DigitalSkola

LEARNING PROGRESS REVIEW (LPR) - WEEK 11

By: Omicron

TEAM OMICRON MEMBERS

1

Anugrah Yazid Ghani

https://www.linkedin.com/in/anugrah-yazid-7253bb221/

2

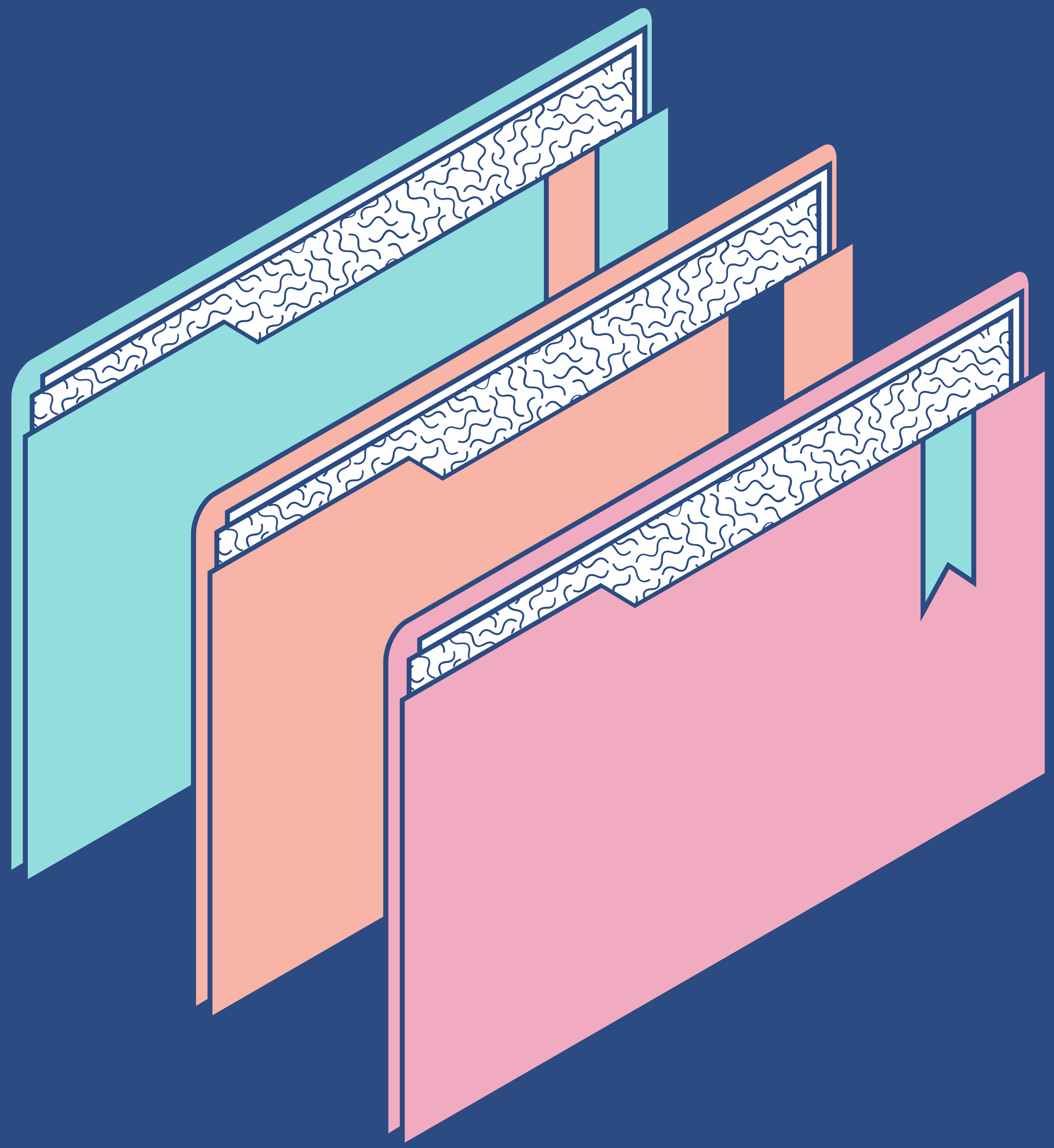
Edo Mohammad Hadad Gibran

https://www.linkedin.com/in/edo-gibran-38505a142/

3

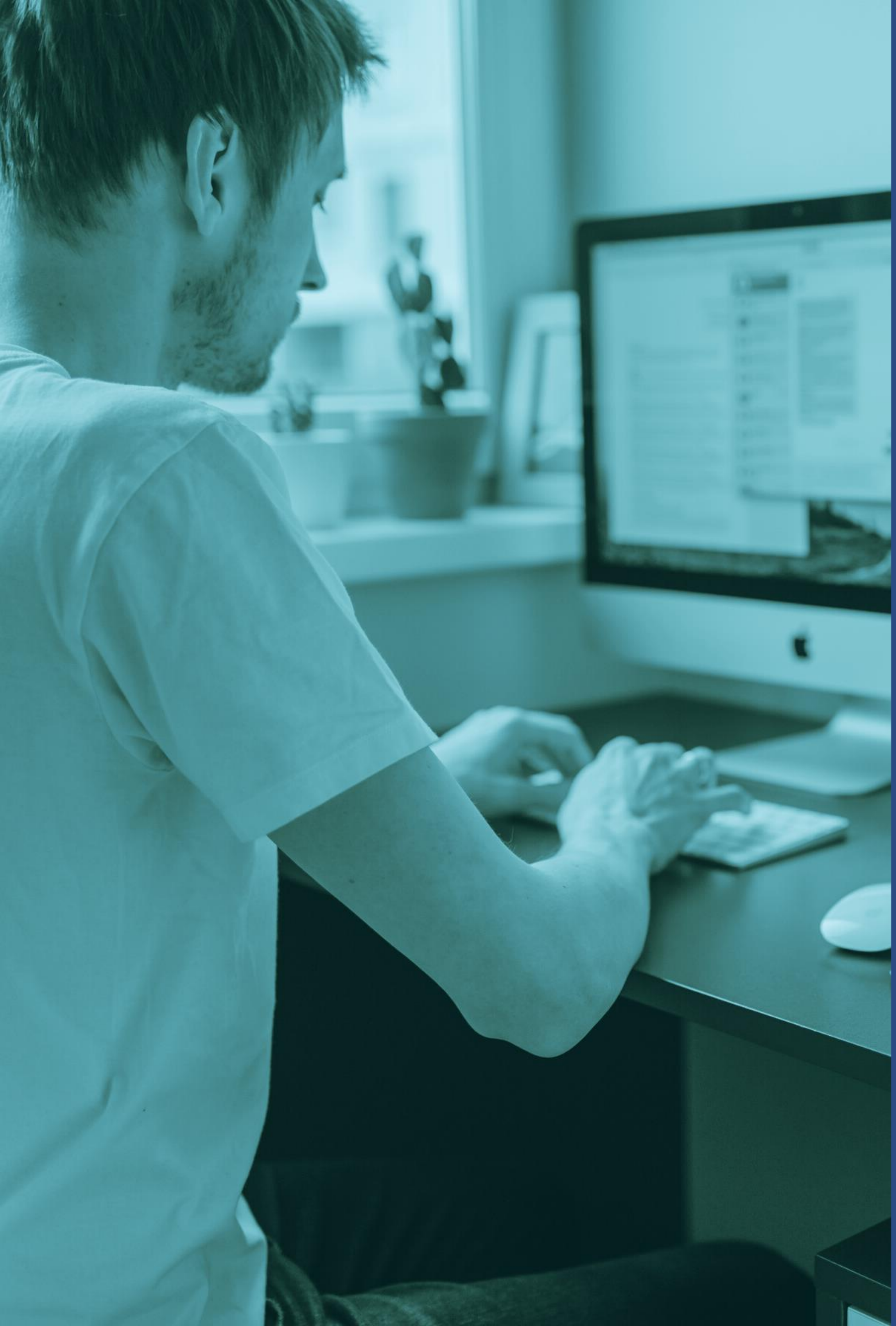
Muhammad Fikri Fadila

https://www.linkedin.com/in/muhammad-fikri-fadila-a551161a6/



DAFTAR ISI

1. Advanced-Data Preprocessing for ML
2. Classification I
3. Classification II

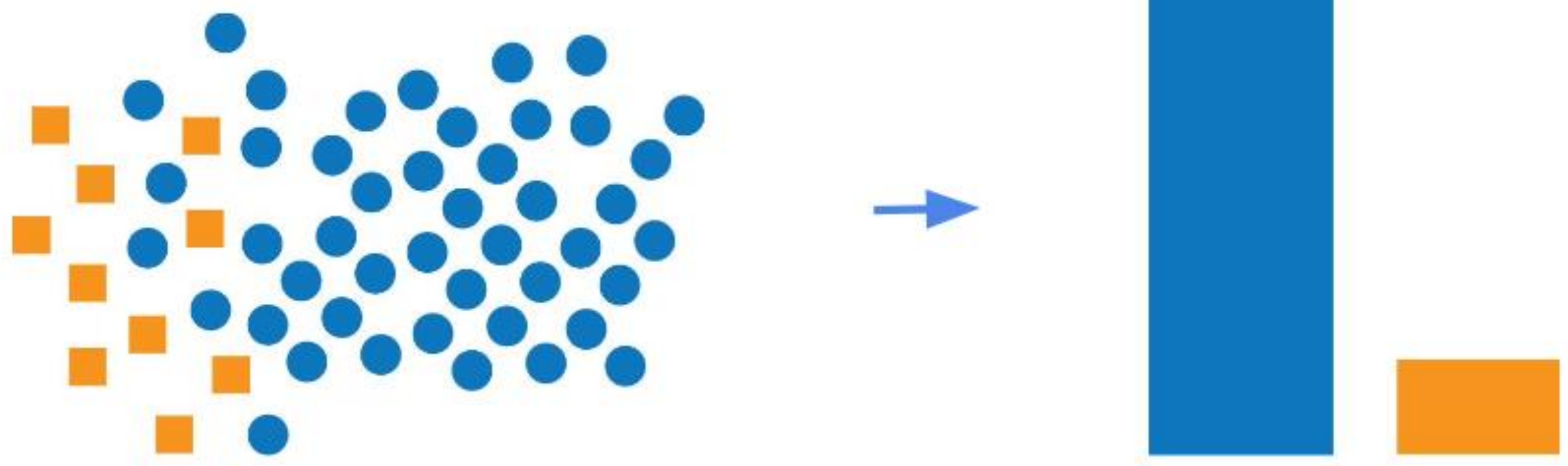


DigitalSkola

