विद्याधनं सर्वधन प्रधानम्

IIT JAMMU

Mini-Project

Department of Mechanical Engineering

ML for Bubble Growth

Prof.: Pothukuchi Harish

Team Members:-

1. Manas Gupta(2022UEE0137)
2. Meet Borisagar(2022UCS0088)
3. Muthres Gurjar(2022UCS0097)
4. Vibhav Bhagat(2022UEE0155)

Outline-

- Project introduction
- Data collection
- Model selection
- Result and Conclusion

Introduction

The primary objective of this mini-project is to collect experimental data and develop a machine learning (ML) model for bubble growth in flowing liquid that incorporates physical features such as pressure, mass flux, heat flux, subcooling, and other relevant parameters.

Data collection

- **Data Collection Approach:**

1. **Literature-Informed Data Extraction**

- Conducted extensive literature review.
- Attempted data extraction from graphical representations.
- Limitations: Sparse data points, insufficient coverage.

Data collection

2. **Relation-Based Data Generation**

- Utilized mathematical relations from literature.
- Generated synthetic data points considering experimental conditions.
- Employed multiple formulae to ensure diversity and mitigate overfitting.

Formulas:-

$$r(t) = \frac{2b}{\sqrt{\pi}} \mathrm{Ja} \sqrt{\alpha t}$$

$$\frac{D_b}{D_{bm}} = 1 - 2^K \left| \frac{1}{2} - \left(\frac{t}{t_b} \right)^N \right|^K$$

$$a(t) = \frac{2}{3} \frac{B^2}{A} \left[(t^+ + 1)^{3/2} - (t^+)^{3/2} - 1 \right]$$

- https://www.sciencedirect.com/science/article/pii/S1359431111003462
- B. B. Mikic, W. M. Rohsenow and P. Grffith, On bubblegrowth rates, In. J. Heat Mass Transfer 13, 657-666(1970).
- https://www.researchgate.net/publication/329254680_Subcooled_flow_boiling_in_a_flat_mini-channel_under_local_heating

Data set

	pressure(bar)	heat flux(kW/m2)	mass fluxkg/(m2-s)	sub cooling	channel dia(mm)	d/dMax	t/tMax
0	1.11	173.000	495.0	6.5	13.33	0.154977	0.004673
1	1.11	173.000	495.0	6.5	13.33	0.215351	0.009346
2	1.11	173.000	495.0	6.5	13.33	0.260160	0.014019
3	1.11	173.000	495.0	6.5	13.33	0.296914	0.018692
4	1.11	173.000	495.0	6.5	13.33	0.328518	0.023364
...	...	...	...	...	...	...	...
09	1.00	0.045	30.0	24.0	3.75	0.916238	0.824840
10	1.00	0.045	30.0	24.0	3.75	0.939426	0.880710
11	1.00	0.045	30.0	24.0	3.75	0.953692	0.913930
12	1.00	0.045	30.0	24.0	3.75	0.992806	0.960740
13	1.00	0.045	30.0	24.0	3.75	0.545288	1.000000

Reference

1. https://docs.google.com/spreadsheets/d/1_l8KEdNzFS5rsSjwH3FBdQi0dvkT8B3y/edit#gid=826066449

Model Selection and Training Approach:

- Explored various regression models
- XGBoost chosen for superior performance.
- Adopted hybrid training approach.

Challenges with Other Models:

- Linear Models: Not suitable for non-linear relationships.
- Decision Trees: Prone to overfitting.
- Gradient Boosting: Slower compared to XGBoost.

XGboost Regressor

- The objective function contains loss function and a regularization term. It tells about the difference between actual values and predicted values.
- Ensemble learning involves training and combining individual models (known as base learners) to get a single prediction.

XGboost Regressor

$$\text{Output value} = \frac{\sum \text{Residuals}}{N + \lambda}$$

$$\text{Similarity Score} = \frac{\left(\sum \text{Residuals}\right)^2}{N + \lambda}$$

N = No. of Residuals

λ = Regularization Parameter

New prediction = Previous Prediction + Learning rate x Output

GAIN = Left Similarity + Right Similarity - Root Similarity

Examples of XGboost Regressor

Example : Bubble Growth ML

XG Boost Regressor.

Saathi

Date ___/___/___

Mass Flux	t/tmax	D/Dmax	Resid	Res1 O/P	Res2
1	0.2	0.1	−0.3	0.325	
3	0.4	0.3	−0.1	0.375	
5	0.6	0.6	0.2		
7	0.8	0.5	0.1		
9	1	0.45	0.05		

$\boxed{\text{Base Model}} \rightarrow$ avg. $\rightarrow \dfrac{0.1 + 0.3 + 0.6 + 0.5 + 0.3}{5}$

≈ 0.4

$[-0.3,\ -0.1,\ 0.2,\ 0.1,\ 0.05]$

S.W = 0.0004

$\boxed{\text{Mass flux}}$

<= 1

$[-0.3]$

$[-0.1,\ 0.2,\ 0.1,\ 0.05]$

Similarity Weight $= \dfrac{(-0.3)^2}{1+1}$

$= 0.045$

S.W $= \dfrac{(0.25)^2}{4+1}$

$= 0.0125$

Gain $= 0.045 + 0.0125 - 0.0004$

$= 0.0571$

Examples of XGboost Regressor

Date ___/___/___

Saathi

[Moss Flow]

<=3 | >3

[−0.3, −0.1] | [0.2, 0.1, 0.05]

$SW = \frac{0.16}{3} = 0.053$

$S.W = \frac{(0.35)^2}{3+1} = 0.03$

Gain = 0.053 + 0.03 − 0.0005

= 0.0876

Gain ↑ ⇒ Better to cut/slice for decision tree

[Moss Flow]

<=3

[−0.3, −0.1] | [0.2, 0.1, 0.05]

<= 0.2 | > 0.2 | > 0.2

[−0.3] | [−0.1]

$O/P = \frac{0.2+0.1+0.05}{5}$

= 0.0875

$O/P = \frac{-0.3}{1+1}$ | $O/P = \frac{-0.1}{1+1}$

= −0.15 | = −0.05

Learning Rate

α

New Prediction = 0.4 + (0.5) (−0.15)

= 0.325

NP = 0.4 + (0.5)(−0.05) = 0.375

= 0.375

Page No.

Why XGBoost was Chosen:

- Performance: Outperformed other models.
- Robustness: Handles noise and overfitting well.
- Feature Importance: Provides valuable insights.
- Scalability: Efficient for large datasets.

