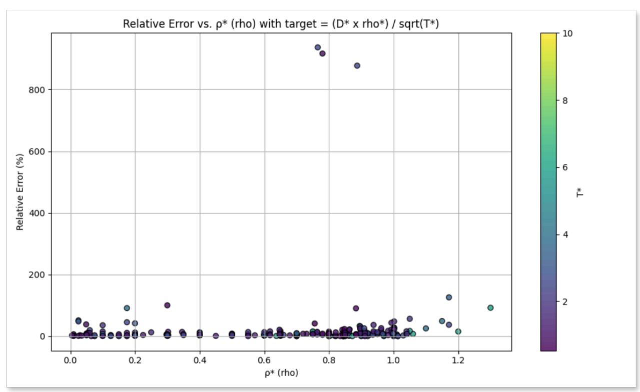
Training	Testing
mse: **0.0001**	mse: **0.0135**
mae: **0.0025**	mae: **0.0256**
relative error: **2.64%**	relative error: **31.78%**



- ***D*ρ*/√T and ρ****

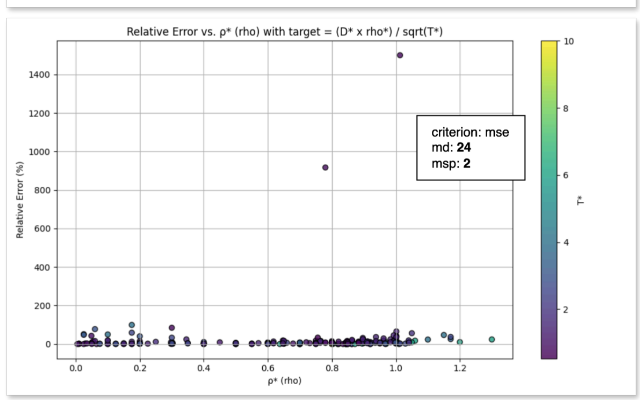
custom

Training	Testing
mse: **0.0001**	mse: **0.0059**
mae: **0.0047**	mae: **0.0184**
relative error: **5.18%**	relative error: **21.79%**



pre_trained

Training	Testing
mse: **0**	mse: **0.0014**
mae: **0.0018**	mae: **0.0116**
relative error: **2.64%**	relative error: **23.91%**



When change target from ***D** to **D*ρ**** or ***D*ρ*/√T and ρ**** *then have better result (about **5 - 6 %**)*

Part data

- ***D***

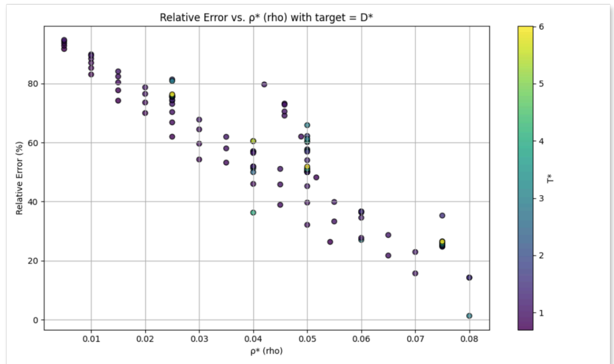
custom

Training

mse: **0.0107**
mae: **0.0287**
relative error: **9.51%**

Testing

mse: **53.5575**
mae: **4.7282**
relative error: **56.25%**



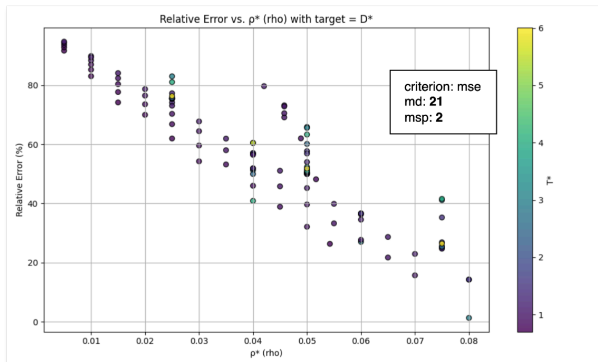
pre_trained

Training

mse: **0.0052**
mae: **0.0108**
relative error: **3.11%**

Testing

mse: **53.8770**
mae: **4.7500**
relative error: **56.52%**



We can see that when we only take part data to train and test a different part. Relative error is significant higher than full data from **26%** to **56%**

