Applied AI in Chemical and Process Engineering

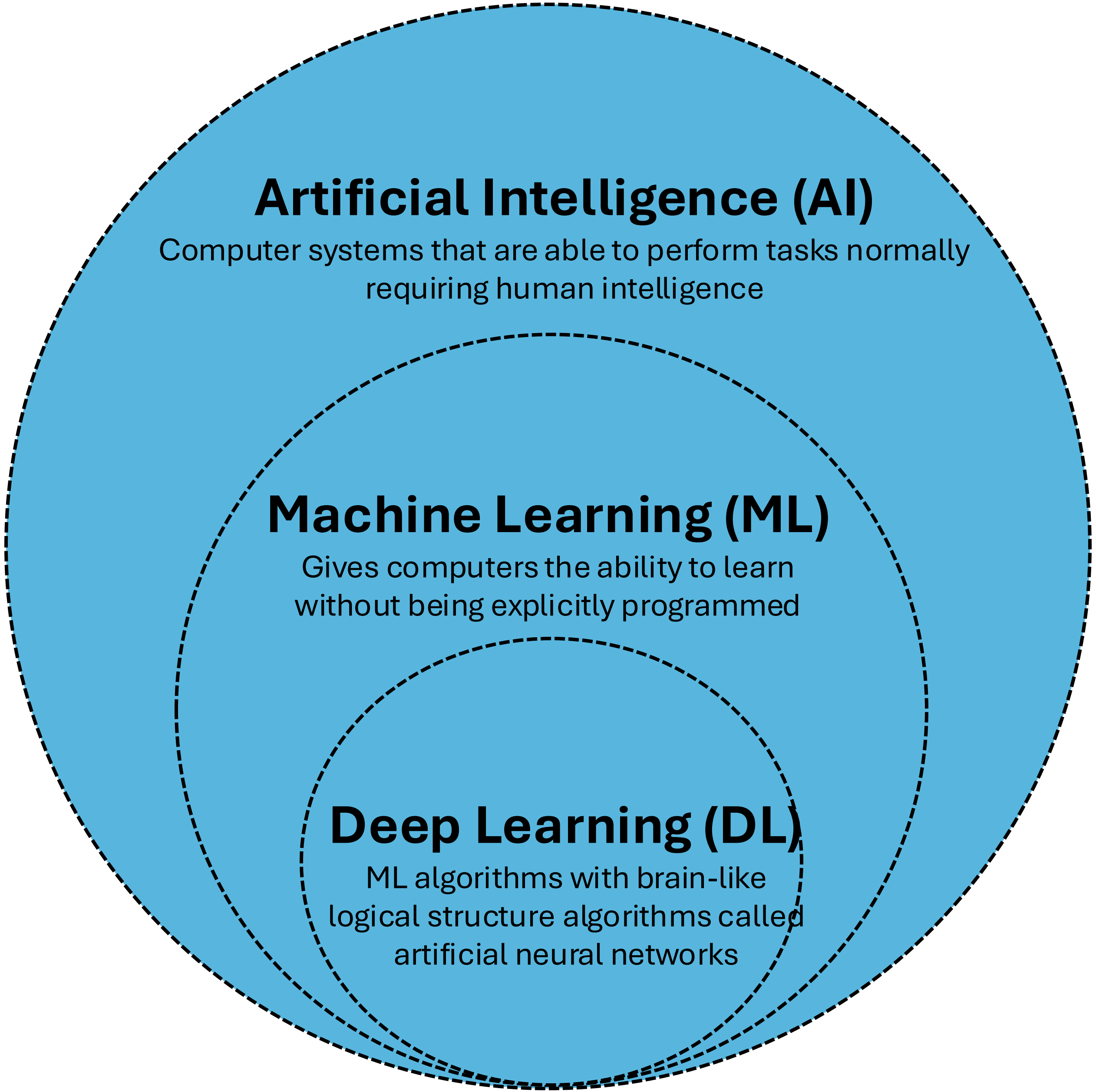
DMMB Dissanayake

DEPARTMENT OF
CHEMICAL AND PROCESS ENGINEERING
UNIVERSITY OF MORATUWA

What you are going to learn

- Machine Learning – Core Concepts
- Supervised Learning
- Common Supervised Learning Algorithms
- Vibe Coding Intro

Machine Learning/ Deep Learning/ Artificial Intelligence



Machine Learning

AI that learns from data without explicit instructions

E.g. Email spam filters, recommendation systems

Machine Learning – Broad Categorization

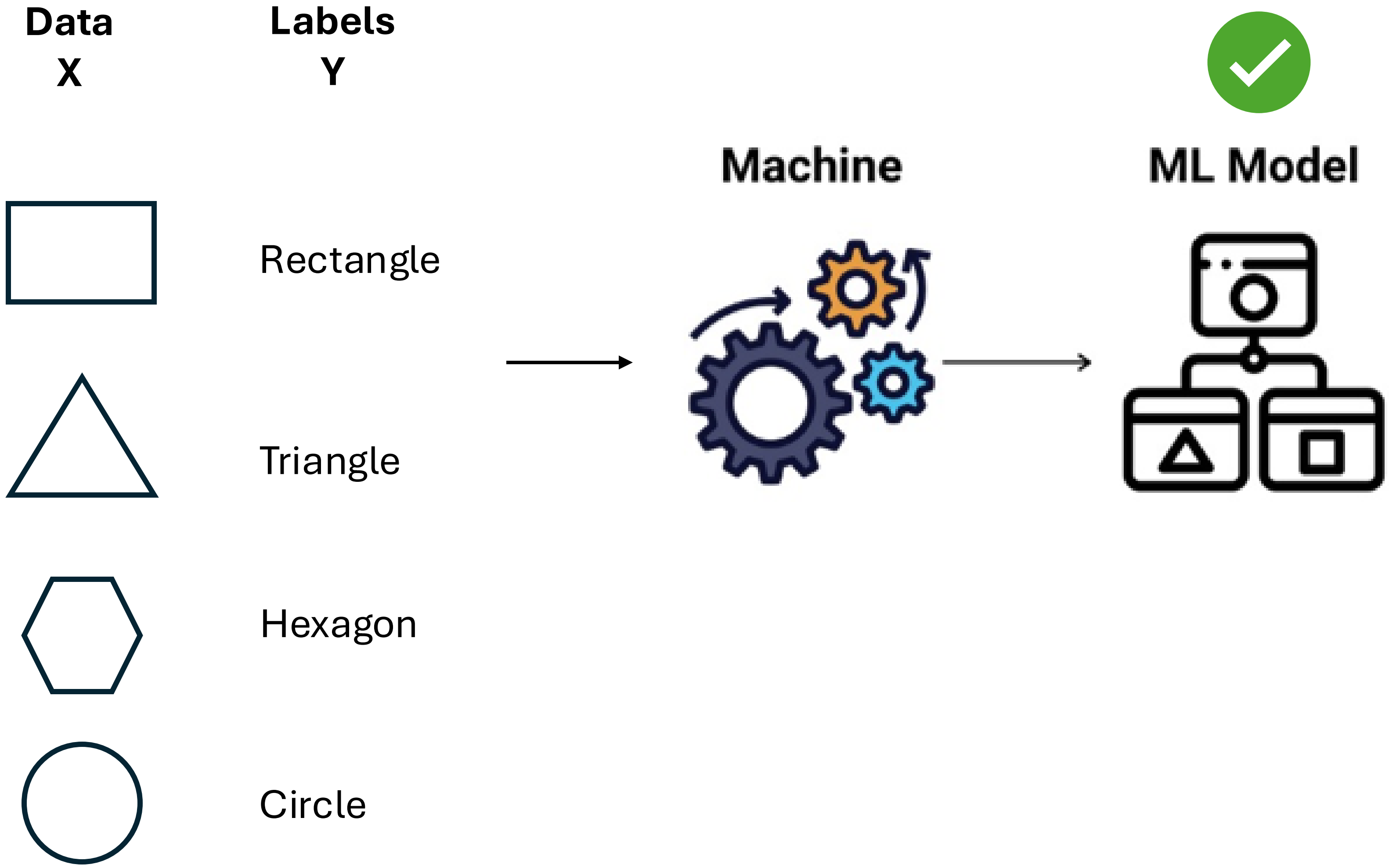
Supervised Learning

- ***Regression***
- *Classification*

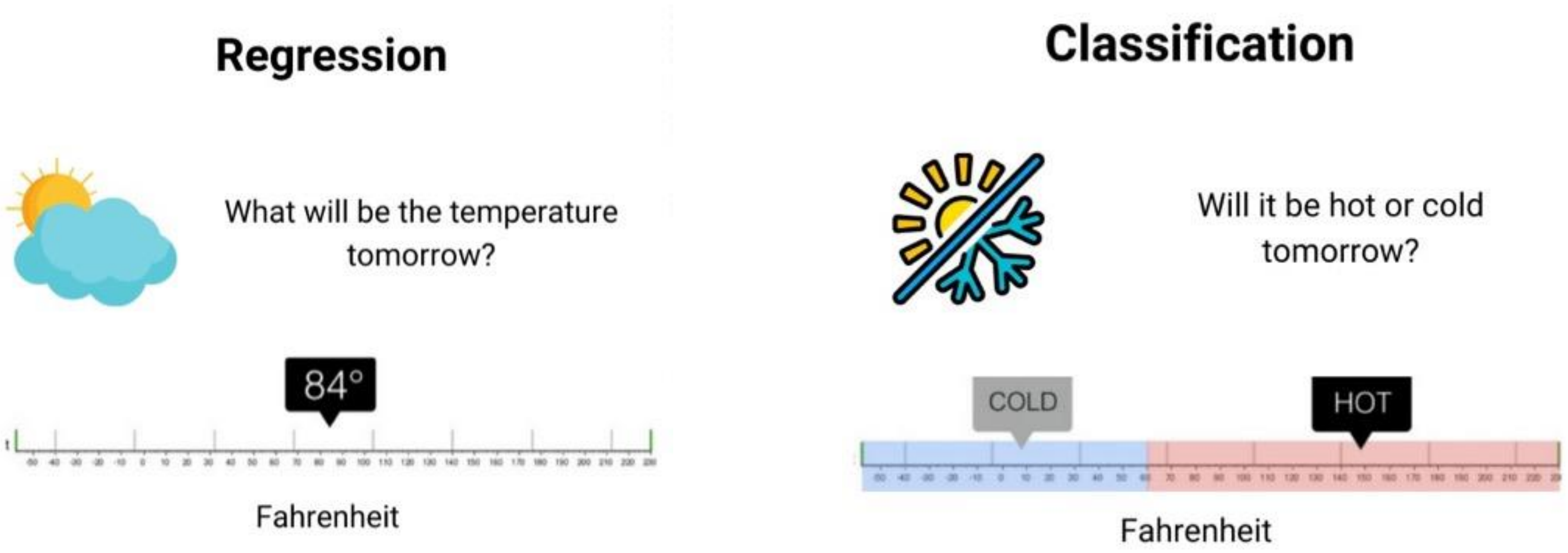
Unsupervised Learning

- *Clustering*
- *Dimensionality reduction*

Supervised Learning



Regression and Classification



Regression - Example

In a cement plant, we want to predict CO_2 emissions based on operational parameters

