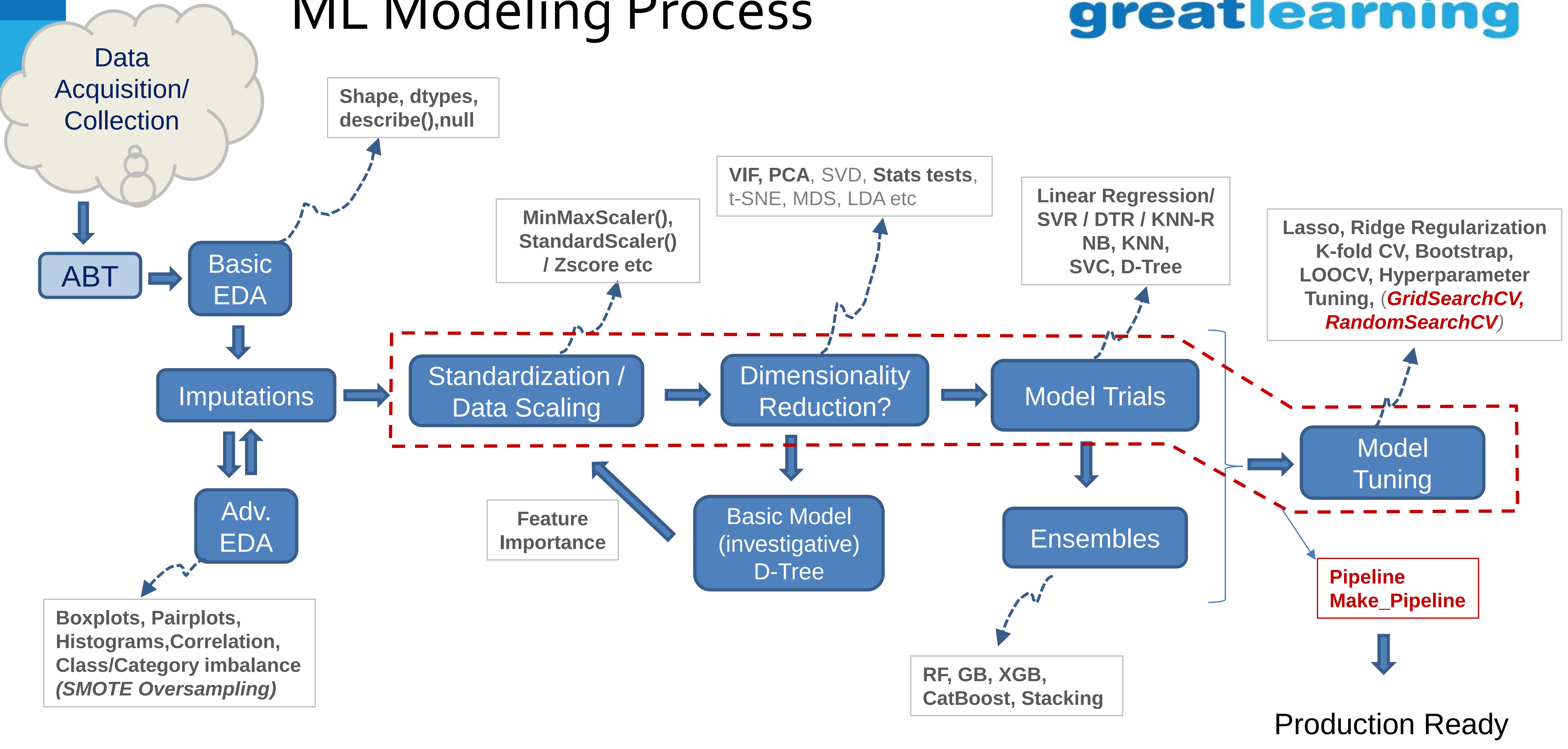
M6W2:

Model Performance Measures, ML Pipeline and Hyperparameter Tuning

Agenda:

- An Overview of **ML modeling process**
- Quick recap of **Classification metrics, SL cheat sheets**
- **Use of a Pipeline object**
- **The Train, Validation and Test sets**
- **Hyperparameter tuning – GridSearchCV and RandomizedSearchCV**
- **Case Studies (3)**
- **Q&A**