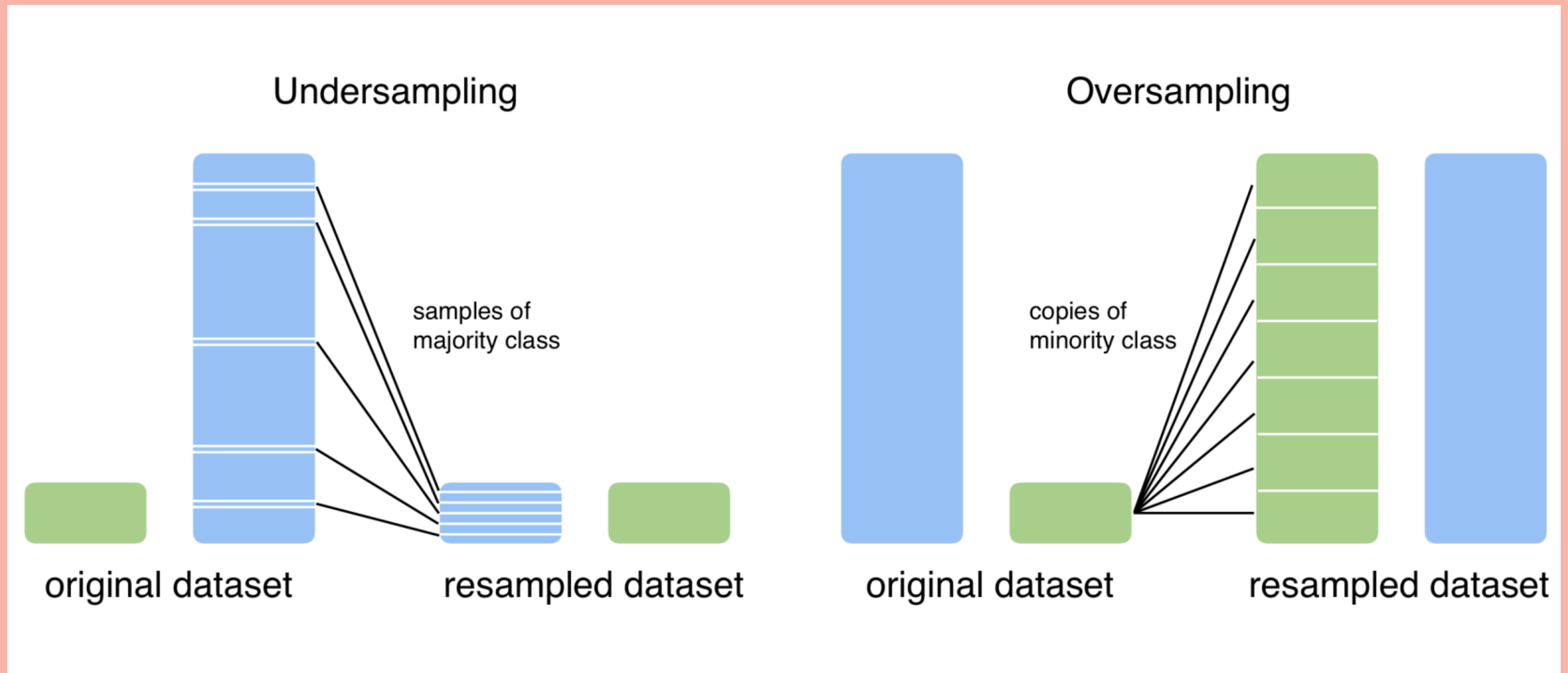
Advanced-Data Preprocessing for ML

Imbalanced Dataset

Masalah klasifikasi di mana jumlah data per kelas tidak terdistribusi merata.



How To Handle Imbalanced Dataset



Undersampling

Menyeimbangkan distribusi kelas dengan menghilangkan data dari kelas mayoritas secara acak.

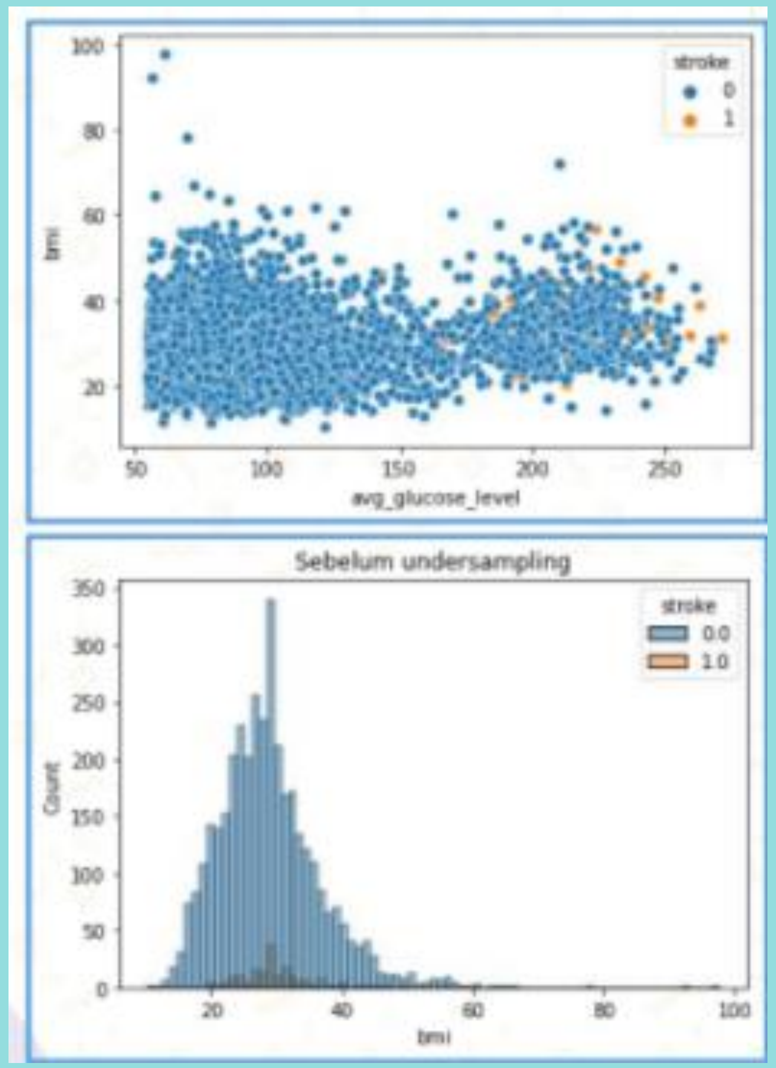
Oversampling

Meningkatkan jumlah instance di kelas minoritas dengan mereplikasikannya secara acak.

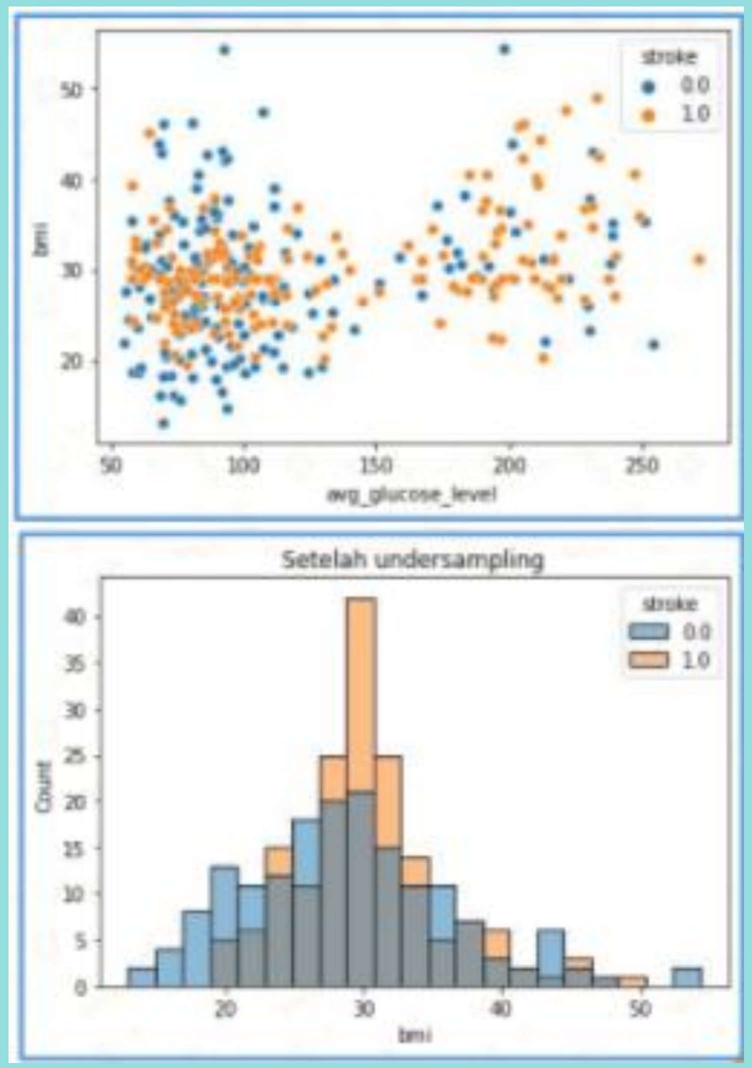
Random Undersampling

Membandingkan training set sebelum dan sesudah dilakukan random undersampling.