- ***D*ρ****

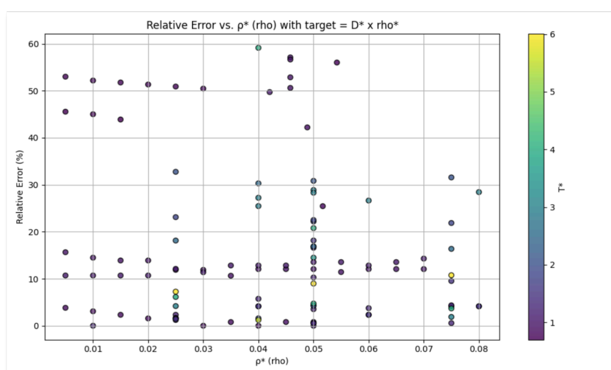
custom

Training

mse: **0.0038**
mae: **0.0171**
relative error: **9.89%**

Testing

mse: **0.0027**
mae: **0.0355**
relative error: **16.87%**



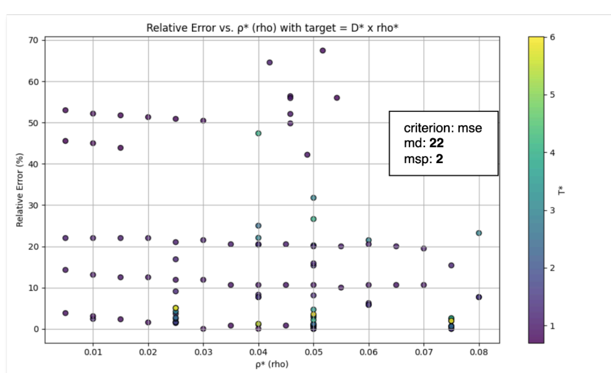
pre_trained

Training

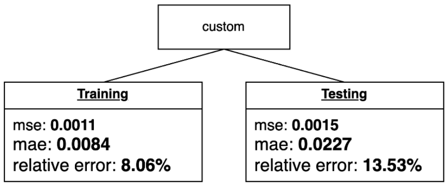
mse: **0.0001**
mae: **0.0030**
relative error: **3.11%**

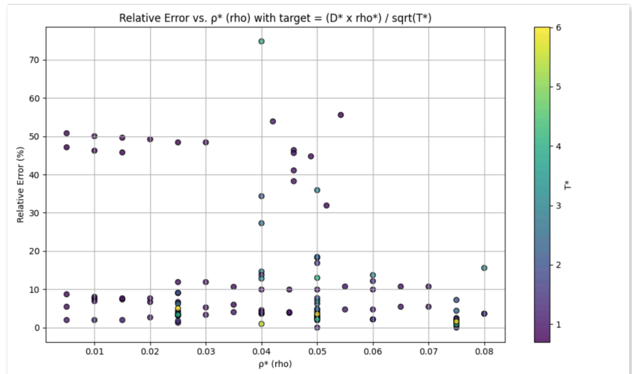
Testing

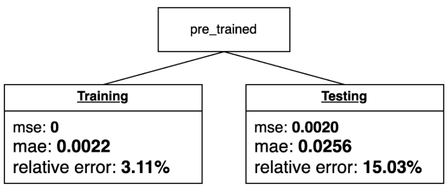
mse: **0.0024**
mae: **0.0302**
relative error: **15.86%**

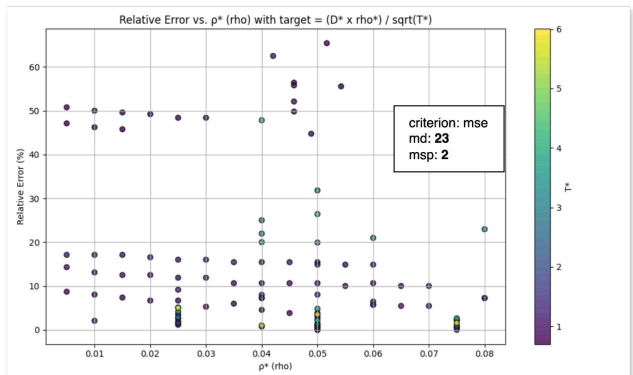


- ***D*ρ*/√T and ρ****









Amazing! we observe that when we change target from ***D** to **D*ρ*** or ***D*ρ*/√T and ρ**** , the result improve dramatically better than full data, reducing relative error from **56%** to **13%** (more **40%**). Custom model still prefers to be selected to pre_trained model.

Part data with augmented data (at rho = 0)

New dataset is calculated by formula (rho = 0): here's formula to calculate ***D*ρ**** *at rho = 0*

$$(D\rho)_0(T) = \frac{3}{8}\sqrt{\frac{kT}{m\pi}}\frac{f_{\mathrm{D}\rho}}{\sigma^2\Omega^{(1,1)*}}$$

Note: Using custom model

- ***D*ρ****

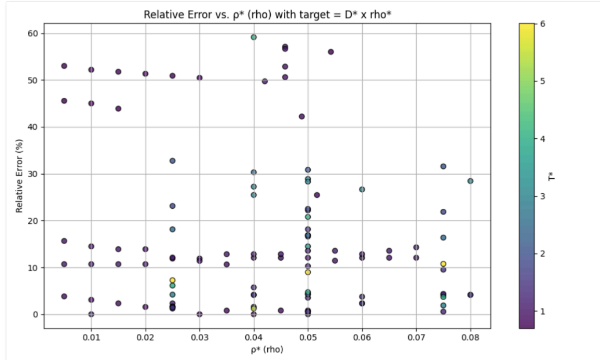
I observe that when adding a new dataset to train model then result improve better (about **3%**). But both adding a new dataset and adding weight (8) in training phase then result even better.