ML Modeling Process



Supervised Learning – Cheat Sheet (Prepare your own!)

greatlearning

Model	Type (Regr/ CLF)	Loss-Function	Parameters	Hyper-parameters	Model Performance Metrics
Linear Regression	Regr	$\sum_i (Y_i - \Sigma_j \boldsymbol{\beta}_j X_j - \boldsymbol{\beta}_0)^2 + \boldsymbol{\lambda}_L \Sigma_j \lvert\beta_j\rvert$	**(β_0, β_j)** *(Intercept, coeffs)*	**Lasso/ Ridge** λ_L , λ_R, degree of polynomial **α**, learning rate	R2-score, Adj. R2-score
Logistical Regression	CLF	$\sum_i -\{Y_i * (Log\ \hat{Y}_i) - (1-Y_i) * Log(1-\hat{Y}_i)\}$ $\hat{Y}_i = \frac{1}{1+e^{-(A+\Sigma_j B_j X_{ji})}}$	**(A, B_j)**	Solver: 'liblinear' Penalty type: l1/l2 Penalty param C	**Confusion Matrix, Recall, Precision, F1-Score, ROC- AUC,** K-S, Gain/Lift chart, CR/DR
KNN	CLF/ Regr.	Non-parametric		K (n_neighbors), Dist metric: 'euclidean', 'minkowski' Weights: 'uniform' Algorithm: 'Auto','brute' 'KDTree', 'Ball-Tree'	
NB	CLF	Non-parametric			
SVM	CLF/ Regr.	$\min_{w,b,\varsigma} \frac{1}{2} \lVert w\rVert^2 + C \sum_{i=1\,to\,k} \zeta_i$	**(w,b)**	C, Gamma, Kernel = 'rbf', 'poly','sigmoid'	

Supervised Learning – Cheat Sheet

greatlearning

Model	Type (Regr/ CLF)	Loss-Function	Parameters	Hyper-parameters	Model Performance Metrics
DTree	CLF	Non-parametric		Criterion:′ gini′, ′entropy′ Max_depth, Max_features, Ccp_alpha	**Confusion Matrix, Recall, Precision, F1-Score, ROC- AUC,** K-S, Gain/Lift chart, CR/DR
…	…	….	...	…	….

Topics covered in Week 2

greatlearning

- Summarize & Clear Doubts on Performance measures (esp. Classification)
  - ROC-AUC
- Concept of Pipeline
- Building a Pipeline
- Performance on train vs test data
- Hyperparameter tuning
- Grid Search and Random Search
- Hands-on Exercises

Confusion matrix:

		Predicted	
		1	0
Observed	1	**TP**	**FN**
	0	**FP**	**TN**

$$\textit{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\textit{Recall}, R = \frac{TP}{TP + FN} = Sensitivity = \boldsymbol{TPR}$$

$$\textit{TNR} = \frac{TN}{TN + FP} = Specificity = 1 - \boldsymbol{FPR}$$

$$\textit{Precision}, P = \frac{TP}{TP + FP}$$

$$\textit{F1 Score} = \frac{2 * P * R}{P + R}$$

F1 Score

A single metric is not sufficient for the evaluation of classification models. We have seen that we need to use recall and precision together along with accuracy to evaluate our model.

Let us consider another metric that puts together the recall and precision metrics. We call it F1 Score.