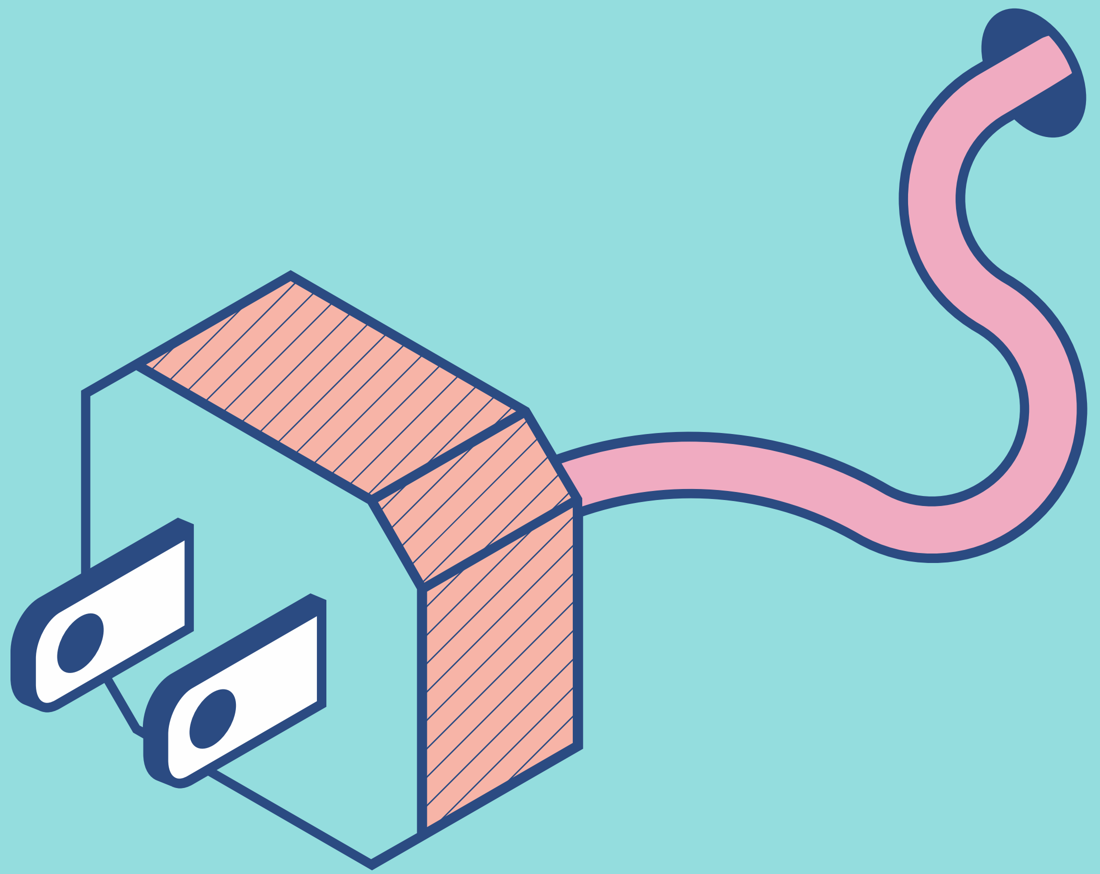
Sebelum



Sesudah



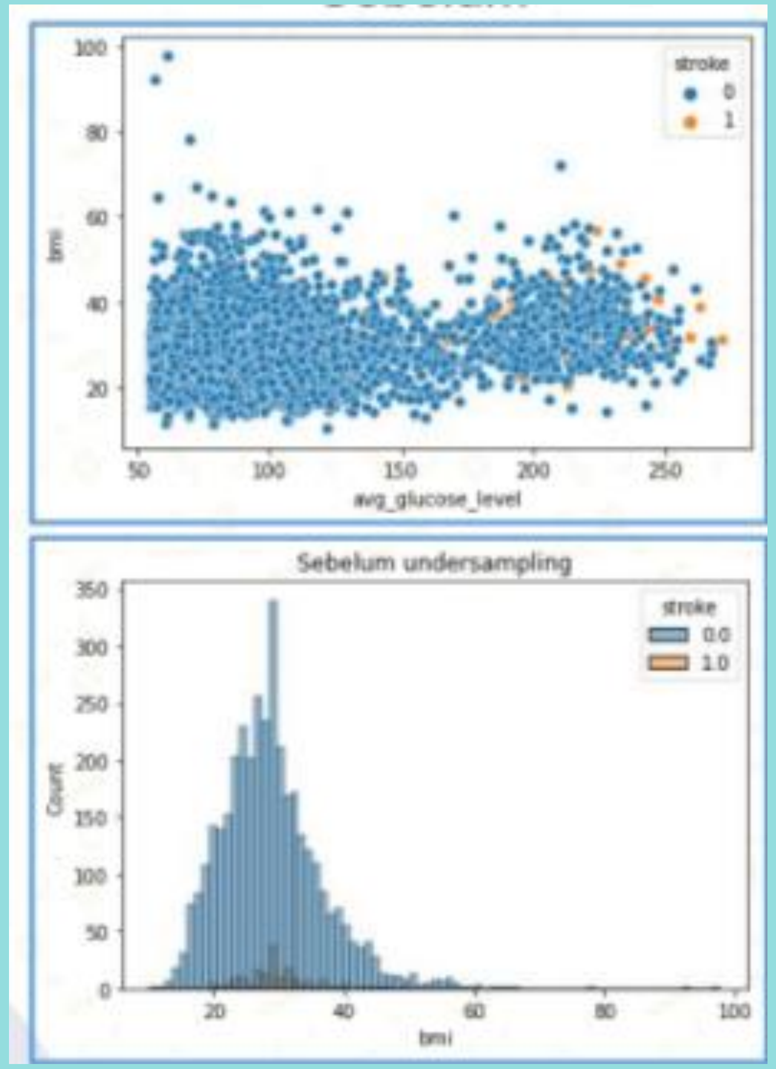

```
Sebelum undersampling
0.0    3663
1.0     169
dtype: int64
Setelah undersampling
1.0    169
0.0    169
dtype: int64
```



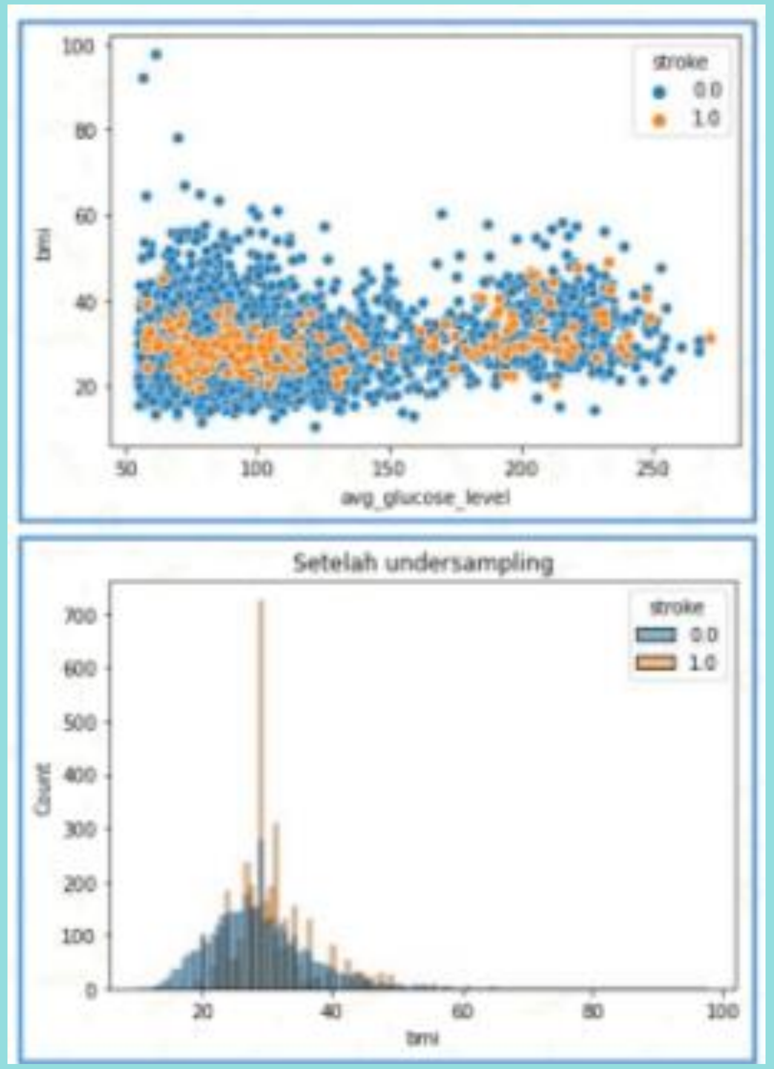
Random Oversampling

Membandingkan training set sebelum dan sesudah dilakukan random oversampling.

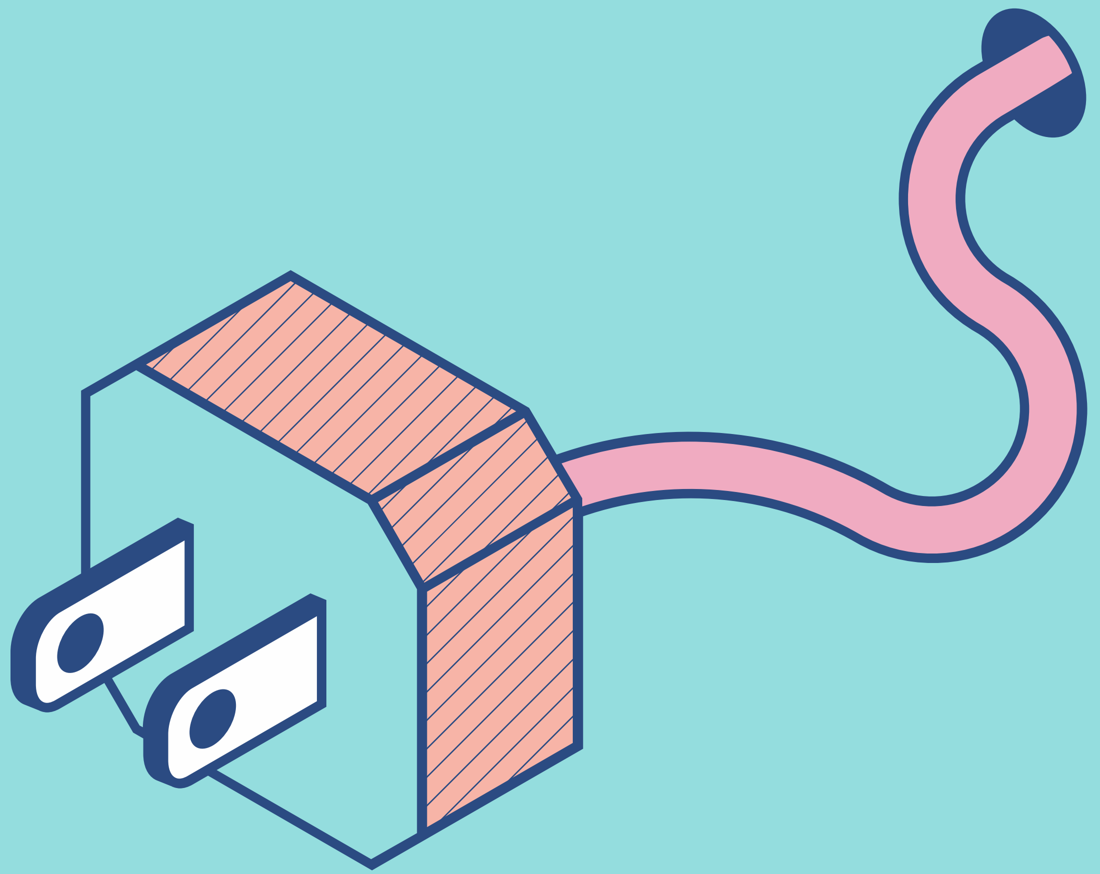
Sebelum



Sesudah




```
Sebelum oversampling
0.0     3663
1.0      169
dtype: int64
Setelah oversampling
1.0     3663
0.0     3663
dtype: int64
```

