Model Performance, Evaluation Method and Hyperparameter Tuning

Outline

ML Objective :

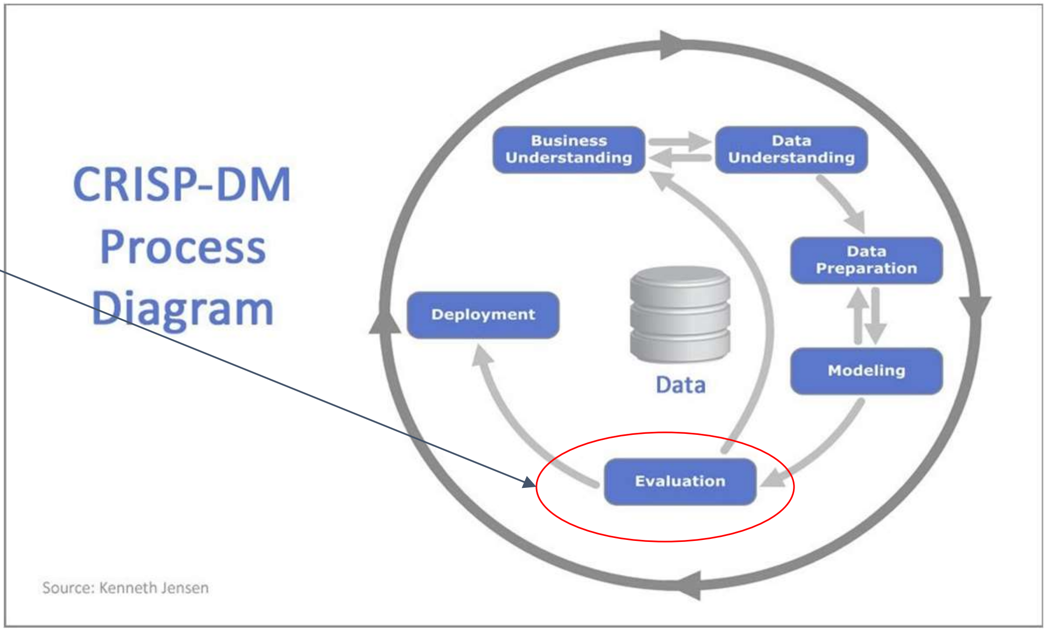
- Classification
- Regression

Evaluation Method :

- Validation and Cross Validation

Hyperparameter Tuning

Algorithm Chains



Source: Kenneth Jensen

Performance Metrics

We need to assess the performance of a Machine Learning Model because it is only prediction and prediction can be wrong

To measure performance, we use some metrics

we already used accuracy for classification problem and R-square or mse for regression problem

We will explore deeper about performance metrics in this section

Classification Performance Metrics

Commonly used:

- Accuracy
- Recall
- Precision
- F1-score
- ROC/AUC
- PR-Curve

Confusion Matrix

Actual	Predicted	
	N	P
N	TN	FP
P	FN	TP

Correct Prediction

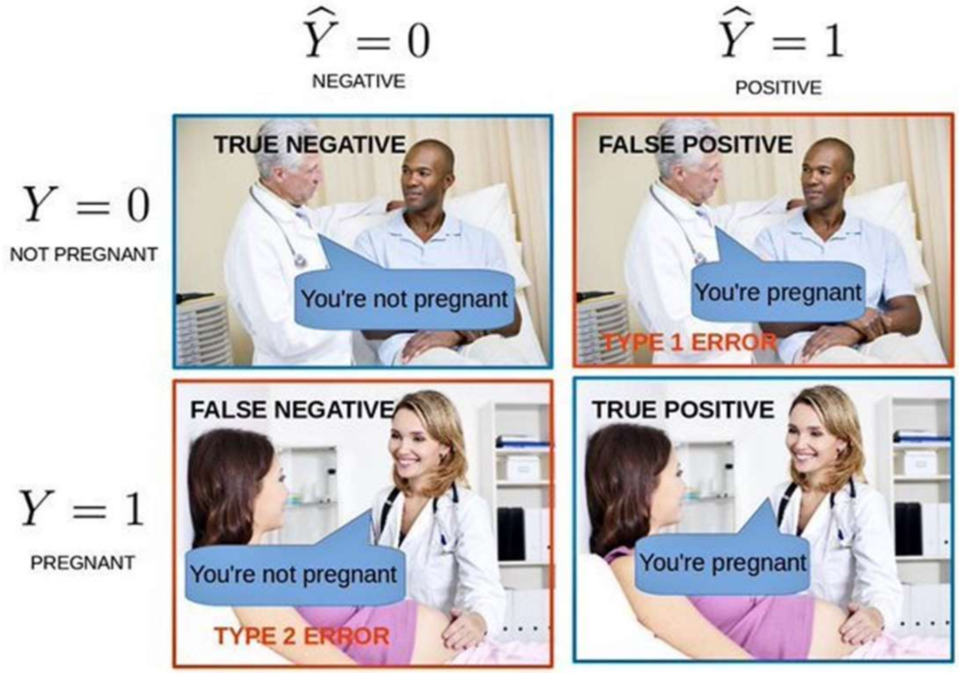
False Prediction

Actual	Prediction	
	Good (N)	Bad (P)
Good (N)	1000	30
Bad (P)	20	100

In credit scoring case :
Bad Usually considered as Positive class while Good as negative class

TP	:	True Positive
TN	:	True Negative
FP	:	False Positive
FN	:	False Negative

Confusion Matrix

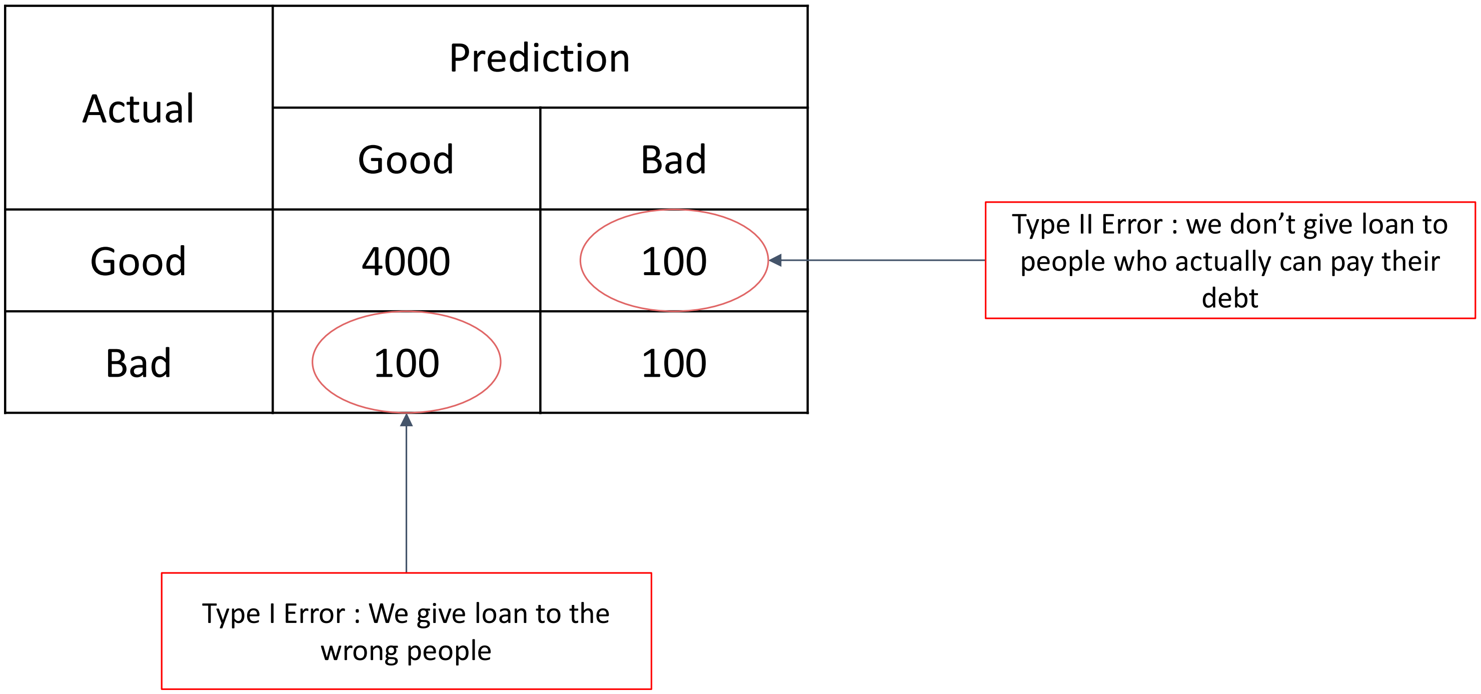


ML Objective

What is your objective :

- Should be about minimize or maximize certain phenomenon
- If can represent money, it would be better