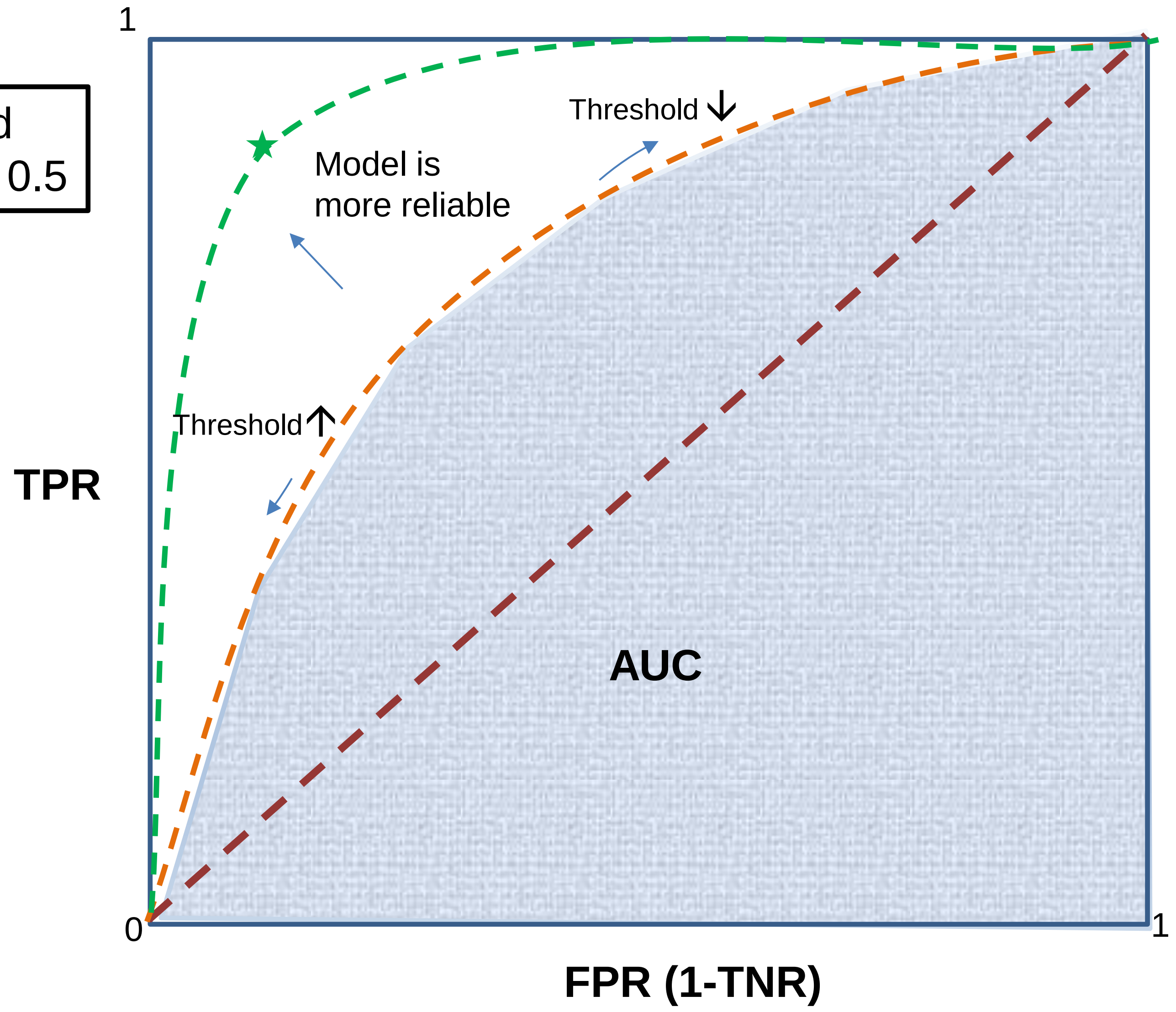
F1 Score = 2(precision*recall)/precision + recall

-which is the harmonic mean of the two metrics.

The F1 score can also be used to evaluate the model.

