2023 Machine Learning Odyssey Pt. 1

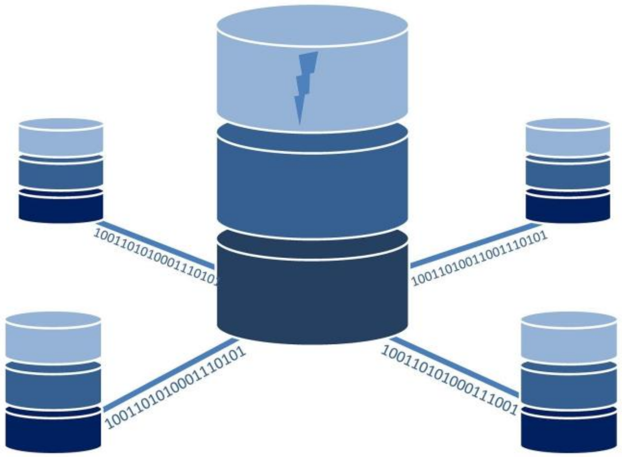
July 25, 2023
Seungeun Lee

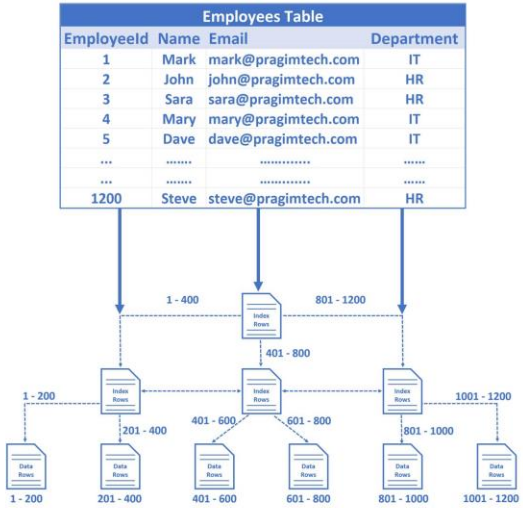
"2023 Machine Learning Odyssey"

Part 1	July 25, 2023	Tabular data, Data preprocessing, Evaluation metrics, Cross validation, imputation
Part 2	TBD (?)	AutoML, Feature selection/extraction, Data Imbalance, Data Preprocessing 2
Part 3	TBD	LightGBM, Hyperparameter tuning, Gridsearch
Part 4	TBD	SVM, RandomForest, Clustering, Dimension reduction
Part 5	TBD	XGBoost, Ensemble Models
Part 6	TBD	ML v.s. DL (TabNet, …), Future of Machine Learning (Causal Inference, Bayesian)
Part 7	TBD	Meta Learning & Meta Reinforcement Learning
Part 8	TBD	MLOps, Multi-modal analysis

Tabular Data

- Structured Data
- Data represented in rows and columns, which is extracted from the Database (DB)





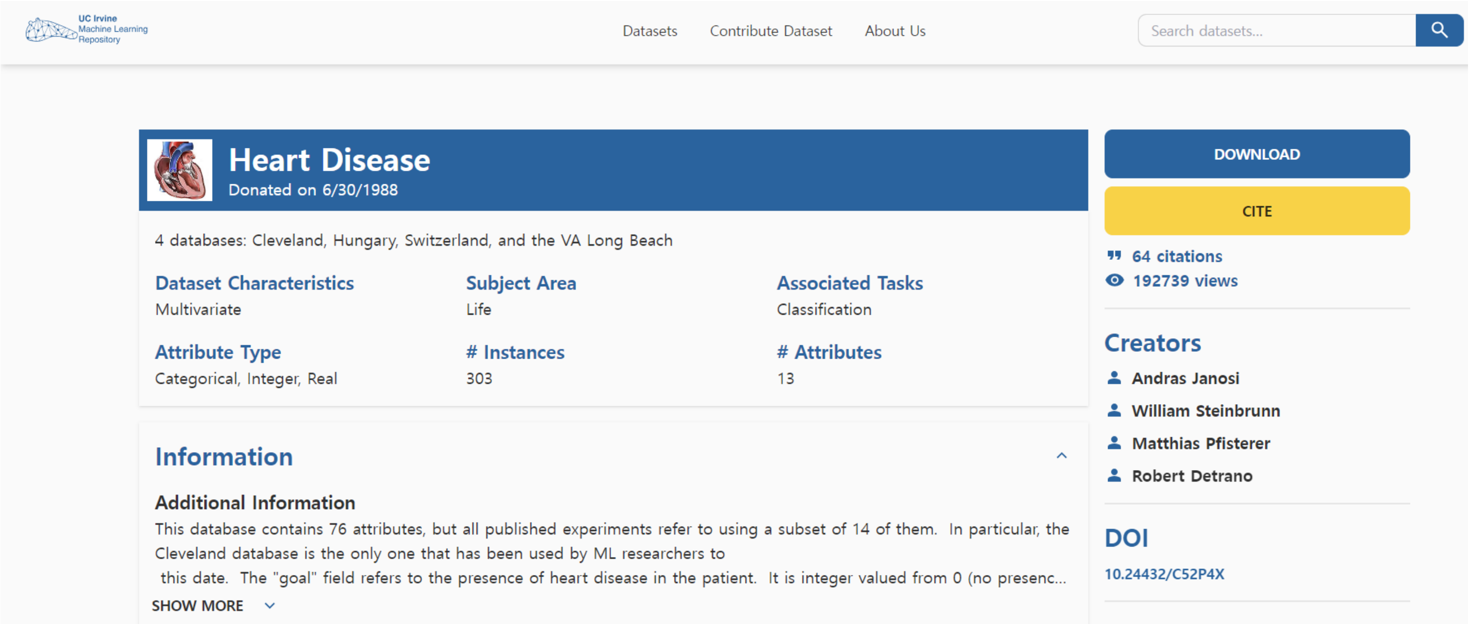
DataBase

Query (SQL)