without_augmentation	→	with_augmentation	→	with_augmentation_weight
16.87 %		**13.82 %**		**9.54 %**

without augmentation

Training
mse: **0.0038**
mae: **0.0171**
relative error: **9.89%**

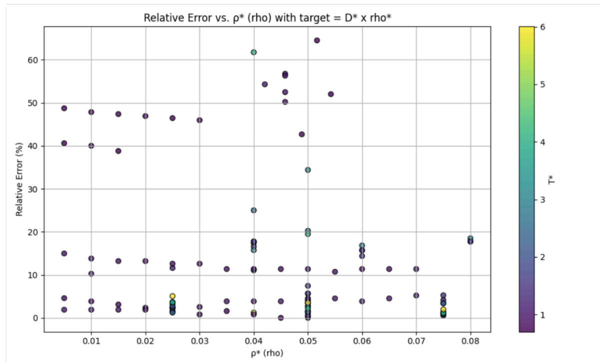
Testing
mse: **0.0027**
mae: **0.0355**
relative error: **16.87%**



with augmentation

Training
mse: **0.0001**
mae: **0.0055**
relative error: **4.21%**

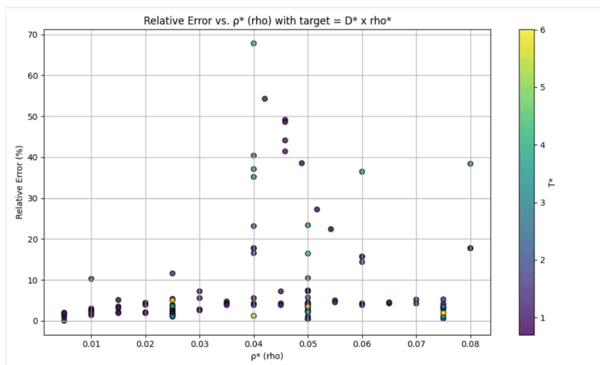
Testing
mse: **0.0022**
mae: **0.0269**
relative error: **13.82%**



with augmentation weight (8)

Training
mse: **0.0002**
mae: **0.0059**
relative error: **4.39%**

Testing
mse: **0.0017**
mae: **0.0215**
relative error: **9.54%**



- ***D*ρ*/√T and ρ****

I observe that when change target from ***D*ρ**** *to* ***D*ρ*/√T and ρ**** then the result improve better (about **3%**). Afterward adding a new dataset and adding weight (8) in training phase then results even better.

without_augmentation → with_augmentation → with_augmentation_weight
13.53 % **12.78 %** **8.74 %**

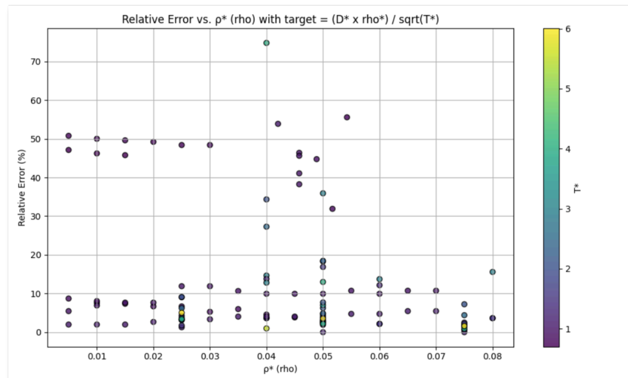
without augmentation

Training

mse: **0.0011**
mae: **0.0084**
relative error: **8.06%**

Testing

mse: **0.0015**
mae: **0.0227**
relative error: **13.53%**



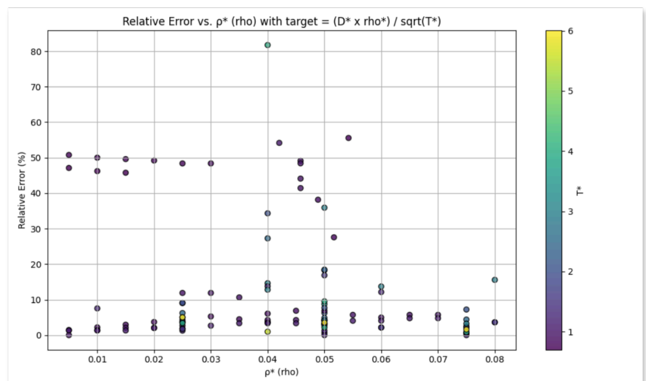
with augmentation

Training

mse: **0.0001**
mae: **0.0031**
relative error: **3.72%**

Testing

mse: **0.0015**
mae: **0.0217**
relative error: **12.78%**



with augmentation weight (8)

Training

mse: **0.0001**
mae: **0.0033**
relative error: **3.83%**

Testing

mse: **0.0012**
mae: **0.0163**
relative error: **8.74%**