ROC Curves, AUC and Gini Coefficient

Threshold
Default = 0.5



$$Recall, R = \frac{TP}{TP + FN}$$

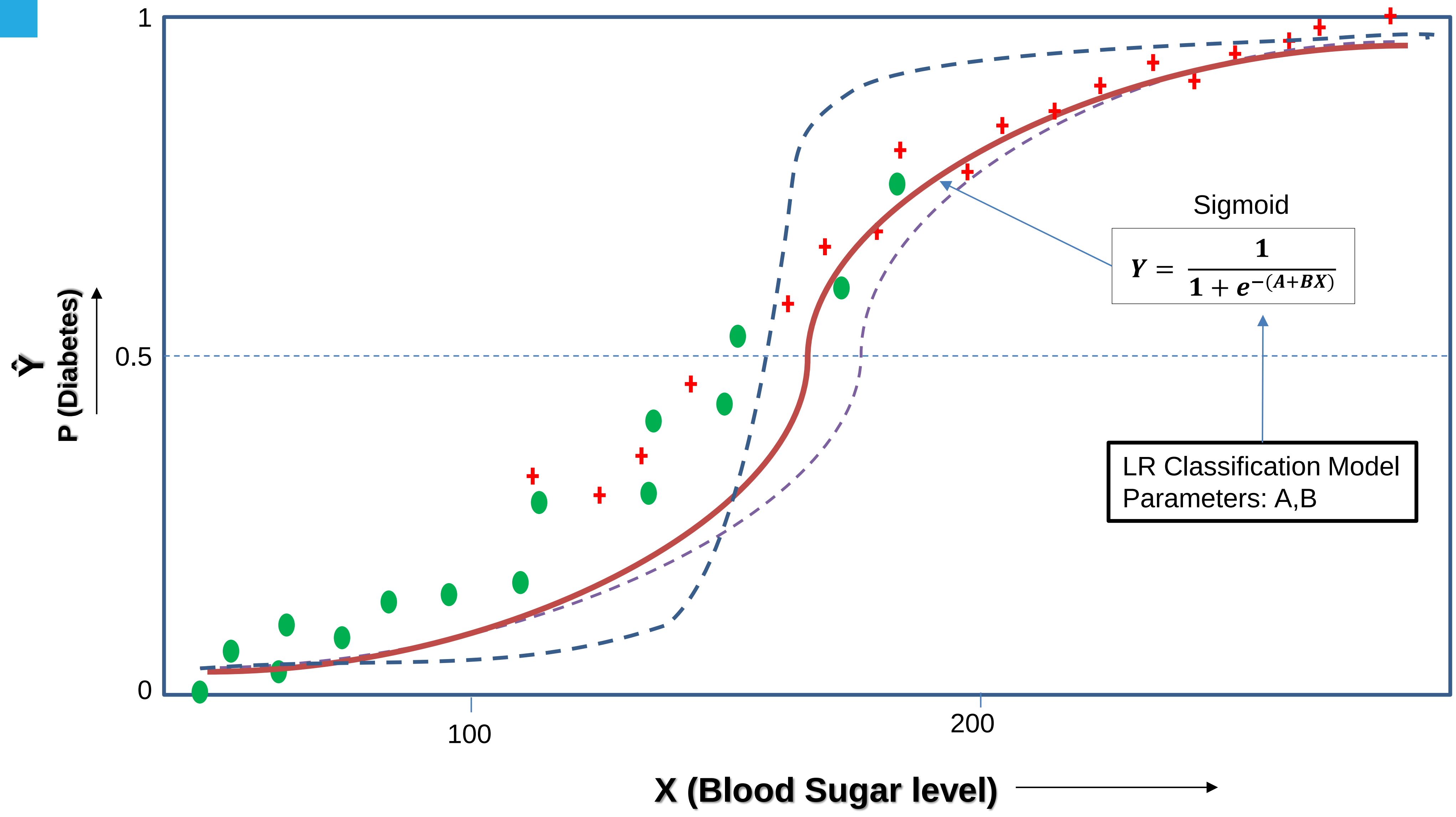
$$= Sensitivity = TPR$$

$$FPR = 1\text{-}TNR = \frac{FP}{TN + FP}$$

$$= 1 - Specificity$$

$$GI = 2*AUC - 1$$

LR Classification Example



ROC and Gini Coefficient &Threshold

- Roc is a curve which allows us to compare models.
- It is plot between TPR(true positive rates) and FPR(false positive ratio).
- The area under the ROC curve (AUC) is a measure of the how good a model is.

Gini Coefficient:

- It is also used to measure the goodness of a fit.
- It is the ratio of areas in a roc curve and is scaled version of the AUC.
- GI = 2*AUC -1

Model Metrics

Case Study: Covid Testing

- **100 Patients** randomly tested using regular RTPCR testing (24-48 hrs for estimation) during 1st wave in NYC, USA, showed **25 positive cases**.

- Several Patients were tested with a Rapid test procedure (**R**apid **A**ntibody **T**esting) and the data was used to build a classification ML model. The probabilities of the above 100 patients for Covid positivity was estimated using the ML model.

- The probabilities of the 100 patients were sorted in descending order in deciles and Tabulated.

