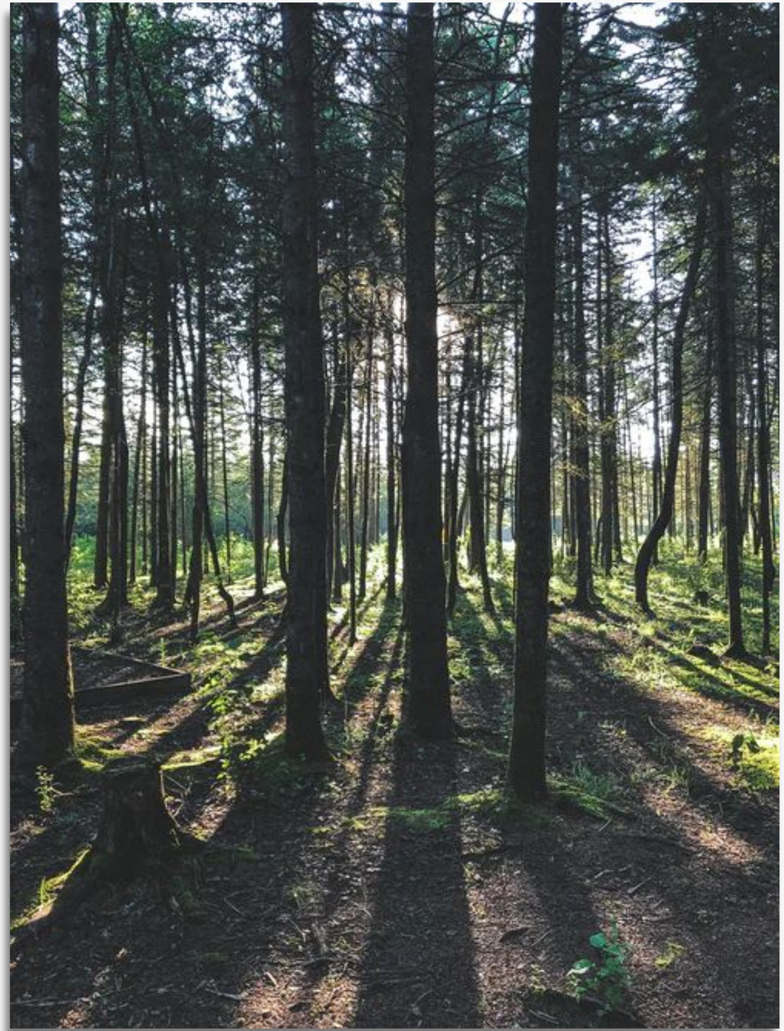
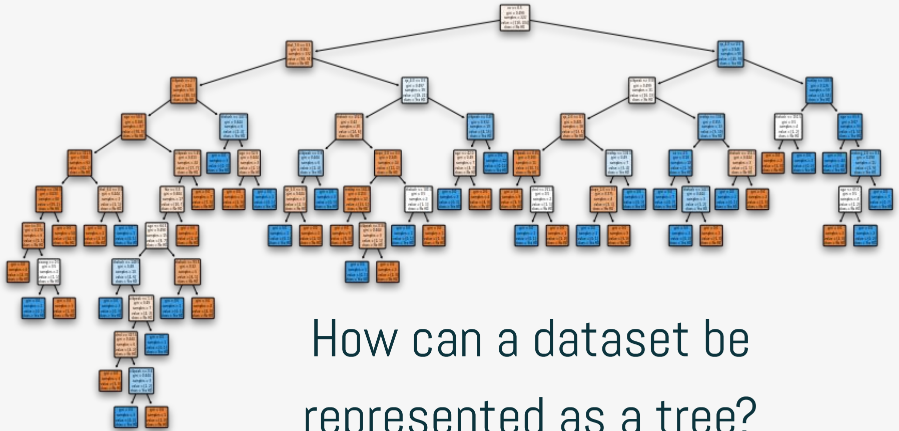
Aprendizado de Máquina

Árvores de Decisão

Ivanovitch Silva
ivanovitch.silva@ufrn.br



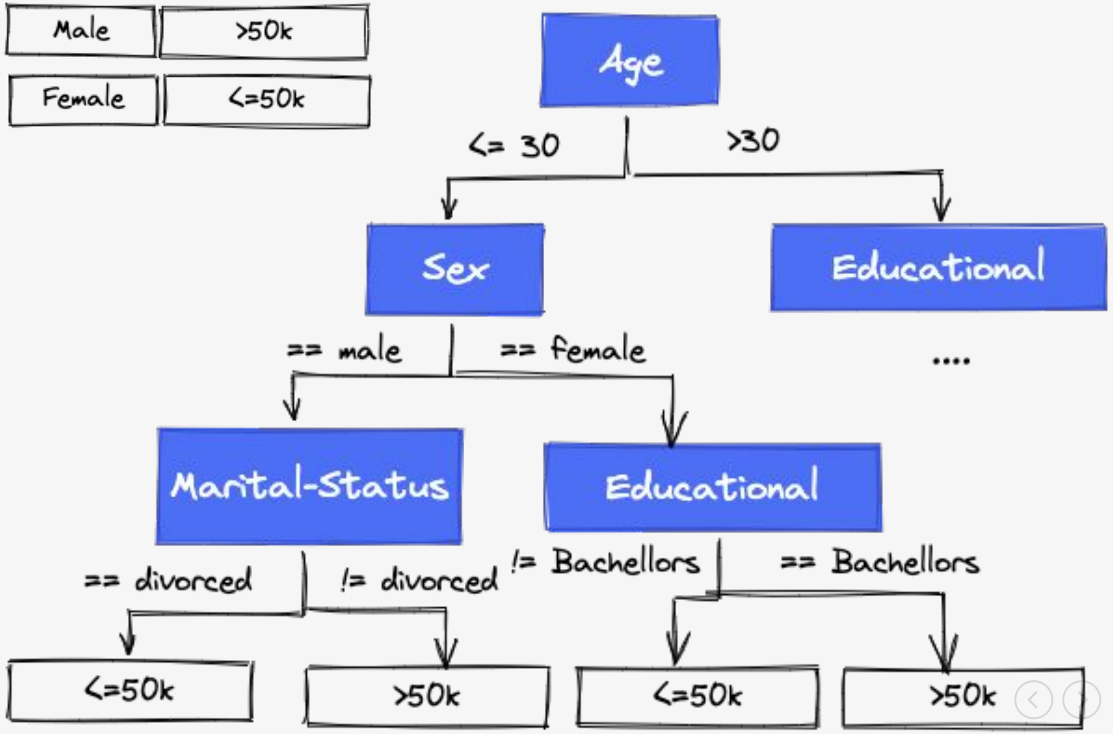
Decision Trees



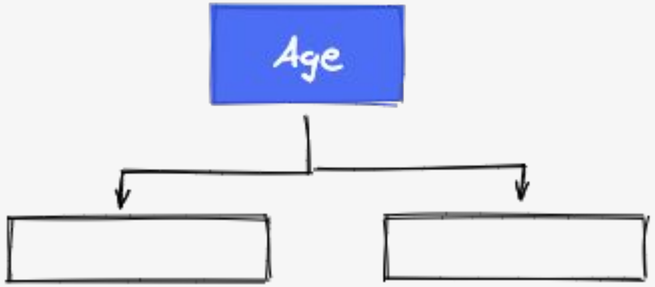
How can a dataset be represented as a tree?

Age	Marital-Status	Educational	Sex	High-Income
28	Never-Married	Bachelors	Female	<=50k
46	Never-Married	Assoc-acdm	Female	<=50k
35	Married-civ-spouse	Some-college	Male	<=50k
27	Married-civ-spouse	Bachelors	Male	>50k
59	Divorced	Some-college	Female	<=50k

Decision Tree (classification)



How can we split the tree?

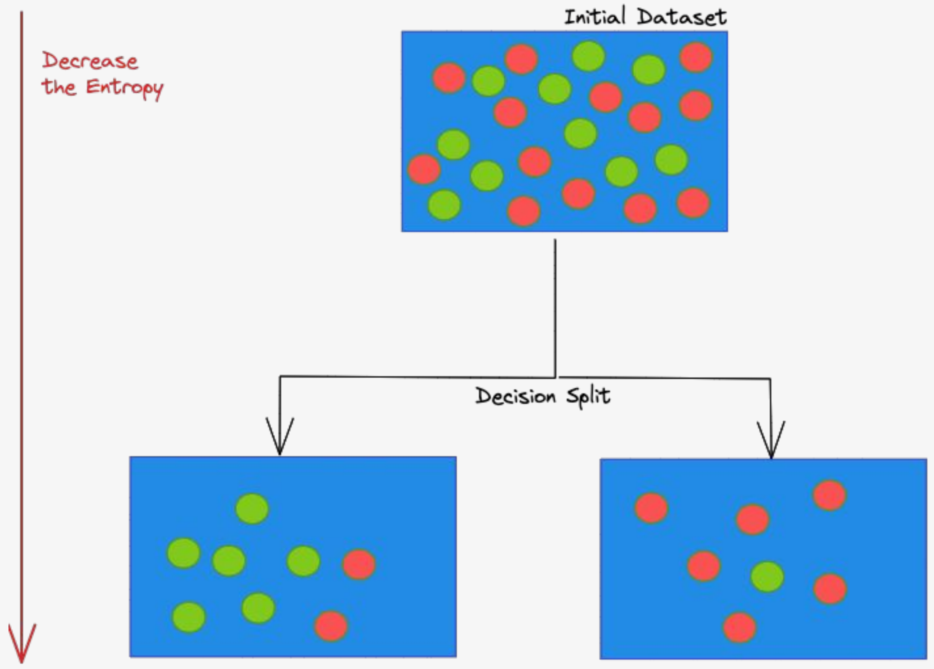


Algorithm used in Decision Trees

1. ID3 (Entropy)
2. Gini Index
3. Chi-Square
4. Reduction in Variance
    a. C4.5, pruning
5. ...

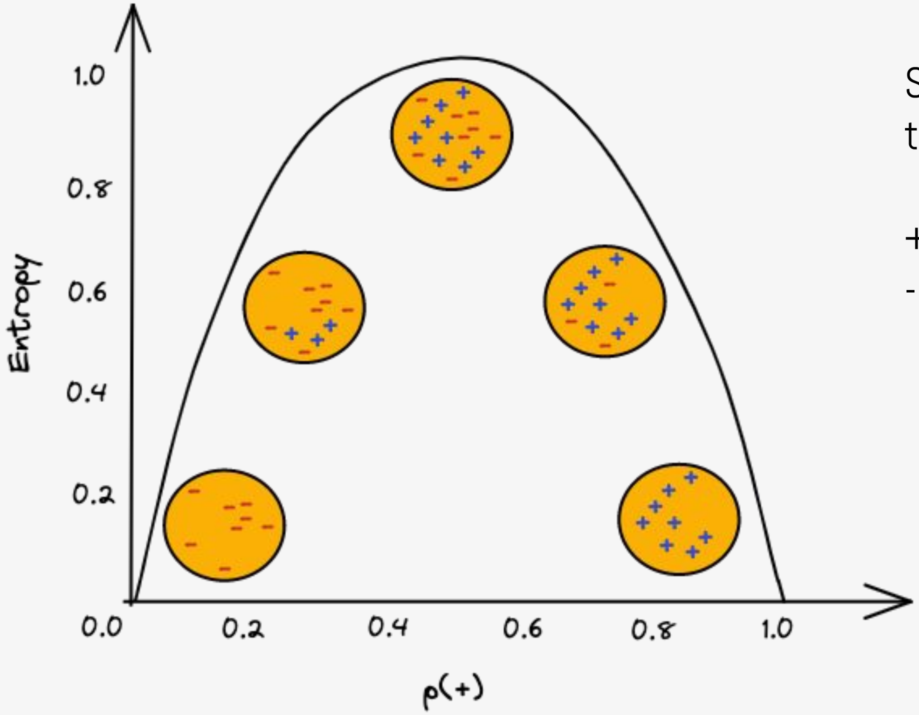
Entropy is an indicator of how messy your data is.