Fuel Consumption (tons/hour)	Production Rate (tons/hour)	Combustion Efficiency (%)	Clinker-to-Cement Ratio	**CO_2 Emissions (tons/hour)**
2.5	100	85	0.90	1.80
3.0	120	80	0.95	2.10
3.1	125	83	0.93	2.05

Features

Target

Regression - Example

AI

Filling volume

Regression

Classification - Example

In a chemical plant, we want to predict if a pump will fail based on sensor data

Vibration Level (mm/s)	Temperature (°C)	Pressure (bar)	Runtime (hours)	**Failure Status** 0-No failure, 1-Failure
2.5	70	5.0	500	No failure
4.8	85	6.2	1200	Failure
3.0	72	5.1	600	No failure
5.5	90	6.5	1500	Failure
2.8	68	4.8	450	Failure
4.2	82	5.8	1100	No failure
3.2	75	5.2	700	No failure
6.0	95	6.8	1600	Failure
2.7	69	4.9	400	No failure
4.5	88	6.0	1300	Failure

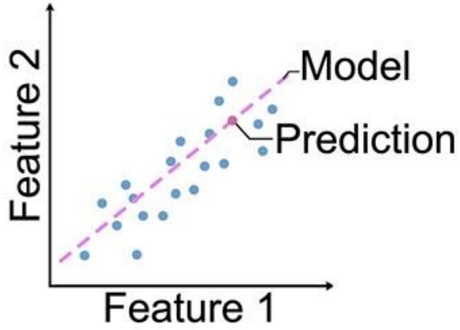
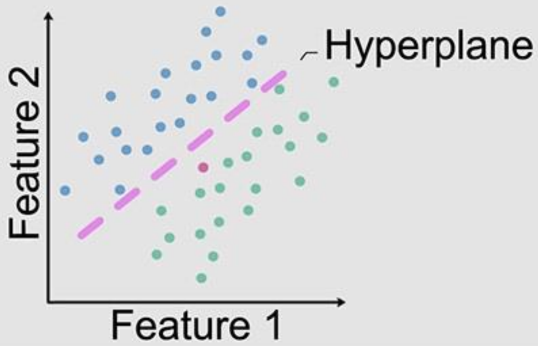
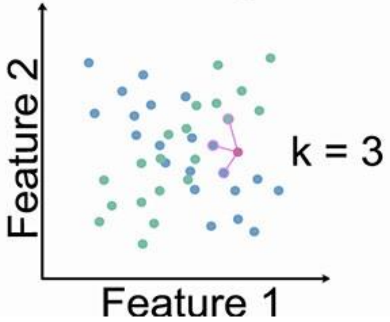
Classification – Example

AI

Pass or fail

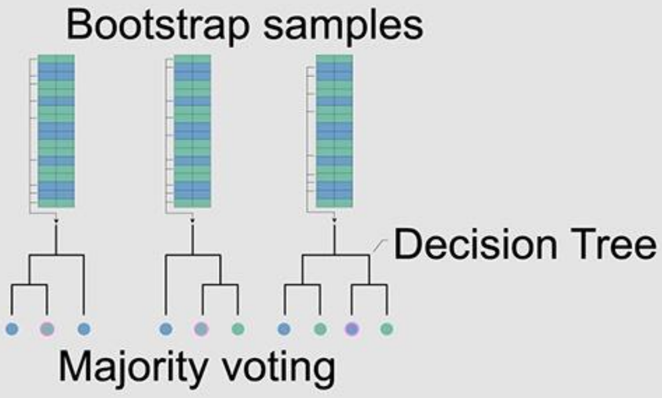
Classification

Summary of Common ML algorithms

Machine learning algorithms	Description	Data Type
LASSO, Ridge regression: 	Regression models with regularized coefficients (Appendix S1.2.1): + highly interpretable + few observations - limited flexibility	Tabular data: - Classification - Regression
Support vector machines: 	Hyperplane is optimized to separate response classes (Appendix S1.2.2): + fast and memory efficient + high dimensional data - kernel dependent - no probabilities	Tabular data: - Classification - Regression
k-nearest neighbor: 	k nearest neighbors in feature space decide response (e.g. by majority voting)(Appendix S1.2.3): + simple + no training - scales poorly - high dimensional data	Tabular data: - Classification - Regression

https://doi.org/10.1111/2041-210X.14061

Random forest:



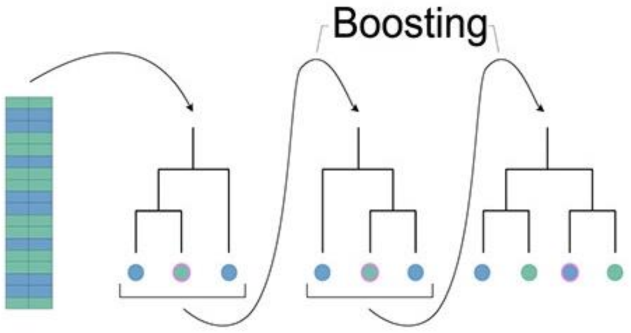
N decision (regression) trees are fitted on bootstrap samples. Split variable is selected from random subset of variables (Appendix S1.2.4):

+ flexible
+ robust (e.g. outliers)
+ few hyper-parameters
(+) variable importance
- scales poorly

Tabular data:

- Classification
- Regression

Boosted regression trees:



N trees are fitted sequentially to minimize an overall loss function (Appendix S1.2.5):

+ flexible
(+) variable importance
- many hyper-parameters
- high complexity

Tabular data:

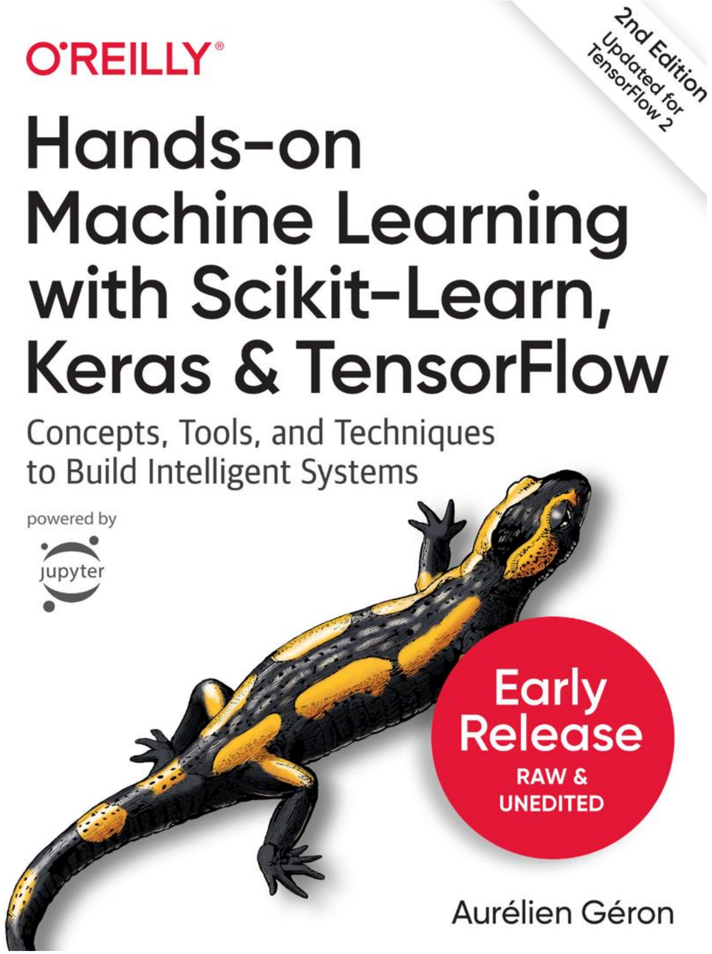
- Classification
- Regression

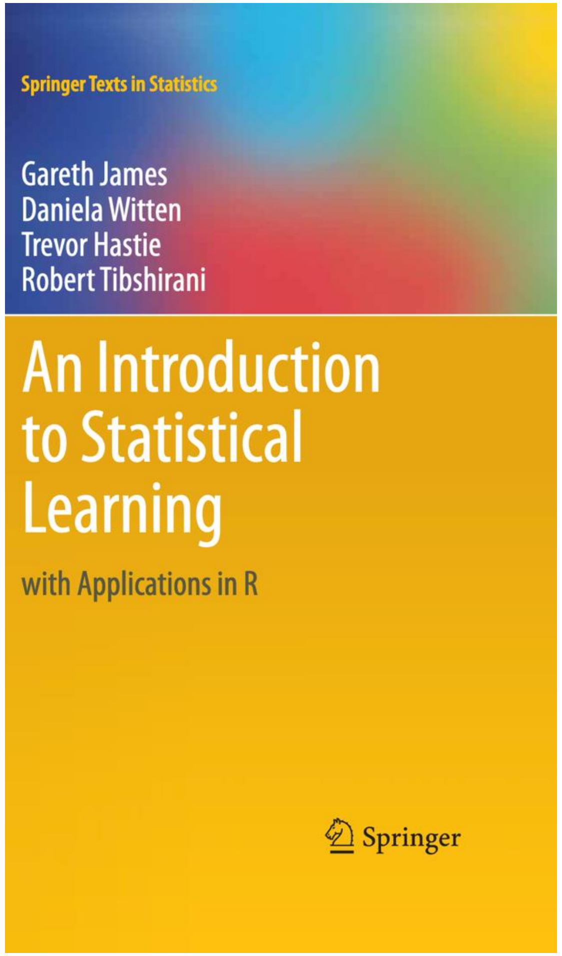
Further Learnings

scikit-learn

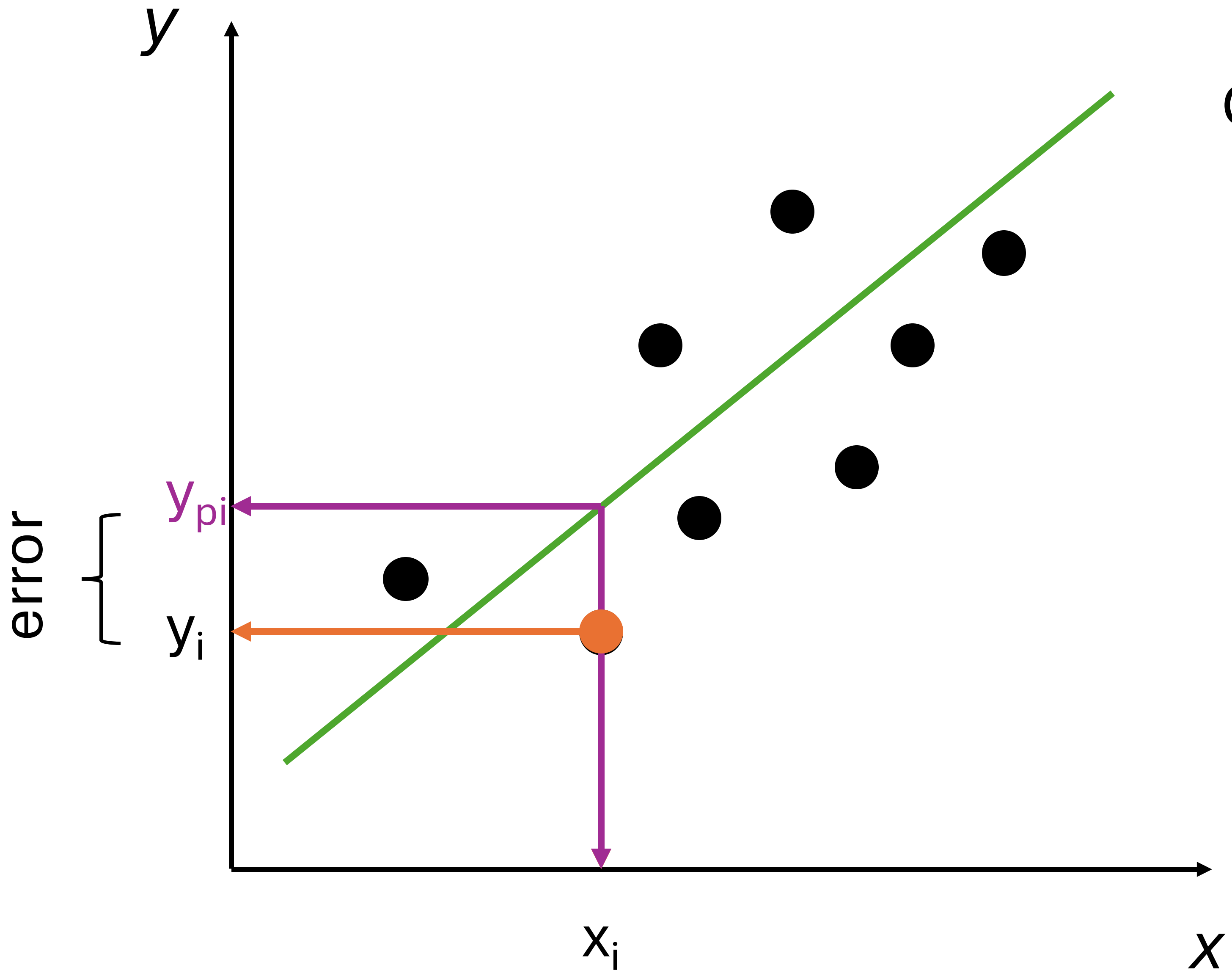
Machine Learning in Python

https://scikit-learn.org/stable/





Ordinary Least Square Regression (OLS)



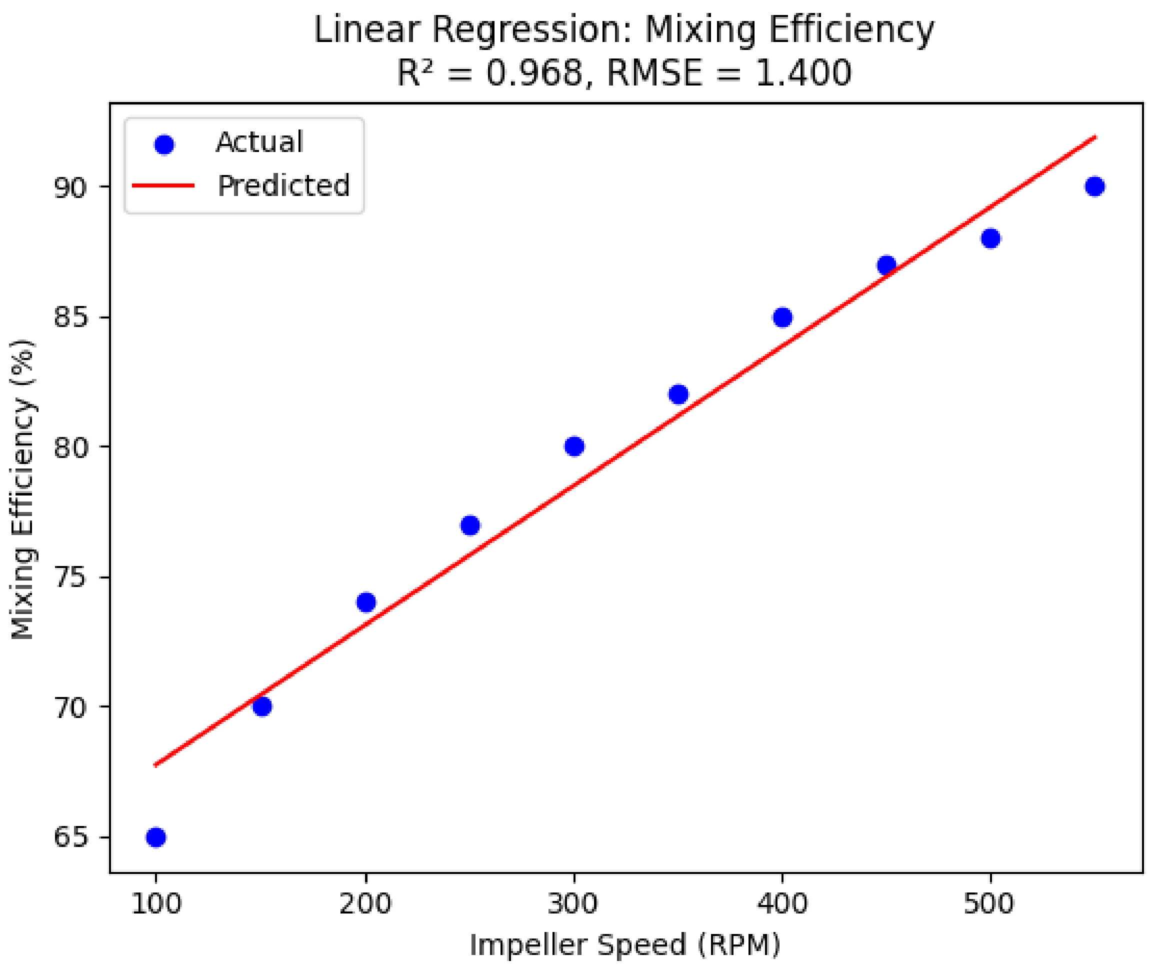
Goal is to minimize the Residual Sum or Squares (RSS)

$$RSS = \sum [y_i - y_{pi}]^2$$

$$y = mx + c$$

Example – Simple system

Impeller Speed (RPM)	Mixing Efficiency (%)
100	65
150	70
200	74
250	77
300	80
350	82
400	85
450	87
500	88
550	90



$$Mixing\ efficacy = 62.4 + 0.05 \times ImprellerSpeed$$

Coefficient of Determination (R^2)

R^2 measures how well a **regression model** explains the **variability** of the target variable.

$$R^2 = 1 - \frac{\mathrm{SS}_{\mathrm{res}}}{\mathrm{SS}_{\mathrm{tot}}}$$

Where:

- $\mathrm{SS}_{\mathrm{res}} = \sum(y_i - \hat{y}_i)^2$ → Residual Sum of Squares
- $\mathrm{SS}_{\mathrm{tot}} = \sum(y_i - \bar{y})^2$ → Total Sum of Squares
- $\hat{y}_i$ is the predicted value, and $\bar{y}$ is the mean of actual values

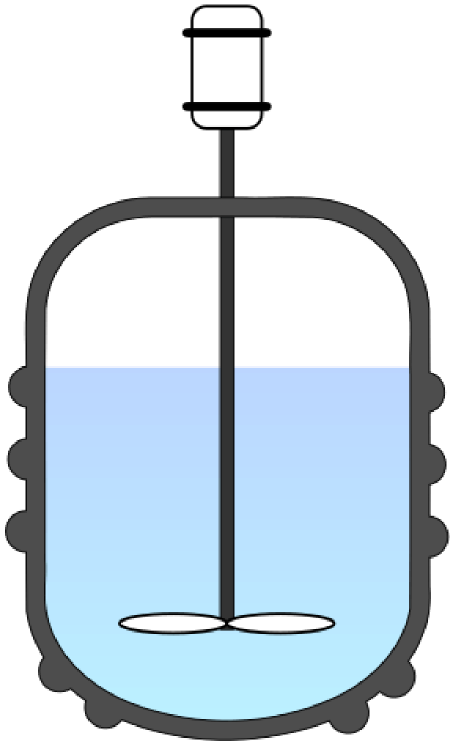
R^2 Interpretation

R^2 Value	Meaning
1	Perfect fit – model explains **100%** of the variability in data
0	Model explains **none** of the variability (predictions are as good as mean)
< 0	Model performs **worse than a horizontal line** (i.e., worse than just predicting the mean)
~0.7–0.9	Indicates a **strong model**, but context-dependent (high R^2 doesn't always mean good generalization)

Measuring the Error in Models

Metric	Formula	Description
ME (Mean Error)	$\mathrm{ME} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)$	Measures the **average error** between actual values y_i and predicted values $\hat{y}_i$. A value close to 0 indicates little bias, but it can hide large errors due to cancellation.
MSE (Mean Squared Error)	$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$	Calculates the **average of squared errors**, giving more weight to larger errors. Commonly used to evaluate regression models.
RMSE (Root Mean Squared Error)	$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$	Square root of MSE. Keeps the unit of the original variable, making it easier to interpret. Sensitive to outliers.