Data

UCI Heart Disease

- UC Irvine Machine Learning Repository
- https://archive.ics.uci.edu/dataset/45/heart+disease



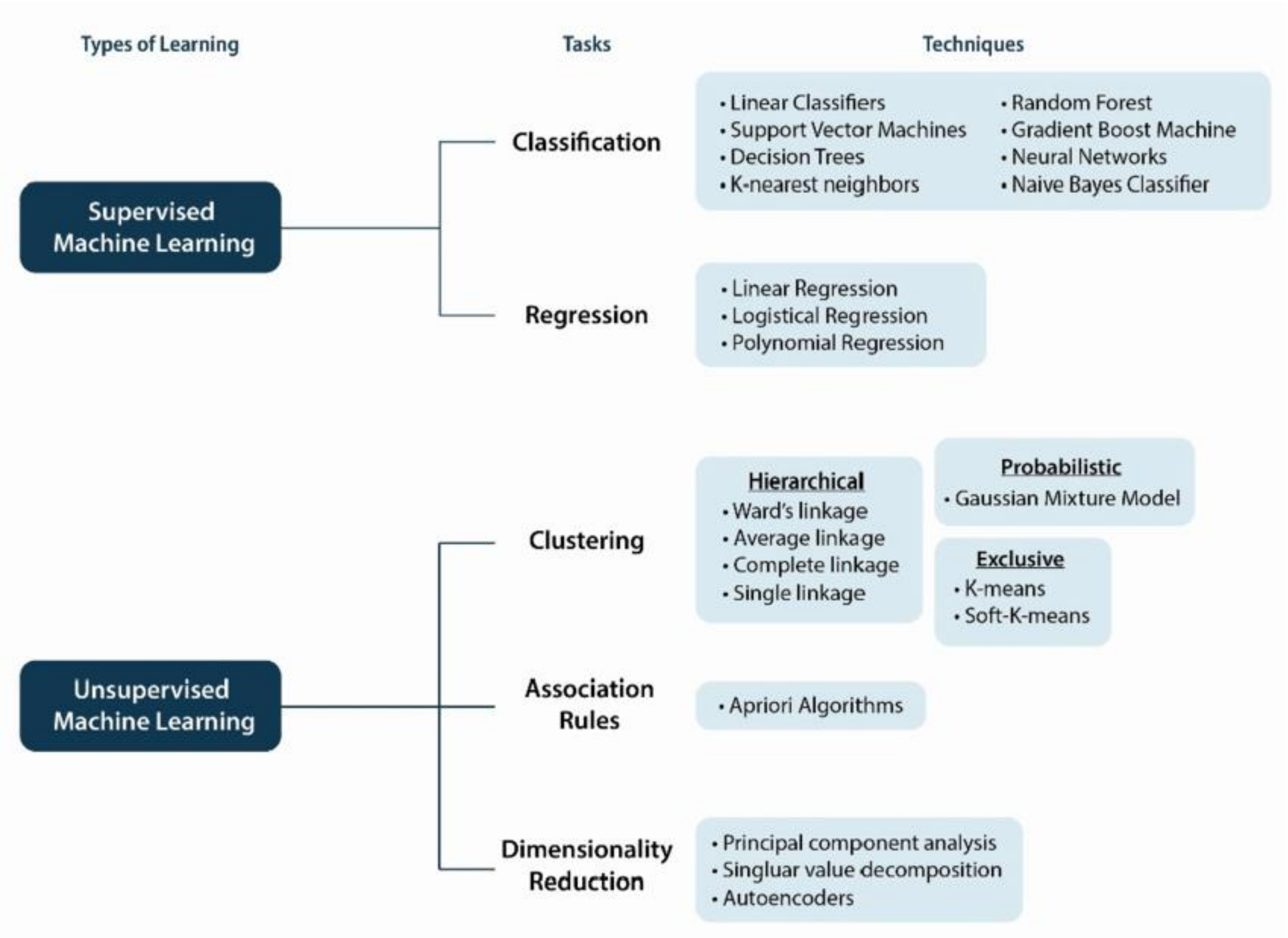
UCI Heart Disease

Features

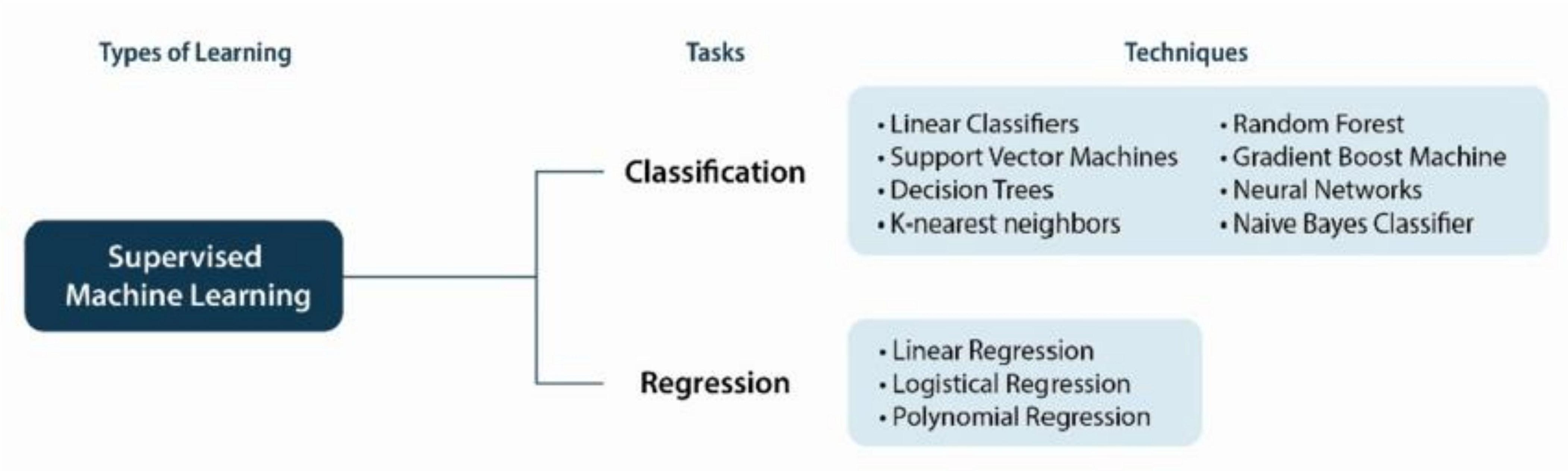
Attribute Name	Role	Type	Demographic	Description	Units	Missing Values
age	Feature	Discrete			years	false
sex	Feature	Categorical				false
cp	Feature	Categorical				false
trestbps	Feature	Discrete		resting blood pressure (on admission to the hospital)	mm Hg	false