Why Entropy in Decision Trees?



- The goal is to tidy the data.
- You try to separate your data and group the samples together in the classes they belong to.
- You maximize the purity of the groups as much as possible each time you create a new node of the tree
- Of course, at the end of the tree, you want to have a clear answer.

Mathematical Definition of Entropy



Suppose a set of N items, these items fall into two categories:

+ gain > 50k (k)
- gain <= 50k (m)

$$p = \frac{k}{N}, q = \frac{m}{N}$$
$$Entropy = -p\log p - q\log q$$

Generalization

$$E(\overset{\text{Feature X}}{x}) = -\sum_{i=1}^{c} \underbrace{P(x_i)}_{P(x_i)\text{ is the fraction of examples in a given class } i} \log_b P(x\ )$$

<= 50k.	17288
> 50k.	5487

```
from scipy.stats import entropy
entropy(df_train.high_income.value_counts(), base=2)
0.7965702796015677
```

Entropy using the frequency table of two attributes

		High Income		
		<= 50k	> 50k	
Age	<=37	7206	3883	11089 (48%)
	>37	10082	1604	11686 (52%)

$$E(T \mid X) = \sum_{c \in X} \frac{|X_c|}{|X|} E(T \mid X_c)$$

```
cross = pd.crosstab(
          df_train.age <= df_train.age.median(),
          df_train.high_income)
```

```
0.486894 * entropy(cross.iloc[0], base=2) \
+ 0.513106 * entropy(cross.iloc[1], base=2)
0.7509335429830957
```

Information Gain

$$IG(T,X) = E(T) - E(T|X)$$

Information Gain from X on T

The information gain is based on the **decrease in entropy after a dataset is split** on an attribute.

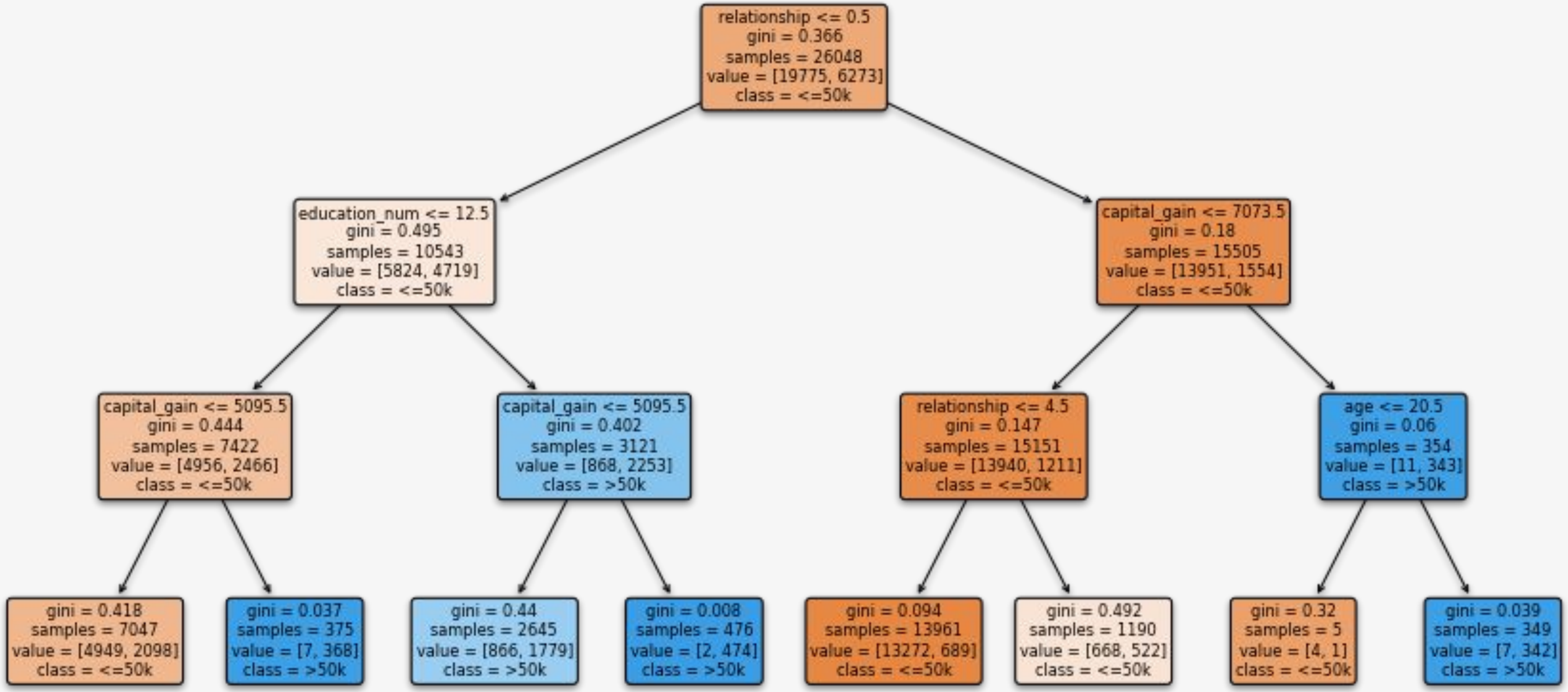
Constructing a decision tree is all about finding attribute that returns the **highest information gain** (i.e., the most homogeneous branches).

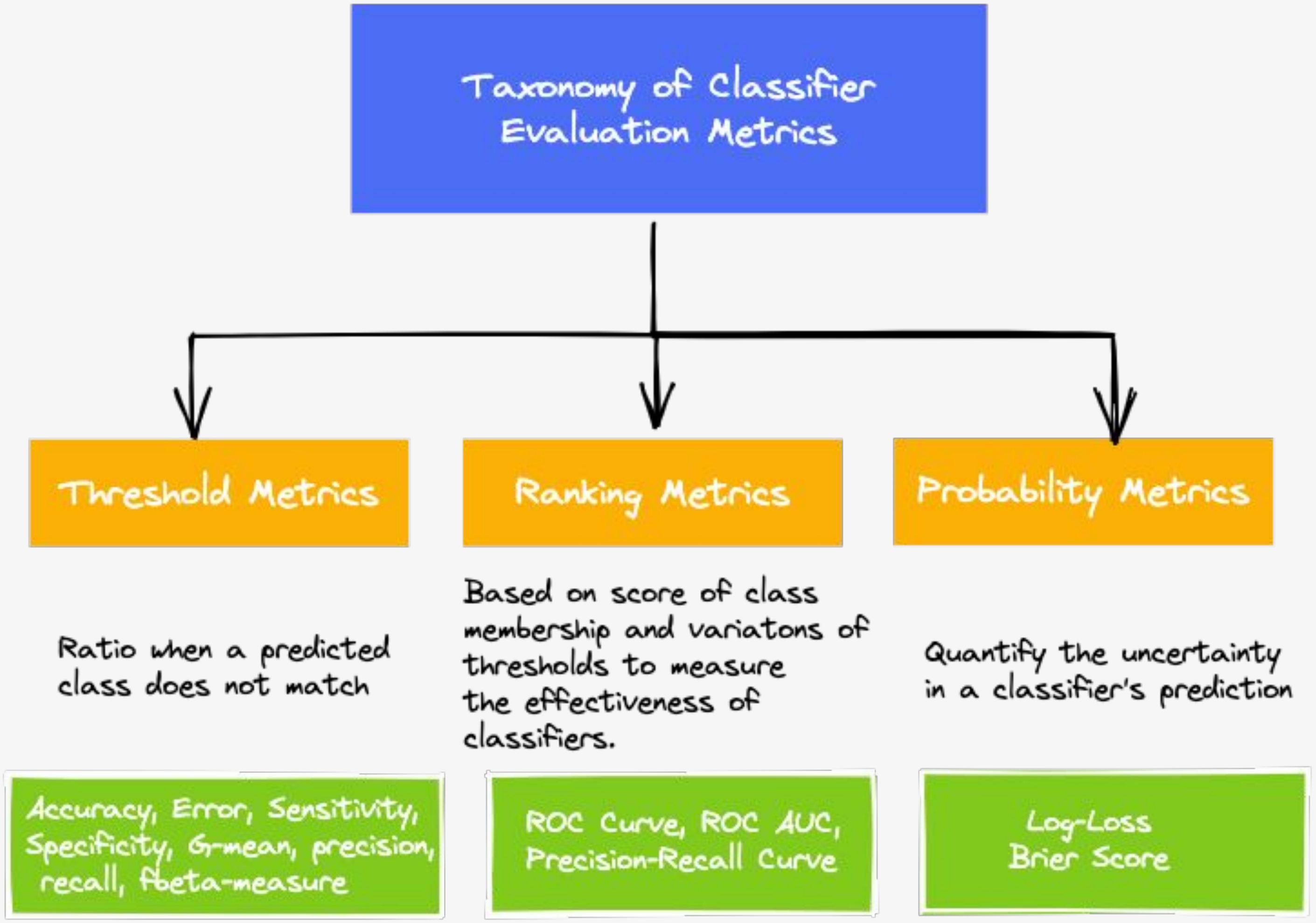
$$Gini(x) = 1 - \sum_{i=1}^{c} P(x_i)^2$$

$$Entropy(x) = -\sum_{i=1}^{c} P(x_i) \log_b P(x_i)$$

Gini index or Entropy is the criterion for calculating **Information Gain**. Both of them are measures of impurity of a node.

```python
from sklearn.tree import plot_tree
```