Example – Complex (somewhat) System



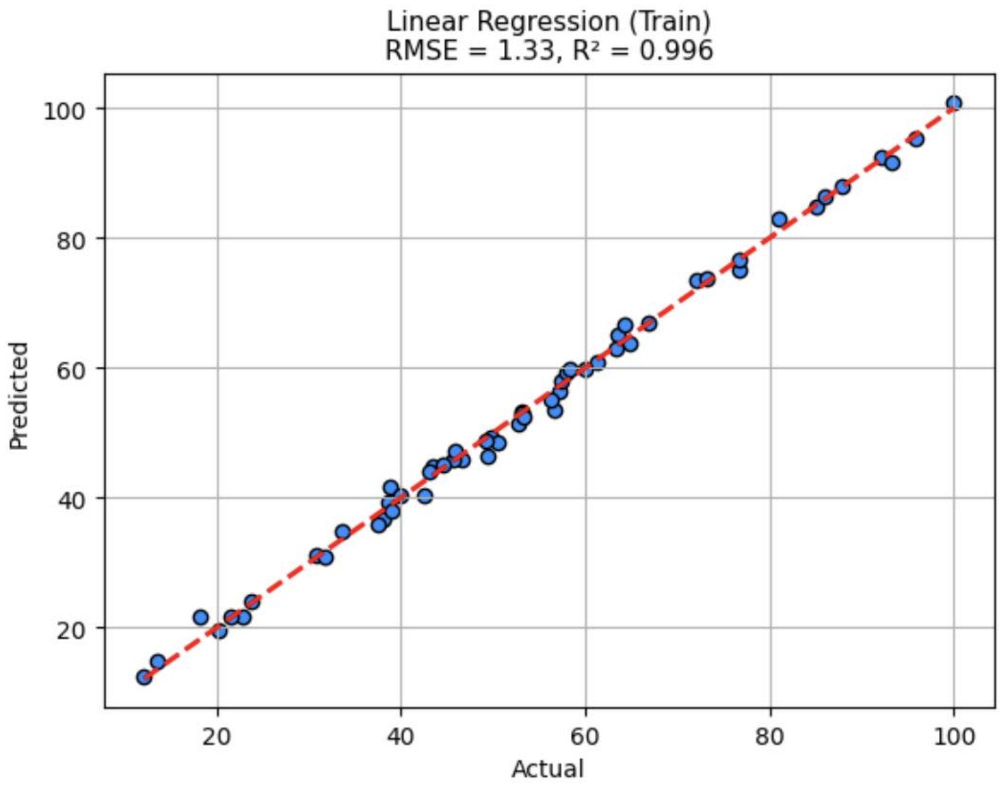
Can I predict the Yield?

Y= Yield

X (features) =
['reactor_temp', 'feed_flow', 'reactor_pressure', 'catalyst_conc', 'residence_time', 'sensors']

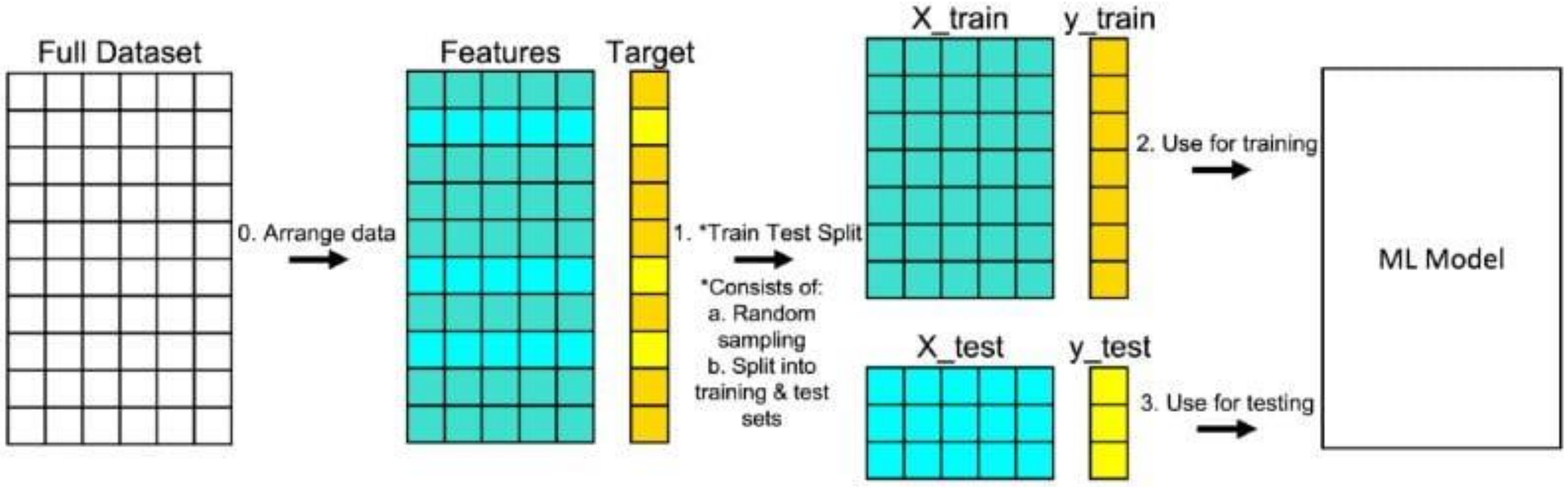
Why not OLS regression?

OLS regression model - evaluation



Can I trust it?