Model

```
import numpy as np
import pandas as pd
import sklearn
from xgboost import XGBRegressor
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_absolute_error
from sklearn.metrics import median_absolute_error
from sklearn.metrics import mean_poisson_deviance
from sklearn.metrics import r2_score
from sklearn.metrics import explained_variance_score
```

Model

```
df_ex = pd.read_csv("mix data.csv")

df_ex.dropna(inplace=True)

xe_o = df_ex.drop("d/dMax", axis=1)
ye_o = df_ex["d/dMax"]

xe = xe_o[1079:]
ye = ye_o[1079:]

np.random.seed(41)
param_grid = {
    'n_estimators': [100, 300],
    'learning_rate': [0.01, 0.1],
    'max_depth': [3, 5, 7],
    'min_child_weight': [1, 3, 5],
    'subsample': [0.5, 0.7],
    'colsample_bytree': [0.5, 0.7],
    'gamma': [0, 0.1]
}
```

Model

```
from sklearn.model_selection import GridSearchCV

m_ex = GridSearchCV(o_m_ex , param_grid , cv = 2 , verbose = 2)
m_ex.fit(xe, ye)
%timeit

count = 0
x = 0
for i in xe["t/tMax"]:
    x = x+1
    if i == 1:
        print(x)
        count = count+ 1
        if count == 5:
            break
```

.

Model

```python
def evaluation(y_true , y_preds ):
    mean_absolute = mean_absolute_error(y_true , y_preds)
    median_absolute = median_absolute_error(y_true , y_preds)
    mean_poisson = mean_poisson_deviance(y_true , y_preds)
    r2_scor = r2_score(y_true , y_preds)
    explained_variance = explained_variance_score(y_true , y_preds)

    print(f"the mean absolute error is  {mean_absolute }")
    print(f"the median absolute error is  {median_absolute }")
    print(f"the mean poisson deviance is  {mean_poisson}")
    print(f"the r2_score is  {r2_scor}")
    print(f"the explained_variance is  {explained_variance}")
```

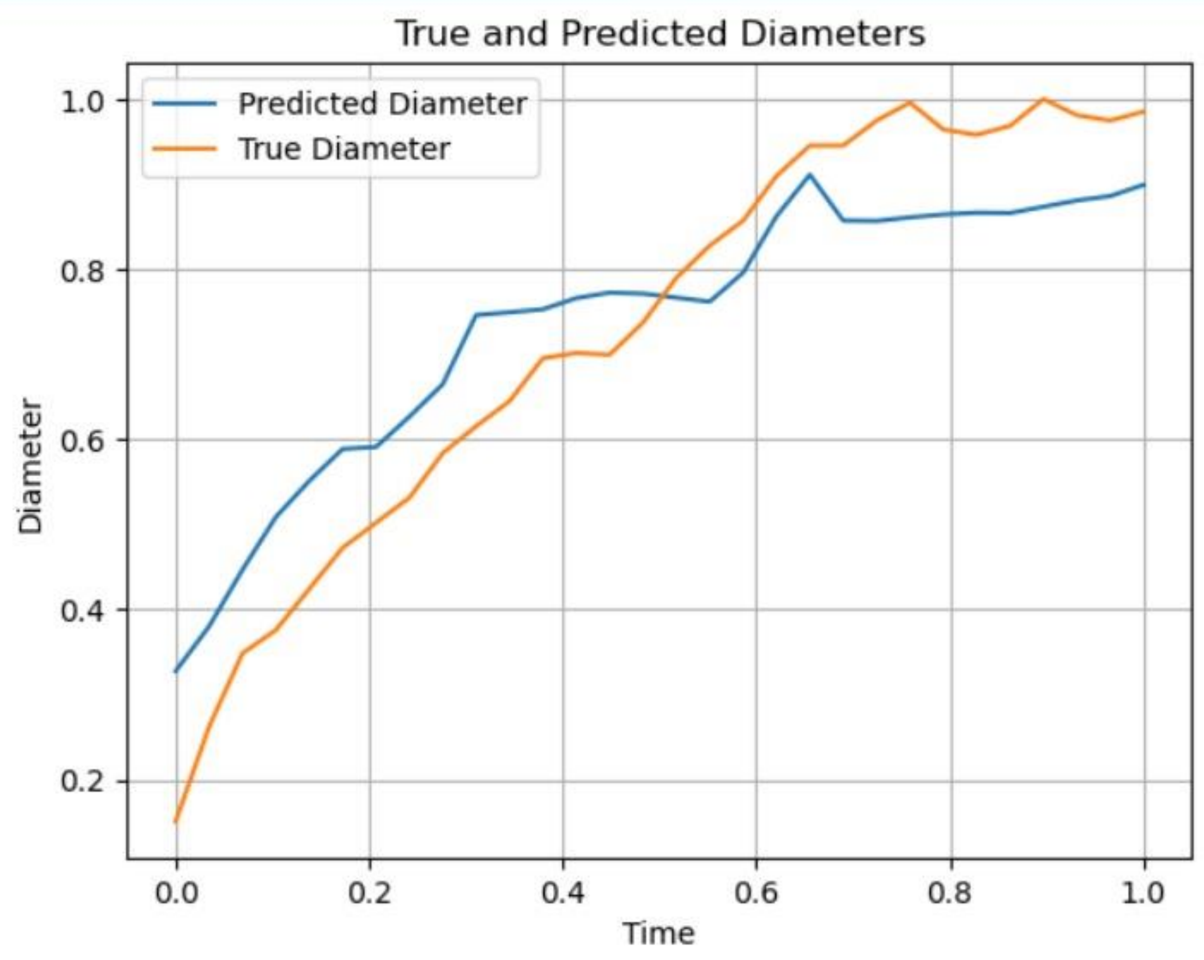
.