Ratio when a predicted class does not match

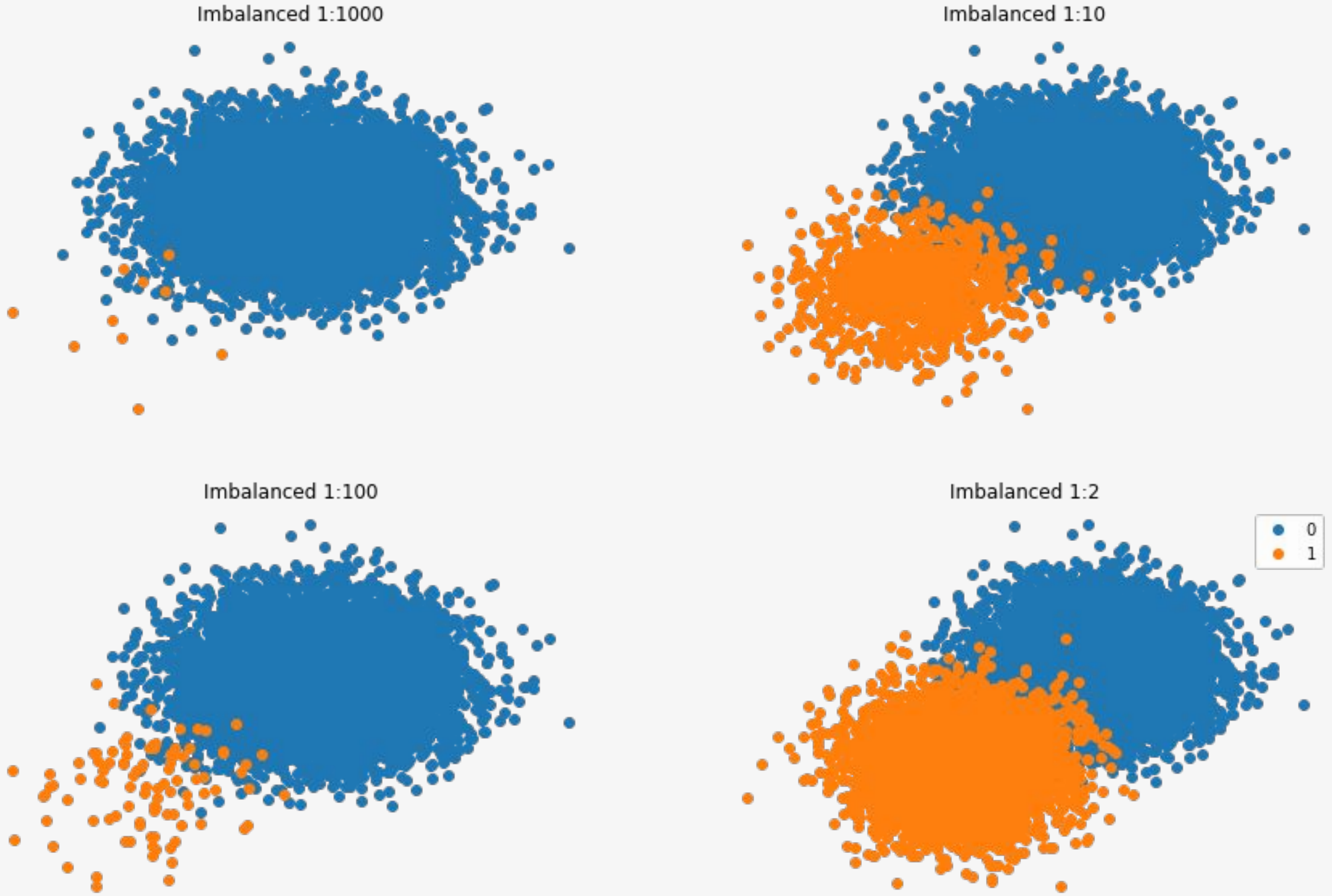
Based on score of class membership and variatons of thresholds to measure the effectiveness of classifiers.

Quantify the uncertainty in a classifier's prediction

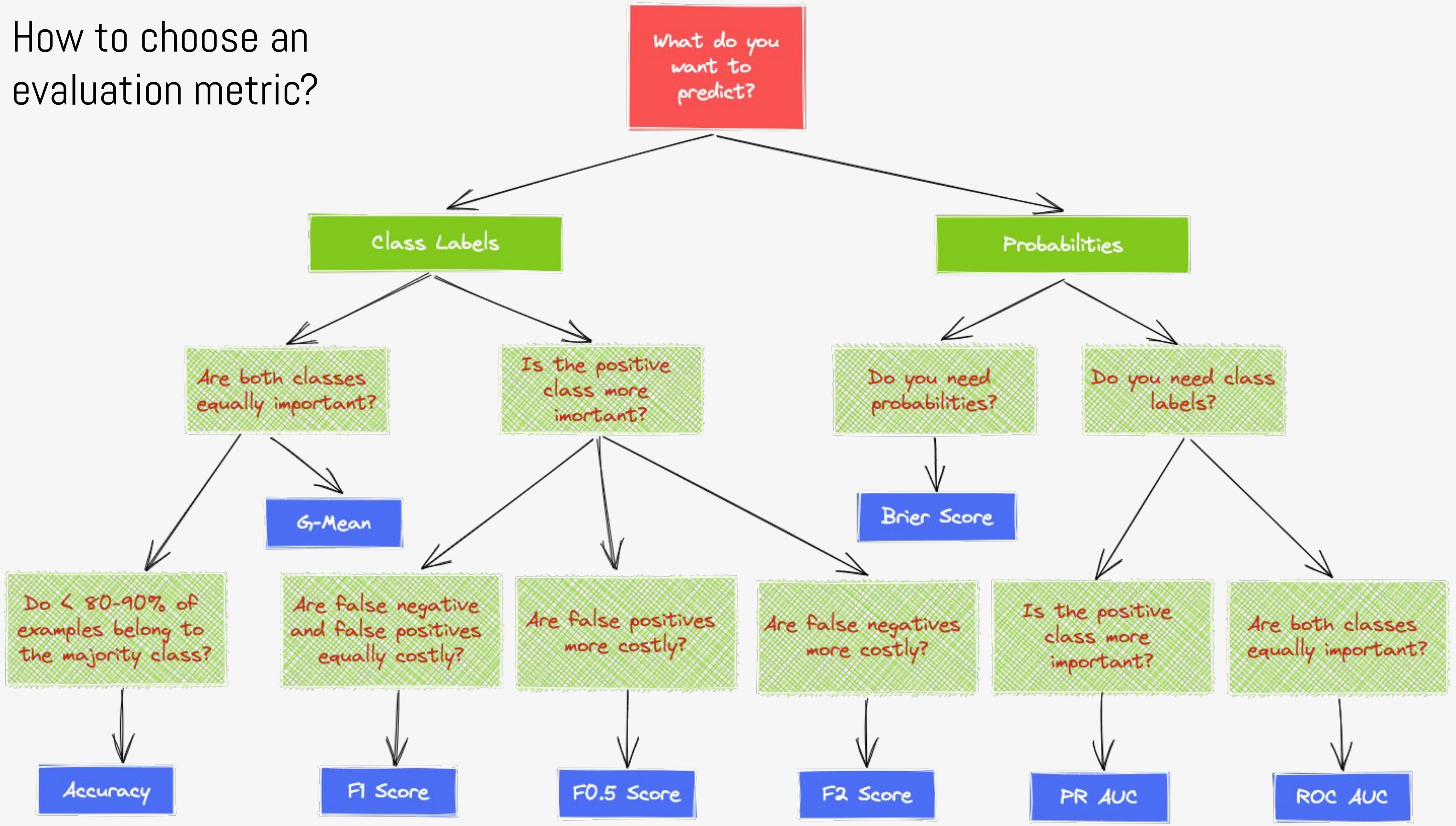
Accuracy, Error, Sensitivity, Specificity, G-mean, precision, recall, fbeta-measure

ROC Curve, ROC AUC, Precision-Recall Curve

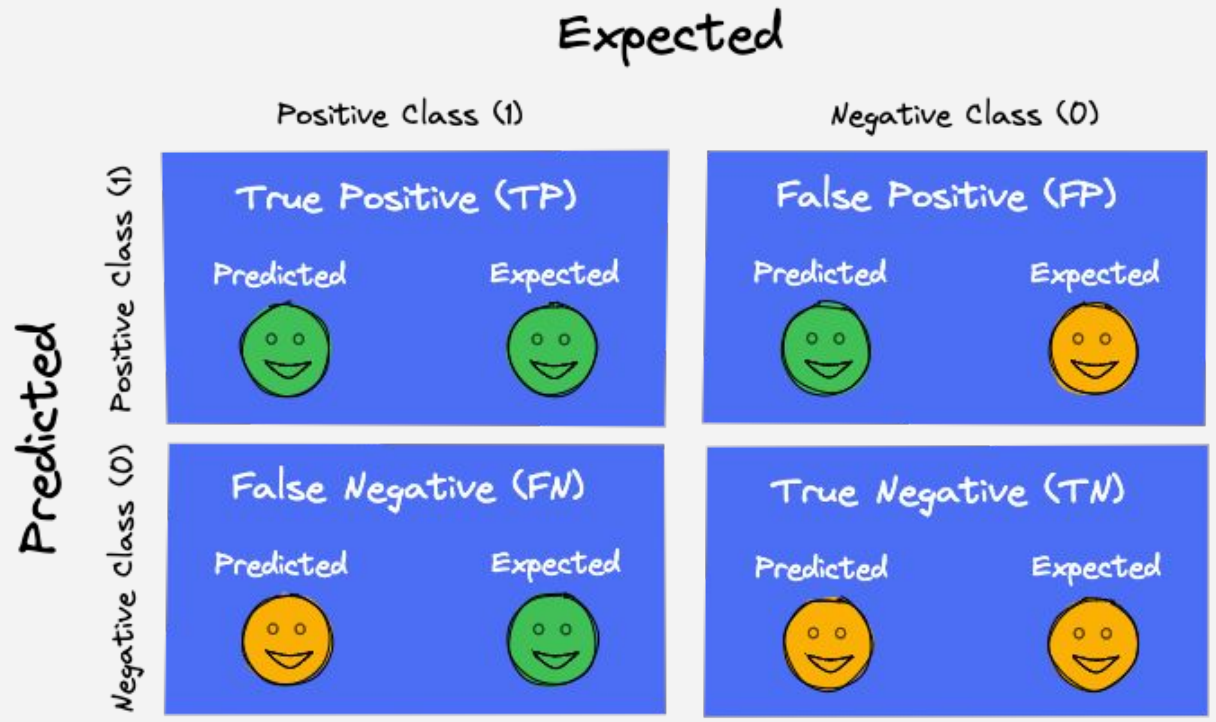
Log-Loss
Brier Score


Imbalanced 1:1000
Imbalanced 1:10
Imbalanced 1:100
Imbalanced 1:2
0
1

How to choose an evaluation metric?