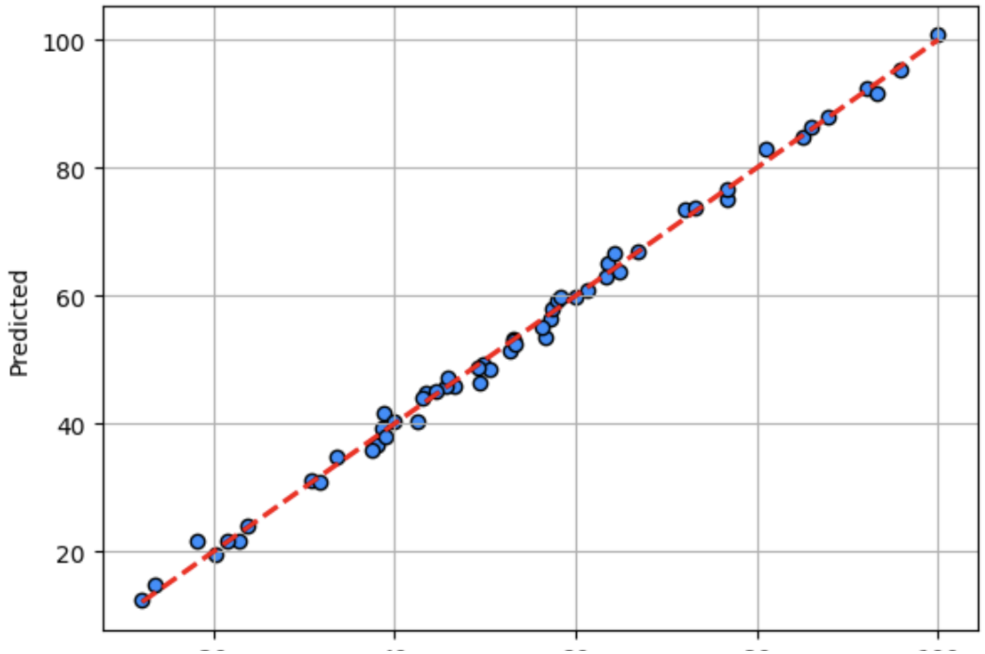
Training strategy with train/test split



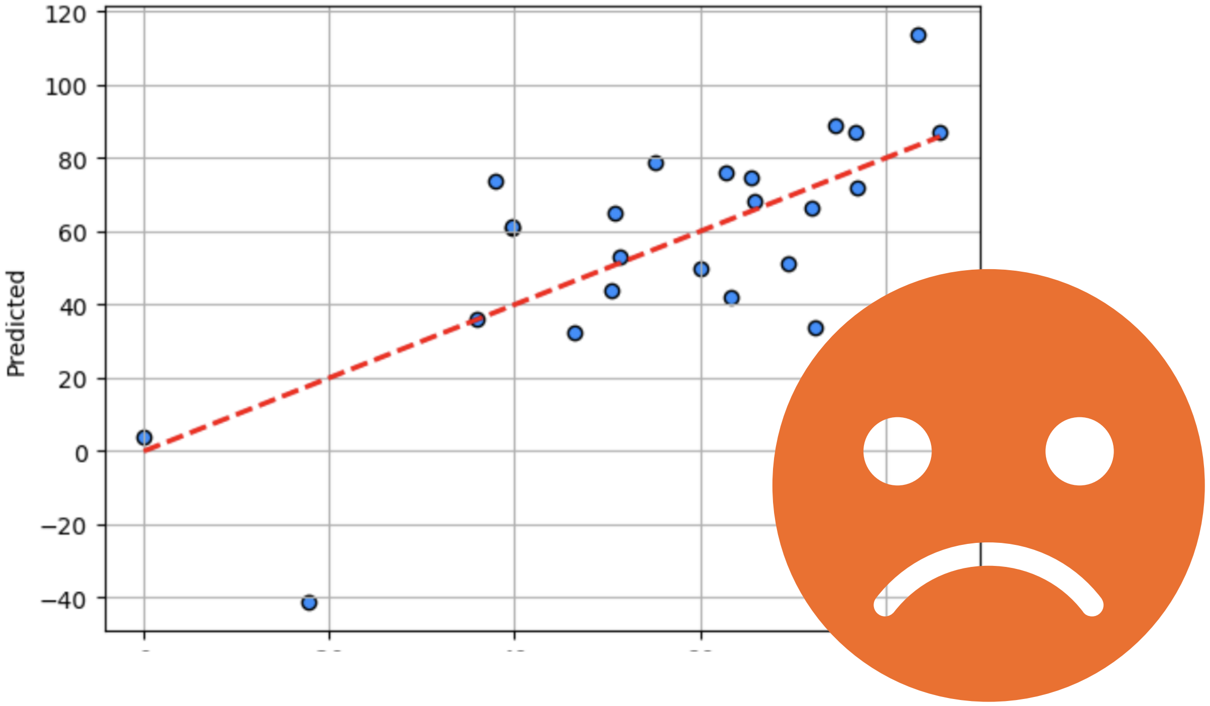
70/30, 80/20 are commonly used strategies in ML

Can I trust?