ML Objective : Credit Scoring

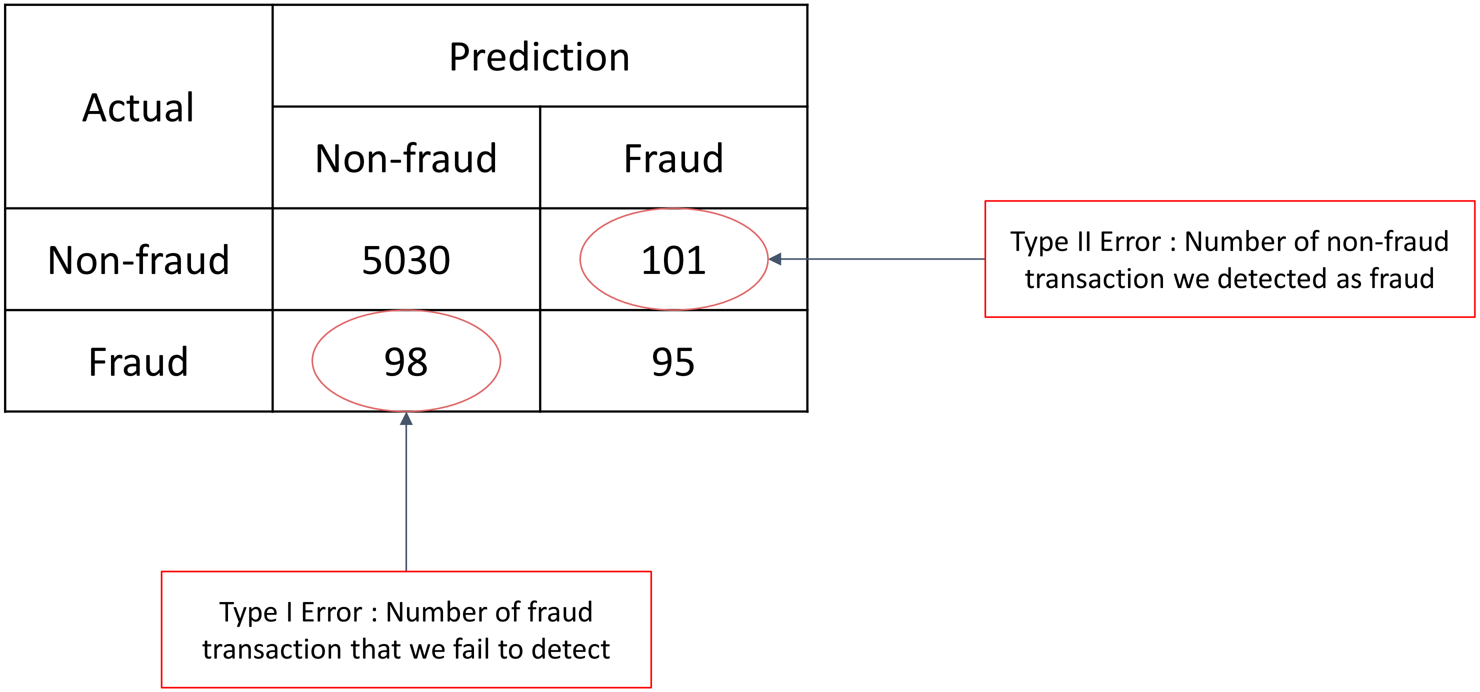


Actual	Prediction	
	Good	Bad
Good	4000	100
Bad	100	100

Type II Error : we don’t give loan to people who actually can pay their debt

Type I Error : We give loan to the wrong people

ML Objective : Fraudulent Credit Card



Actual	Prediction	
	Non-fraud	Fraud
Non-fraud	5030	101
Fraud	98	95

Type II Error : Number of non-fraud transaction we detected as fraud

Type I Error : Number of fraud transaction that we fail to detect

Accuracy

Actual	Prediction	
	N	P
N	TN	FP
P	FN	TP

Accuracy = (TP + TN) / (TP + TN + FP + FN)

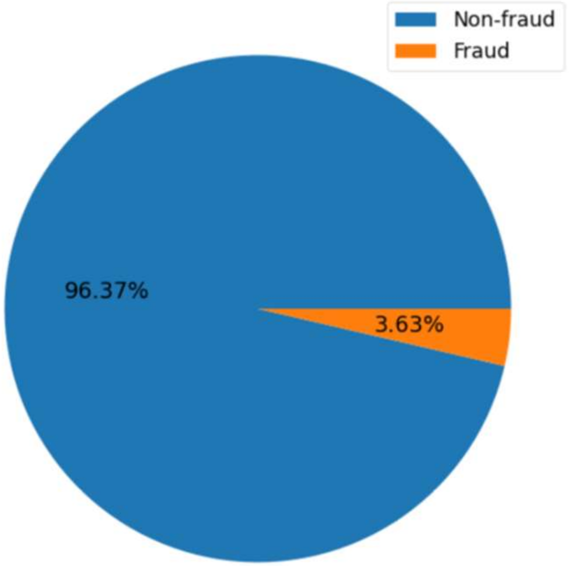
We are interested in both classes

Percentage of the Correct Prediction:

(correct prediction / prediction) x 100%

Accuracy Problem

Actual	Prediction	
	Non-fraud	Fraud
Non-fraud	5030	101
Fraud	98	95



Accuracy = (95 + 5030) / (95 + 5030 + 98 + 101) = 0.9626

- accuracy 96.26 % is meaningless
- in this case we want to detect fraud transaction as much as possible and the model only able to detect below 50%

Recall(P) / Sensitivity

Actual	Prediction	
	N	P
N	TN	FP
P	FN	TP

Recall(P) = TP / (TP + FN)

We are interested in positive class only

Completeness:
You want to put all guilty person in prison

Ex. we want to minimize:

1. Number loan given to the wrong people
2. Number of fraud transaction that we fail to detect
3. etc

Precision(P)

Actual	Prediction	
	N	P
N	TN	FP
P	FN	TP

Precision(P) = TP / (TP + FP)

We are interested in positive class only

Exactness:
You make sure that people that you put in prison are guilty

Ex. we want to minimize:

1. Number of people that actually can pay but we didn’t give loan
2. Number of non-fraud transaction we detected as fraud
3. etc

F1-Score (P)

Actual	Prediction	
	N	P
N	TN	FP
P	FN	TP

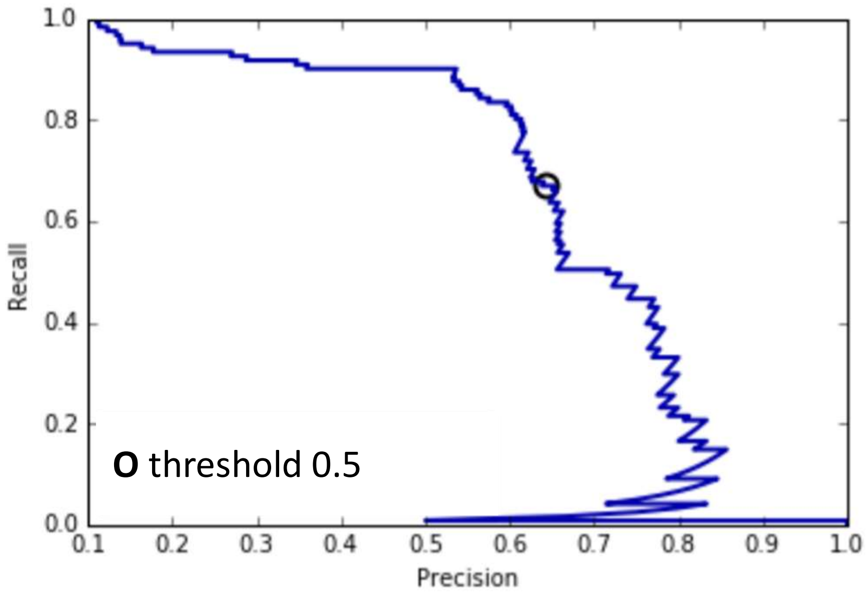
Combination of exactness and completeness:
You want to put all guilty in prison
and
You make sure that people that you put in prison are guilty

F1(P) = 2 x (Precision x Recall)/(Precision + Recall)

We are interested in positive class only

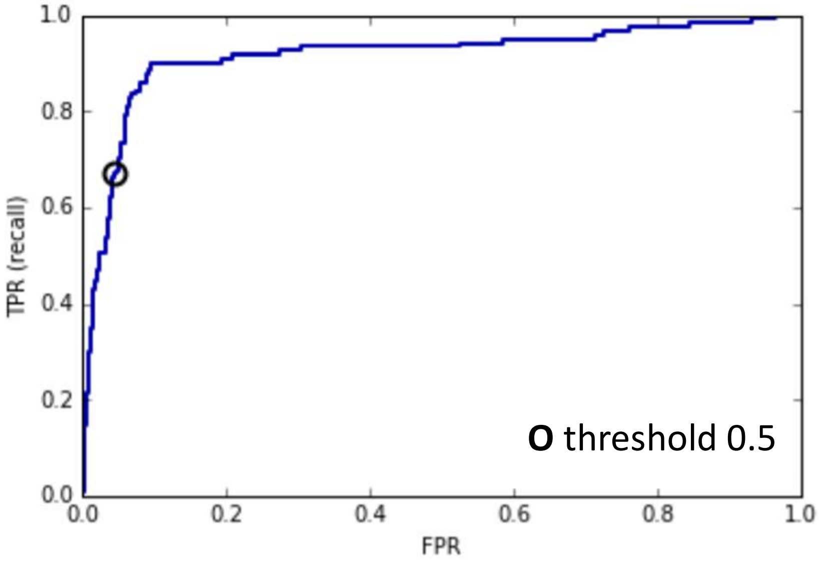
Precision - Recall Curve

- There is always a trade-off between recall and precision if we adjust threshold for classification
- adjusting the the threshold depend on the business goal
- we can see all possible threshold performance using precision - recall curve
- summarize all possible threshold using average precision
- Suitable when we are interested in probability and we want to focus on one of the class