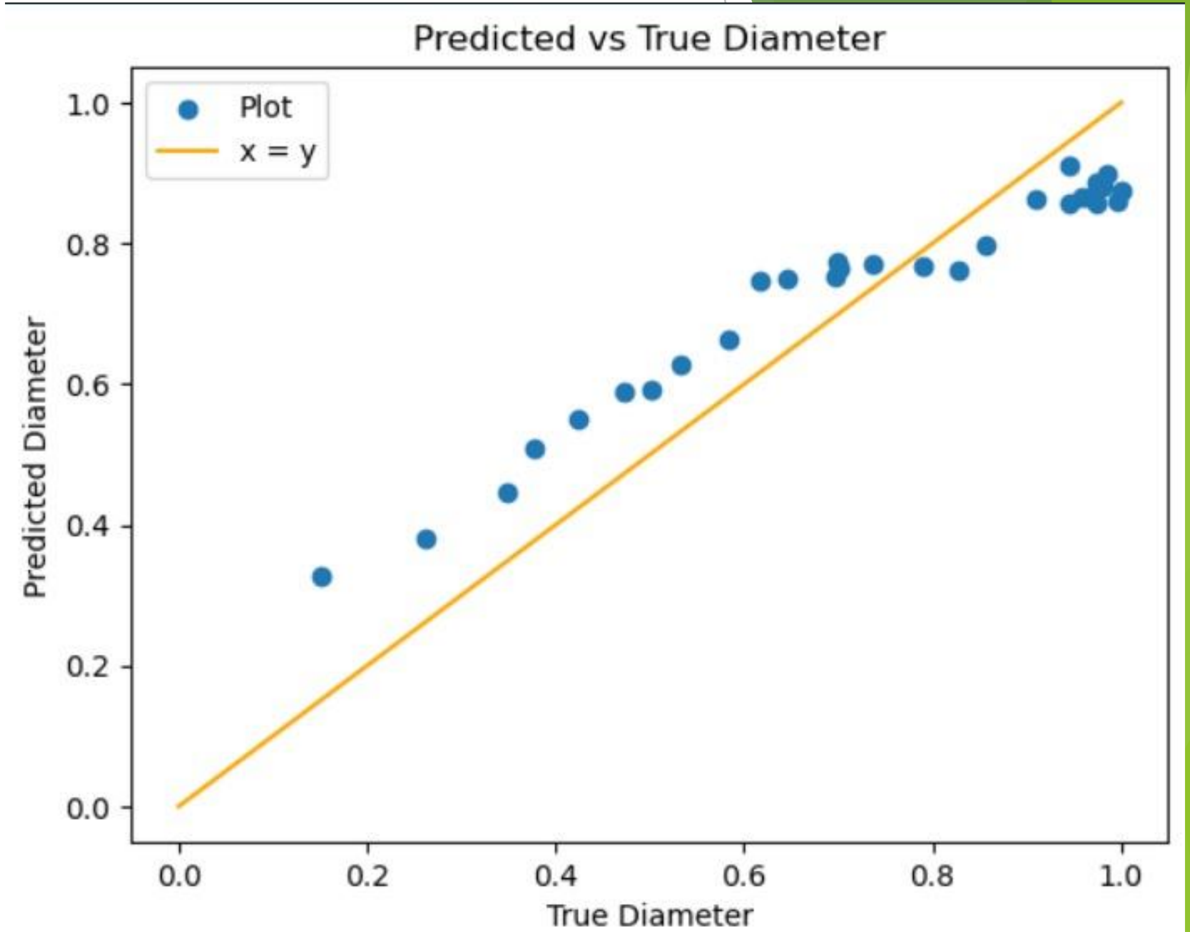
Model

```
testingx1 = xe_o[:213]
testingy1 = ye_o[:213]
testingx2 = xe_o[214:452]
testingy2 = ye_o[214:452]
testingx3 = xe_o[453:679]
testingy3 = ye_o[453:679]
testingx4 = xe_o[680:1017]
testingy4 = ye_o[680:1017]
testingx5 = xe_o[1018:1078]
testingy5 = ye_o[1018:1078]

yp1 = m_ex.predict(testingx1)
yp2 = m_ex.predict(testingx2)
yp3 = m_ex.predict(testingx3)
yp4 = m_ex.predict(testingx4)
yp5 = m_ex.predict(testingx5)
```