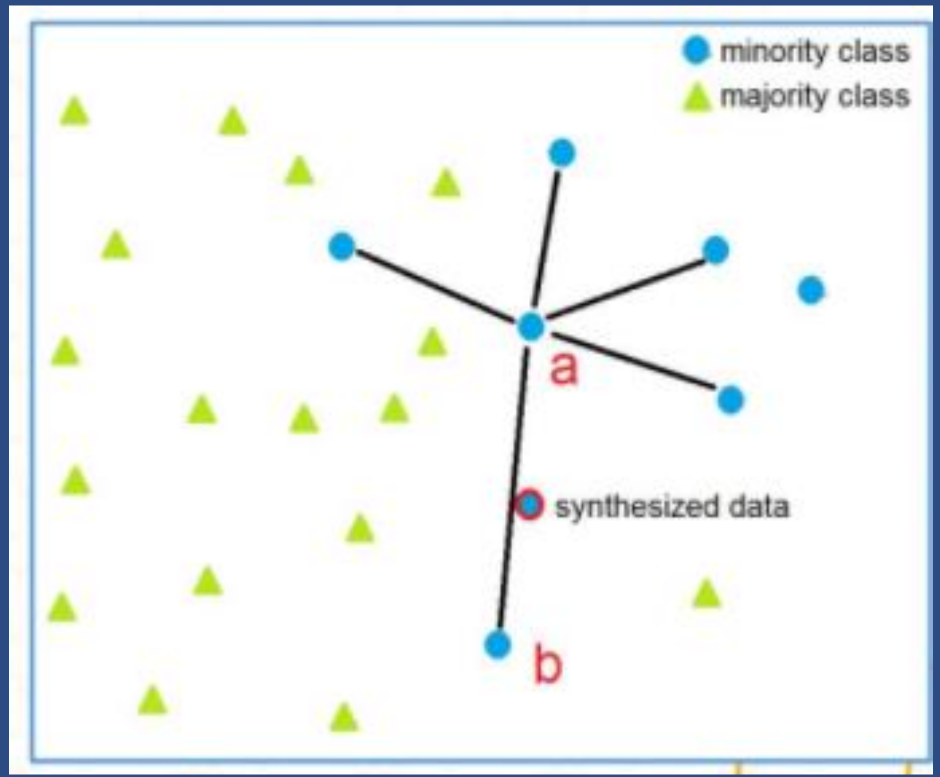


Synthetic Minority Oversampling Technique (SMOTE)

Pendekatan oversampling yang menciptakan sample kelas minoritas secara sintetik.

Cara Kerja :

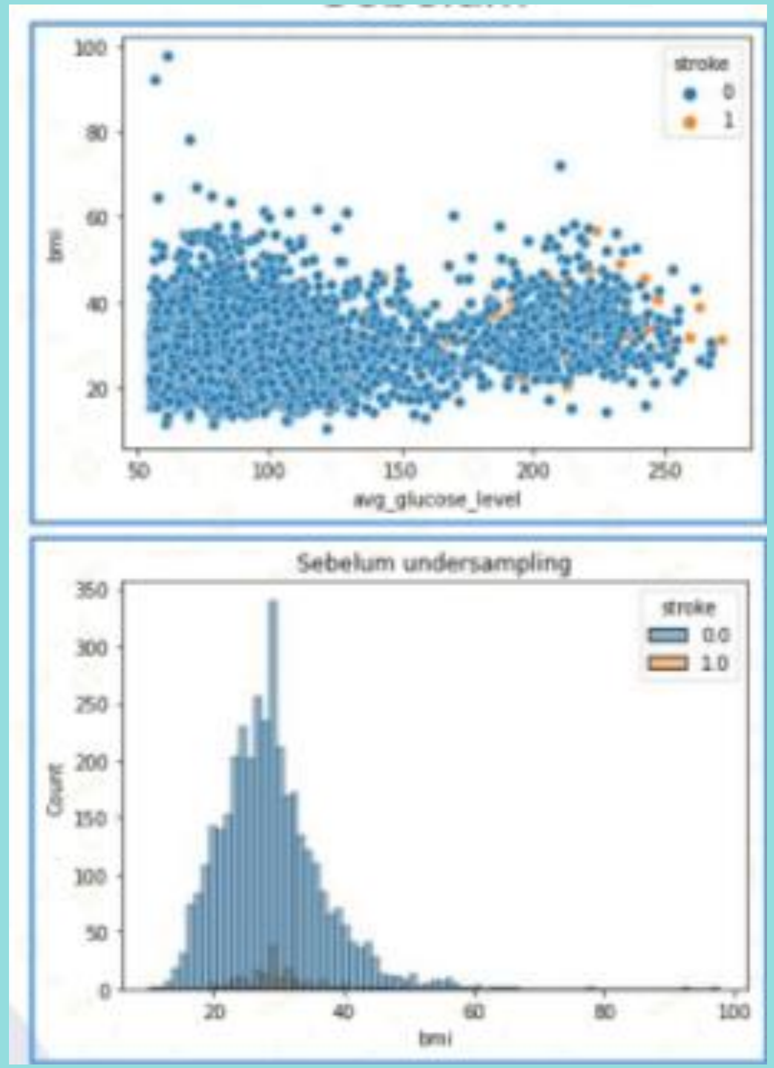
1. Contoh acak dari kelas minoritas **a** dipilih terlebih dahulu.
2. Kemudian k dari tetangga terdekat (*nearest neighbour*) untuk contoh tersebut ditentukan. (biasanya k=5).
3. Tetangga **b** yang dipilih secara acak.
4. Data sintetik **c** dibuat pada titik yang dipilih secara acak di antara dua data tersebut (**a** dan **b**).



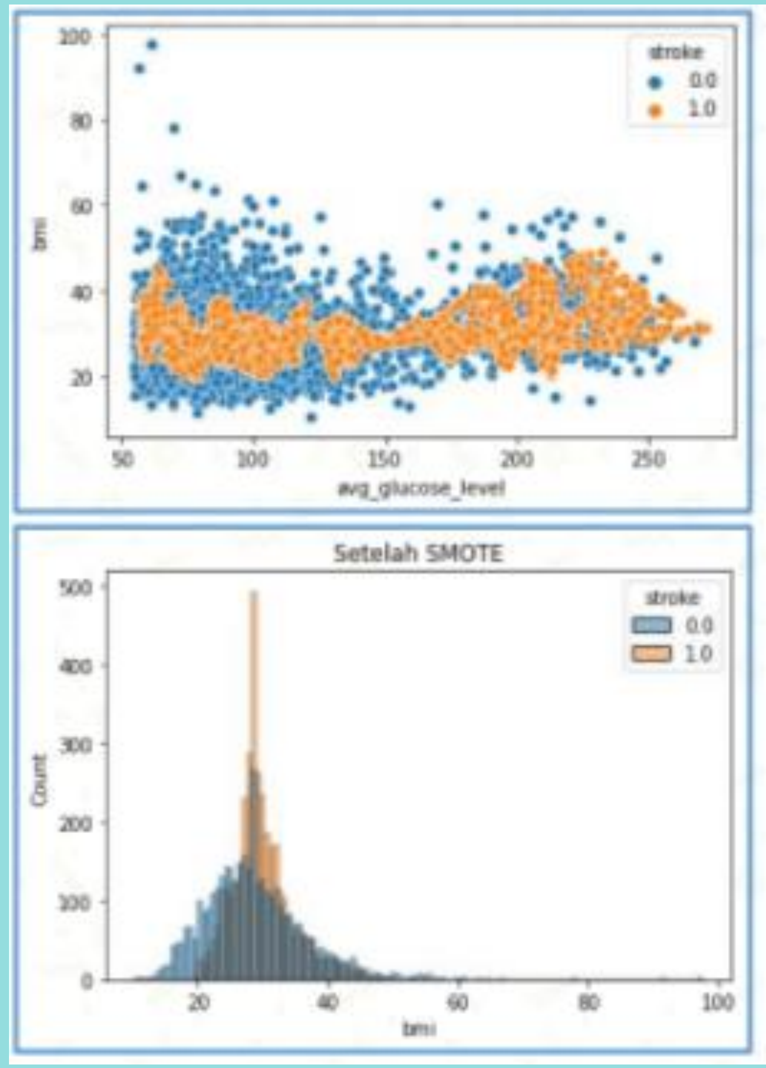
SMOTE

Membandingkan training set sebelum dan sesudah dilakukan SMOTE.