What do you want to predict?
Class Labels
Probabilities
Are both classes equally important?
Is the positive class more imortant?
Do you need probabilities?
Do you need class labels?
G-Mean
Brier Score
Do < 80-90% of examples belong to the majority class?
Are false negative and false positives equally costly?
Are false positives more costly?
Are false negatives more costly?
Is the positive class more important?
Are both classes equally important?
Accuracy
F1 Score
F0.5 Score
F2 Score
PR AUC
ROC AUC

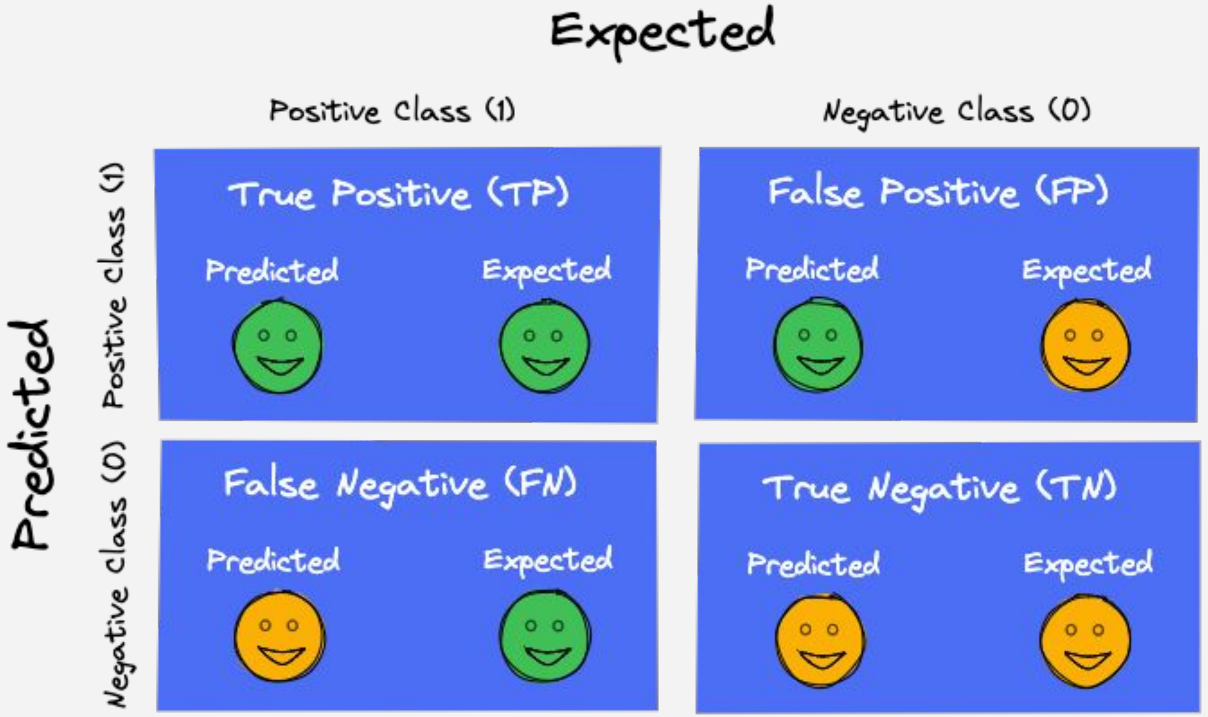
Confusion Matrix



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

$$\text{Error} = 1 - \text{Accuracy}$$

Confusion Matrix

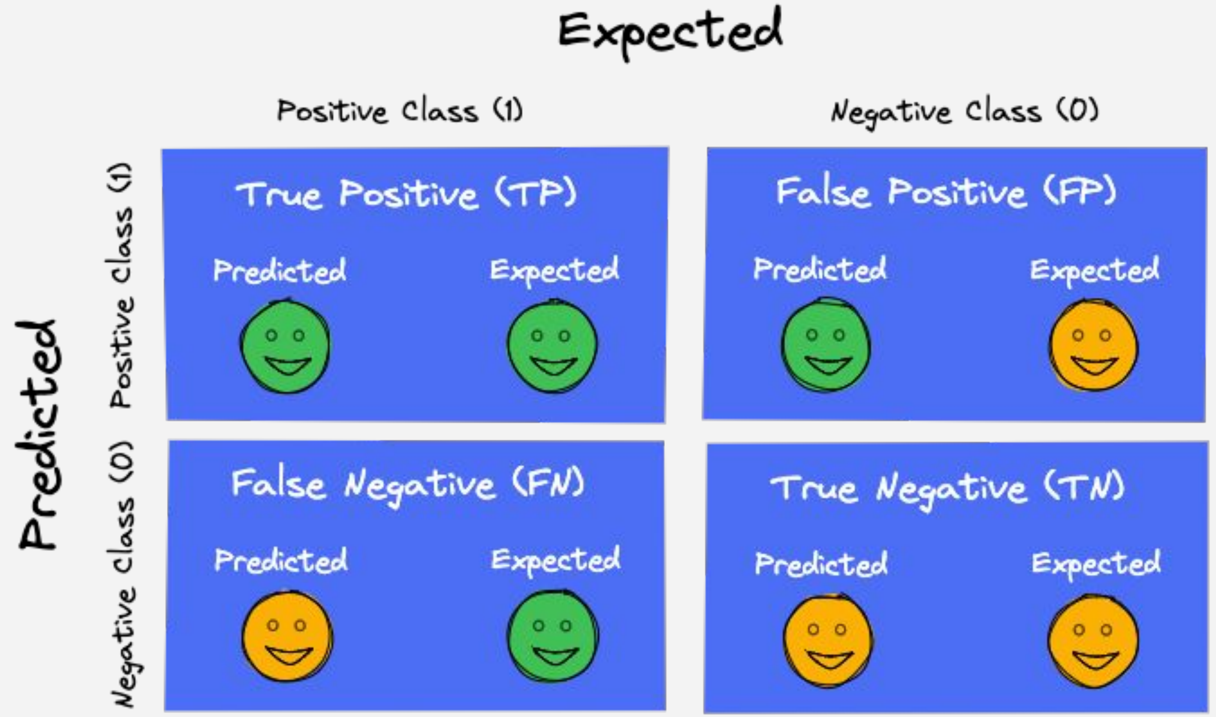


$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

$$\text{Specificity} = \frac{TN}{FP + TN}$$

$$\text{G-mean} = \sqrt{\text{Sensitivity} \times \text{Specificity}}$$

Confusion Matrix

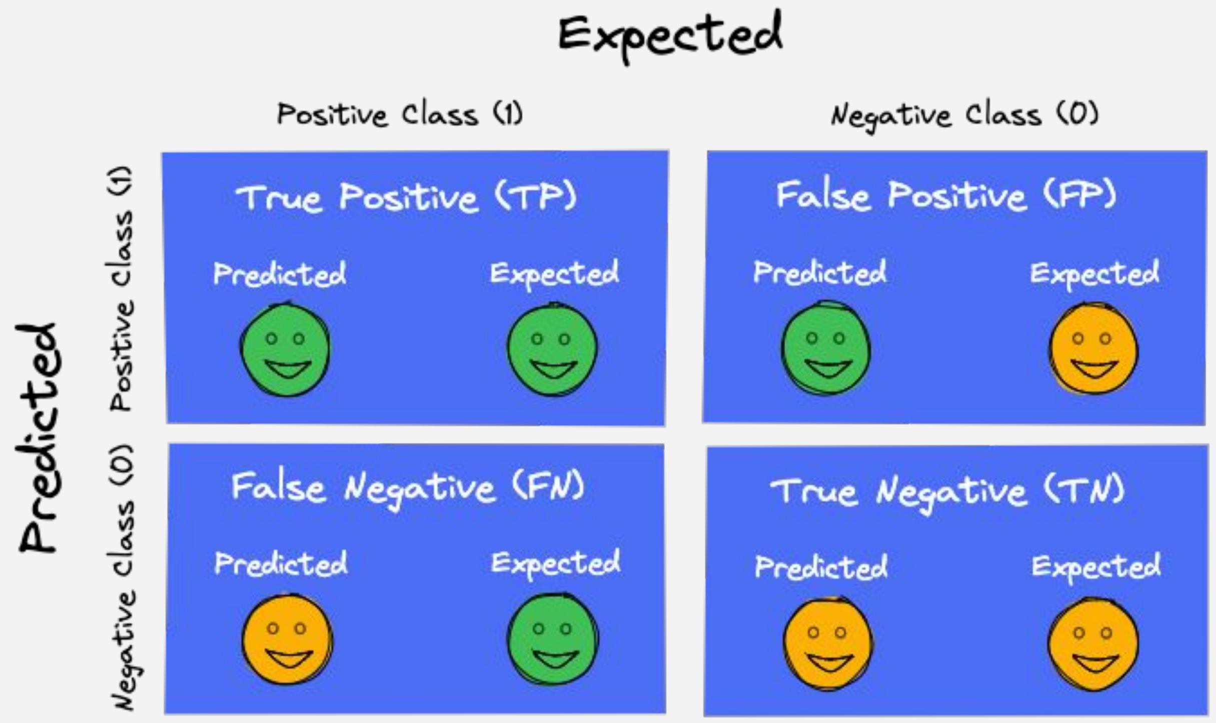


Precision = $\frac{TP}{TP + FP}$
(positive predicte value - PPV)

Precision = $\frac{TN}{TN + FN}$
(negative predicte value - NPV)

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

Confusion Matrix



$$\text{Fbeta-measure} = \frac{(1 + \beta^2) \times \text{Precision} \times \text{Recall}}{\beta^2 \times \text{Precision} + \text{Recall}}$$