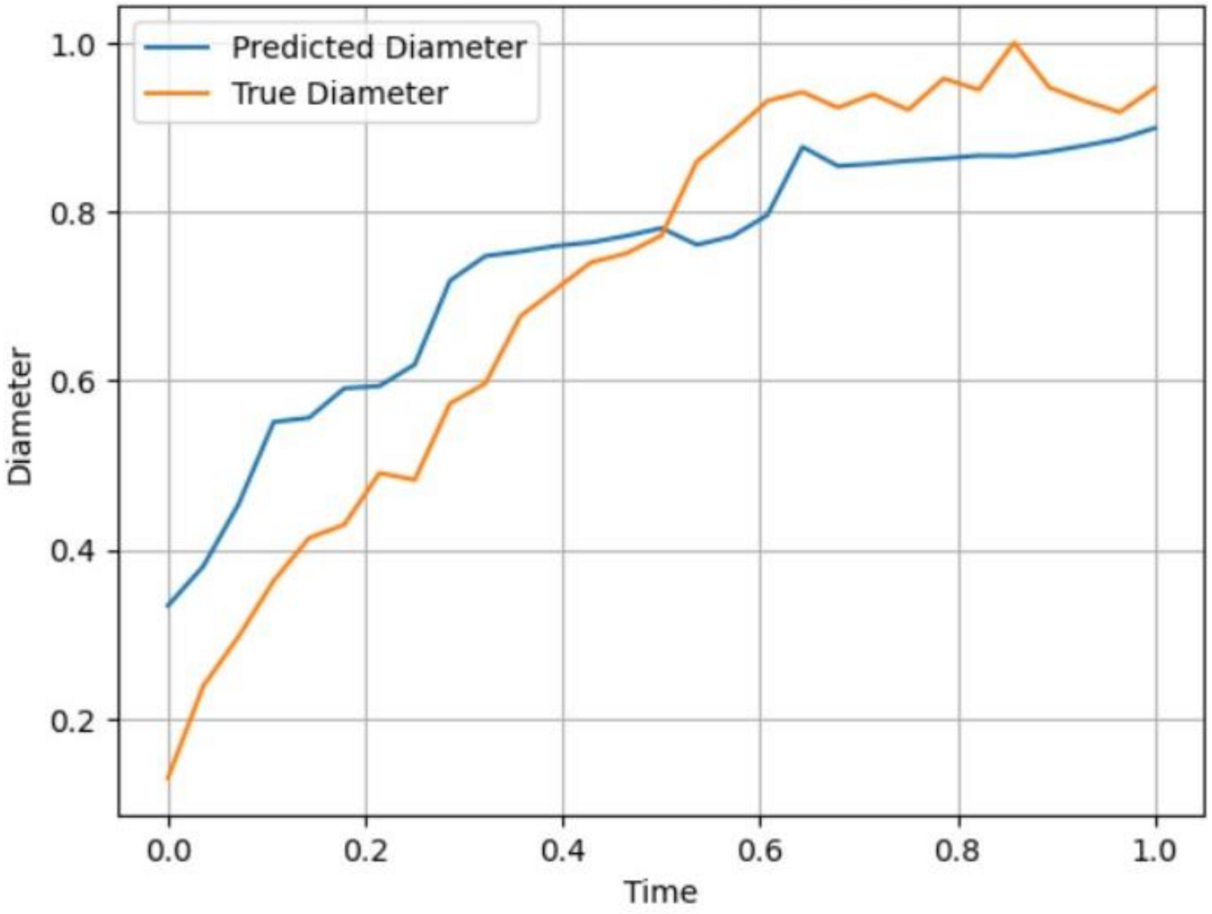
Approach 1



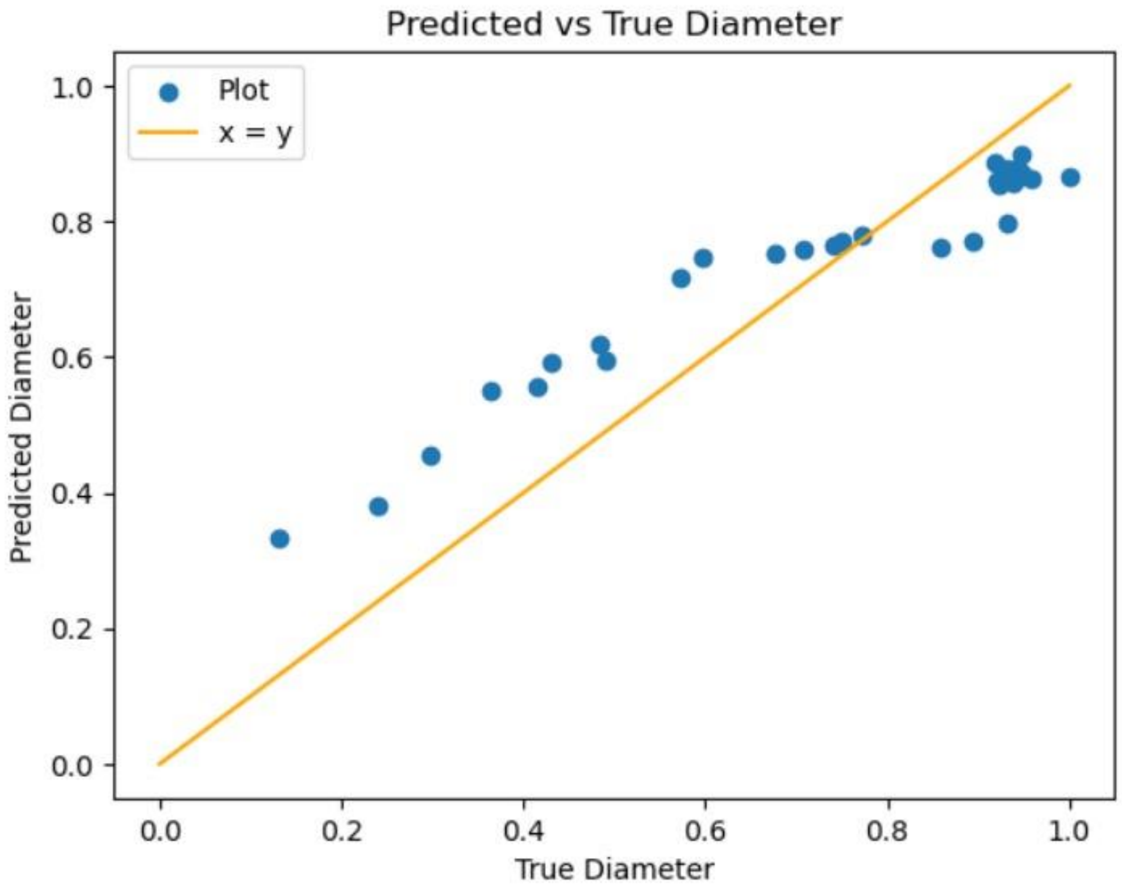


graph1

Approach 1



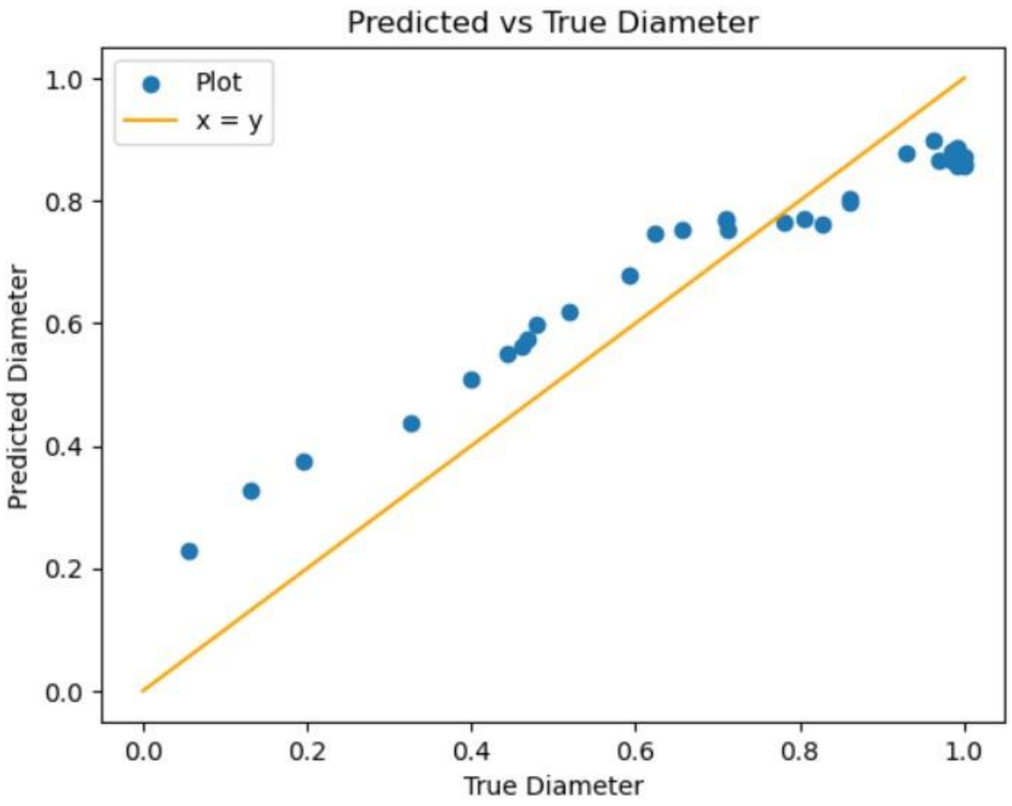
Predicted Diameter
True Diameter
Diameter
Time
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0