Case Study: Covid Testing
(Results from ML Algorithm based on RAT model)

Default Threshold
for Classification

Probability

0.96
0.85
0.69
0.5
0.45
0.39
0.31
0.28
0.23
0.18
0.12

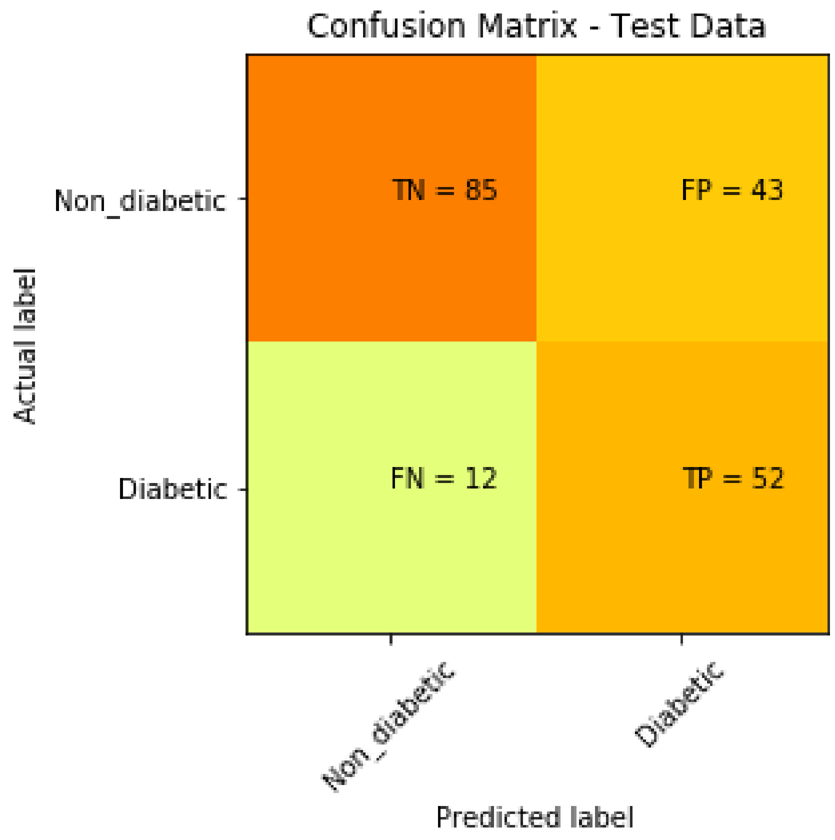
Deciles (sorted by Prob.)	Covid +ve Actuals	Covid –ve Actuals
1	9	1
2	7	3
3	6	4
4	2	8
5	1	9
6	0	10
7	0	10
8	0	10
9	0	10
10	0	10

Pls. refer the excel workbook
Attached in the mentor deck

Classification metrics

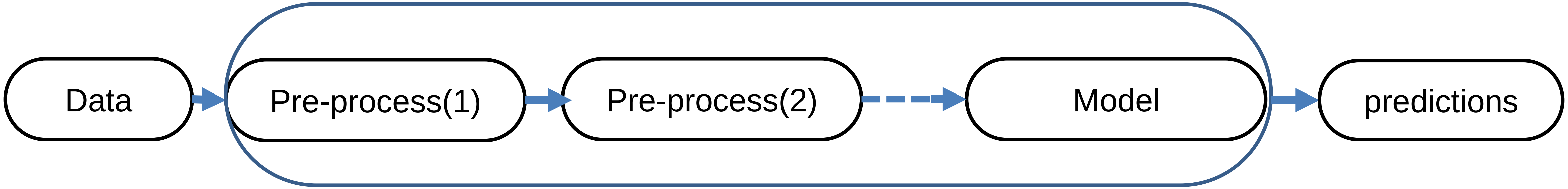
For the given Confusion matrix, what is the F1 score?