chol	Feature	Discrete		serum cholestoral	mg/dl	false
fbs	Feature	Categorical		fasting blood sugar > 120 mg/dl		false
restecg	Feature	Categorical				false
thalach	Feature	Discrete		maximum heart rate achieved		false
exang	Feature	Categorical		exercise induced angina		false
oldpeak	Feature	Discrete		ST depression induced by exercise relative to rest		false

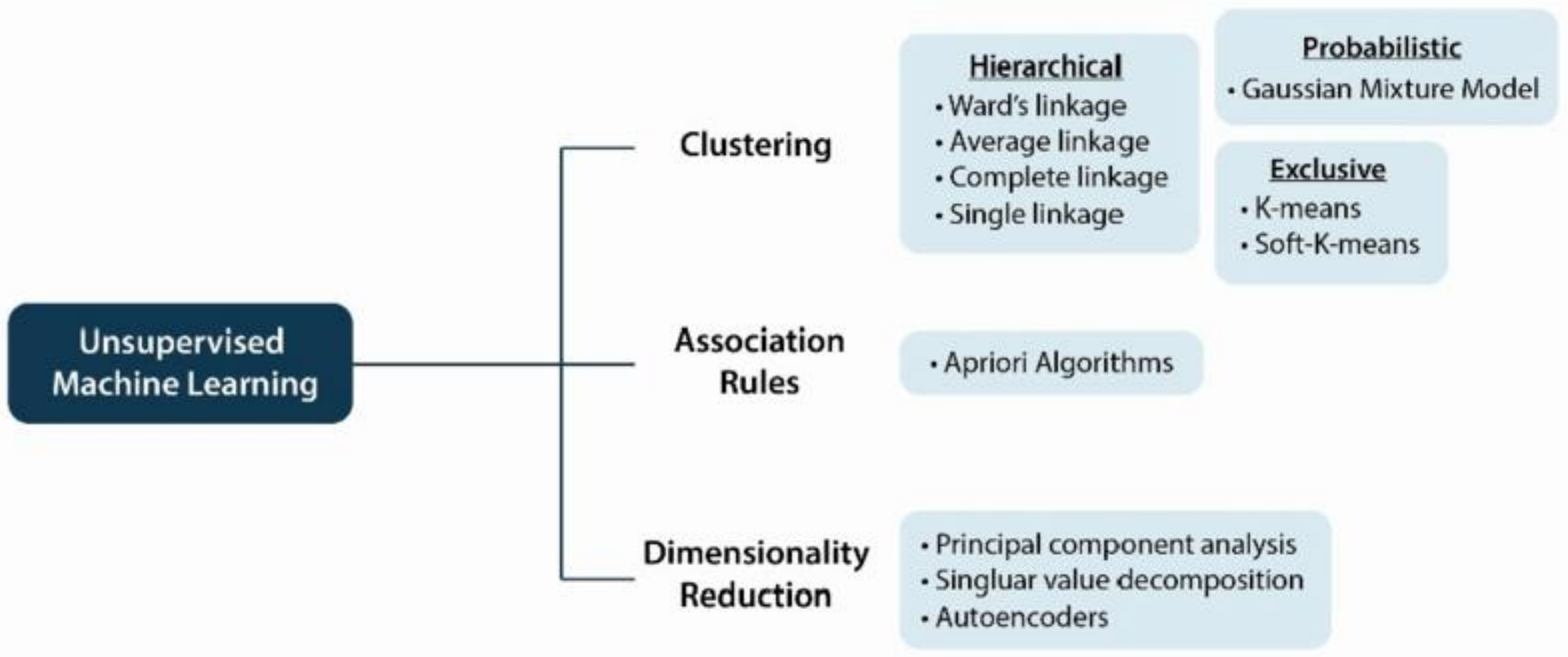
Why is Tabular Data important?