Train



Test

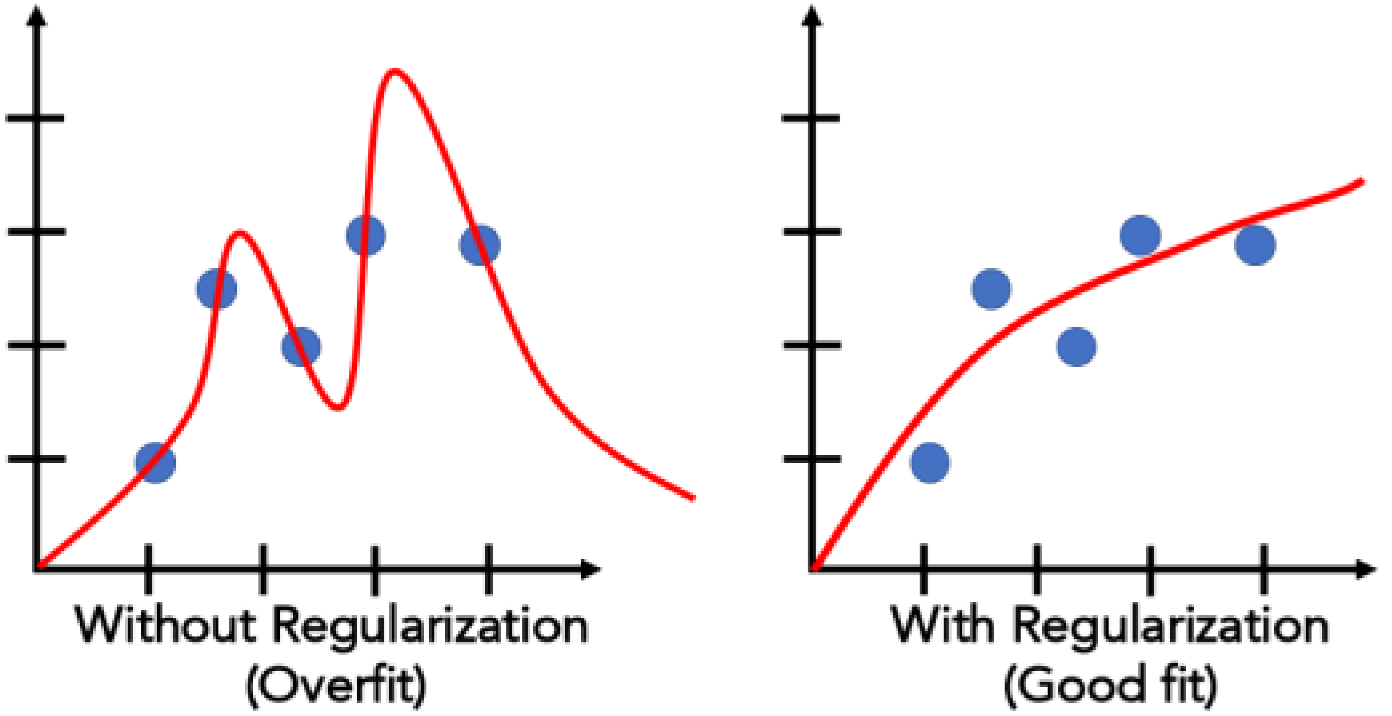


The model is very weak when unseen data presented

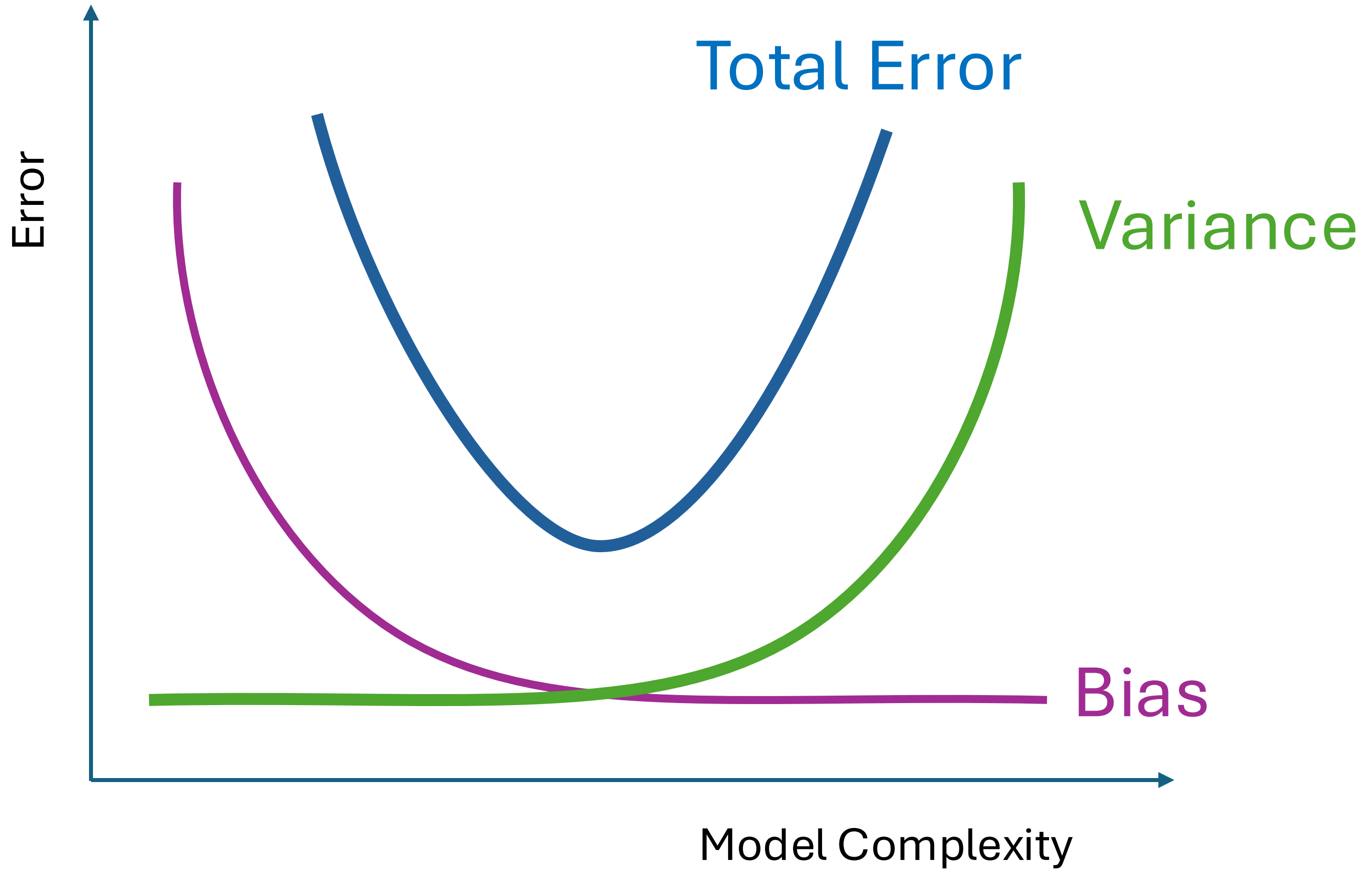
Underfitting/Overfitting

Regularization

Regularization helps stop overfitting by penalizing complex models



Lasso Regression



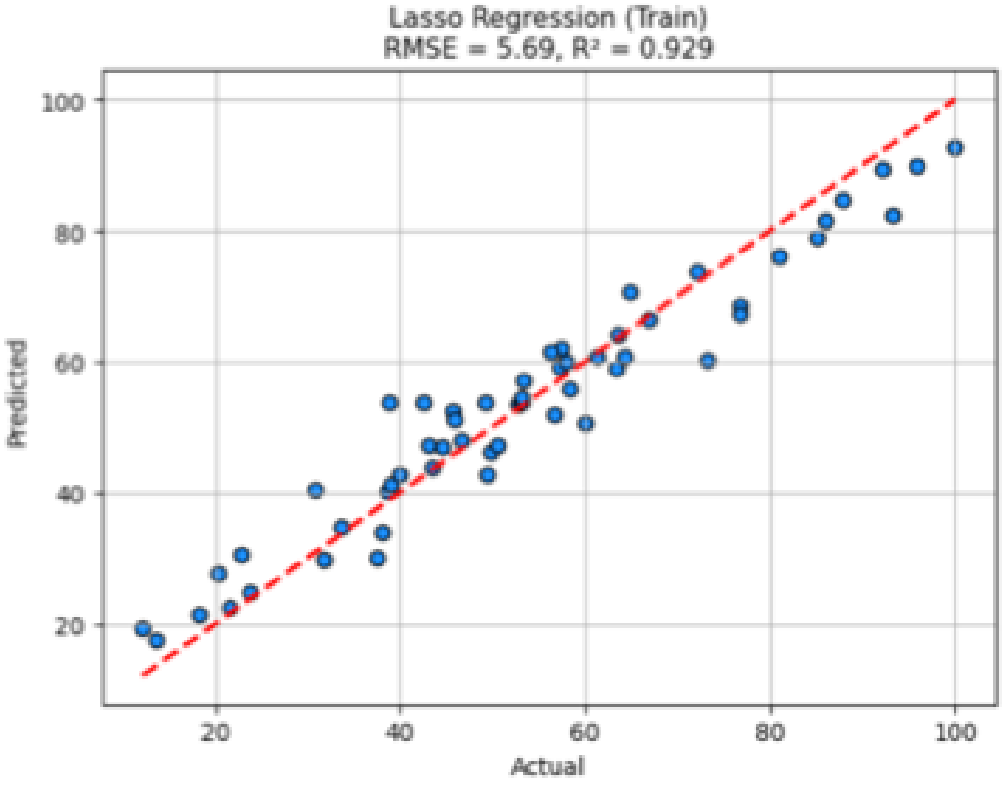
OLS $\min RSS = \Sigma(y_i - \hat{y}_i)^2$

Lasso $RSS + \lambda \times \sum |\beta_i|$

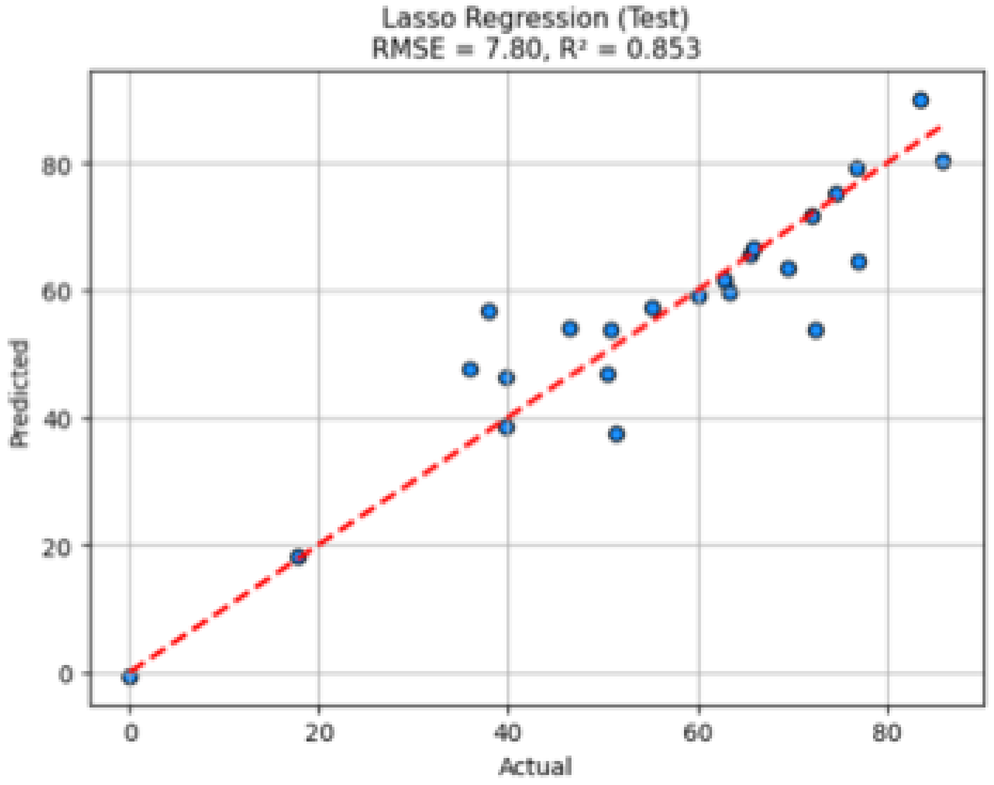
β_i represents the coefficients of the predictors

λ is the tuning parameter that controls the strength of the penalty. As λ increases more coefficients are pushed towards zero

Previous example with Lasso regression



Train



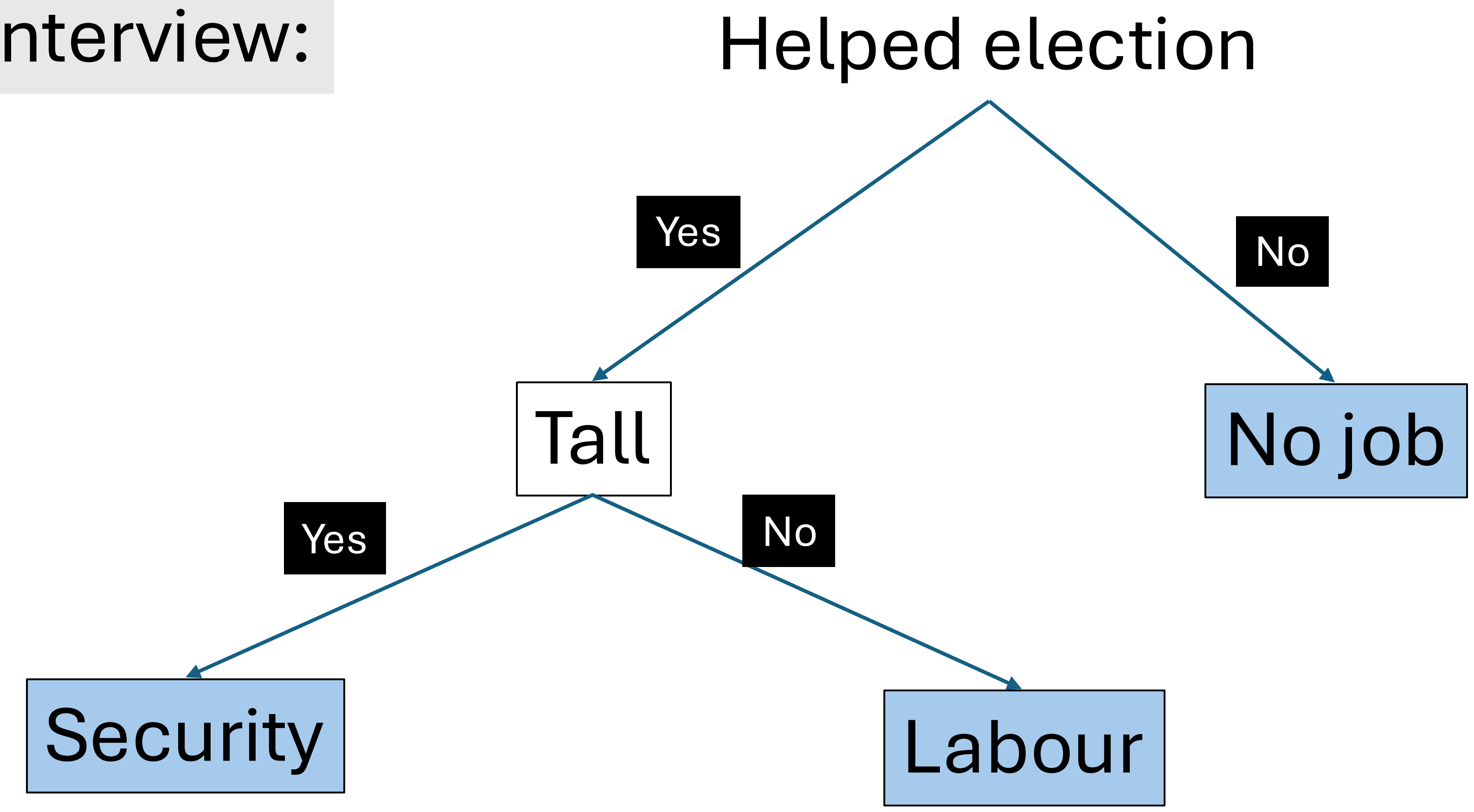
Test

Regularization addressed the overfitting issue in OLS

Feature	Linear Regression	Lasso Regression
X0	17.78	14.30
X1	-2.83	-5.93
X2	7.28	7.02
X3	3.03	2.68
X4	-13.23	-10.47
X5	2.46	0.00
X6	0.92	-0.00
X7	0.44	-0.82
X8	4.24	0.00
X9	4.16	0.00
X10	3.02	0.00
X11	0.62	0.70
X12	-2.13	-0.00
X13	-0.38	0.00
X14	3.85	0.00

Decision Tree

Job Interview:



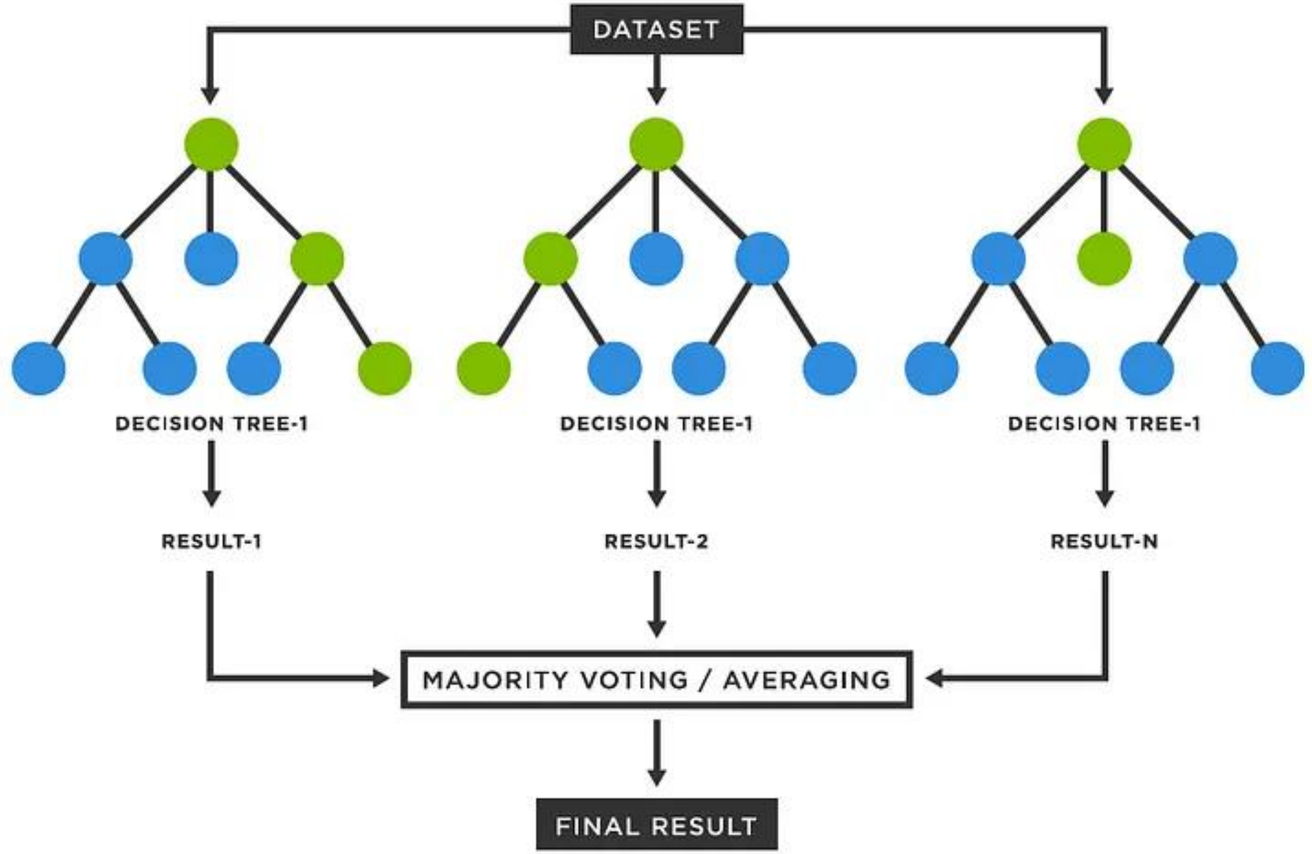
ありがとうございます

Random Forest

Collection of Decision Trees

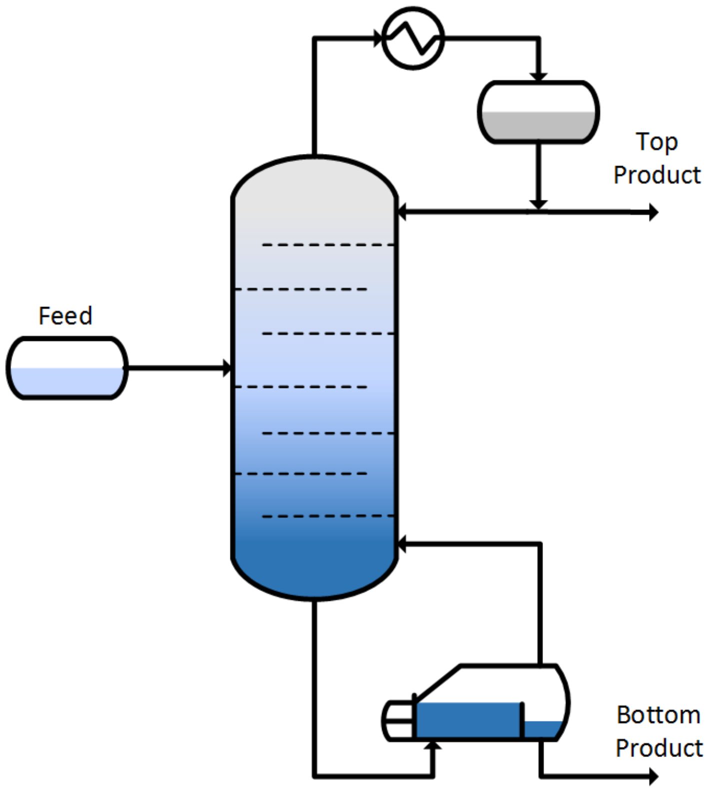
5	9	4
12	8	3
1	7	15

5	9	4
12	9	4
1	7	15



12	15	4
12	8	3
1	7	15

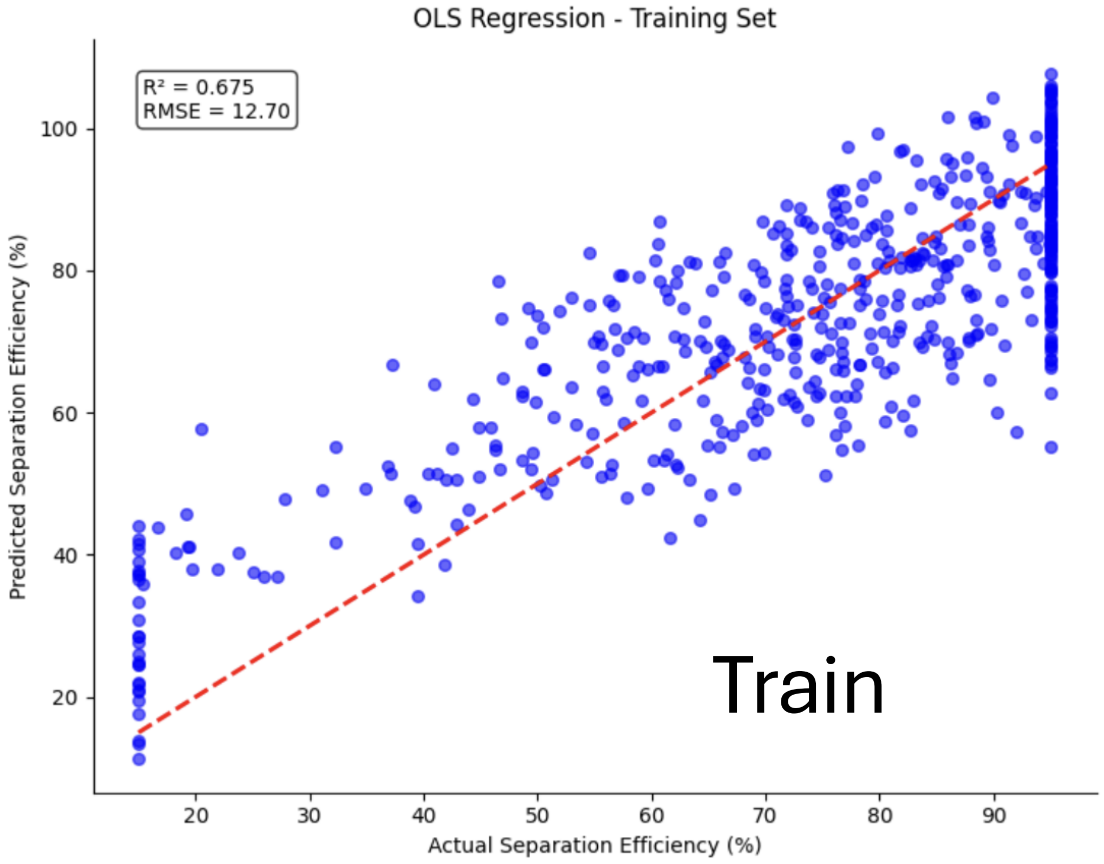
Random Forest Example

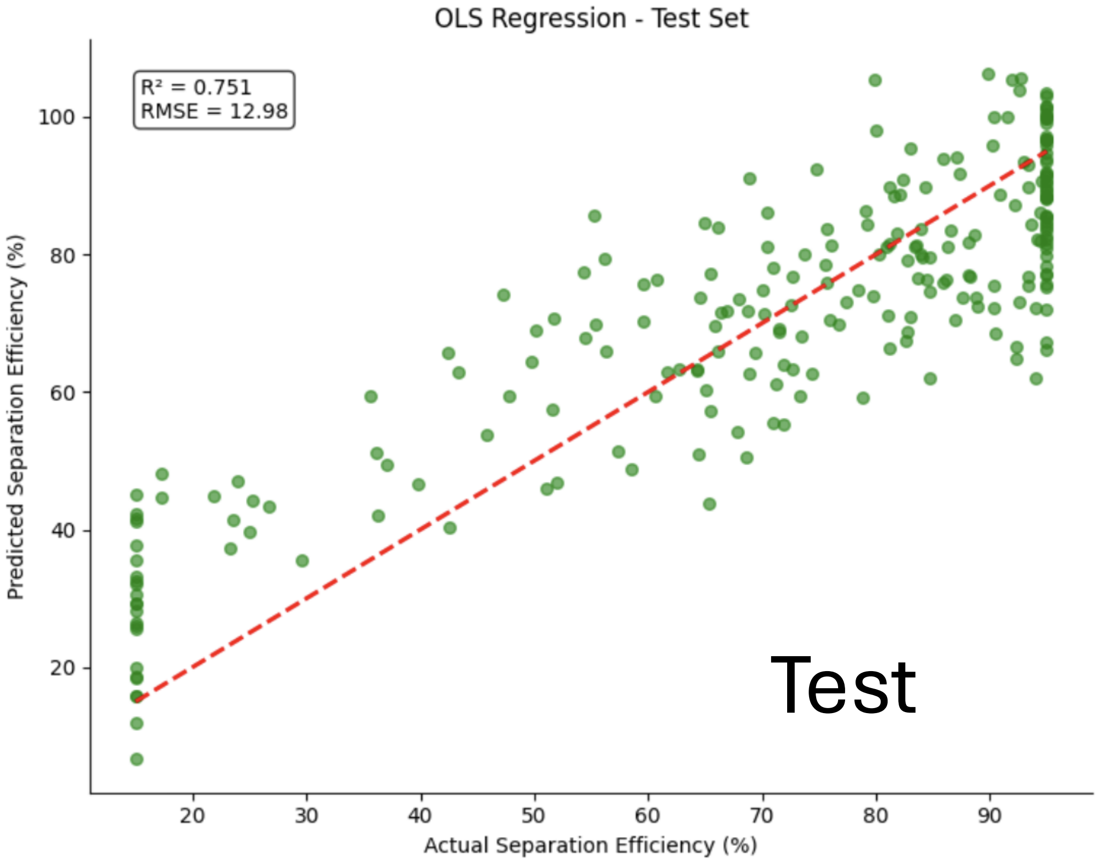


Y=Separation efficiency

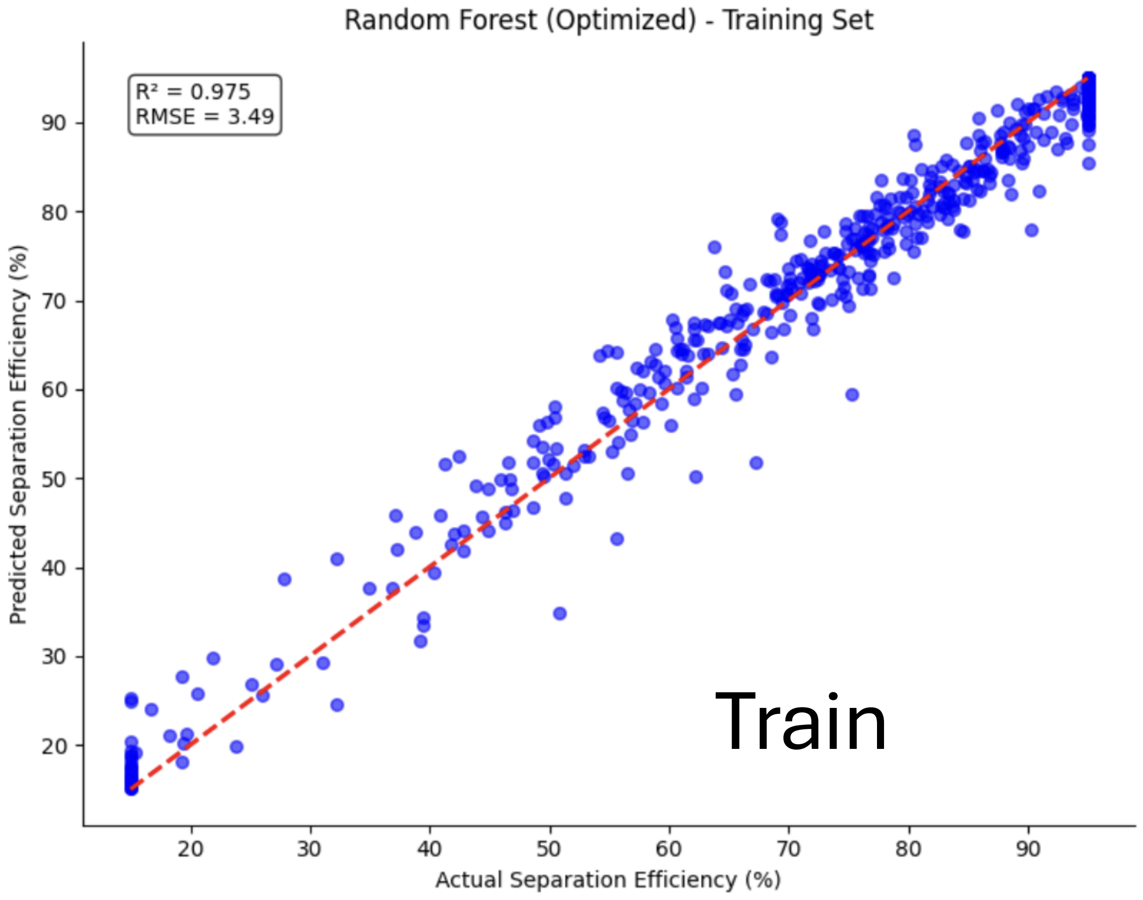
X (features)= [reflux_ratio, feed_temperature, column_pressure, feed_composition vapor_velocity]

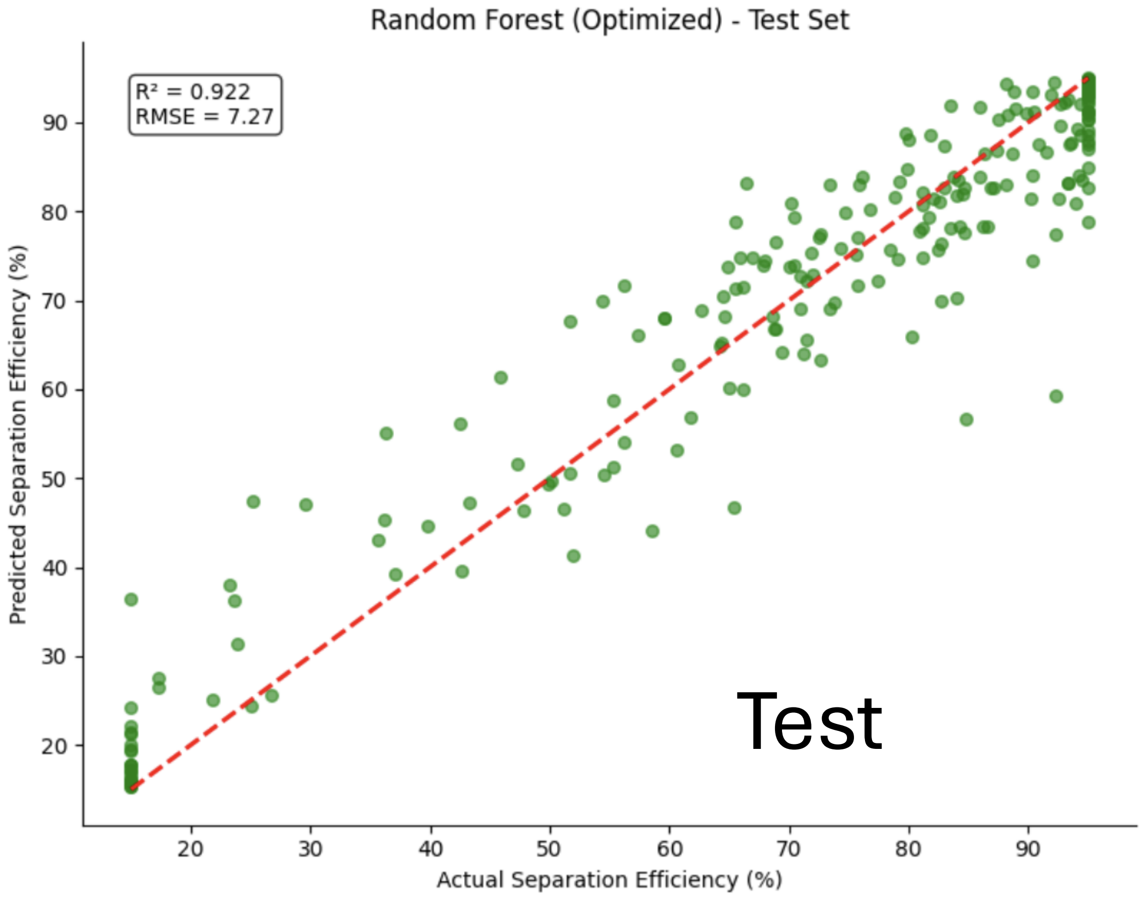
Let's fit OLS


OLS Regression - Training Set
R² = 0.675
RMSE = 12.70
Predicted Separation Efficiency (%)
Actual Separation Efficiency (%)
20
40
60
80
100
20
30
40
50
60
70
80
90
Train


OLS Regression - Test Set
R² = 0.751
RMSE = 12.98
Predicted Separation Efficiency (%)
Actual Separation Efficiency (%)
20
40
60
80
100
20
30
40
50
60
70
80
90
Test

Random Forest Regressor





Can we do even better? Perhaps yes

Rule of thumb

Scenario	Recommended Model(s)	Notes
Linear + Few variables	Linear/Logistic Regression	High accuracy if relationship is linear; highly interpretable