65.4%



Need for a Pipeline

- Streamlines the process of transforming data, training an estimator and using it for prediction



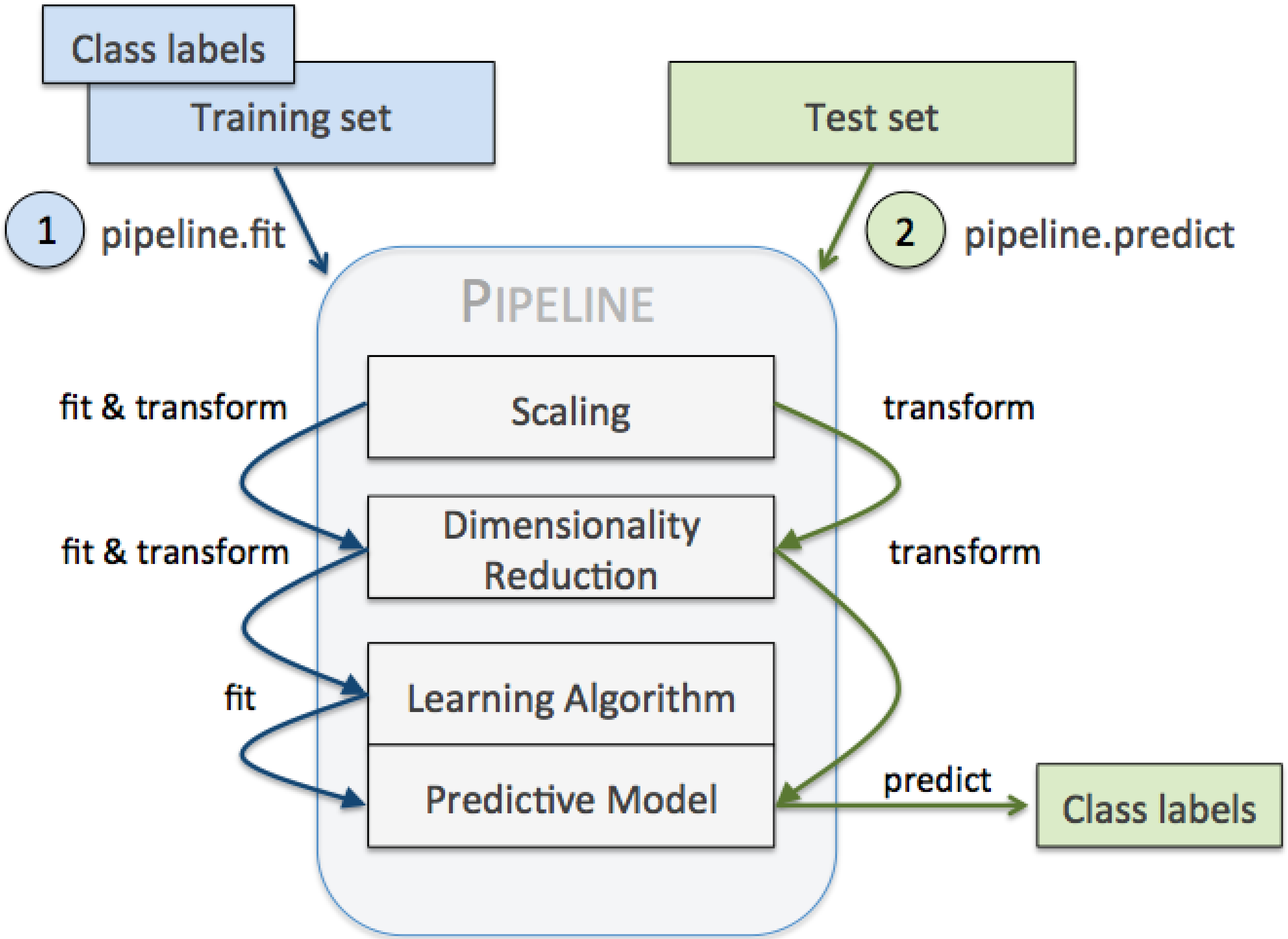
Pipe= Pipeline(steps=[(‘Step1’, Process1()), (‘Step2’, Process2()),….,(‘Model’, CLF(params))])

Examples:

```
Pipe1=Pipeline(steps=[('scaler', StandardScaler()), ('PCA',
PCA(n_components=9)), ('model', SVC(gamma='auto',
random_state=1))])
Pipe2=make_pipeline(StandardScaler(), GaussianNB(priors=None))
```