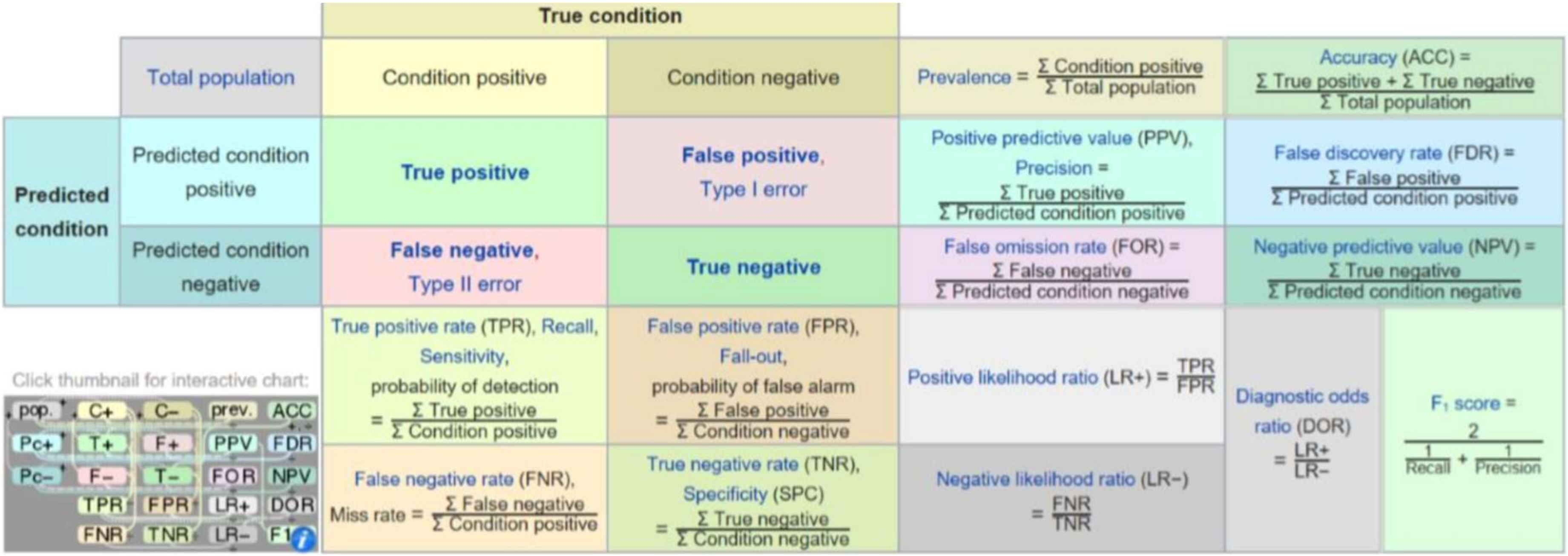


ROC Curve

- There is always a trade-off between false positive rate (FPR)/(1-recall(N)) and true positive rate (TPR)
  - FPR = 1 - recall(N)
  - TPR = recall(P)
- adjusting the the threshold depend on the business goal
- we can see all possible threshold performance using receiver operating operator (ROC)
- summarize all possible threshold using area under curve (AUC)
- Suitable when we are interested in probability and we want to focus on both classes



More Complete Classification Performance Metrics



<table>
<tr><td></td><td></td><td colspan="2"><b>True condition</b></td><td></td><td></td></tr>
<tr><td></td><td>Total population</td><td>Condition positive</td><td>Condition negative</td><td>Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$</td><td colspan="2">Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$</td></tr>
<tr><td rowspan="2"><b>Predicted condition</b></td><td>Predicted condition positive</td><td><b>True positive</b></td><td><b>False positive</b>,
Type I error</td><td>Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$</td><td colspan="2">False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$</td></tr>
<tr><td>Predicted condition negative</td><td><b>False negative</b>,
Type II error</td><td><b>True negative</b></td><td>False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$</td><td colspan="2">Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$</td></tr>
<tr><td colspan="2" rowspan="2">Click thumbnail for interactive chart:</td><td>True positive rate (TPR), Recall, Sensitivity, probability of detection $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$</td><td>False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$</td><td>Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$</td><td rowspan="2">Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$</td><td rowspan="2">$\text{F}_1$ score = $\frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$</td></tr>
<tr><td>False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$</td><td>True negative rate (TNR), Specificity (SPC) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$</td><td>Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$</td></tr>
</table>

Regression Performance Metrics

Commonly used:

- Root Mean Squared Error (rmse) or Mean Squared Error (mse)
- R-square
- Mean Absolute Error (mae)
- Mean Squared Percentage Error (mspe)
- Mean Absolute Percentage Error (mape)
- Mean Squared Log Error (msle)

MSE, RMSE and R-Square

Y	Y-Pred	Residuals
12	13	-1
14	12	2
10	11	-1
11	12	1
15	15	0
16	17	-1
17	18	-1
…	…	…
14	12	2

Residual = Y - Y-pred

When to use ?

1. Small variance in Y
2. No outliers or very few outliers in residuals

RMSE is very sensitive to outlier

$$RMSE = \sqrt{MSE} = \sqrt{\frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n}}$$

Mean Absolute Error

Y	Y-Pred	Y-Y-pred
10	13	-3
14	10	4
16	11	5
12	11	1
15	15	0
16	17	-1
17	18	-1
…	…	…
14	32	-16

Outlier

Residual = Y - Y-pred

When to use ?

1. Small variance in Y
2. Too many outliers exists in residuals

$$MAE = \frac{\sum_{i=1}^{n} |Y_i - \hat{Y}_i|}{n}$$

Root Mean Squared Percentage Error

Same residual value

Relatively different in %

Y	Y-Pred	Residuals	Residuals(%)
1000	1050	-50	-5%
250	300	-50	-20%
1000	900	100	10%
110	120	-10	-9%
770	870	-100	-13%
760	780	-20	-3%
…	…	…	…
650	560	90	14%
10	50	-40	-400%

But be careful when this happens

Relative error/Residual(%) = (Y - Y-pred)/Y

When to use ?

1. You are more interested in relative error
2. No outliers or very few outliers in absolute residuals

RMSPE is also very sensitive to outlier

$$RMSPE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \hat{Y}_i}{Y_i}\right)^2}$$

Mean Absolute Percentage Error

Y	Y-Pred	Residuals	Residuals(%)
1000	1050	-50	-5%
100	150	-50	-50%
1000	900	100	10%
110	120	-10	-9%
150	150	0	0%
770	870	-100	-13%
760	780	-20	-3%
…	…	…	…
650	350	300	46%

Relative error/Residual(%) = (Y - Y-pred)/Y

When to use ?

1. You are interested in relative error
2. Too many outliers in relative error

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \hat{Y}_i}{Y_i}\right|$$

Outlier

Root Mean Squared Log Error

Y	Y-Pred	Residuals	Residuals(log)
1000	1050	-50	-5
130	150	-20	-14
1000	900	100	11
110	120	-10	-9
150	150	0	0
770	870	-100	-12
760	780	-20	-3
...	...	...	
0	10		undefined
1	-12		undefined

But be careful when this happens: we can't compute value of and log -1 or below and we log 0 is infinite, so we need to add any constant (c) so the value become greater than zero

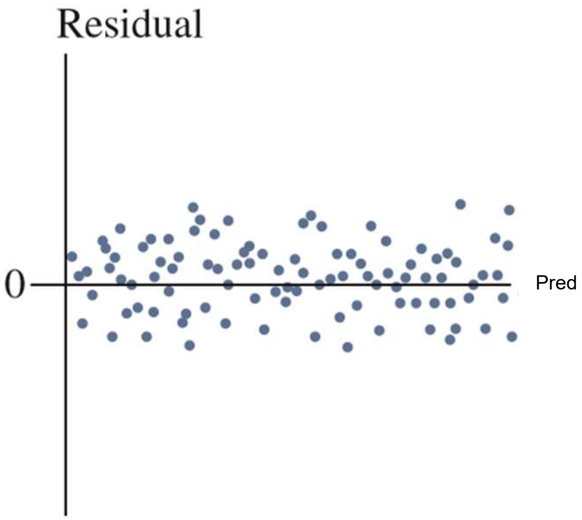
Residual(log) = log(Y + c) - log(Y-pred + c)

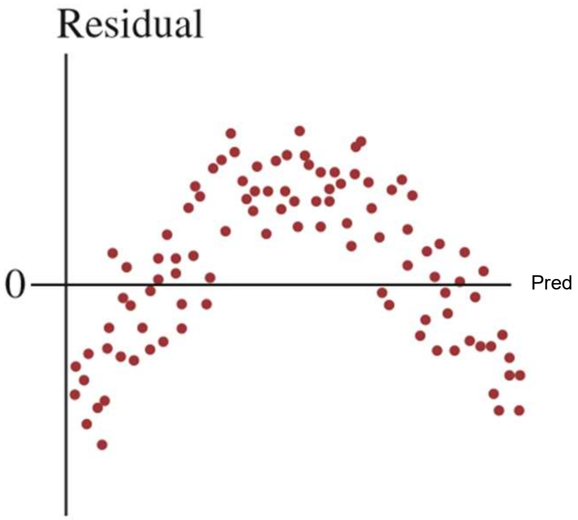
RMSPE and MAPE are used if we are interested in relative error but the drawback of these method is tend to represent prediction result when the actual value are smaller. The prediction will be more accurate only for observation that has smaller actual value. The alternative metrics for this drawback is RMSLE.