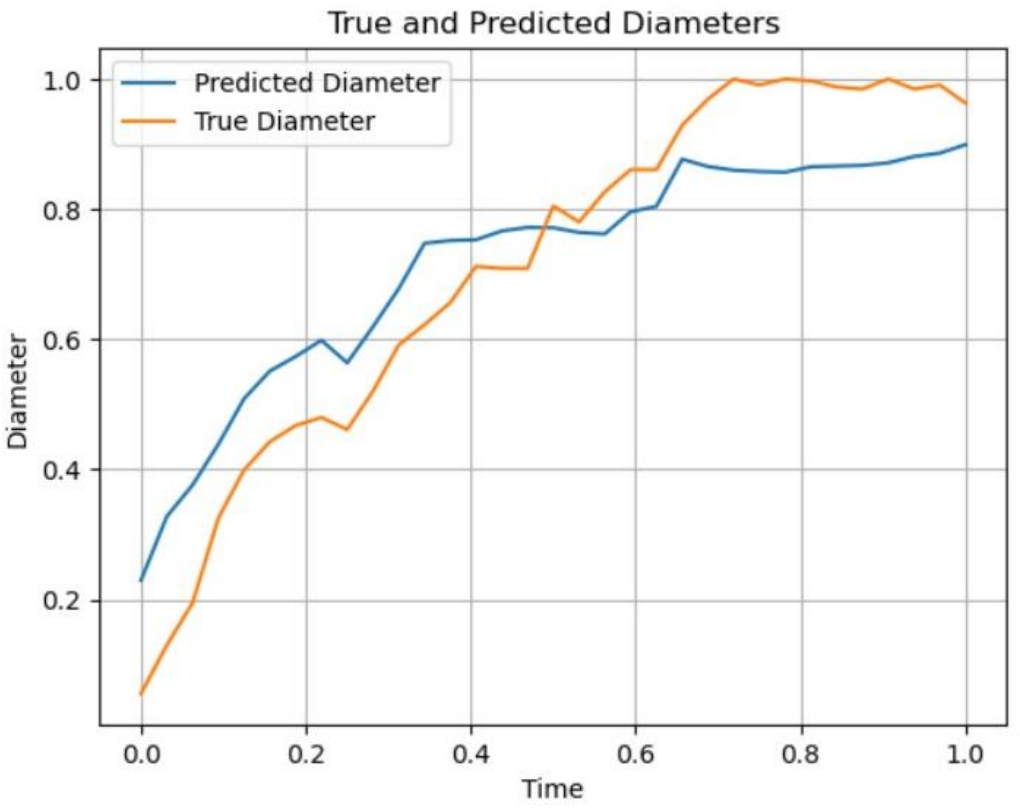
Predicted vs True Diameter
Plot
x = y
Predicted Diameter
True Diameter
0.0
0.2
0.4
0.6
0.8
1.0

.

Approach 1



Predicted vs True Diameter
Plot
x = y
Predicted Diameter
True Diameter
0.0
0.2
0.4
0.6
0.8
1.0



True and Predicted Diameters
Predicted Diameter
True Diameter
Diameter
Time
0.0
0.2
0.4
0.6
0.8
1.0

.

Model evaluation

```
✓ 0.0s                                                    Python

the mean absolute error is  0.0985424080518318
the median absolute error is  0.09457955436540533
the mean poisson deviance is  0.024132951039125123
the r2_score is   80.76
the explained_variance is  0.8134929854530497
```

```
evaluation(testingy3, yp3)
```

```
✓ 0.0s                                                    Python

the mean absolute error is  0.1017634808199585
the median absolute error is  0.10467985069852292
the mean poisson deviance is  0.026222351027837914
the r2_score is   84.39
the explained_variance is  0.8444240018504391
```