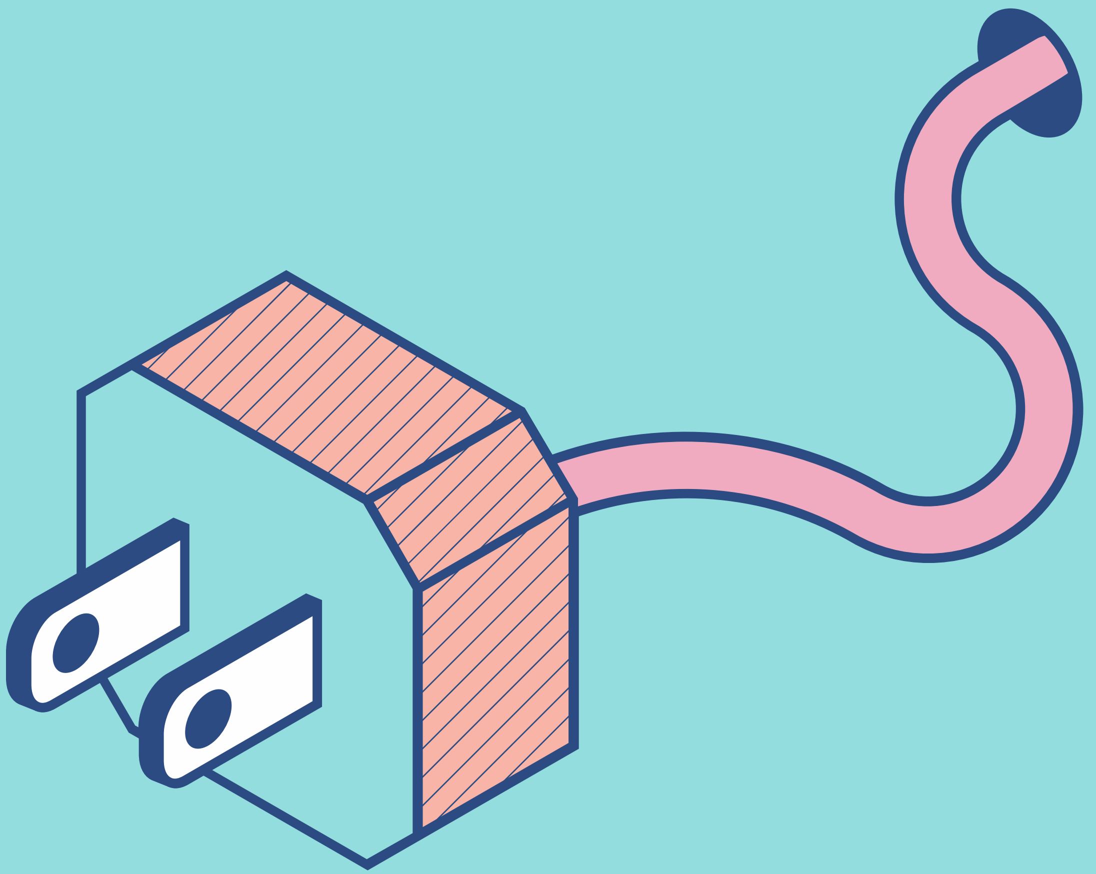
Sebelum



Sesudah

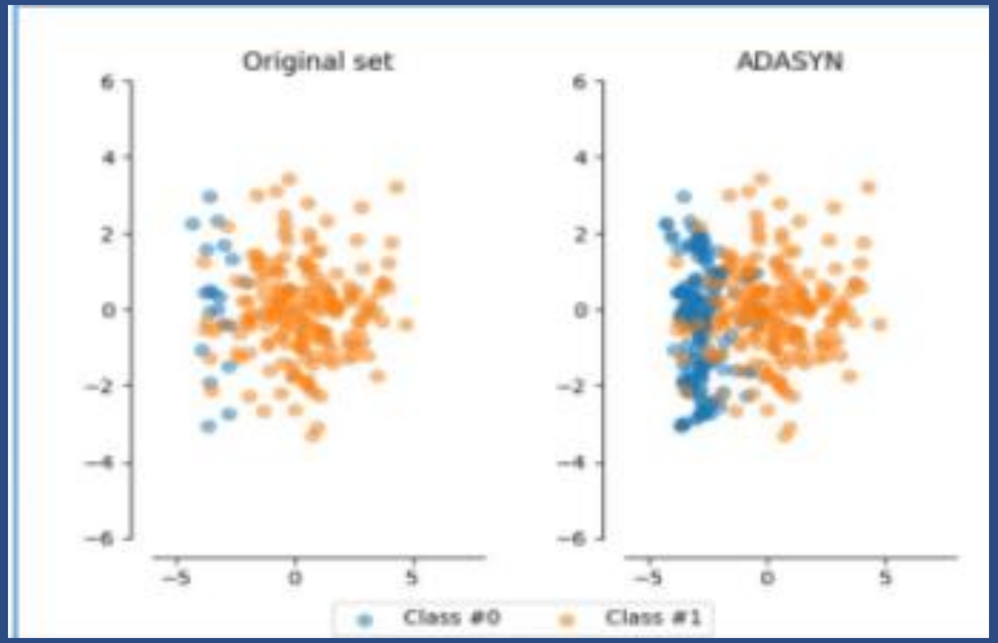



```
Sebelum SMOTE
0.0    3663
1.0     169
dtype: int64
Setelah SMOTE
1.0    3663
0.0    3663
dtype: int64
```

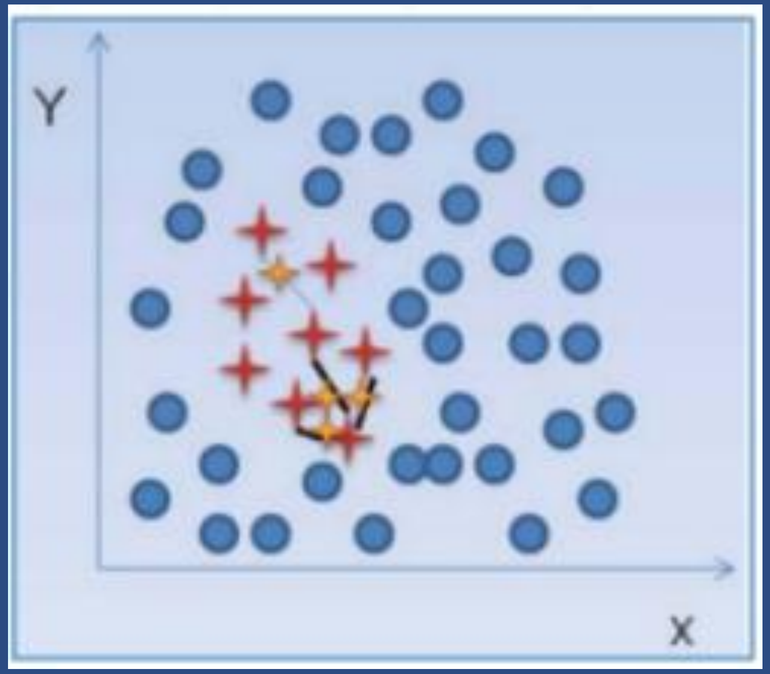


ADASYN

(Adaptive Synthetic Sampling Approach for Imbalanced Learning)

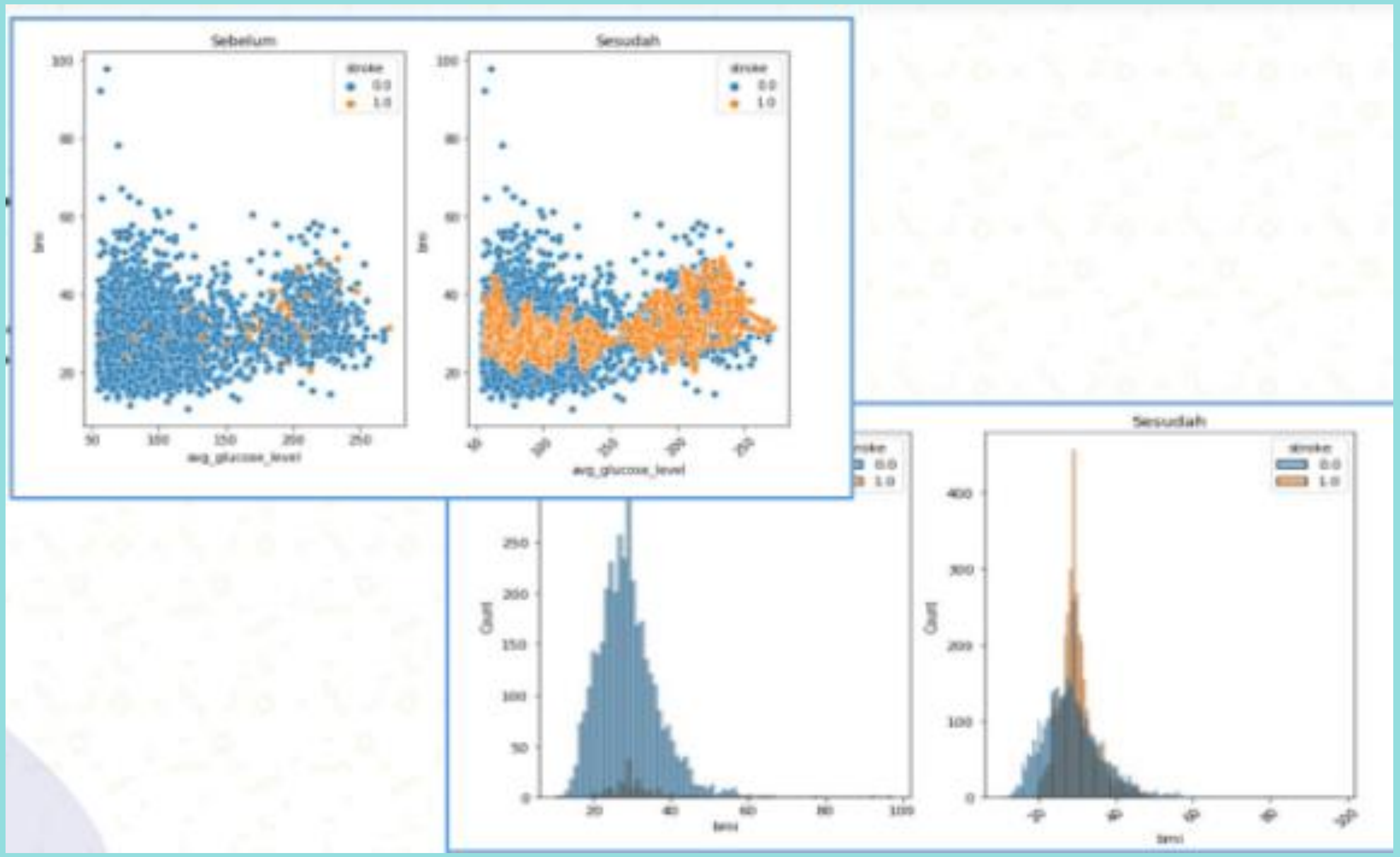


Mirip seperti SMOTE, perbedaannya adalah ADASYN menggunakan sistem pembobotan untuk memilih contoh kelas minoritas dimana sample yang sulit diklasifikasikan memiliki bobot yang lebih tinggi.

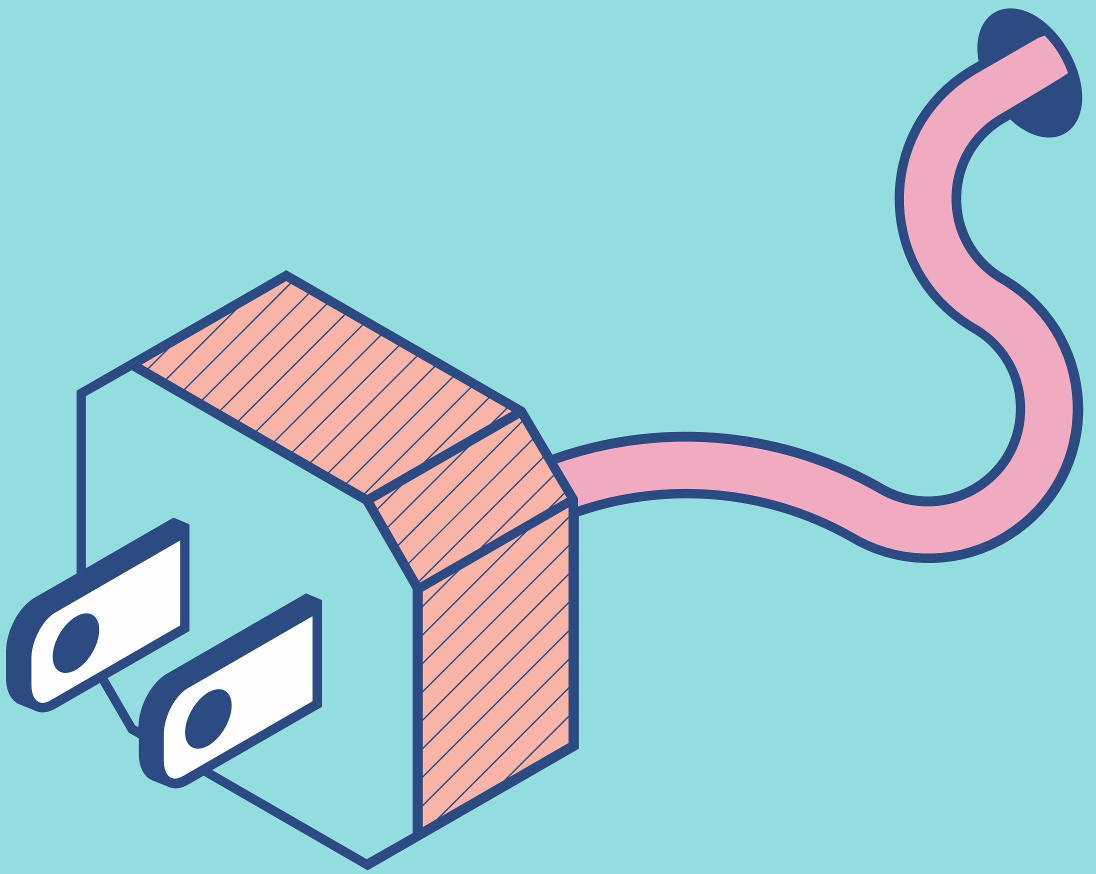


ADASYN

Membandingkan training set sebelum dan sesudah dilakukan ADASYN.