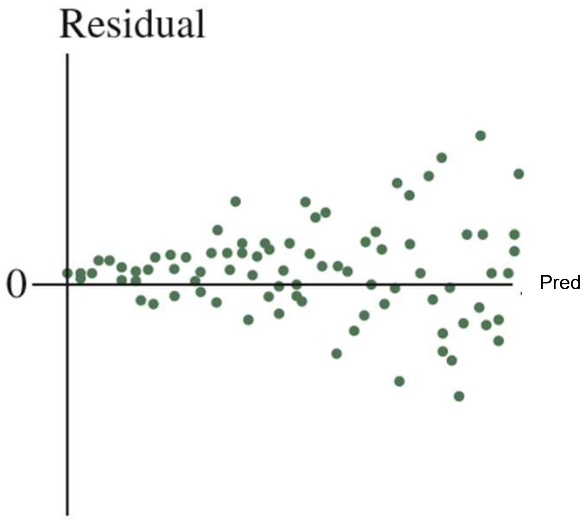
When to use ?
Y distribution is highly right skewed (usually from 0 to very high 1000, 10000, etc)

$$RMSLE = \sqrt{\frac{\sum_{i=1}^{n}(log(Y_i + c) - log(\hat{Y}_i + c))^2}{n}}$$

Residual Analysis







Evaluation Method

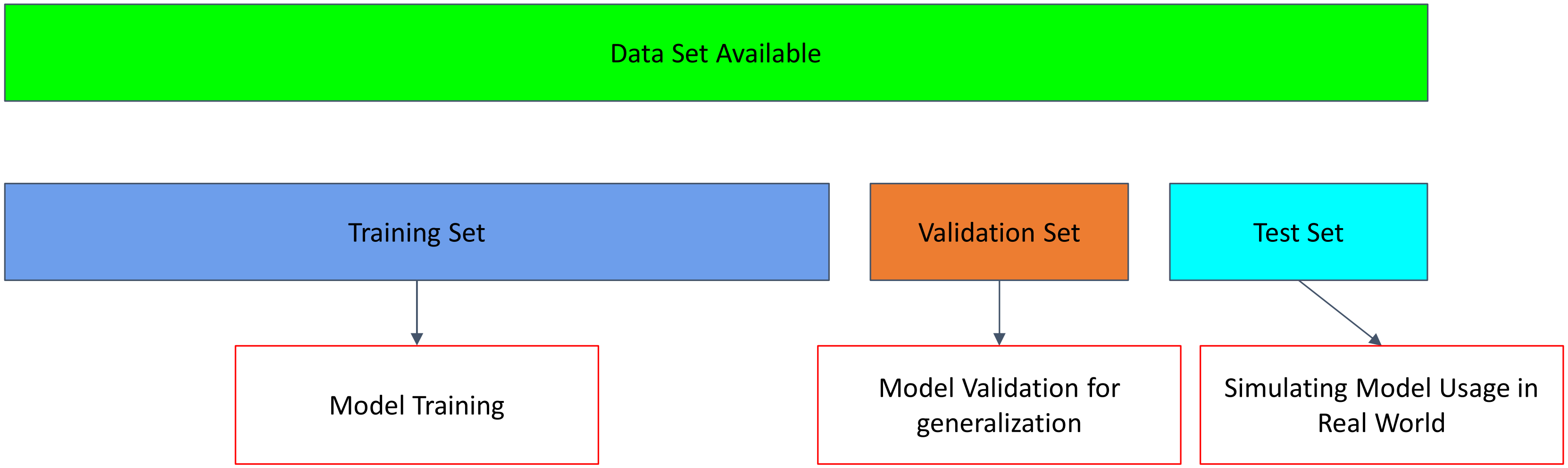
Evaluation Method

The main idea of evaluation method is we want to get model that is able to generalize as accurately as possible

Some terminologies needed to know:

- Training Set
- Validation Set
- Test Set

Validation


Data Set Available
Training Set
Validation Set
Test Set
Model Training
Model Validation for generalization
Simulating Model Usage in Real World

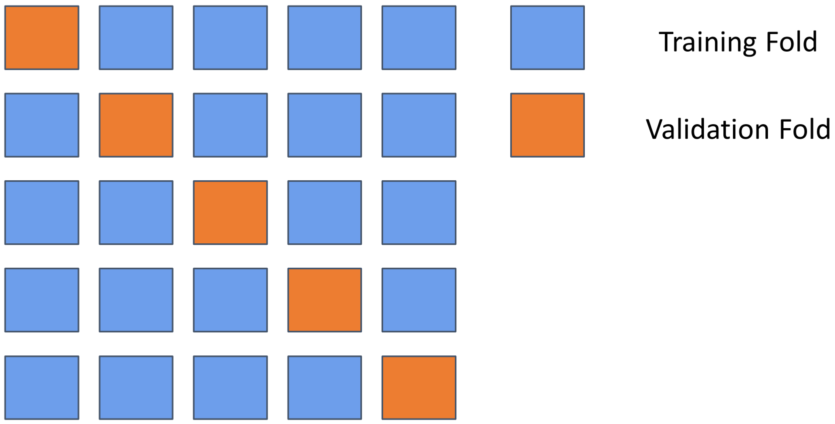
Complete dataset randomly divided into training set, validation set, and test set

Cross Validation

Remainder :

- In supervised learning, we build model on the training data (seen data) and then be able to make accurate predictions on new data (unseen data)
- if a model is able to make accurate predictions on the unseen data, we say it is able to generalize from the training set to the test set.
- In Cross Validation we generalize the unseen data multiple times and then combine them so we can get more robust result

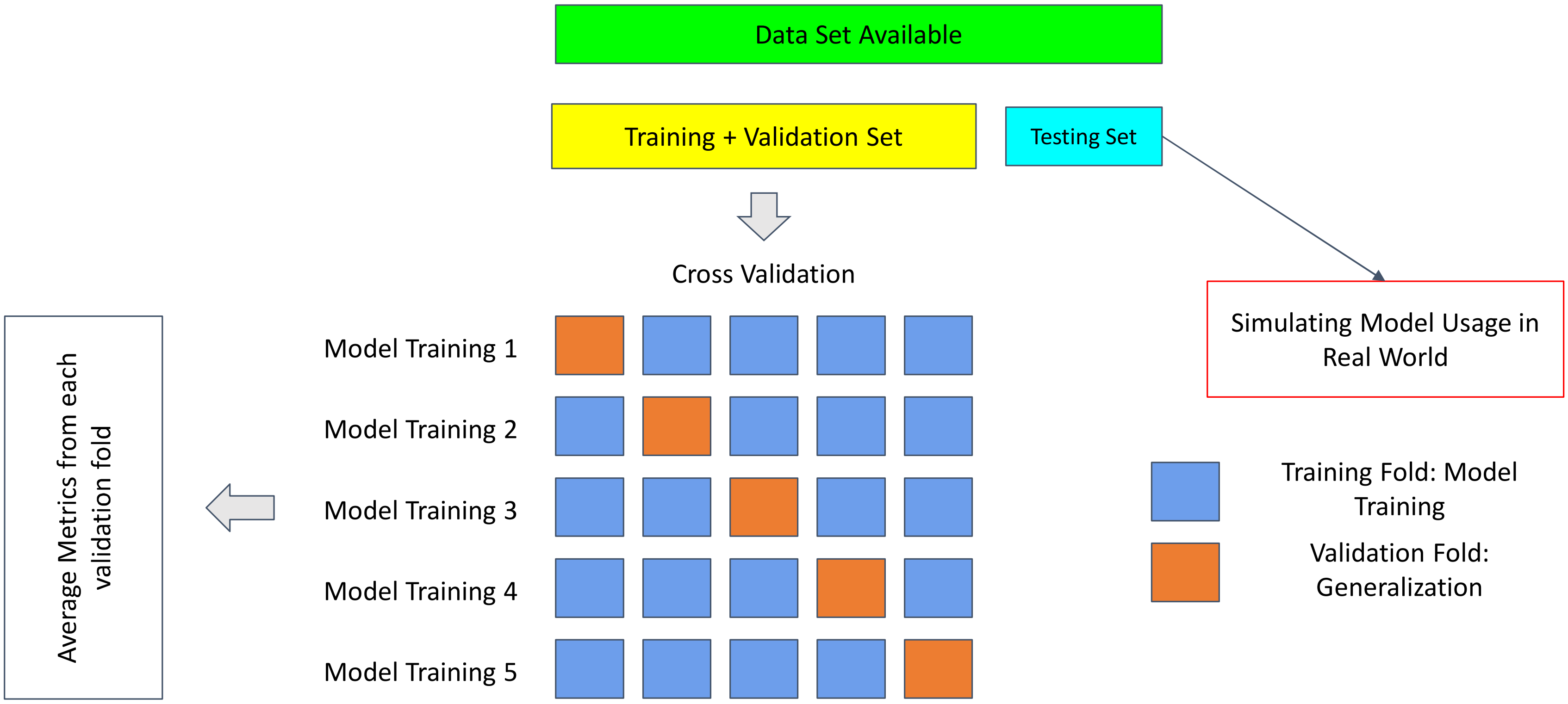
5-fold Cross Validation



Some commonly used version of cross-validation:

- k-fold cross-validation : ex. 2 fold, 5 fold, 10 fold
- Leave One Out Cross-Validation (LOOCV)

Cross Validation



Data Set Available

Training + Validation Set

Testing Set

Cross Validation

Simulating Model Usage in Real World

Model Training 1

Model Training 2

Model Training 3

Model Training 4

Model Training 5

Average Metrics from each validation fold

🟦 Training Fold: Model Training

🟧 Validation Fold: Generalization

Why do we need cross validation ?

- We might get lucky when randomly splitting the data only once because the test set contain “easy” example.
- We might also get unlucky.
- We can see the worst case and best case scenario
- We can see how stable our model

The main disadvantages of cross-validation is increasing computational cost because when we evaluate a model we need to train k times

Cross-Validation is not a way to build a model but only to evaluate how well a given algorithm will generalize unseen data.

Hyperparameter Tuning

Hyperparameter Tuning

- using validation or cross-validation we can measure how well a model can generalize and then compare it with another model.
- using validation or cross-validation we can also improve the model's generalization performance by tuning its hyperparameter
  - Tree : max depth, min samples leaf, min samples split etc
  - Lasso and Ridge : alpha
  - KNN : nearest neighbour
  - etc

Hyperparameter Tuning in Scikit-Learn

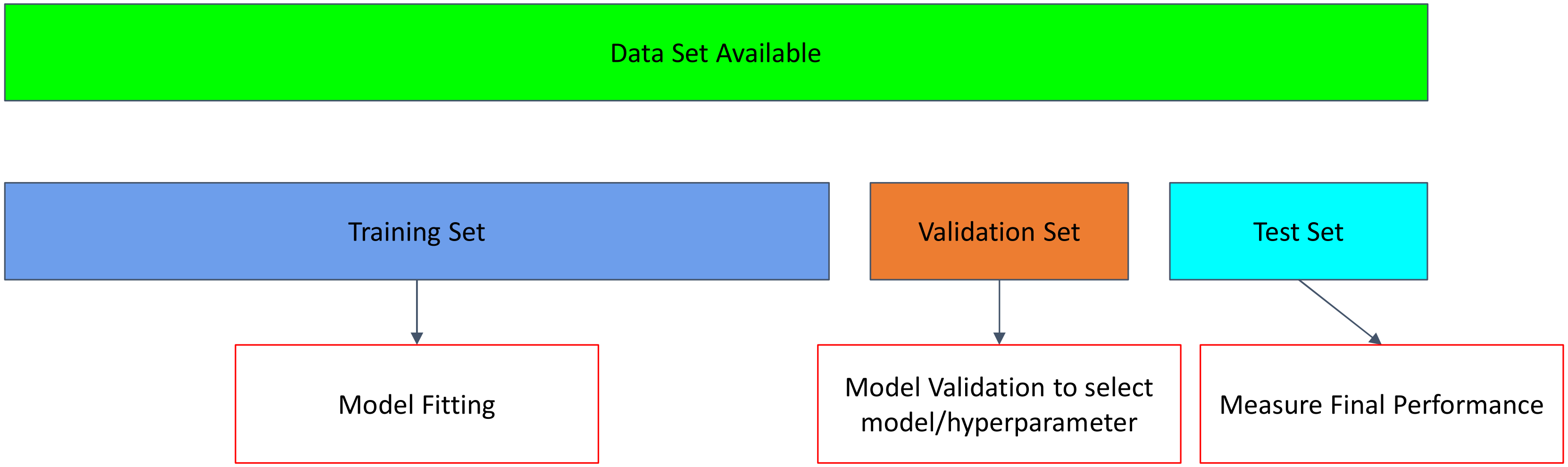
- Grid Search
    - select from all possible combination of candidate model
- Randomized Search
    - select **randomly** from all possible combination of candidate model

Things That Need to be considered

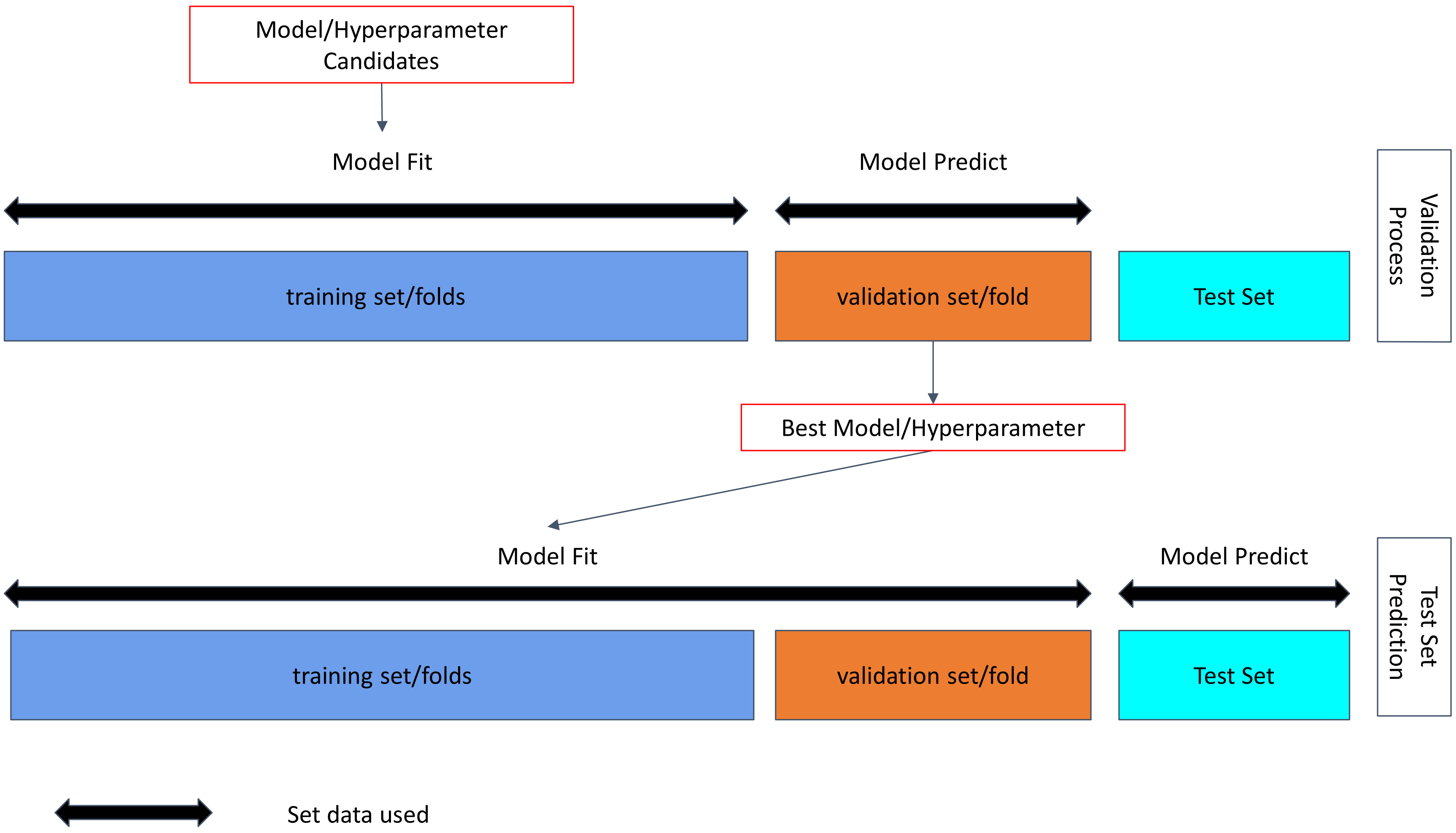
- Your hyperparameter space: some hyperparameter if not chosen correctly may resulted in overfitting
- Your benchmark: you can compare the model performance in test data before and after hyperparameter tuning. from this you can do double checking :
  - is the improvement in the cross validation score significant ?
  - are your choice of hyperparameter resulted in overfitting/unstable model ?
- random_state in some places: you can use this to make sure that any changes in performance (validation fold) because of the hyperparameter not by chance. you need to check in some places :
  - data splitting
  - cv splitting
  - model, etc

How To Do Model Selection ?