Pipeline process Train vs Test

greatlearning



Train, Validation and Test sets

greatlearning

- It is a general practice to split our data into three sets
- The train set
  - The data that we use to train the model
- The validation set
  - The data that we use to 'validate' a model
  - Any hyper-parameter tuning that is done, is based on the performance of the model on the validation set
- The test set
  - The data that is used to simulate real unseen data

Why?
Prevent "Data Leak"

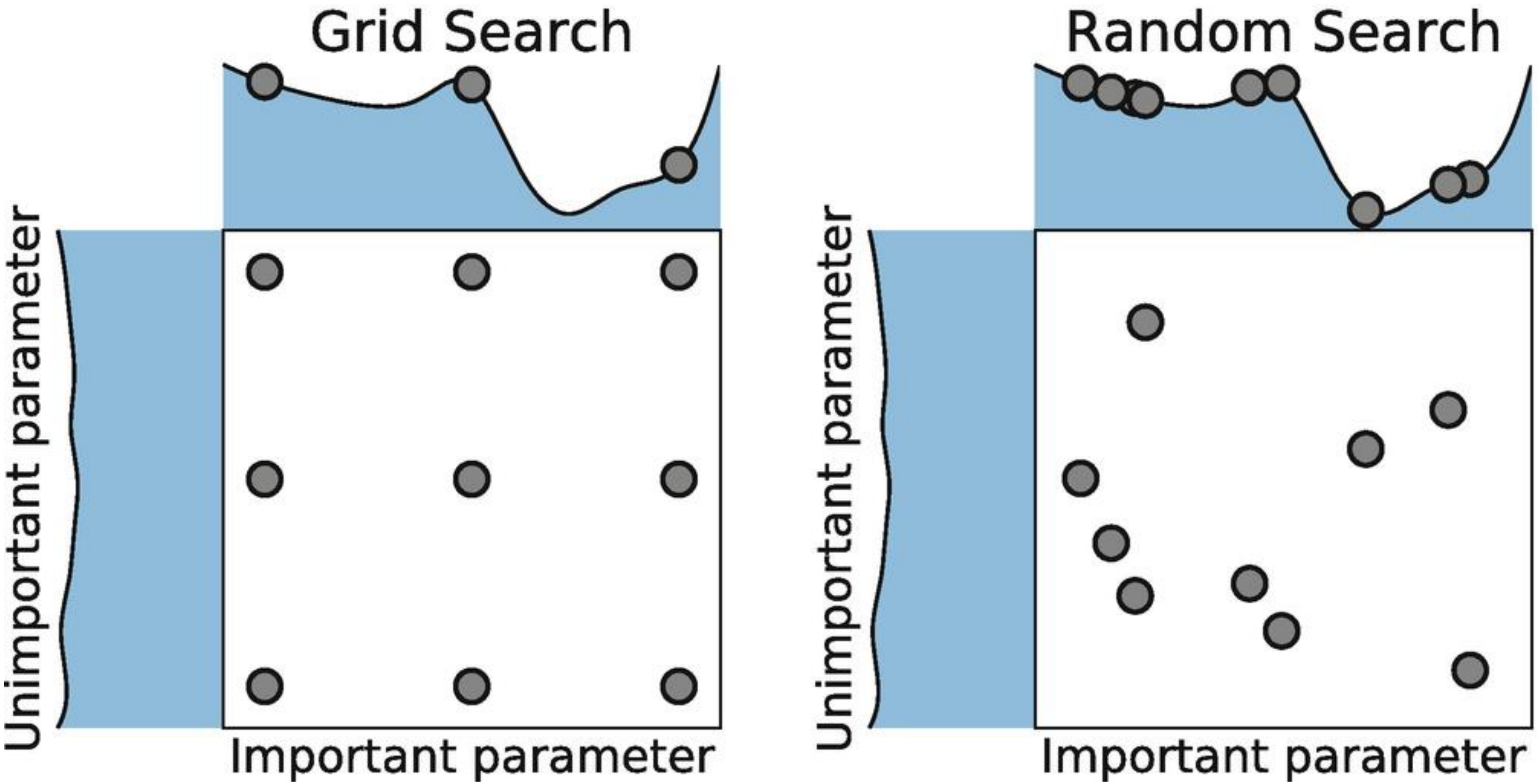
Train, Validation and Test sets

- Always tune the model based on the performance on the validation set, once the model is trained on the Train set
- Never fine-tune a model based on its performance on the test set
- Test set is meant to aid in assessing a model's performance in production before the model hits production