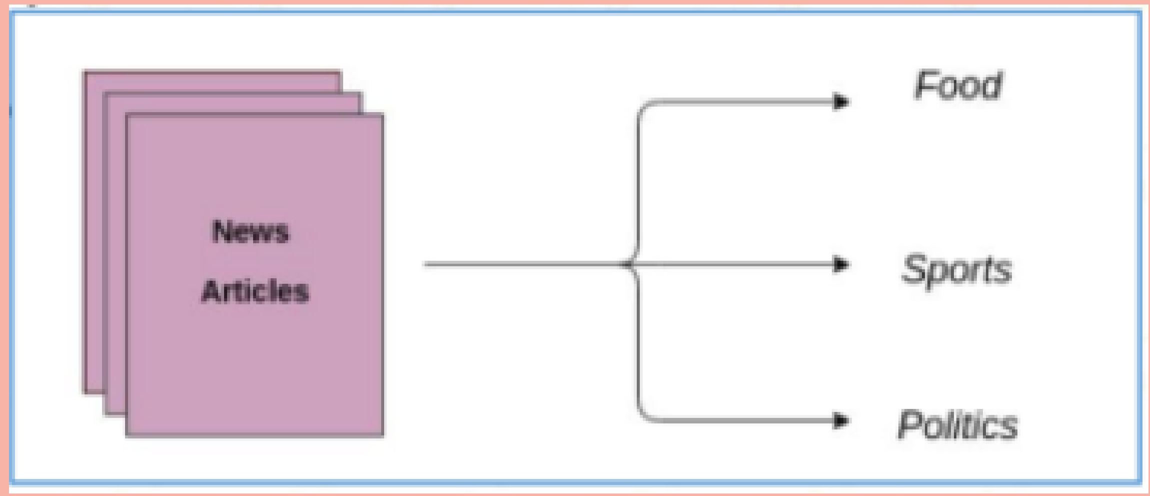

```
Sebelum ADASYN
0.0     3663
1.0      169
dtype: int64
Setelah ADASYIN
1.0     3676
0.0     3663
dtype: int64
```

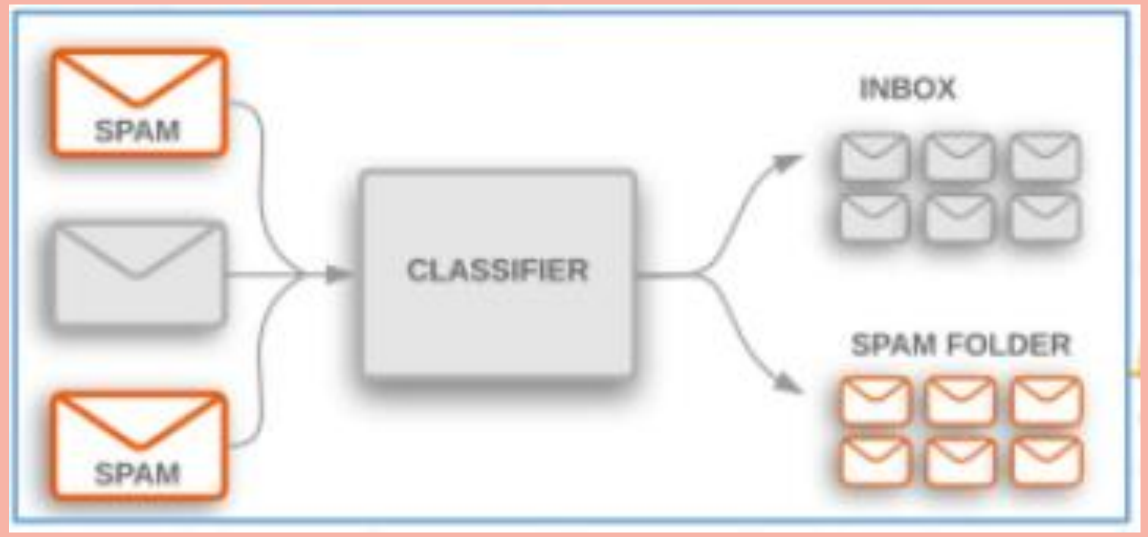


Text Classification

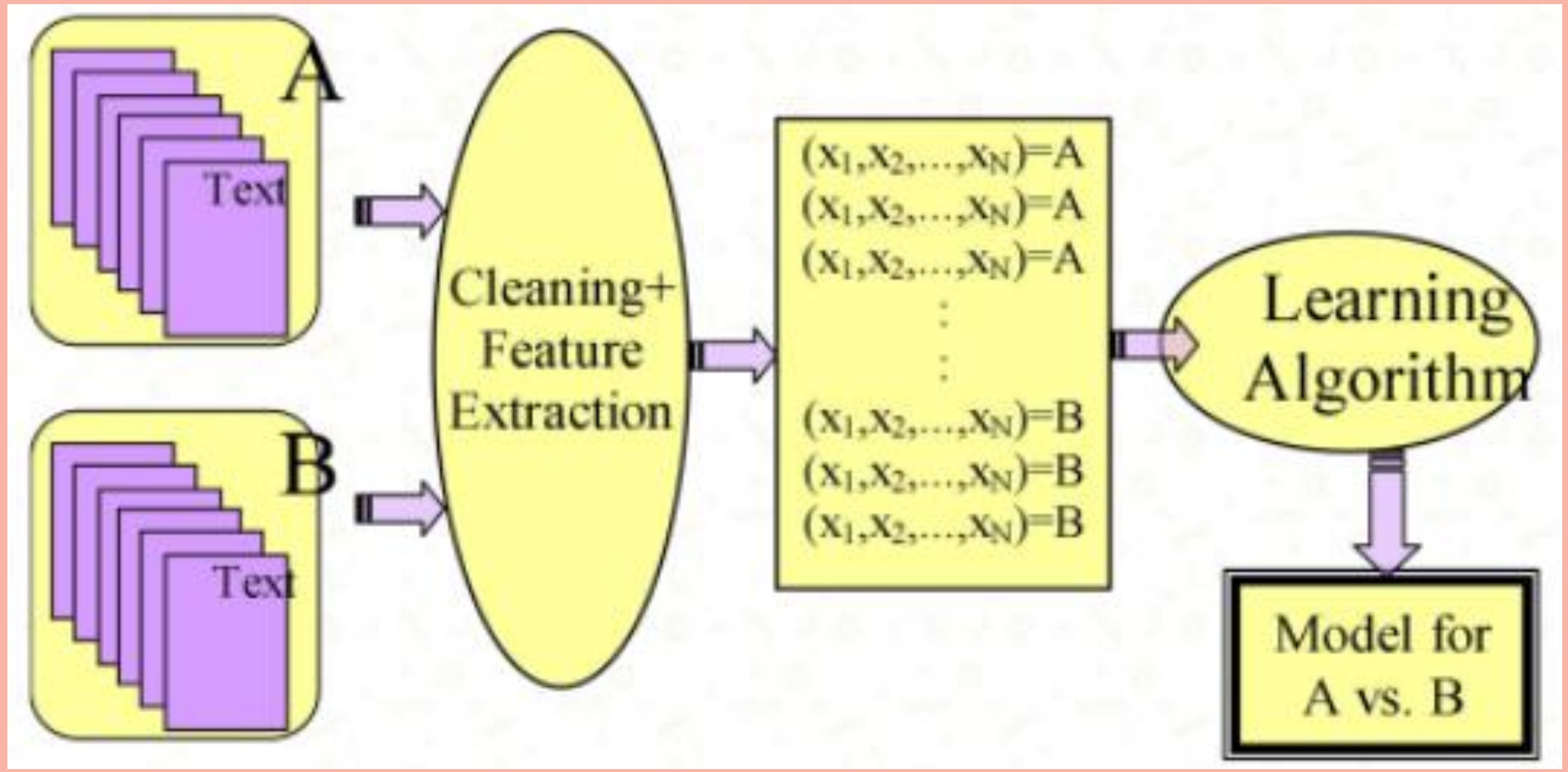
Dikenal juga sebagai ***text tagging / text categorization***, yaitu proses mengkategorikan teks ke dalam kelompok tertentu.

Text classification adalah salah satu tugas dasar dalam *natural language process* (NLP) dengan aplikasi yang luas, seperti **sentiment analysis, topic labelling, spam detection,& intent detection**.

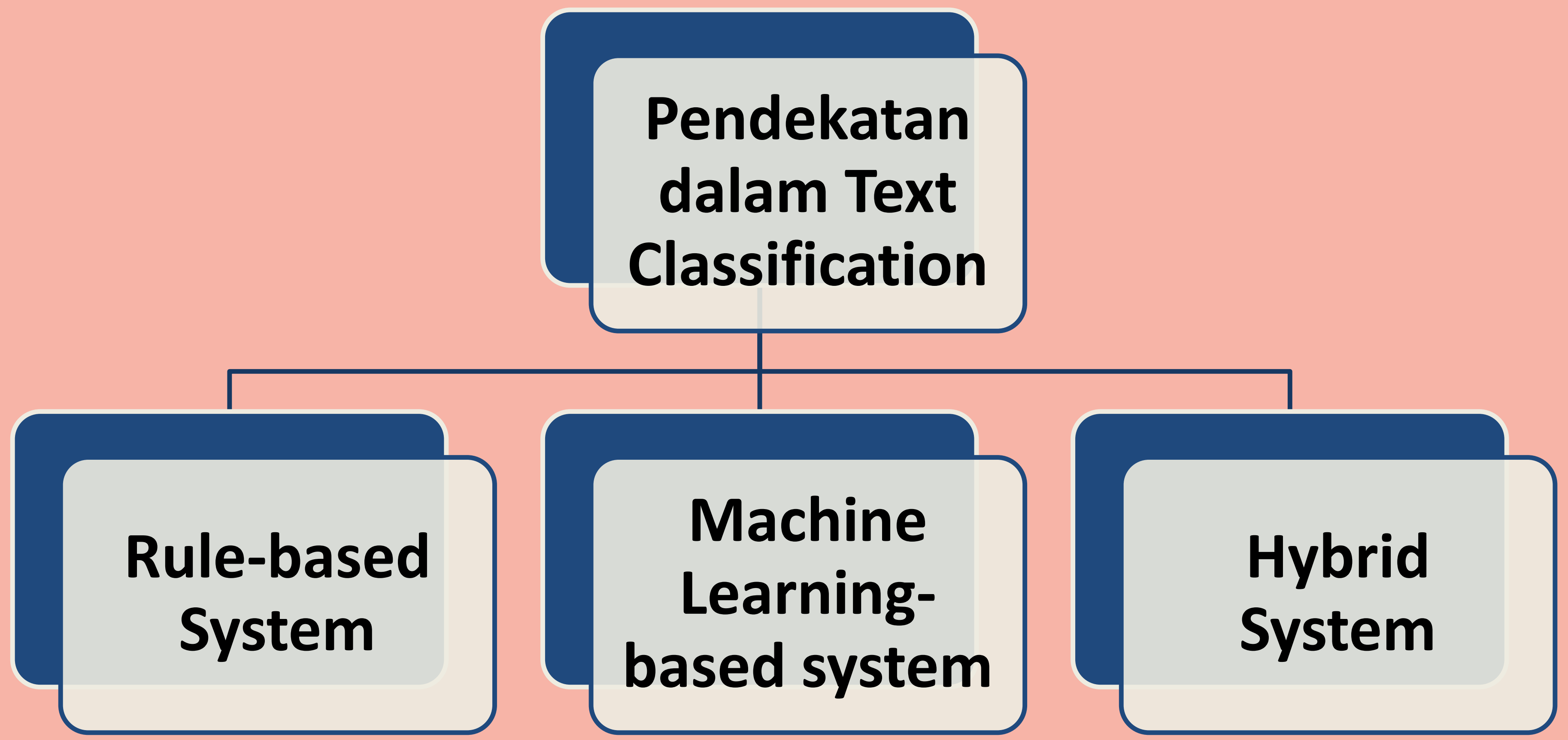




How Text Classification Works ?