What is Your Model Candidates ?


Data Set Available
Training Set
Validation Set
Test Set
Model Fitting
Model Validation to select model/hyperparameter
Measure Final Performance

We can’t use validation set to measure final performance because it’s already used to select best candidate model



Model/Hyperparameter Candidates

Model Fit

Model Predict

training set/folds

validation set/fold

Test Set

Validation Process

Best Model/Hyperparameter

Model Fit

Model Predict

training set/folds

validation set/fold

Test Set

Test Set Prediction

Set data used

Python Exercise : Model Evaluation 1

Analyze data ‘bankloan.csv’

- employ, debtinc, creddebt, othdebt as Features
- default as Target

Random state 2020, splitting 80:20 stratified

1. modeling compute accuracy, recall, and another metrics using Stratified CV 5 fold:
    a) logistic regression (solver liblinear)
    b) KNN (k = 5)
    c) tree (criterion entropy, max_depth 5)
2. compute recall, precision, f1 score and make ROC, PRC from logistic regression(solver liblinear) in test
3. Simple hyperparameter tuning : (optimize C) optimized by f1 and using training 60% validation 20% test 20%
4. compare the result (before and after)
5. Grid Search CV hyperparameter tuning : (optimize C and max_iter) optimized by f1 and using stratified CV 5 fold
6. compare the result (before and after)

Python Exercise : Model Evaluation 2

Analyze tips data from seaborn

- Total Bill, sex, smoker, day and time as Features
- Tips as Target

Preprocess

1. one hot encoding : smoker, day, time
2. no treatment : numerical

Random state 2020, splitting 80:20

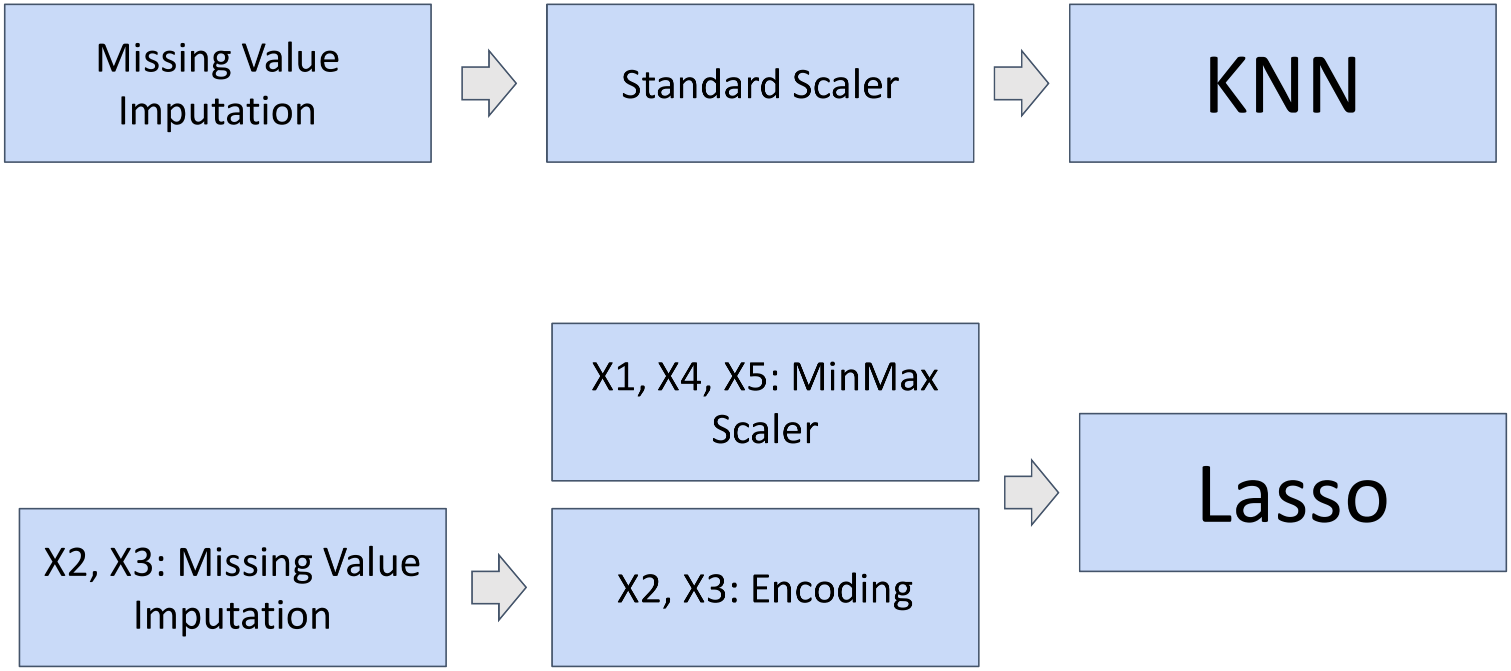
1. model linear regression and tree (criterion mse, max_depth 5) compute R2 using 5-fold CV
2. model decision tree (criterion mse, max_depth 5) compute mse in test set
3. do hyperparameter tuning (Randomized Search) for decision tree (optimize criterion, min sample leaf, max depth) optimized by mse and using CV 5 fold
4. compare the result (before and after) ini test set

Algorithm Chains

Algorithm Chains

- Most machine learning application require not only the application of a single algorithm
- Chaining together of many different preprocessing steps and machine learning algorithm
- we already illustrated these in the Data Preparation and Feature Engineering topic
- Pipeline function from scikit-learn can simplify this problem
- This will be very useful when we evaluating using cross validation and making prediction