Hyper-parameter tuning

- As opposed to parameters (like the ones in linear regression slope and constant term) which change based on the data for a given parametric model, hyper-parameters are preset values even before a non-parametric model gets trained on the data
- Parameters change during the training process
- Hyper-parameters are preset and do not change while training
- The process of setting the right hyper-parameters to get max performance out of a given model, is called <u>Hyper-parameter Tuning</u>

Grid Search and Random Search

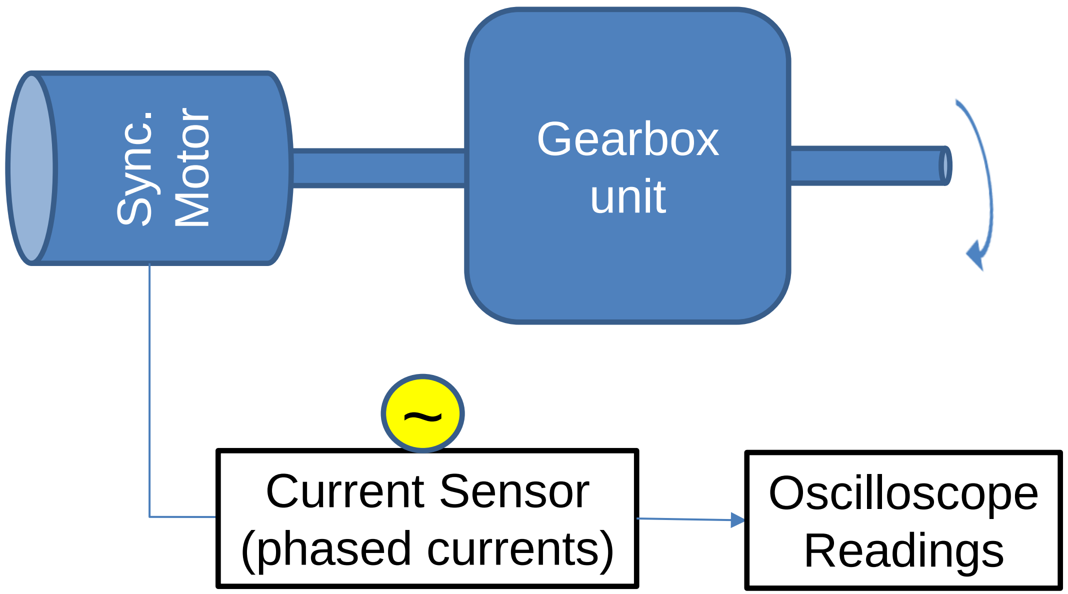


Grid Search and Random Search

- Both are the two most common methods of choosing the right hyper-parameters
- In Grid search, each and every combination of hyper-parameters tested before selecting the 'best' combination of hyper-parameters
- In Random search, only a subset of combinations can be tested before selecting the 'best' combination of hyper-parameters
- We use Random Search when the parameter grid is fairly large and we want to save on processing time
- GridSearchCV and RandomizedSearchCV are included in the sklearn library to perform the same over a parameter grid, that is passed as an argument to the functions along with the estimator

Case study 1

Sensorless diagnosis of Autonomous Electric Drive system



Case study 2

South African Coronary Heart Disease (CHD) Classifier with Model Tuning

Case study 3

Customer Satisfaction survey on flights

References

https://towardsdatascience.com/a-simple-example-of-pipeline-in-machine-learning-with-scikit-learn-e726ffbb6976

https://machinelearningmastery.com/machine-learning-modeling-pipelines/

https://towardsdatascience.com/automated-machine-learning-hyperparameter-tuning-in-python-dfda59b72f8a

https://towardsdatascience.com/gridsearchcv-or-randomsearchcv-5aa4acf5348c

https://towardsdatascience.com/hyperparameters-of-decision-trees-explained-with-visualizations-1a6ef2f67edf

https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/

Q and A

HappyLearning