Pendekatan dalam Text Classification



Rule-based System

Teks dipisahkan ke dalam kelompok secara terorganisir menggunakan *handicraft linguistic rules.*

Machine Learning-based System

ML-based classifier membuat klasifikasi berdasarkan pengamatan sebelumnya dari kumpulan data.

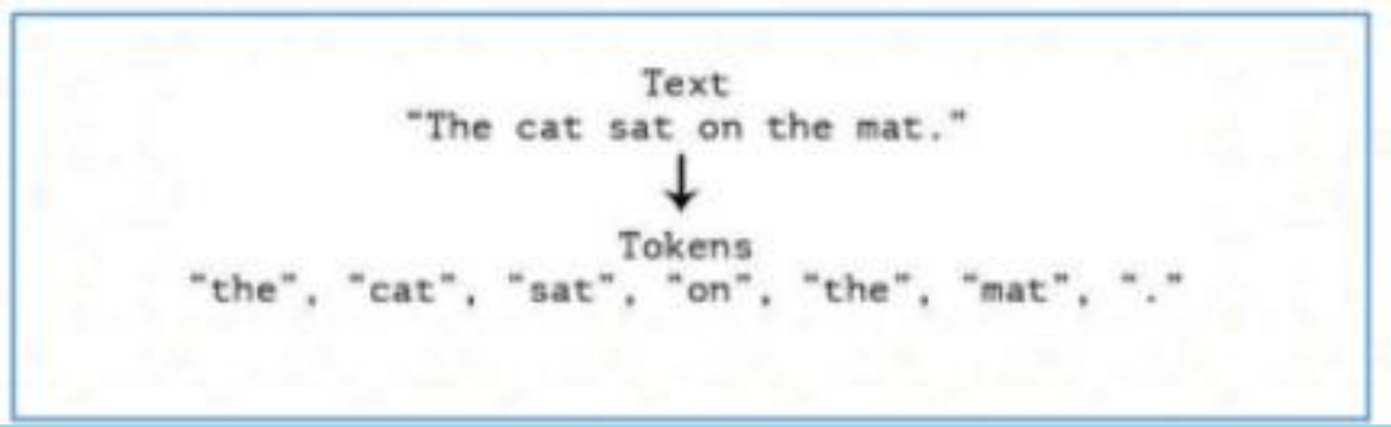
Hybrid System

Menggabungkan pendekatan *machine learning classifier* dengan *rule-based system,* digunakan untuk meningkatkan performa.

Handling Text Dataset

Tokenization

- Memecah teks mentah menjadi kata - kata yang disebut sebagai **tokens**.
- Tokens ini membantu dalam memahami konteks atau mengembangkan model untuk NLP.



Handling Text Dataset

Pre-processing The Text

Removing Stop words	Stemming	Lemmatization
• Tanda baca (*punctuation*). e.g: ?, !, [], (). • Kata depan (preposisi). e.g: in, at, on, of, to.	• Proses mereduksi kata menjadi bentuk kata yang paling dasar (*word stem*)/ akar kata (*root form*). E.g: changes -> chang.	• Proses mengubah kata menjadi kata dasar. E.g: changes -> change.

Feature Extraction

Bag of Words

- Istilah frekuensi (*term*) dalam dokumen.
- Fokus pada skema pengkodean yang mewakili kata - kata, **tanpa informasi tentang urutan**.

```
doc1 = "saya belajar pemrograman dan belajar melukis"
doc2 = "saya membantu adik saya belajar menulis"
doc3 = "ibu belajar menjahit"
```

adik	belajar	dan	ibu	melukis	membantu	menjahit	menulis	pemrograman	saya
0	2	1	0	1	0	0	0	1	1
1	1	0	0	0	1	0	1	0	2
0	1	0	1	0	0	1	0	0	0

TF-IDF

(*Term Frequency-Inverse Document Frequency*)

- *Numerical statistics* yang mencerminkan betapa pentingnya sebuah kata bagi dokumen dalam suatu kumpulan dokumen atau *corpus*.

- TF-DF memperlihatkan skor frekuensi kata yang berguna untuk menonjolkan kata - kata yang lebih menarik (sering muncul tapi tidak di semua dokumen).

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

For a term i in document j:

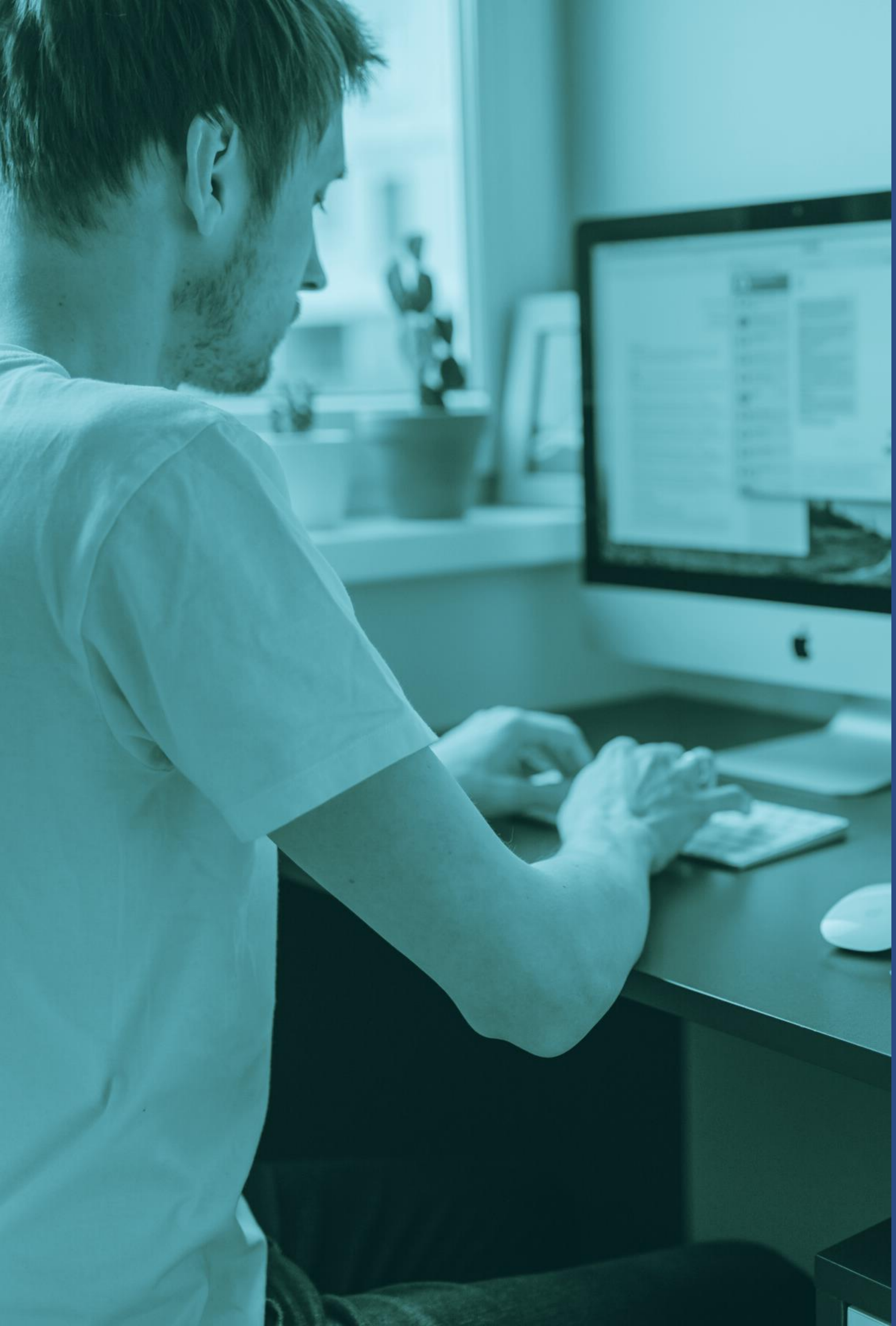
tf_{ij} = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