Algorithm Chain Example



Missing Value Imputation

Standard Scaler

KNN

X2, X3: Missing Value Imputation

X1, X4, X5: MinMax Scaler

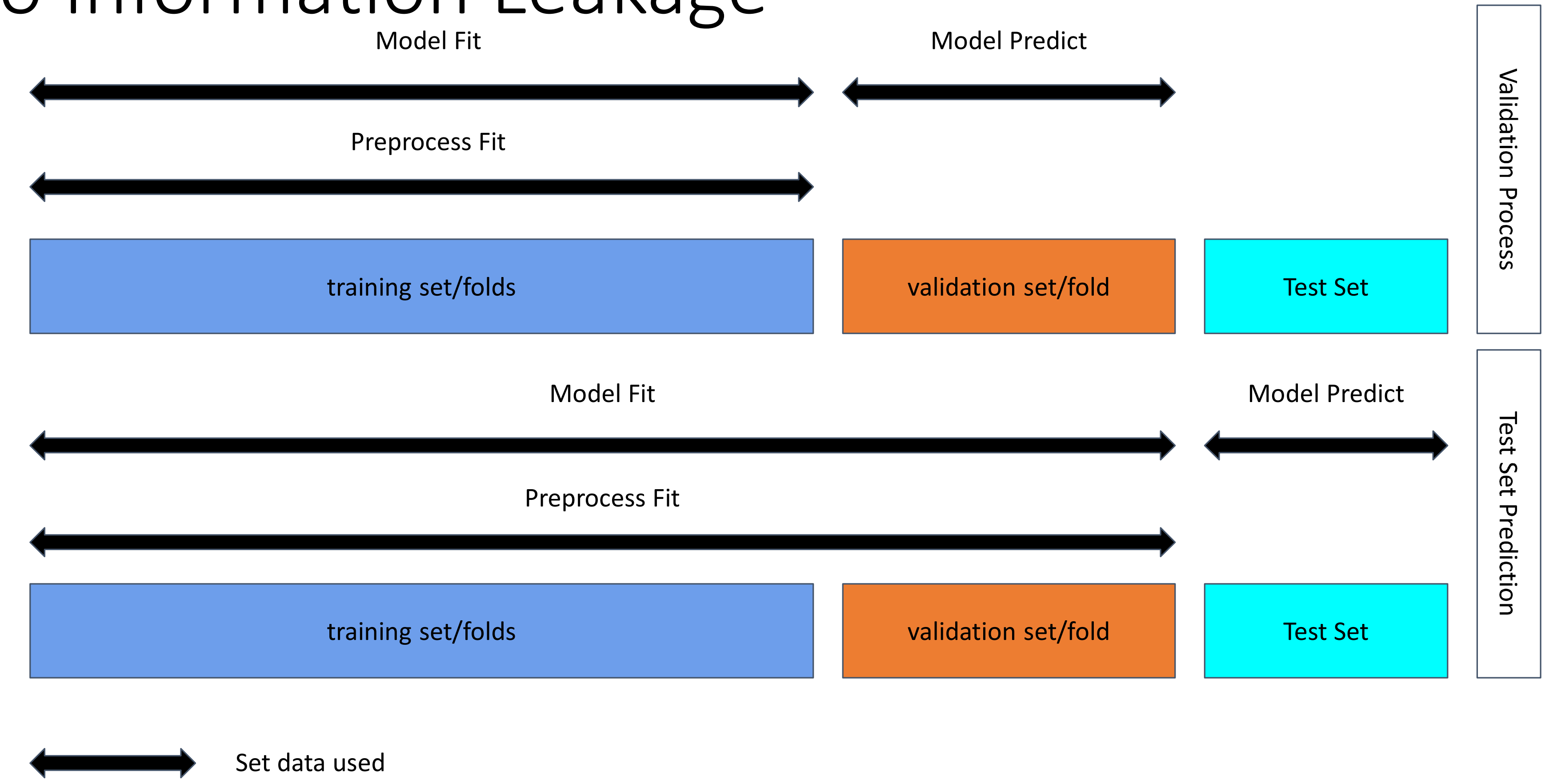
X2, X3: Encoding

Lasso

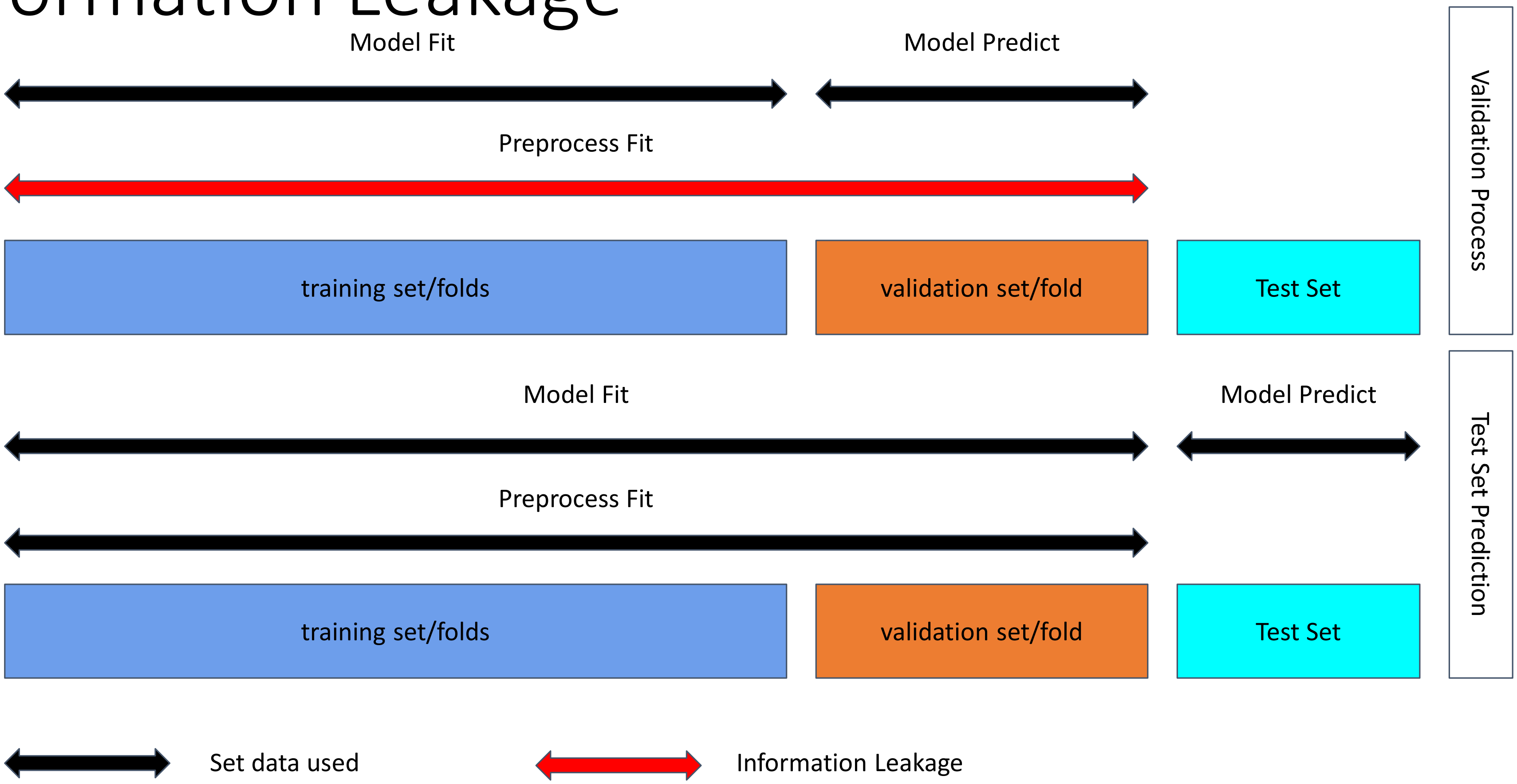
Why Do We Need Algorithm Chains ?

- Automatic preprocessing
- To avoid information leakage
- Information leakage can cause overly optimistic performance in validation set or in test set

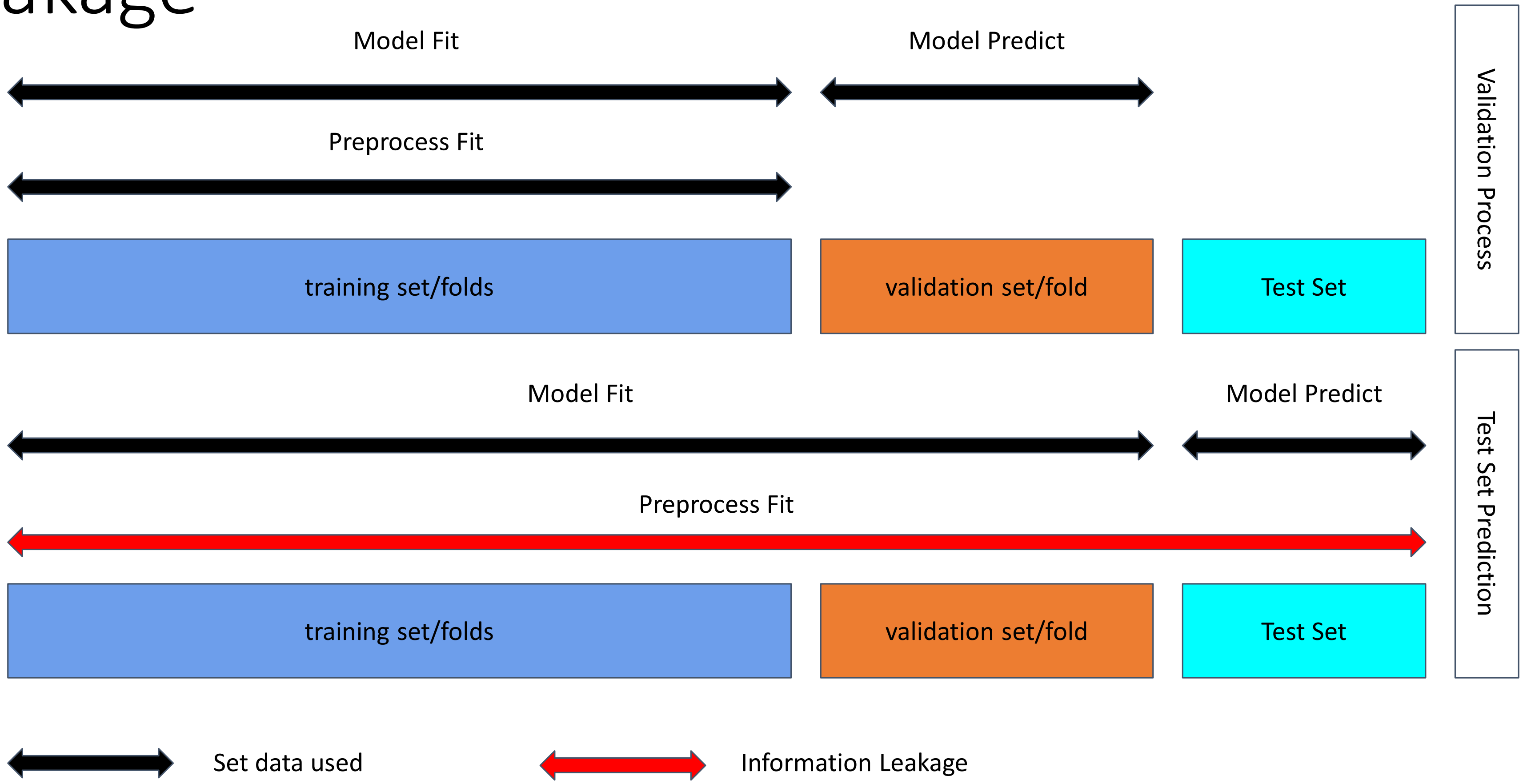
Proper Preprocessing : No Information Leakage