"In the field, structured data is used more often than unstructured data."
"Obtaining unstructured data (images, audio, text) is still challenging, so we are working on cloud improvements."
"Device parameter tuning with ML assistance."
"There are many instances when knowledge of ML and statistics is required."


Types of Learning
Tasks
Techniques
Supervised Machine Learning
Classification
• Linear Classifiers
• Support Vector Machines
• Decision Trees
• K-nearest neighbors
• Random Forest
• Gradient Boost Machine
• Neural Networks
• Naive Bayes Classifier
Regression
• Linear Regression
• Logistical Regression
• Polynomial Regression
Unsupervised Machine Learning
Clustering
Hierarchical
• Ward's linkage
• Average linkage
• Complete linkage
• Single linkage
Probabilistic
• Gaussian Mixture Model
Exclusive
• K-means
• Soft-K-means
Association Rules
• Apriori Algorithms
Dimensionality Reduction
• Principal component analysis
• Singluar value decomposition
• Autoencoders

Random Forest
XGBoost



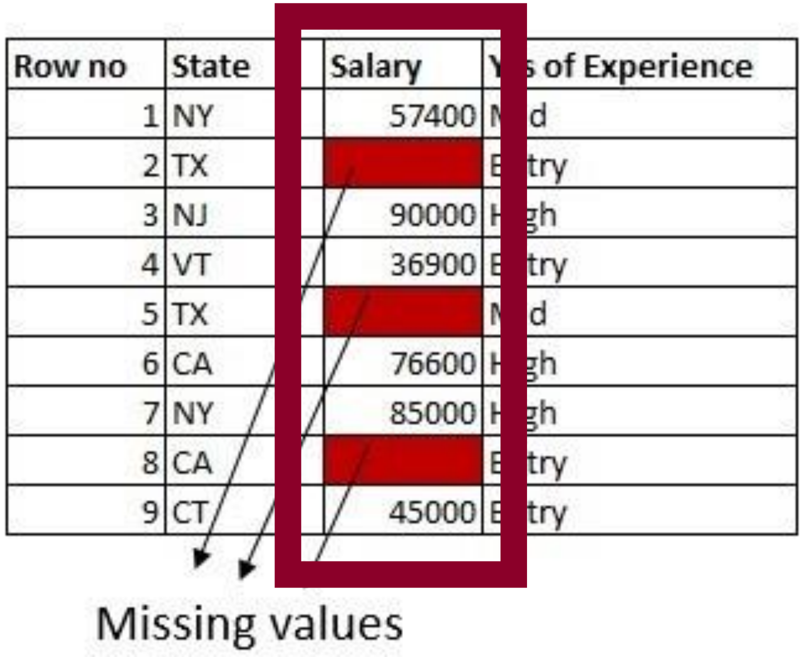

Unsupervised Machine Learning
Clustering
Hierarchical
• Ward's linkage
• Average linkage
• Complete linkage
• Single linkage
Probabilistic
• Gaussian Mixture Model
Exclusive
• K-means
• Soft-K-means
Association Rules
• Apriori Algorithms
Dimensionality Reduction
• Principal component analysis
• Singluar value decomposition
• Autoencoders

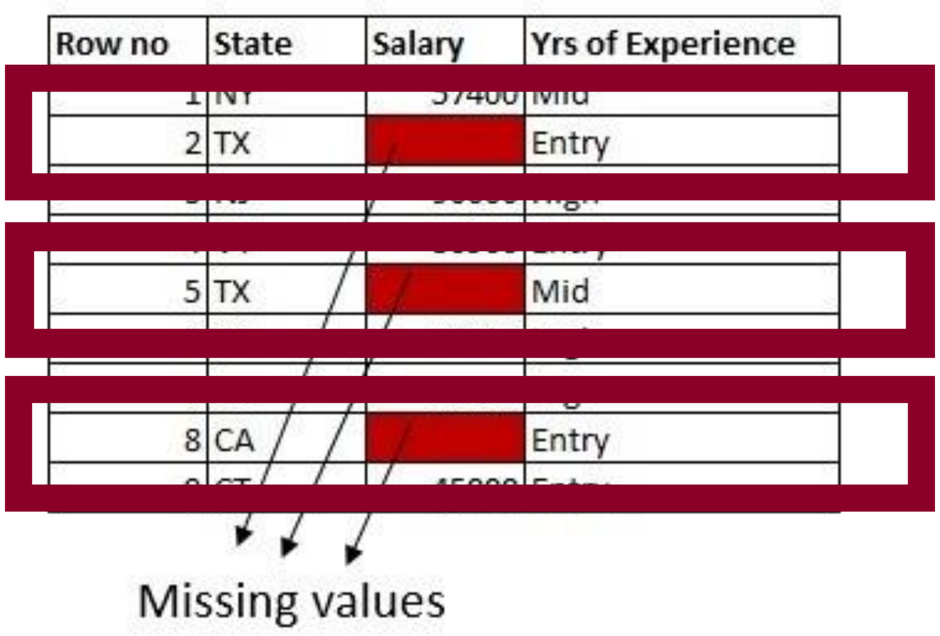
Missing feature (NA, Not Available)

- Handling Missing Values (In DL, the concept of missing values doesn't exist. Tabular data, however, often has partial missing values)

(1) <u>Do nothing</u>: Use models that handle missing values, like XGBoost or LightGBM (covered in later slides) Why did the missing values occur? (Random? Specific reason?)