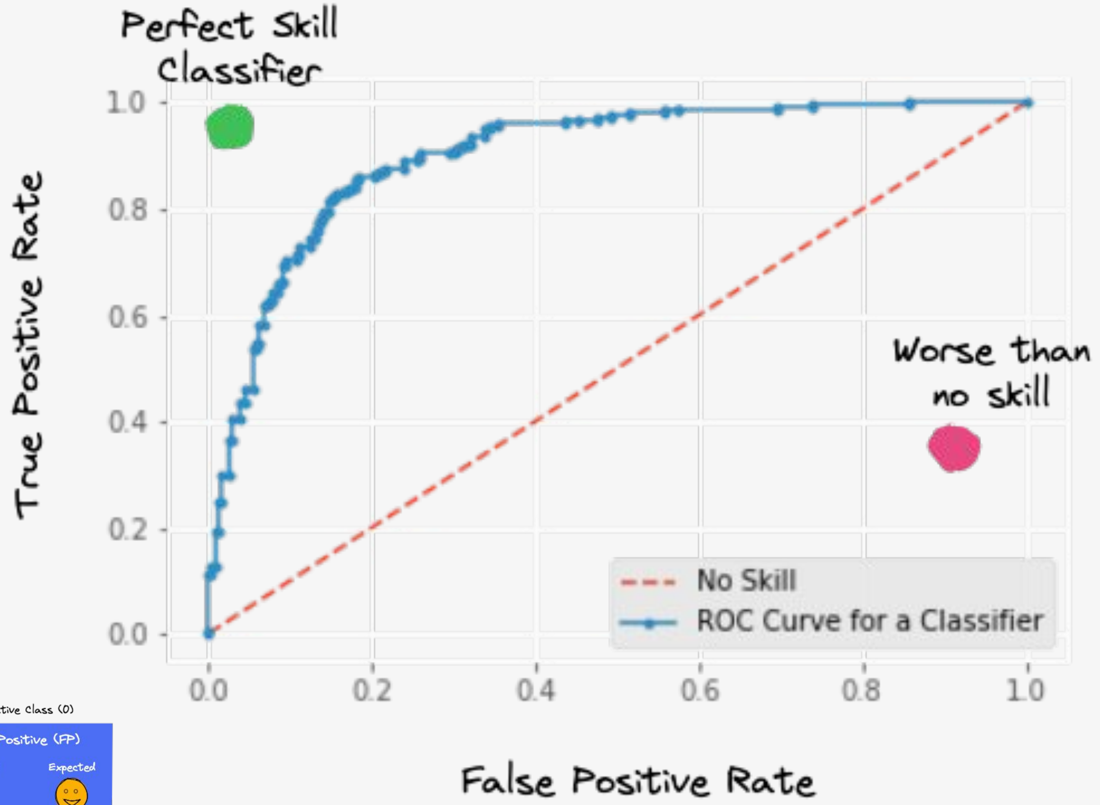
$$\beta == \begin{cases} 0.5, & \text{more weight on precision} \\ 1.0, & \text{balance on weight PR and RE} \\ 2.0, & \text{less weight on precision} \end{cases}$$

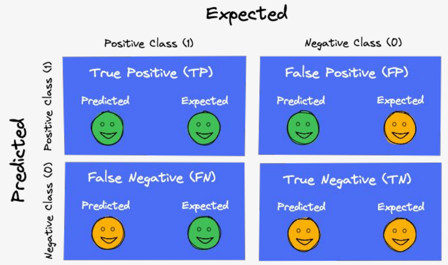
Rank metrics are more concerned with evaluating classifiers based on **how effective** they are at separating classes.

These metrics require that a **classifier predicts a score** or a probability of class membership. From this score, **different thresholds** can be applied to **test the effectiveness of classifiers**. Those models that maintain a good score across a range of thresholds will have good class separation and will be ranked higher.

Receiver Operating Characteristic (ROC)

$$TPR = \frac{TP}{TP + FN}$$

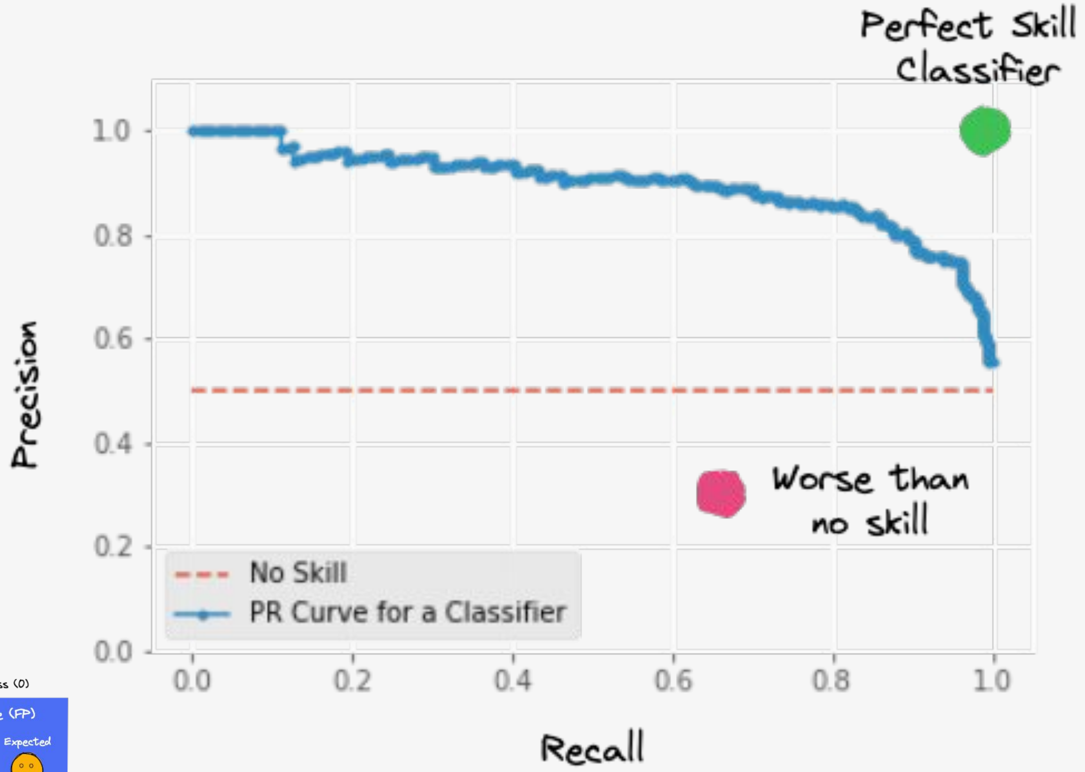




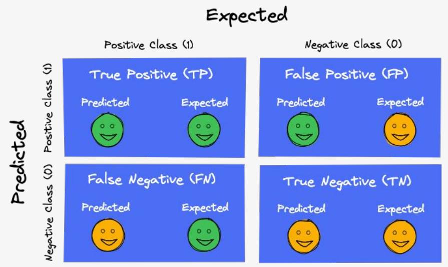
$$FPR = \frac{FP}{FP + TN}$$

Precision-Recall (PR) Curve

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



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



Hands ON

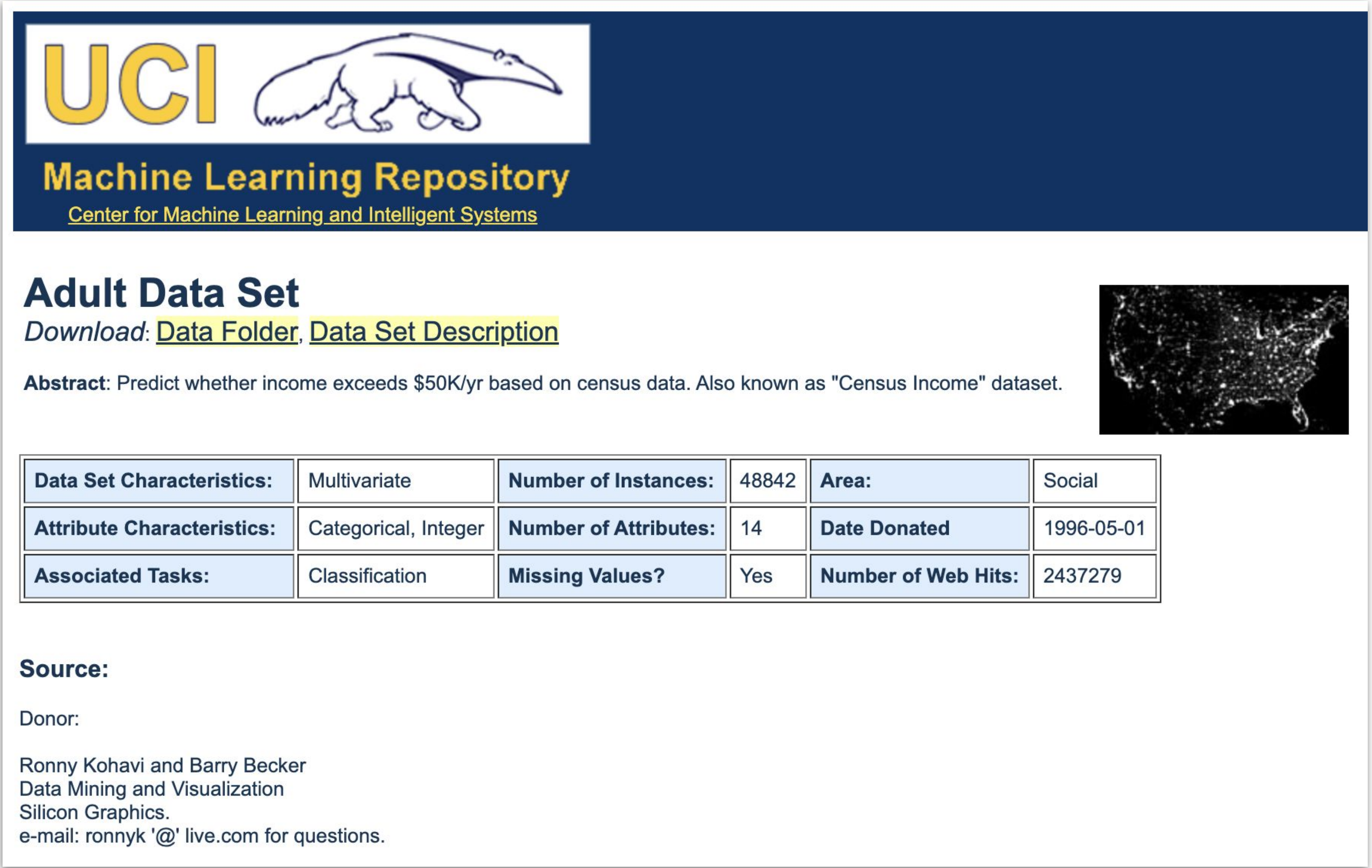
Python

Streamlit ANACONDA CONDA

Optional

matplotlib pandas seaborn plotly

W&B Jupyter scikit learn



UCI Machine Learning Repository

Center for Machine Learning and Intelligent Systems

Adult Data Set

Download: Data Folder, Data Set Description

Abstract: Predict whether income exceeds $50K/yr based on census data. Also known as "Census Income" dataset.