Improper Preprocessing : Information Leakage



Improper Preprocessing : Information Leakage



Model Fit

Model Predict

Preprocess Fit

training set/folds

validation set/fold

Test Set

Validation Process

Model Fit

Model Predict

Preprocess Fit

training set/folds

validation set/fold

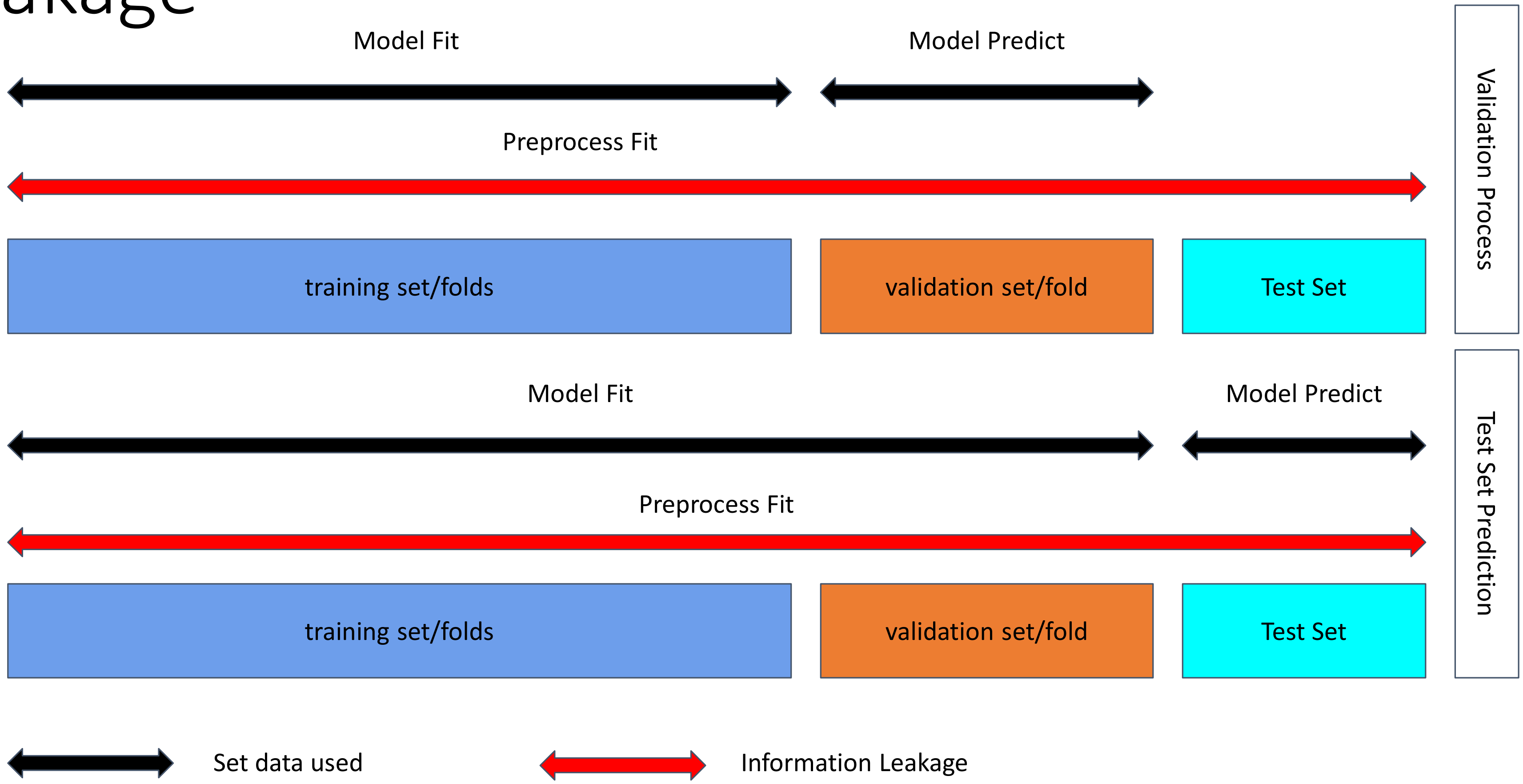
Test Set

Test Set Prediction

Set data used

Information Leakage

Improper Preprocessing : Information Leakage



Validation Process

Model Fit

Model Predict

Preprocess Fit

training set/folds

validation set/fold

Test Set

Test Set Prediction

Model Fit

Model Predict

Preprocess Fit

training set/folds

validation set/fold

Test Set

Set data used

Information Leakage

Information Leakage

data:

- generate data
  - y ~ normal (mean = 0, std = 0, n = 100)
  - X ~ normal (mean = 0, std = 0, n = 100, dimension = 10000)
  - y and X uncorrelated each other (no relationship)
- Information leakage
  - modeling X and y using linear regression + percentile feature selection (f statistics)
  - do feature selection : apply fit and transform to X
  - compute R-square using cross validation
- No Information leakage
  - modeling X and y using linear regression + percentile feature selection (f statistics)
  - make pipeline : feature selection + regression
  - compute R-square using cross validation

Apply Several Preprocessing Method to Modeling at once and do hyperparameter Tuning

data : adult.csv

target : income

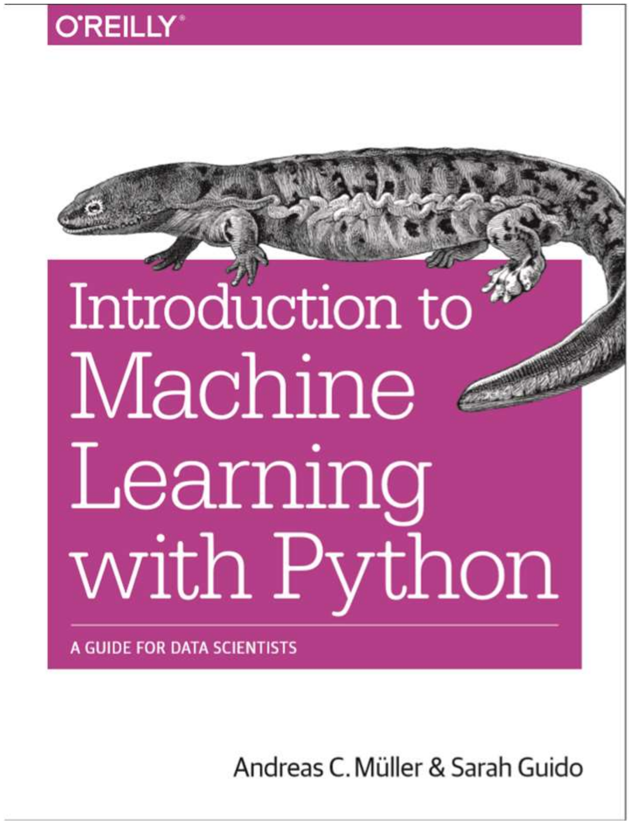
preprocess:

1. missing value : simple imputer with constant
2. one hot encoding : relationship, race, sex
3. binary encoding : workclass, marital status, occupation, native country
4. ordinal encoding : education (already encoded)
5. no treatment : numerical
6. out : fnlwgt

Random state 10, data splitting 70:30

1. model Tree(max depth 5, criterion entropy) and compute f1 using CV
2. Compute another metrics
3. model Logistic Regression (solver liblinear) compute recall +, precision +, f1 +, make ROC and PRC in test set
4. do hyperparameter tuning for logistic regression (optimize C and solver) optimized by f1 and using stratified CV 5 fold
5. compare the result (before and after) ini test set

References

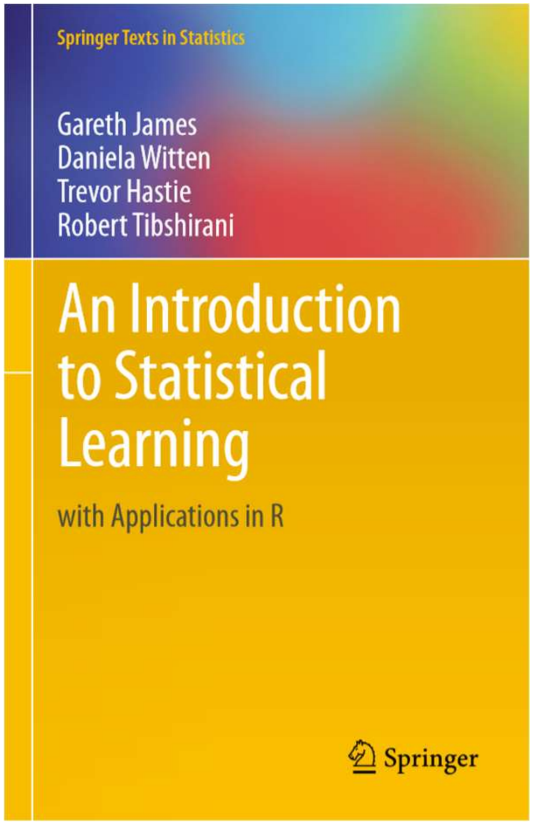


O'REILLY®

Introduction to Machine Learning with Python

A GUIDE FOR DATA SCIENTISTS

Andreas C. Müller & Sarah Guido



Springer Texts in Statistics

Gareth James
Daniela Witten
Trevor Hastie
Robert Tibshirani

An Introduction to Statistical Learning

with Applications in R

Springer

References

https://towardsdatascience.com/metrics-to-understand-regression-models-in-plain-english-part-1-c902b2f4156f

https://towardsdatascience.com/the-ultimate-guide-to-binary-classification-metrics-c25c3627dd0a#a015

https://skillplus.web.id/data-training-validation-dan-test/

https://www.the-modeling-agency.com/crisp-dm.pdf

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