(2) <u>Deletion / Drop (removing missing data)</u>: Caution! (may lead to loss of important data)





Missing feature (NA, Not Available)

(3) Imputation

- Mean, Median imputation
- Most Frequent Value / Zero / Constant Imputation
- Why did the missing values occur? (Random? Specific reason?) -> Imputation based on domain knowledge
- Drawback: Does not consider correlation with other features (columns) and may introduce bias. -> Alternative approaches?

< Guidelines for Handling Missing Values >

Less than 10%: Deletion or Imputation

10% - 50%: Use Regression or Model-based Imputation

Over 50%: Remove the column (variable) itself

(Not a strict rule, just an empirical guideline)

Missing feature (NA, Not Available)

- KNN Imputation
- MICE (Multivariate Imputation by Chained Equation) Imputation
- DL-based imputation

```
from impyute.imputation.cs import fast_knn
np_imputed=fast_knn(df_null.values, k=5)
df_imputed = pd.DataFrame(np_imputed)
```

```
from impyute.imputation.cs import mice
np_imputed=mice(df_null.values)
df_imputed = pd.DataFrame(np_imputed)
```

- Each is a complete paper -> Details are left for future work
- Drawbacks: Takes a long time to impute ($\mathbb{R}^{321\times70}$, more than 2 hours), OOM (Out of Memory), sensitive to outliers

Missing feature (NA, Not Available)

- KNN Imputation
  - Finding the nearest #K data using feature similarity
  - Finding the nearest neighborhood (NN) by generating KDTree -> assigning weighted average according to the distance

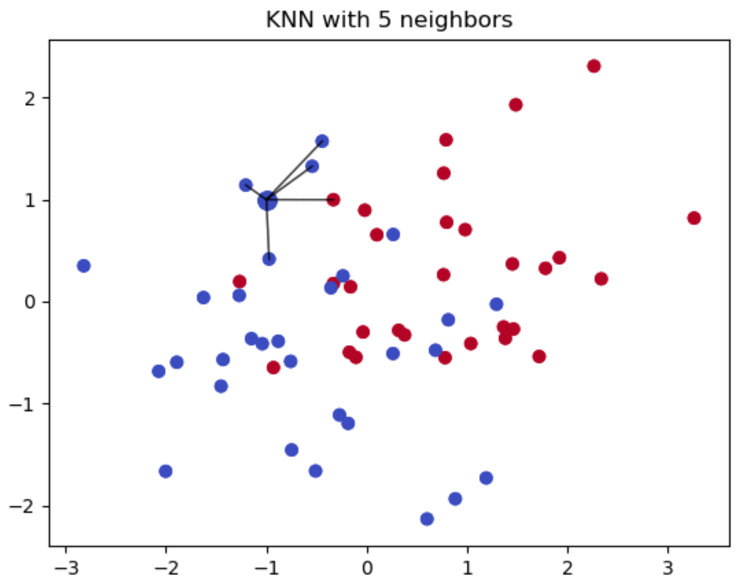
- MICE (Multivariate Imputation by Chained Equation) Imputation

  Multiple Imputation Approach for Handling Missing Data: A chained approach
  - Capable of handling continuous, binary, ordinal types, and survey skip patterns
  - Multiple Imputation manages uncertainty better than Single Imputation

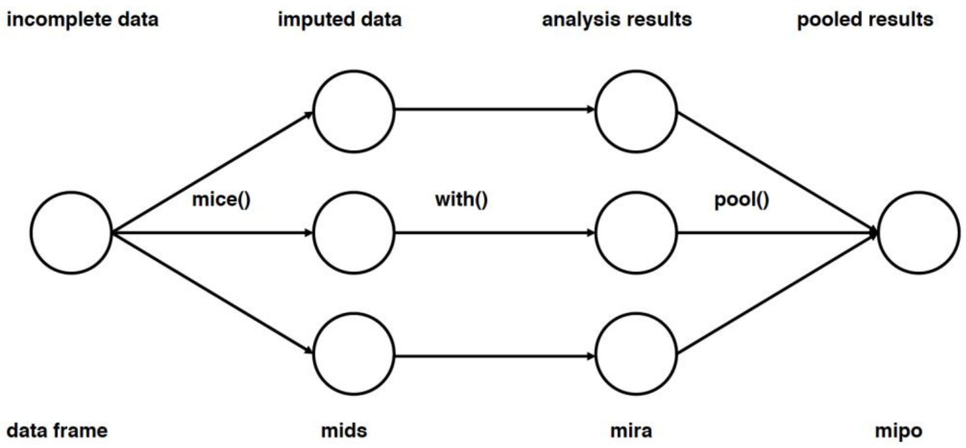
  (1) Imputation: option (mean, median, …); with *m* different methods
  (2) Analysis: Analyze each of the *m* completed datasets
  (3) Pooling: Integrate results by calculating mean, variance, and confidence intervals

- DL-based (It may view as a kind of an overload to use DL to solve ML problems)

Missing feature (NA, Not Available)