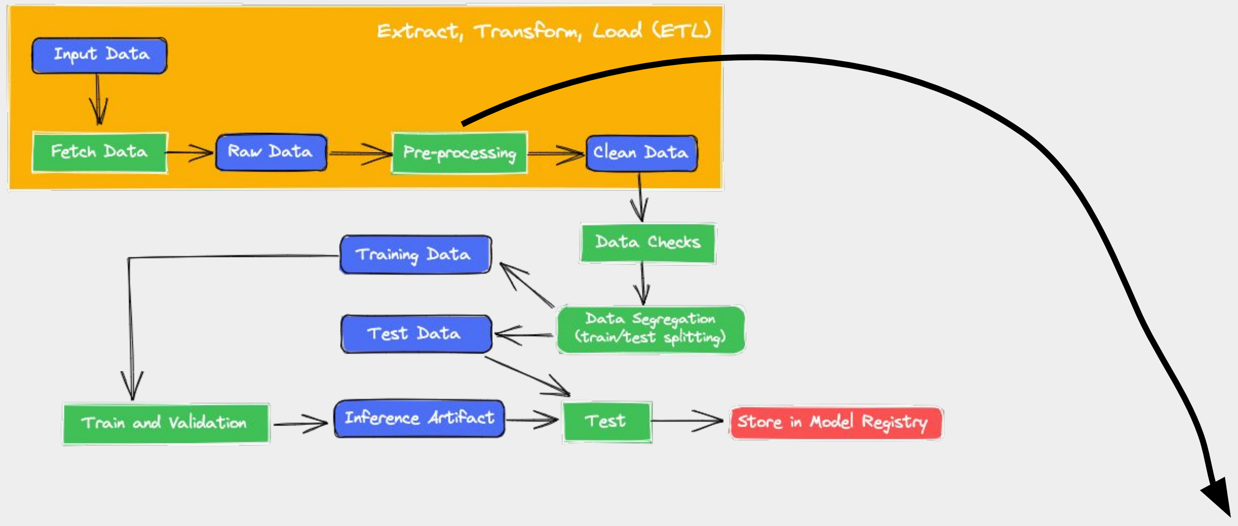
Data Set Characteristics:	Multivariate	**Number of Instances:**	48842	**Area:**	Social
Attribute Characteristics:	Categorical, Integer	**Number of Attributes:**	14	**Date Donated**	1996-05-01
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	2437279

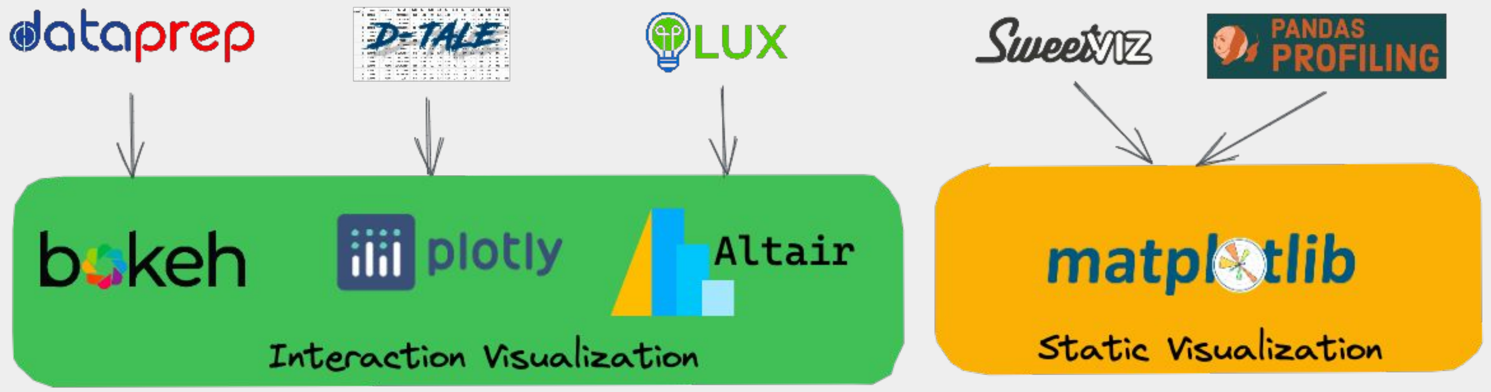
Source:

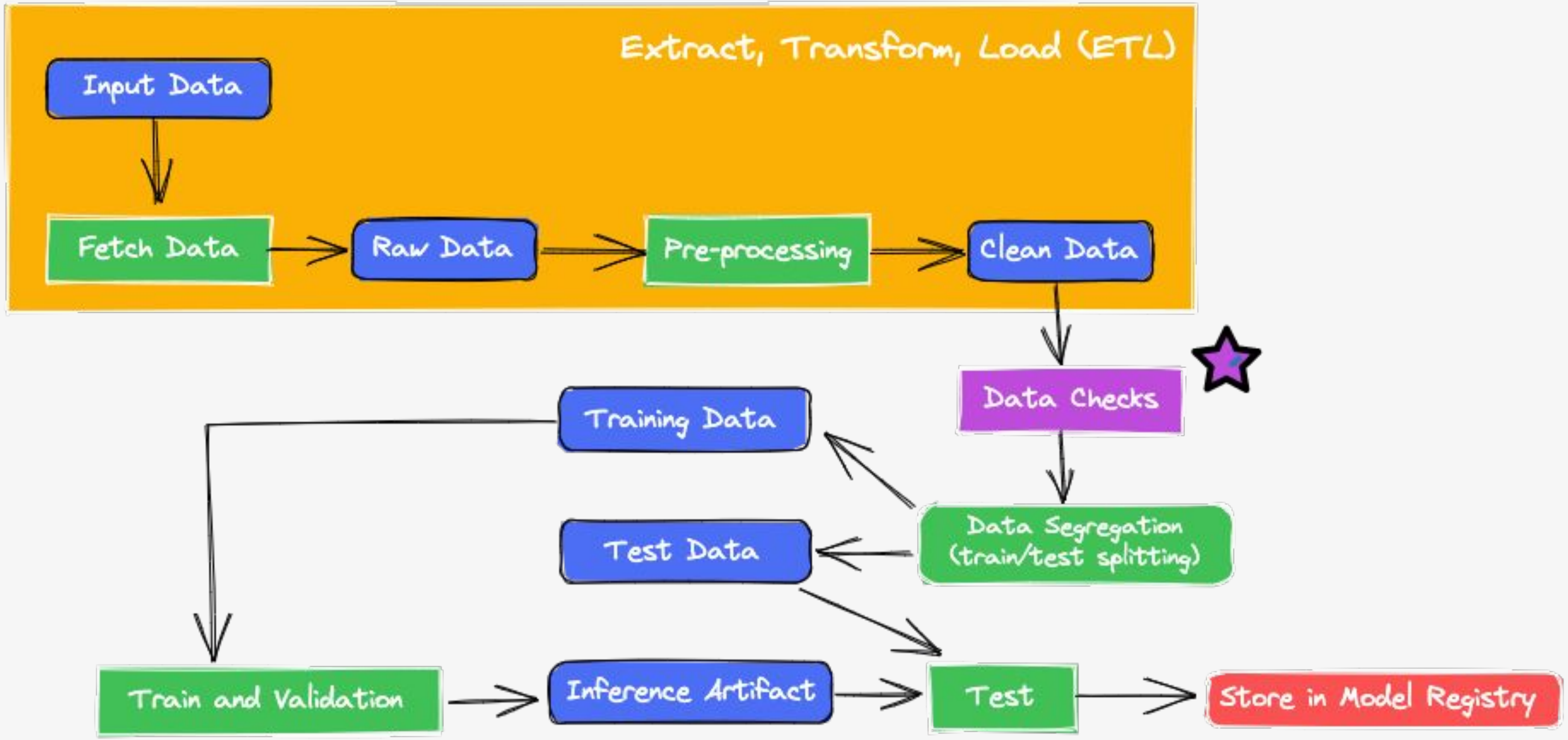
Donor:

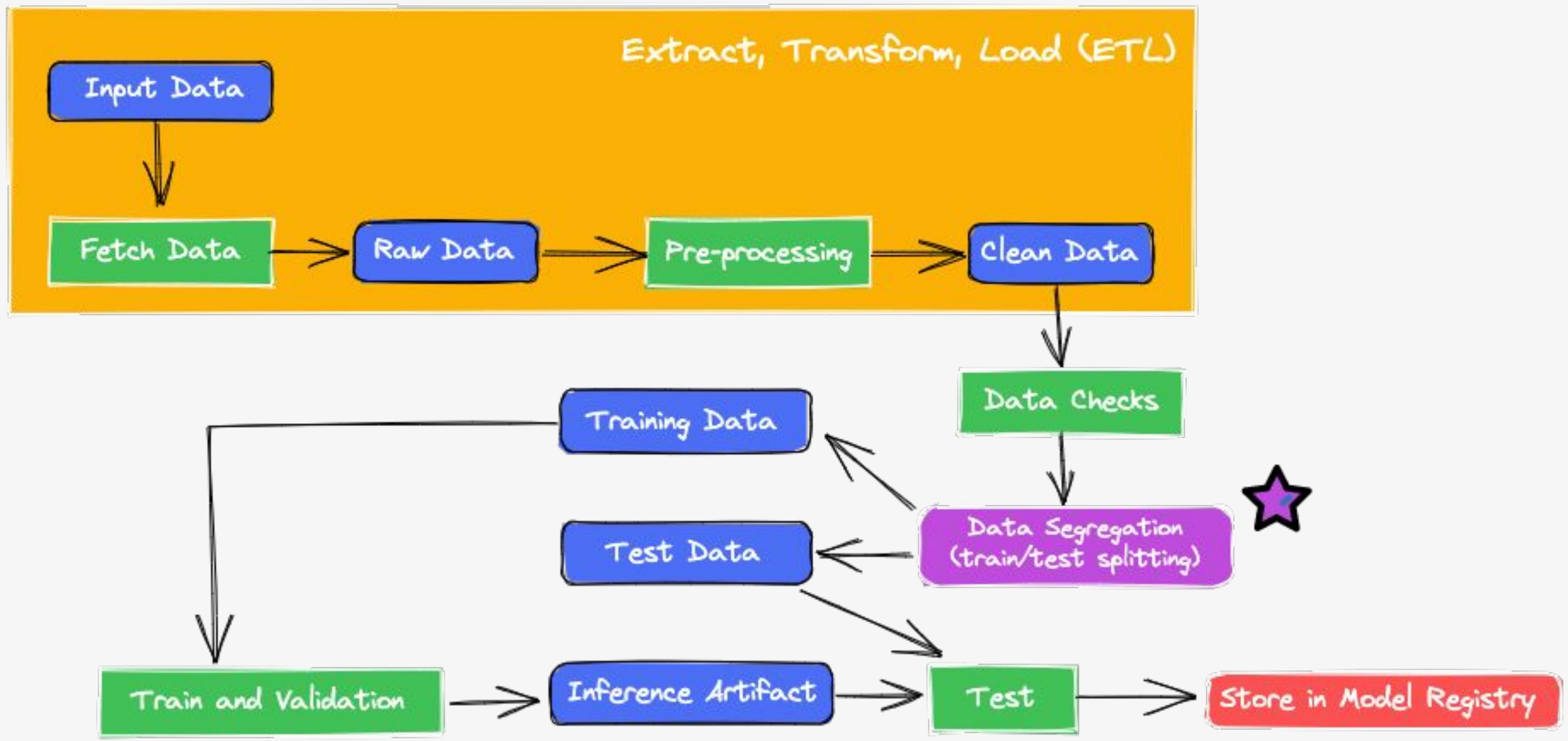
Ronny Kohavi and Barry Becker
Data Mining and Visualization
Silicon Graphics.
e-mail: ronnyk '@' live.com for questions.