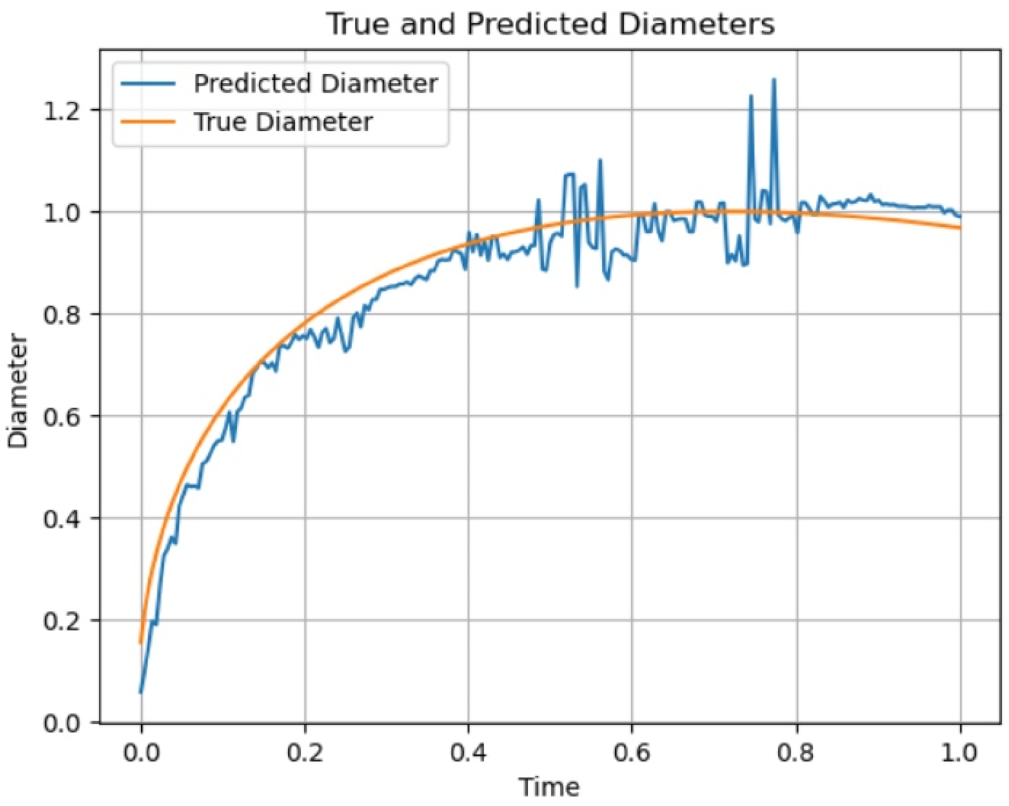
```
evaluation(testingy4, yp4)
```

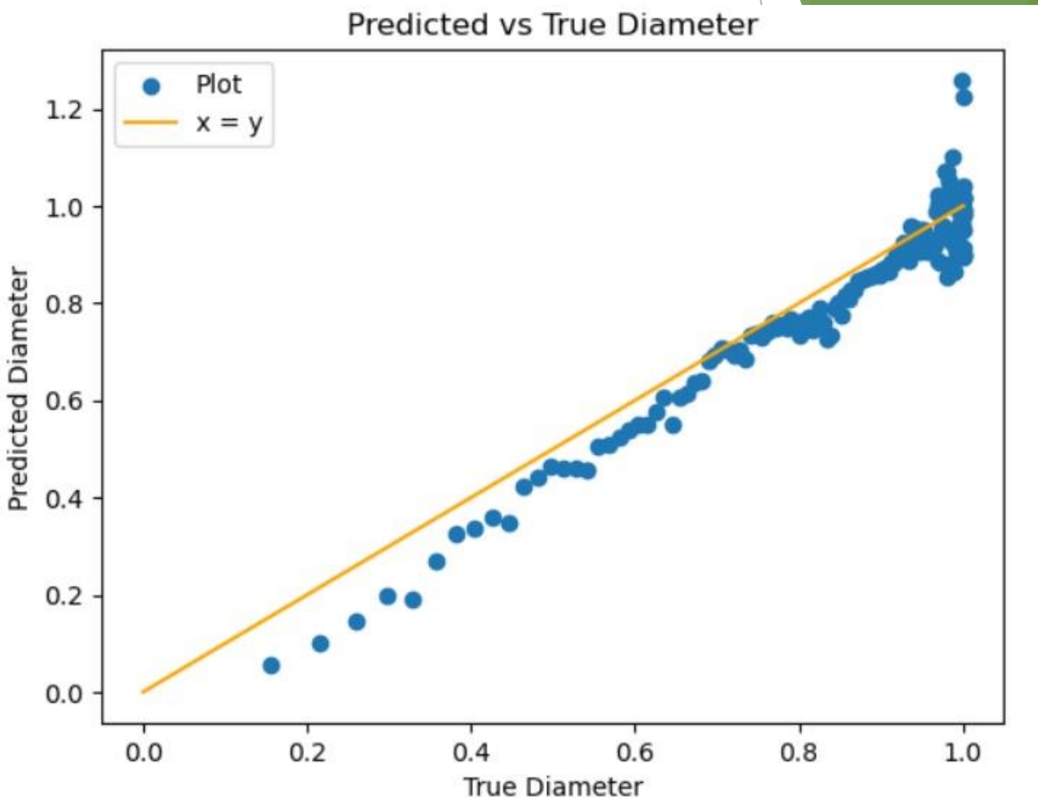
```
✓ 0.0s                                                    Python

the mean absolute error is  0.14215365853704176
the median absolute error is  0.14939844935443725
the mean poisson deviance is  0.04990535551587238
the r2_score is   39.41
the explained_variance is  0.4329414832905185
```

.

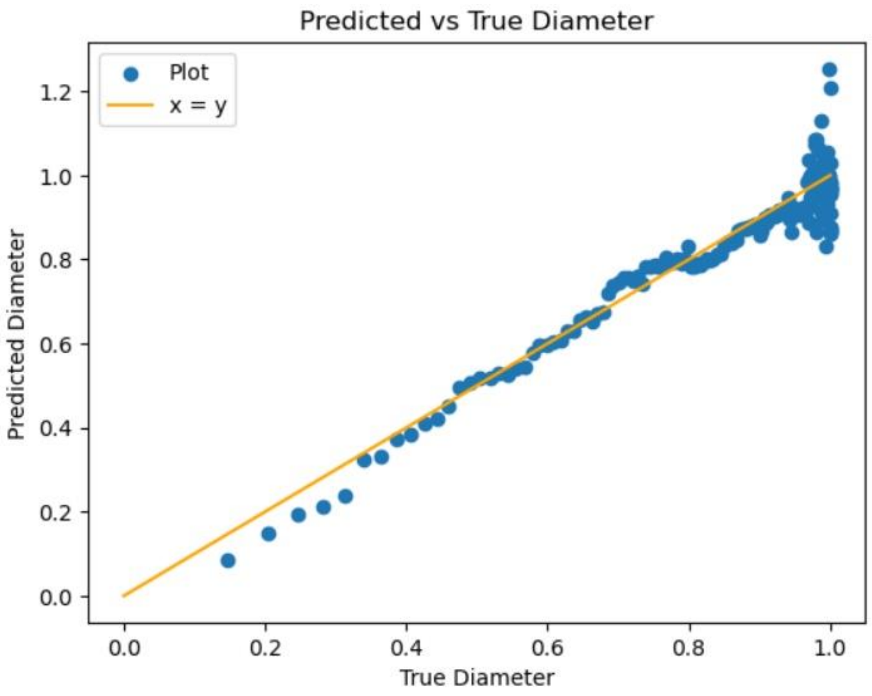
Approach 2

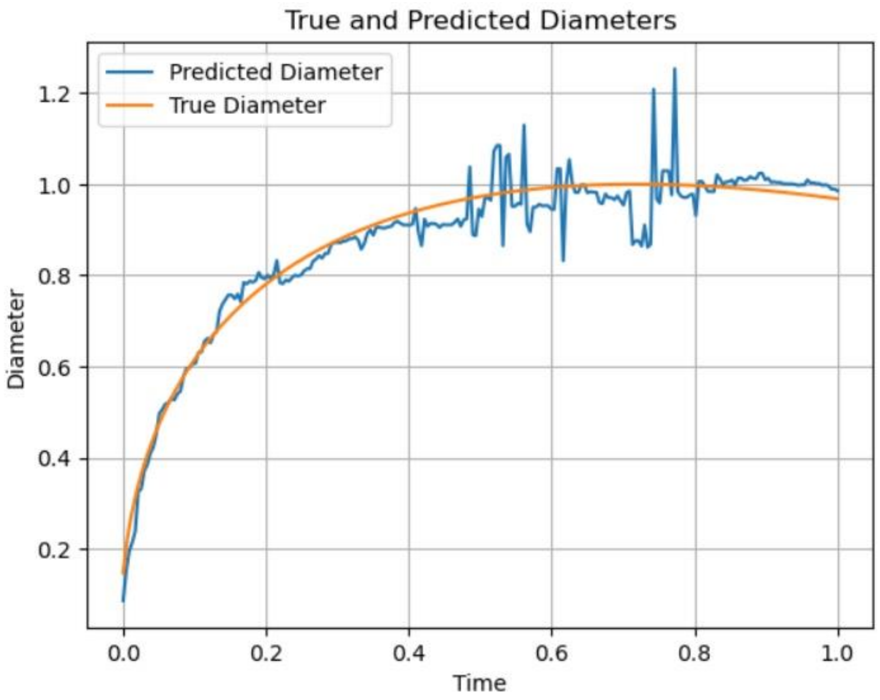

True and Predicted Diameters
Predicted Diameter
True Diameter
Diameter
Time
0.0
0.2
0.4
0.6
0.8
1.0
1.2


Predicted vs True Diameter
Plot
x = y
Predicted Diameter
True Diameter
0.0
0.2
0.4
0.6
0.8
1.0
1.2

.