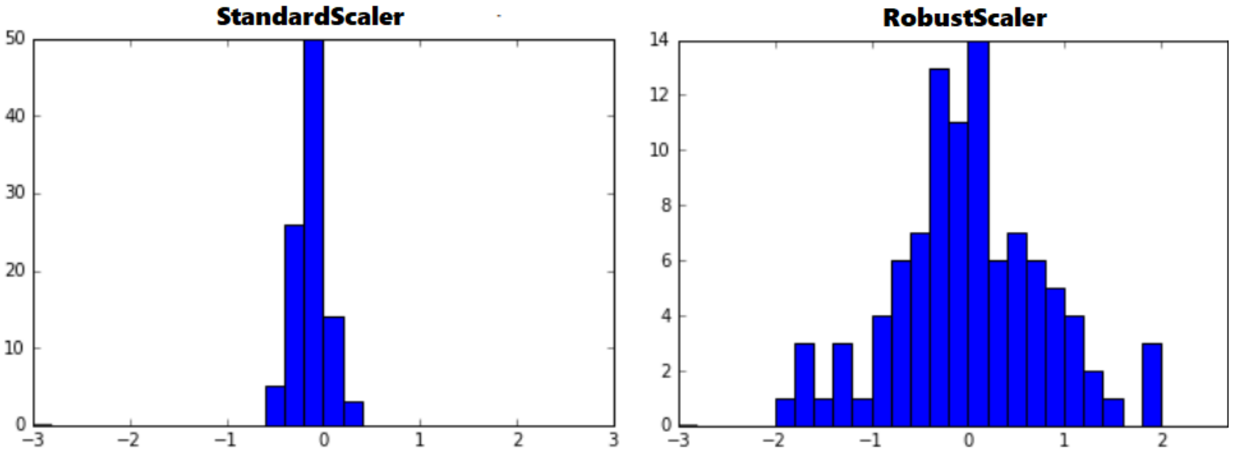
KNN Imputation



MICE Imputation

Data preprocessing (Scikit-learn)

- **StandardScaler**: Removes the mean and scales data to unit variance; sensitive to outliers.
- **MinMaxScaler**: Scales data so that all feature values are within the range of 0-1; sensitive to outliers.
- **MaxAbsScaler**: Scales absolute values between 0 and 1 (i.e., -1 to 1); behaves similarly to MinMaxScaler with positive-only data but is sensitive to large outliers.
- **RobustScaler**: Minimizes the effect of outliers by using the median and interquartile range (IQR), resulting in a broader distribution of standardized values compared to StandardScaler.



Personally, I believe that…

- It is important to make feature scales similar through scaling, but it is not necessary to make all feature distributions identical.
- Depending on the characteristic, maintaining the original data distribution may be meaningful for certain features.
  Example: When standardizing a feature highly concentrated in one area, minor changes may end up representing significant differences. In practice, experiments often show better performance when using the original data as is.

- **No need to overthink data scaling!**
  If scaling is desired, StandardScaler is recommended.

Data preprocessing

! Caution !

- After splitting into training and test datasets, apply scaling only to the training dataset and then apply the same scaling method to the test dataset.
- If training and test data are scaled independently, it may introduce bias specific to each dataset.
- Applying scaling to the entire dataset at once risks data leakage, as information from the test data may inadvertently influence the training data.

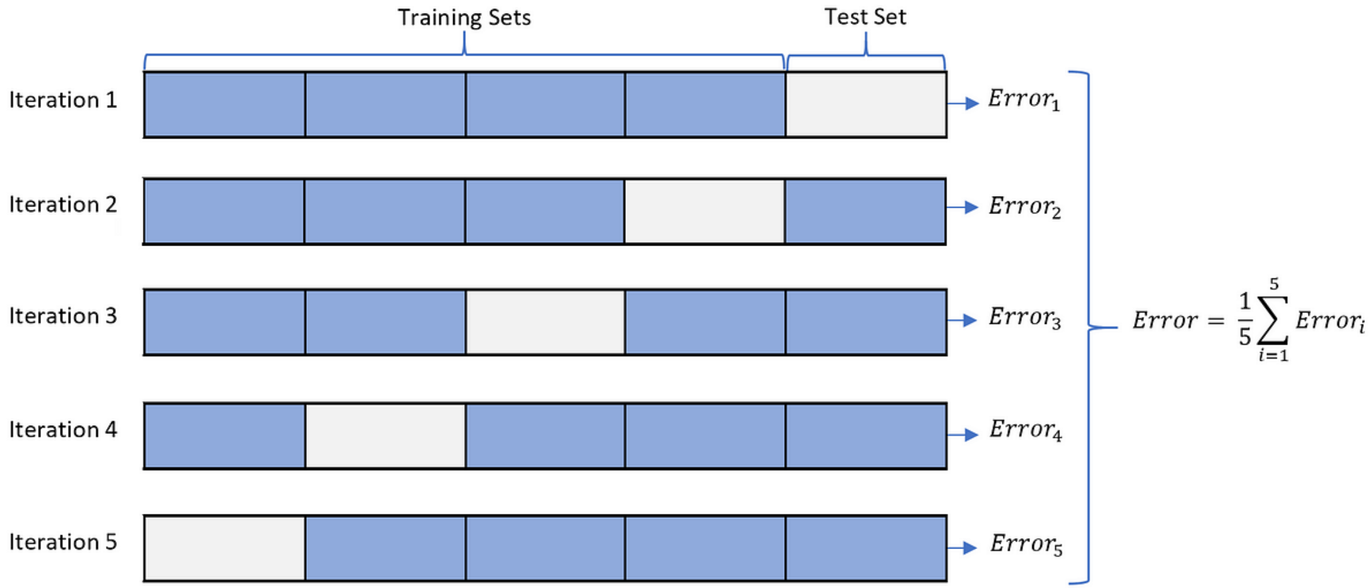
- **Real-time Validation** and **External Validation** are needed.
- However, as mentioned before, there's generally no need to worry excessively about data scaling!