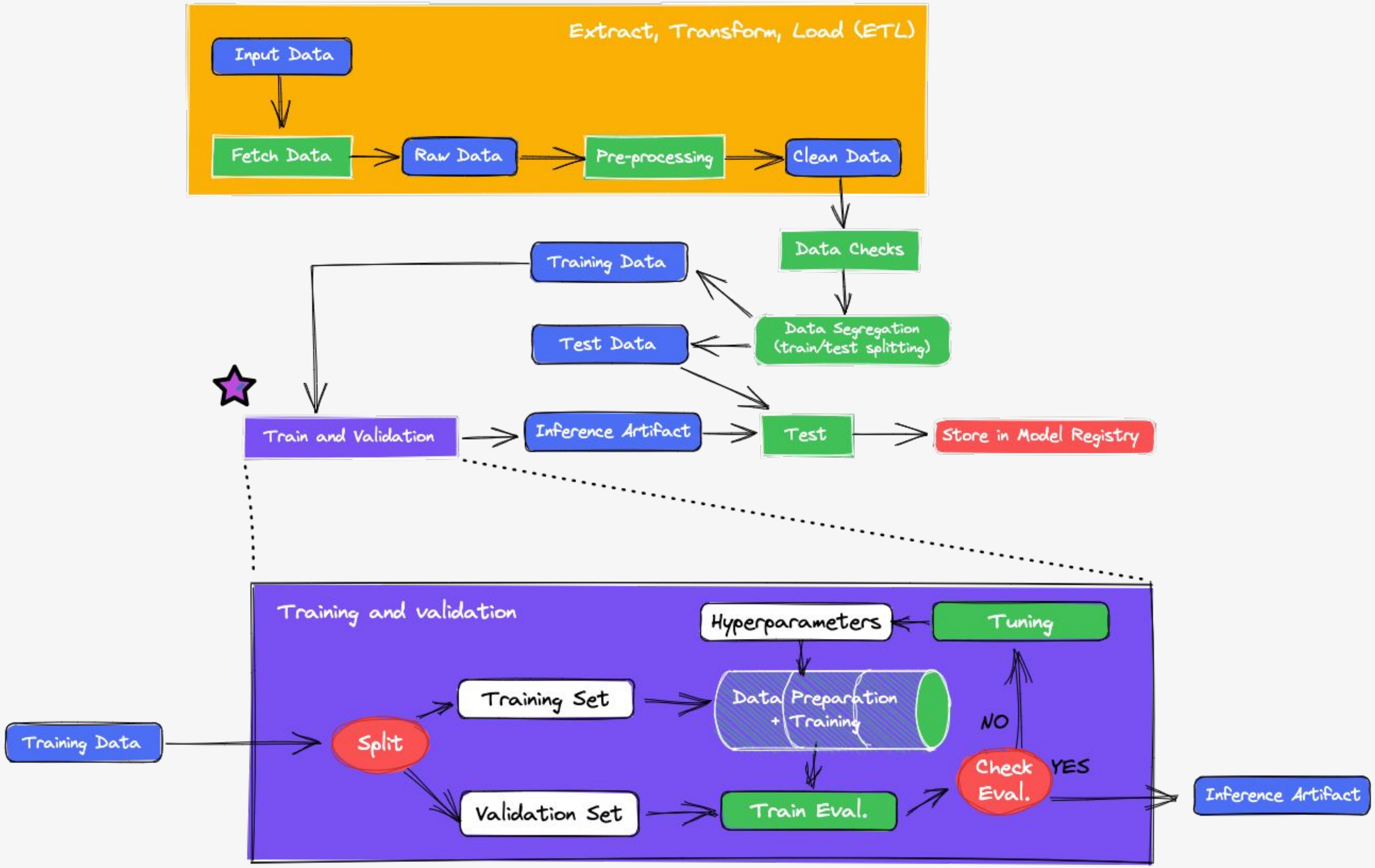
Exploratory Data Analysis

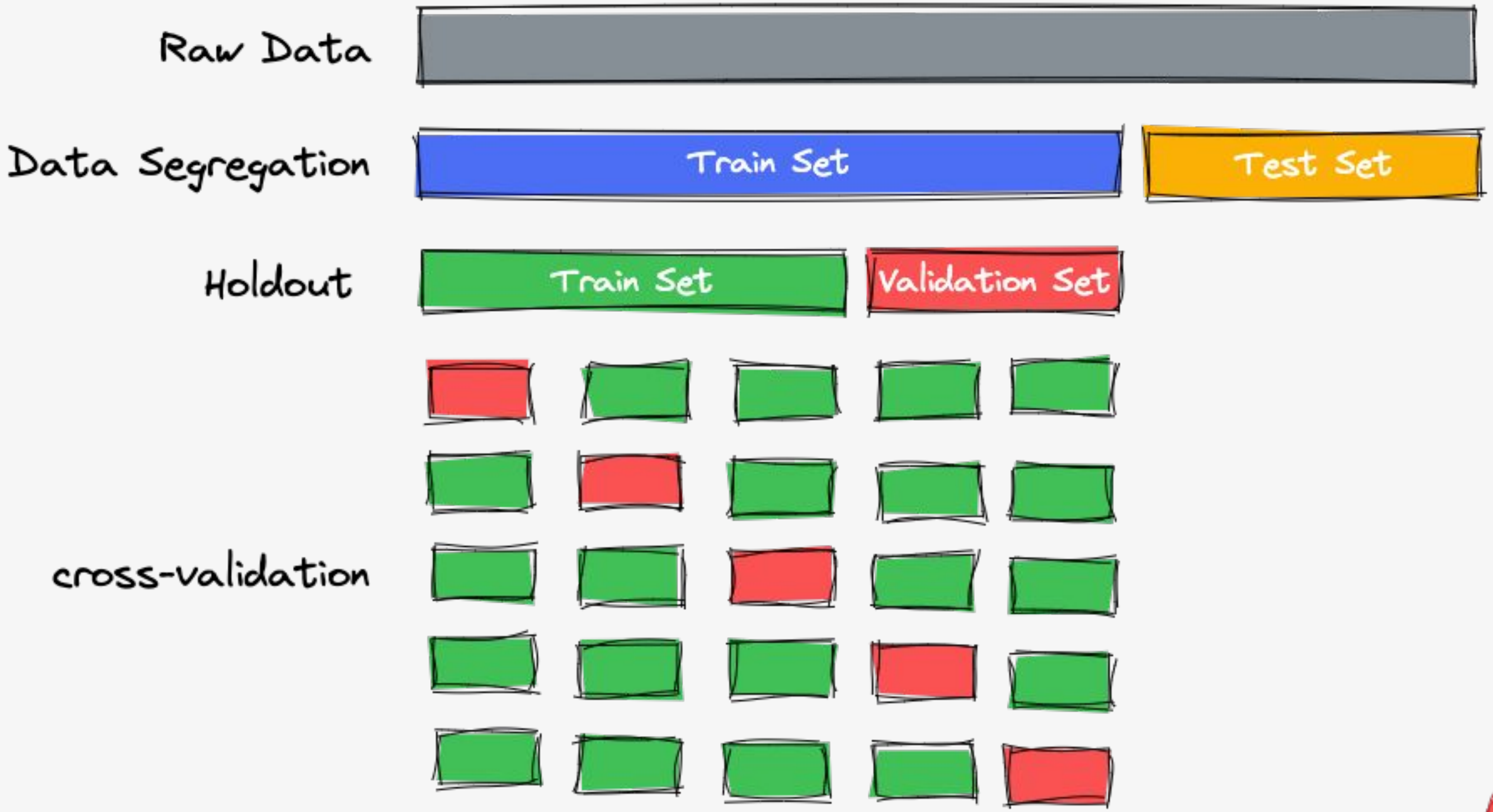




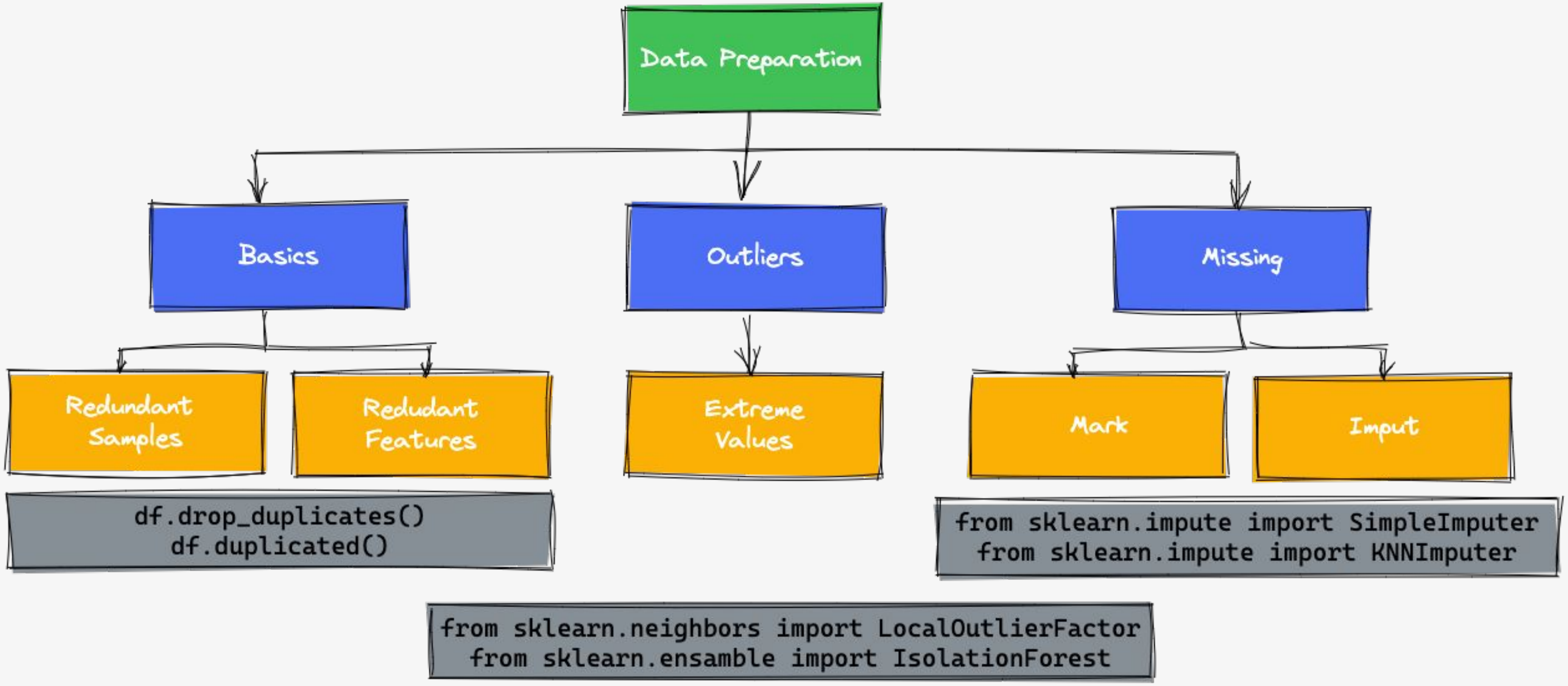

Extract, Transform, Load (ETL)
Input Data
Fetch Data
Raw Data
Pre-processing
Clean Data
Data Checks
Training Data
Test Data
Data Segregation (train/test splitting)
Train and Validation
Inference Artifact
Test
Store in Model Registry