Linear + Many variables	Ridge/Lasso	Handles many predictors; requires tuning
Non-linear + Small data	Random Forest	Robust, good accuracy, moderate interpretability
Non-linear + Large data	**Gradient Boosting**	High accuracy, slower, less interpretable
Very large data + High compute	Neural Networks (deep learning)	Best for complex, big data; low interpretability
Need interpretability	Linear, Random Forest, Gradient Boosting (with SHAP/LIME)	Linear most interpretable; others with tools
Time constraints	Linear > Random Forest > Gradient Boosting > Neural Networks	Linear fastest, Neural Networks slowest

Remember … most of real-life problems are non-linear

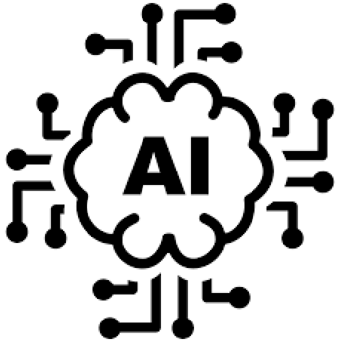
Coding has been a rate limiting factor for AI/ML development..

Hence AI’s penetration has been somewhat limited in Science/Engineering

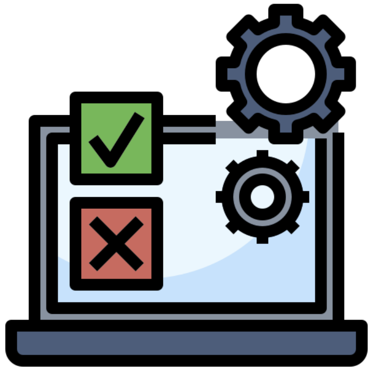
Vibe-Coding

Vibe-Coding (AI-assisted coding)

Describe what you want



AI generating code



Testing and Refining code

Iterate

Toolbox

Favorite

SCAN ME

https://github.com/dissabnd/Applied-AI-in-Chemical-and-Process-Engineering