Data preprocessing

(+)

(1) Removing Outliers (After checking the distribution)
(2) SMOTE-based Over Sampling … a kind of a Data Augmentation

Cross Validation (2/5/10 Fold CV)



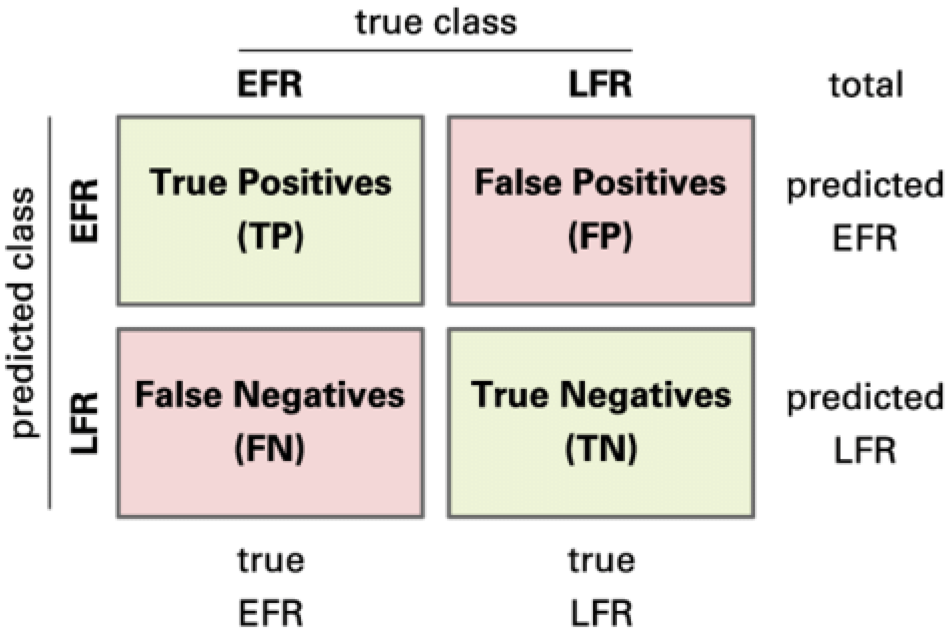
- Used to ensure stability in training (especially necessary for stable learning with small datasets in ML) — commonly required in the medical domain.
- When using Scikit-learn's API, consider the data distribution when splitting into Train/Validation sets.

Evaluation Metrics

(1) Accuracy
(2) Confusion Matrix

$$Accuracy = \frac{Correct\ prediction}{Total\ cases} * 100\%$$

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} * 100\%$$



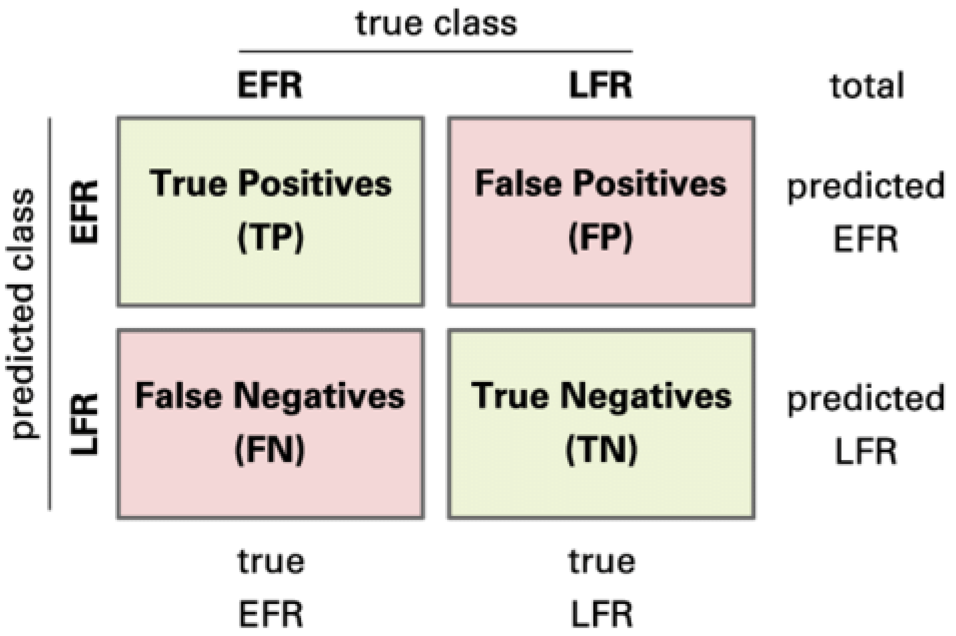
$$\mathrm{PR} = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$$

$$\mathrm{RE} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$$

$$\mathrm{CA} = \frac{\mathrm{TP+TN}}{\mathrm{TP+TN+FP+FN}}$$

$$\mathrm{F}_1 = \frac{\mathrm{2TP}}{\mathrm{2TP+FP+FN}}$$

Evaluation Metrics



$$\mathrm{PR} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$$

$$\mathrm{RE} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$$

$$\mathrm{CA} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}$$

$$\mathrm{F}_1 = \frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$$

(TN) Prediction: Negative (0), Truth: Negative (0)
(FP) Prediction: Negative (1), Truth: Negative (0)
(FN) Prediction: Negative (0), Truth: Negative (1)
(TP) Prediction: Negative (1), Truth: Negative (1)

Evaluation Metrics

(3) F1 score

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Precision: The proportion of predictions that are Positive that are actually Positive
== PPV [Positive Predictive Value

Recall: The proportion of actual Positive cases correctly predicted as Positive by the model
== sensitivity, hit rate, TPR

F1 score: A balanced measure of Precision and Recall, achieving a higher score when neither is overly dominant.