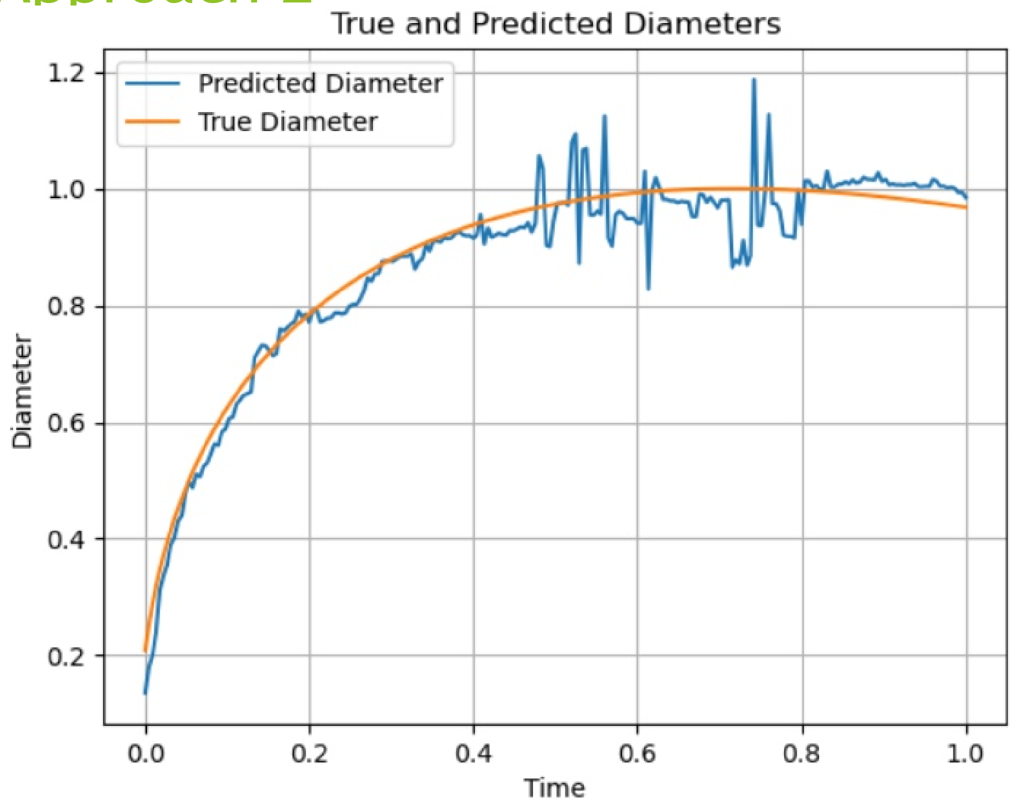
Approach 2

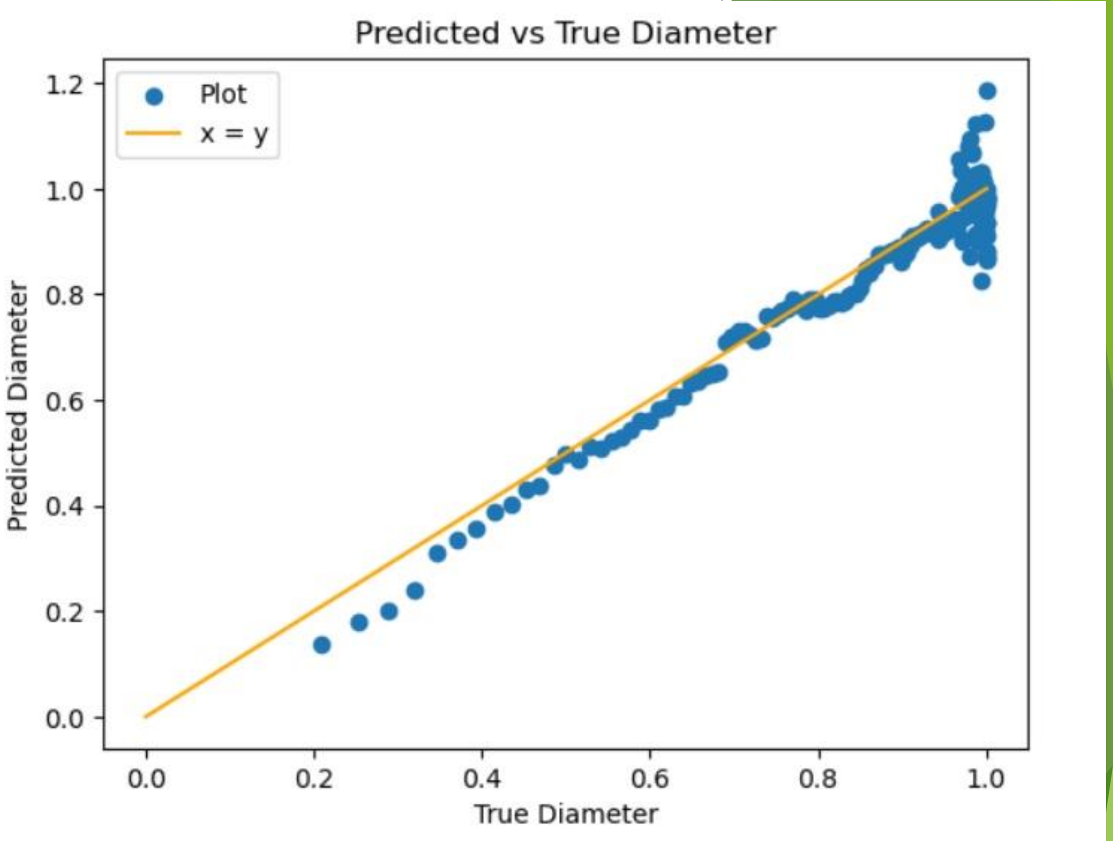

Predicted vs True Diameter
Plot
x = y
Predicted Diameter
True Diameter
0.0
0.2
0.4
0.6
0.8
1.0
1.2


True and Predicted Diameters
Predicted Diameter
True Diameter
Diameter
Time
0.0
0.2
0.4
0.6
0.8
1.0
1.2

.