Extract, Transform, Load (ETL)
Input Data
Fetch Data
Raw Data
Pre-processing
Clean Data
Data Checks
Training Data
Data Segregation (train/test splitting)
Test Data
Train and Validation
Inference Artifact
Test
Store in Model Registry


Extract, Transform, Load (ETL)
Input Data
Fetch Data
Raw Data
Pre-processing
Clean Data
Data Checks
Training Data
Data Segregation (train/test splitting)
Test Data
Train and Validation
Inference Artifact
Test
Store in Model Registry
Training and validation
Hyperparameters
Tuning
Training Data
Split
Training Set
Validation Set
Data Preparation + Training
Train Eval.
NO
Check Eval.
YES
Inference Artifact


Raw Data
Data Segregation
Train Set
Test Set
Holdout
Train Set
Validation Set
cross-validation

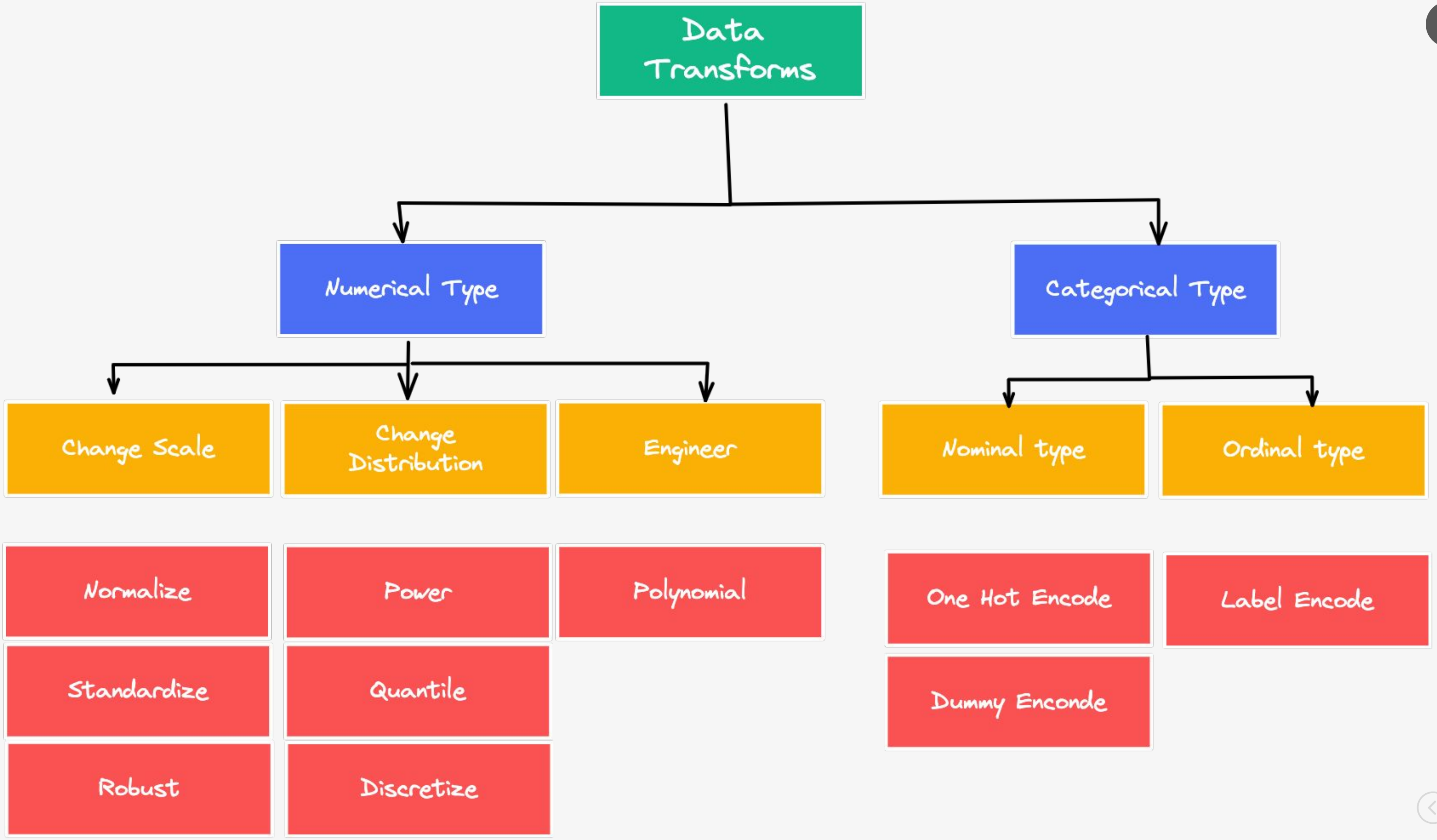
Split

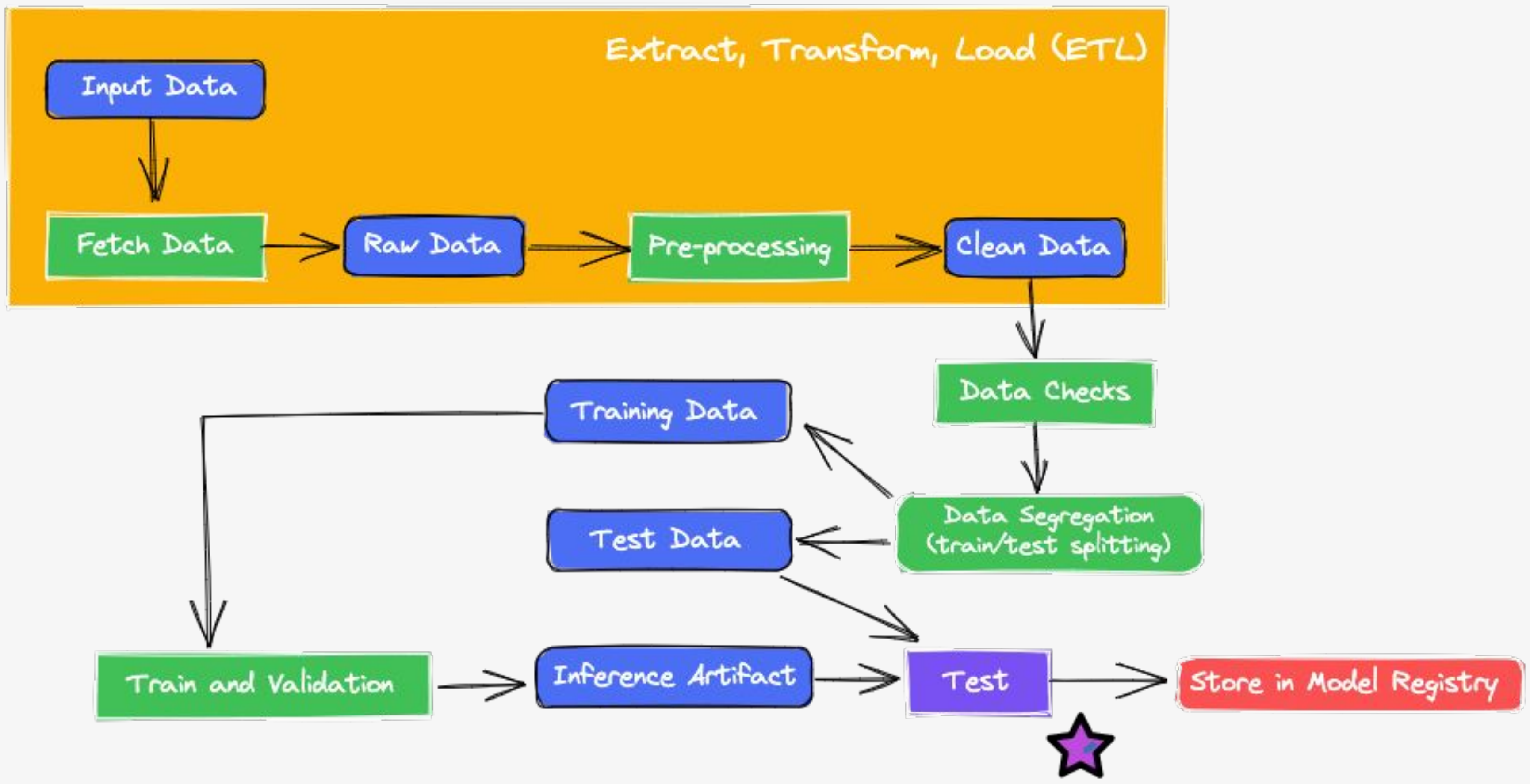


```
df.drop_duplicates()
df.duplicated()
```

```
from sklearn.impute import SimpleImputer
from sklearn.impute import KNNImputer
```

```
from sklearn.neighbors import LocalOutlierFactor
from sklearn.ensamble import IsolationForest
```


Data Transforms
Numerical Type
Categorical Type
Change Scale
Change Distribution
Engineer
Nominal type
Ordinal type
Normalize
Standardize
Robust
Power
Quantile
Discretize
Polynomial
One Hot Encode
Dummy Enconde
Label Encode


Extract, Transform, Load (ETL)
Input Data
Fetch Data
Raw Data
Pre-processing
Clean Data
Data Checks
Data Segregation (train/test splitting)
Training Data
Test Data
Train and Validation
Inference Artifact
Test
Store in Model Registry