Evaluation Metrics

(+)

Specificity: The proportion of actual Negative cases correctly predicted as Negative by the model

NPV [Negative Predictive Value]: The proportion of predictions that are Negative that are actually Negative

>> Accuracy, AUC, Sensitivity, Specificity, PPV, NPV

Data preprocessing

! Caution !

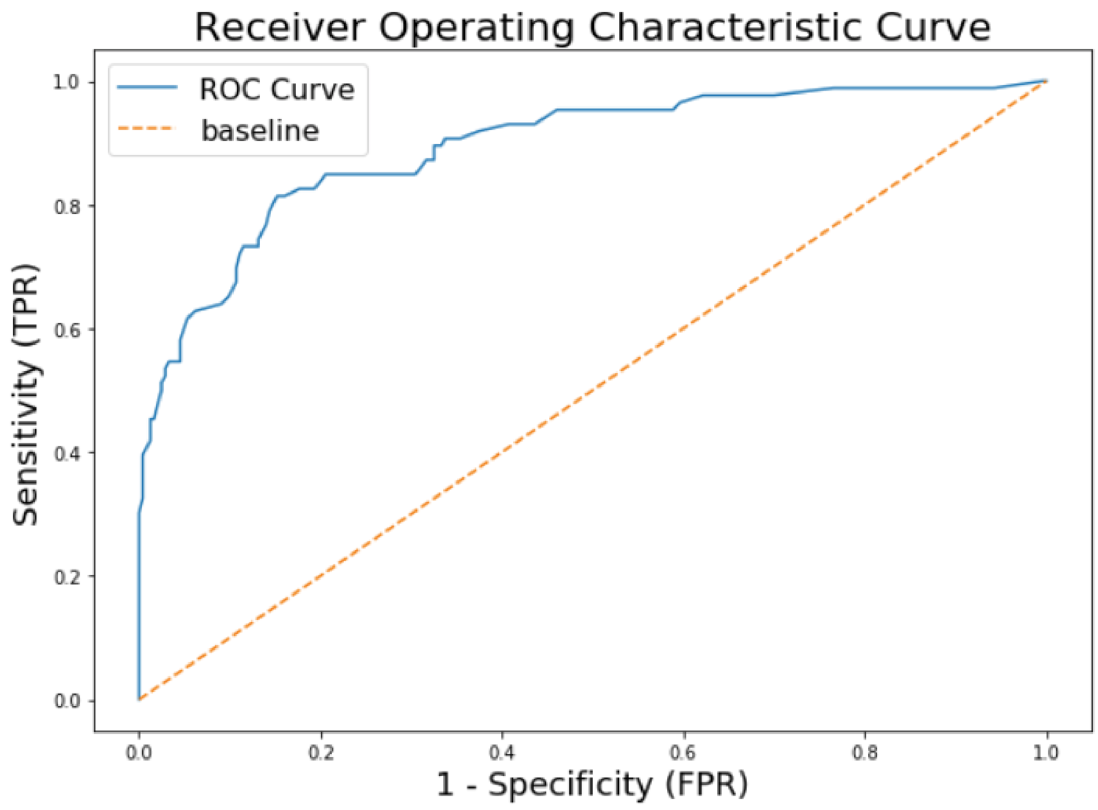
For models that allow threshold adjustments,
e.g., setting a threshold to predict as 0 or 1 -> customizing the threshold is possible.
However, it should not be used simply as a means to boost performance metrics.

? Question ?

Is it possible to evaluate across all thresholds?

Evaluation Metrics

(4) ROC Curve & AUC [Area Under the Curve]



A higher AUC is better, and it's important to achieve a high TPR while keeping the FPR low

FPR: False Positive Rate
TPR: True Positive Rate

(*AUC should be above 0.5; if it's below 0.5 in a typical classifier, there may be a calculation error*)

Data preprocessing

- Question
- ROC Curve for Multi-class classification

- https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html
- One-vs-One [OvO] multiclass ROC / One-vs-Rest [OvR] multiclass ROC
- https://github.com/duneag2/vit-xgboost-imaging-genomics

Evaluation Metrics

(5) MCC

Matthews correlation coefficient (Phi coefficient)
Binary classification (Useful in case of a Class Imbalance)

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

Data preprocessing

- Regression Metrics

RMSE, R Square, MAE, MSLE, RMSLE, Pearson Correlation Coefficient

Reference

(1) https://inhovation97.tistory.com/65
(2) https://dining-developer.tistory.com/19
(3) https://wooono.tistory.com/103
(4) https://mkjjo.github.io/python/2019/01/10/scaler.html
(5) https://ivoryrabbit.github.io/%EC%88%98%ED%95%99/2021/03/12/%EB%A7%A4%ED%8A%9C%EC%83%81%EA%B4%80%EA%B3%84%EC%88%98.html

COMING UP NEXT…!

No need to write the whole python code of ML models...?!! (feat. AutoML)

Q & A