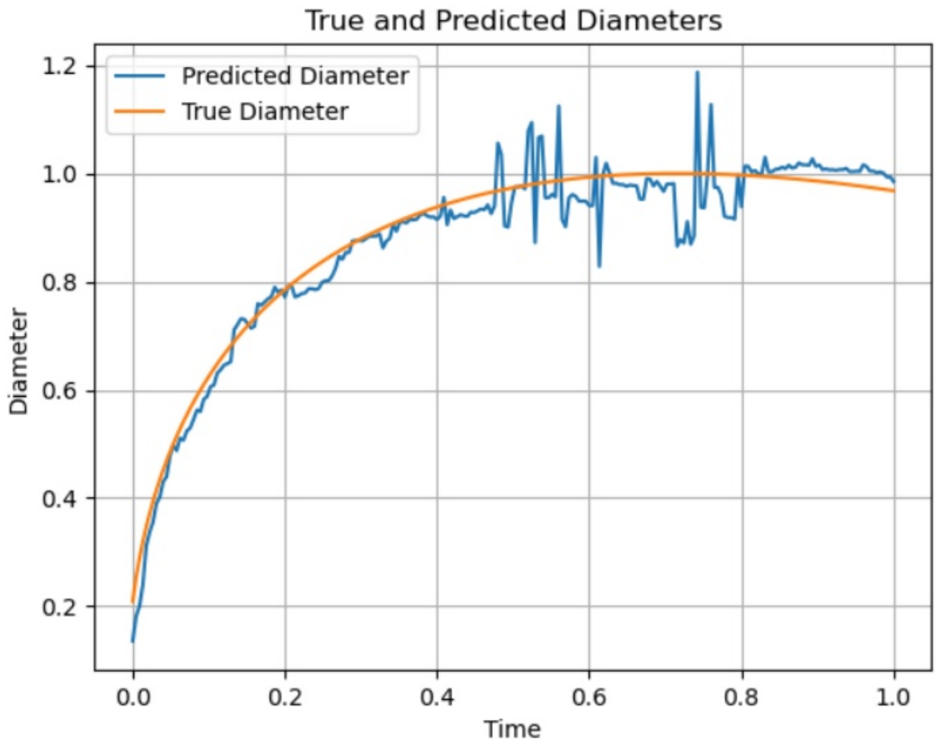
Approach 2


True and Predicted Diameters
Predicted Diameter
True Diameter
Diameter
Time
0.2
0.4
0.6
0.8
1.0
1.2
0.0
0.2
0.4
0.6
0.8
1.0

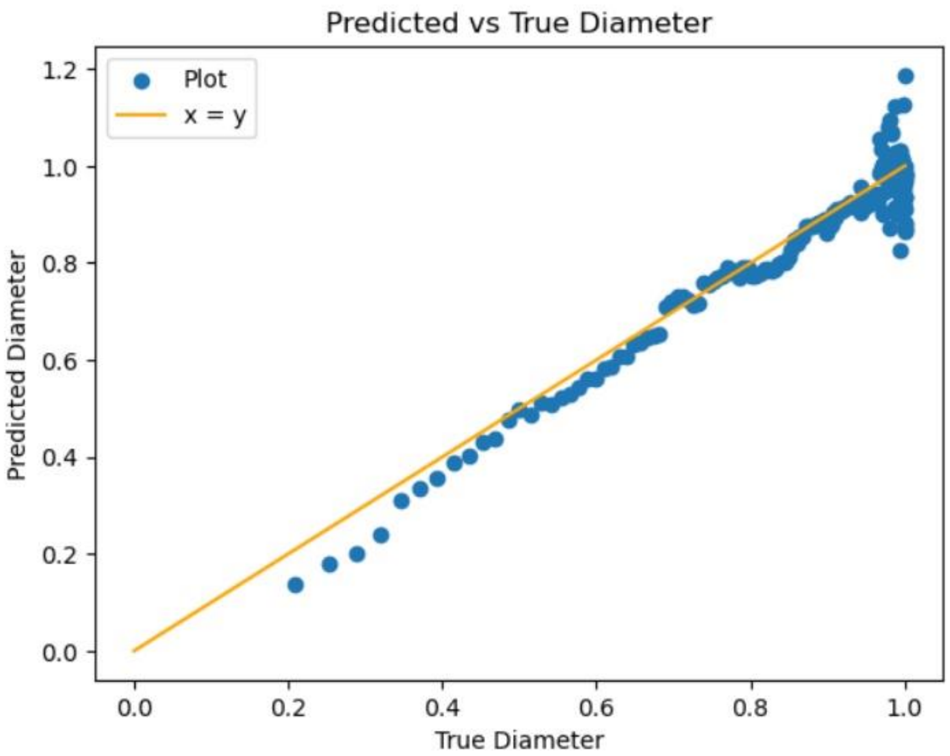

Predicted vs True Diameter
Plot
x = y
Predicted Diameter
True Diameter
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0.0
0.2
0.4
0.6
0.8
1.0

.

Approach 2



True and Predicted Diameters
Predicted Diameter
True Diameter
Diameter
Time
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0



Predicted vs True Diameter
Plot
x = y
Predicted Diameter
True Diameter
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

.

Model evaluation

```
[120]: evaluation(testingy1 , yp1)

the mean absolute error is  0.042640989206342635
the median absolute error is  0.0338384694486511
the mean poisson deviance is  0.005244872100167102
the r2_score is  0.9033060681909382
the explained_variance is  0.9158883248114164

[121]: evaluation(testingy2 , yp2)

the mean absolute error is  0.03322793639751075
the median absolute error is  0.02428819277018429
the mean poisson deviance is  0.0027042539374809837
the r2_score is  0.9287097394718593
the explained_variance is  0.9308258765773424

[122]: evaluation(testingy3 , yp3)

the mean absolute error is  0.03265381644022791
the median absolute error is  0.025876214137683096
the mean poisson deviance is  0.0025899994927438075
the r2_score is  0.9312001600278942
the explained_variance is  0.936237460355679

[123]: evaluation(testingy4 , yp4)

the mean absolute error is  0.03456695201300266
the median absolute error is  0.02164139072255855
the mean poisson deviance is  0.004485152275384383
the r2_score is  0.8918886288953248
the explained_variance is  0.8948556545856442
```

,

Conclusion

- The XGBoost model predicts bubble growth dynamics.
- To further improve model performance, expanding the dataset with more diameter vs. time relations is recommended.
- For handling larger datasets, the use of LSTM (Long Short-Term Memory) models shows promise for future research endeavors.

References:-

Links of literatures:

1.https://www.sciencedirect.com/science/article/pii/S0017931023012036#bib0022
2. B. B. Mikic, W. M. Rohsenow and P. Griffith, On bubble growth rates, Inl. J. Heat Mass Transfer 13, 657-666 (1970).

3. Janani Sree Muralidharan a , B.V.S.S.S. Prasad b , B.S.V. Patnaik

4. Nikhil Chitnavis ,Harish Pothukuchi, B. S. V. Patnaik,Physics of Fluids 35, 053327 (2023),https://doi.org/10.1063/5.0145889

GitHub link:-

https://github.com/nyaksha06/bubble-growth


Thank You