idf(t) = log [n / df(t)] + 1

sklearn formula

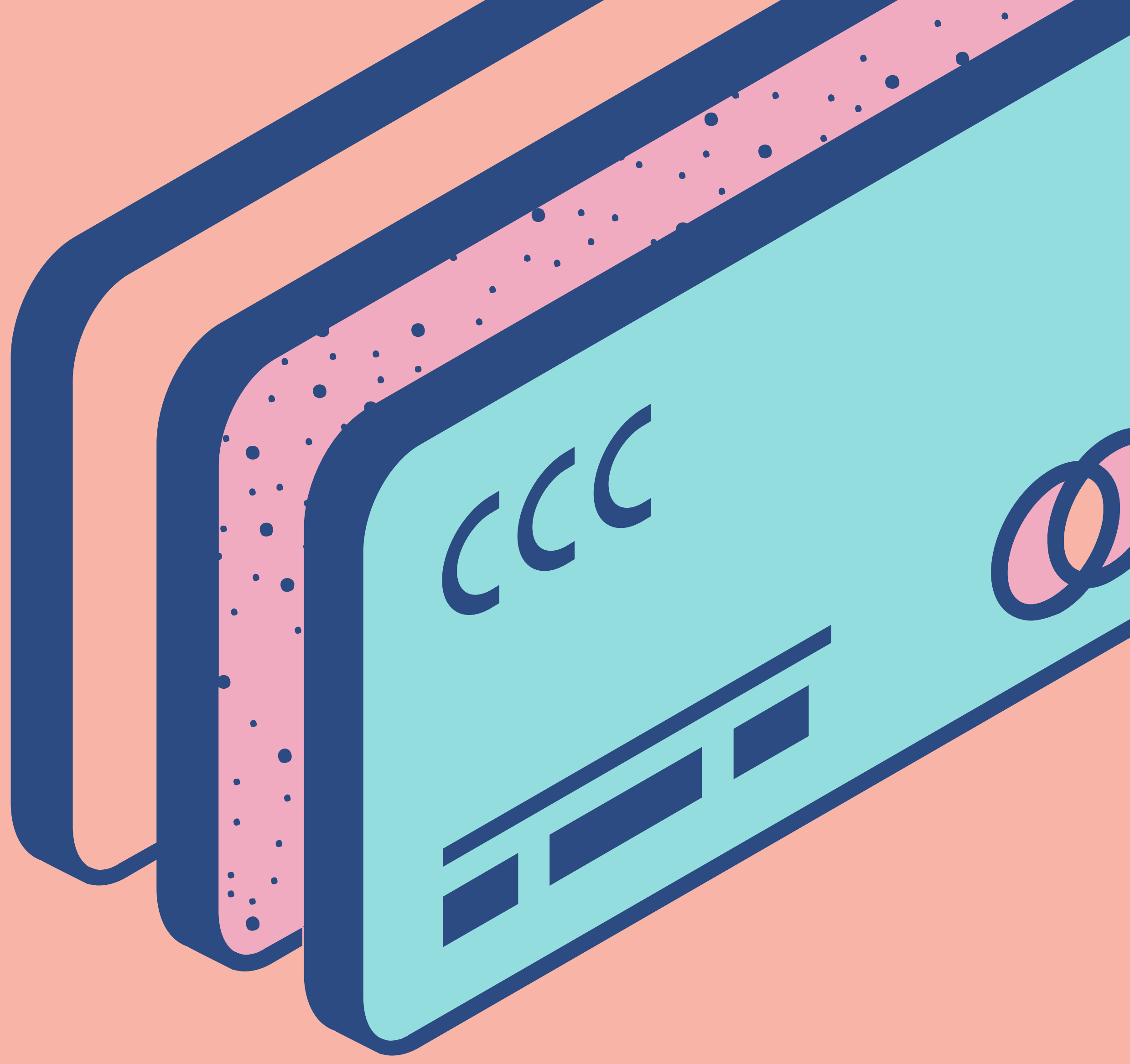
```
doc1 = "saya belajar pemrograman dan belajar melukis"
doc2 = "saya membantu adik saya belajar menulis"
doc3 = "ibu belajar menjahit"
```

adik	belajar	dan	ibu	melukis	membantu	menjahit	menulis	pemrograman	saya
0	2	2.0986123	0	2.0986	0	0	0	2.09861229	1.405465
2.098612	1	0	0	0	2.09861229	0	2.0986123	0	2.81093
0	1	0	2.0986	0	0	2.09861229	0	0	0



CLASSIFICATION I

Classification & Clustering Techniques



Classification

- K-Nearest Neighbour
- Decision Tree
- Ensemble Methods

Clustering

- K-Method
- K-Means
- DBSCAN

Nearest Neighbor Classifier

Membutuhkan tiga hal :

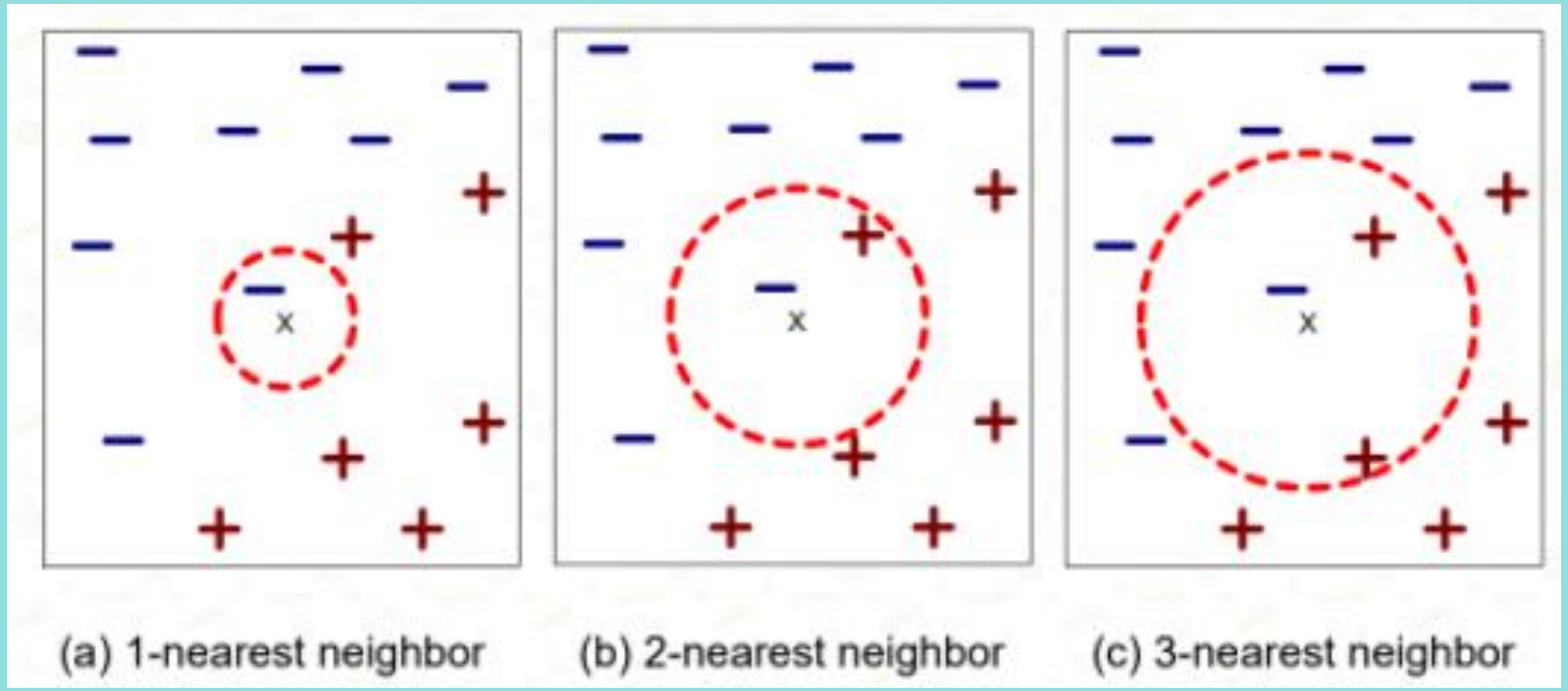
- Kumpulan data yang tersimpan
- Jarak metrik (*distance metric*)
- Nilai k, jumlah tetangga terdekat yang diambil

Untuk Mengklasifikasikan data baru “

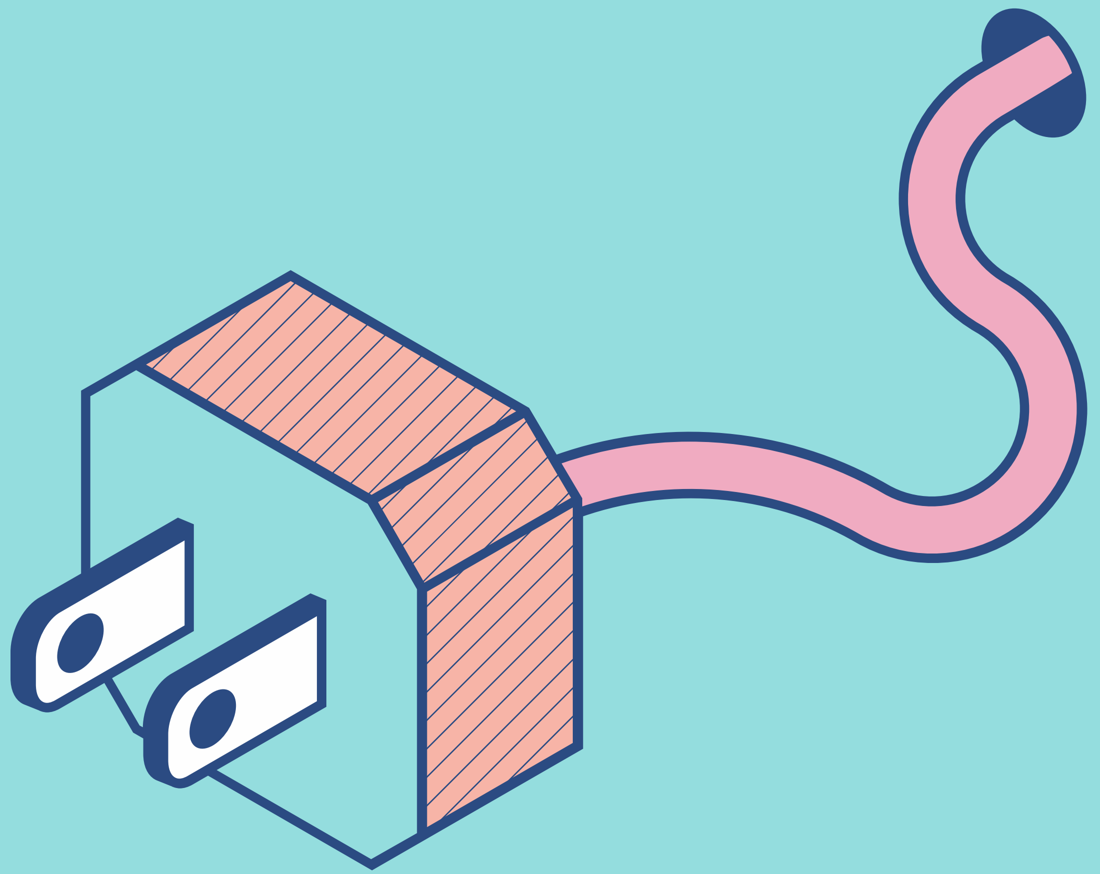
1. Hitung Jarak terhadap data lain
2. Memilih k tetangga terdekat
3. Gunakan label kelas tetangga terdekat untuk menentukan label kelas data baru.

   (misal : ambil suara mayoritas)

Definition of Neighbor Classifier



K-nearest neighbor dari record x adalah k data yang memiliki jarak terdekat ke x



Nearest Neighbor Classification

Hitung jarak antara 2 titik :

Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

Memilih class dari tetangga terdekat :

Ambil suara mayoritas dari label kelas di antara *k-nearest neighbor*

Memberi bobot suara menurut jarak (*distance*).

Nearest Neighbor Classification

Memilih nilai k :

Jika nilai **k** terlalu kecil :
Sensitif terhadap *noise*.

Jika nilai **k** terlalu besar :
neigborhood dapat memasukkan nilai dari kelas lain.

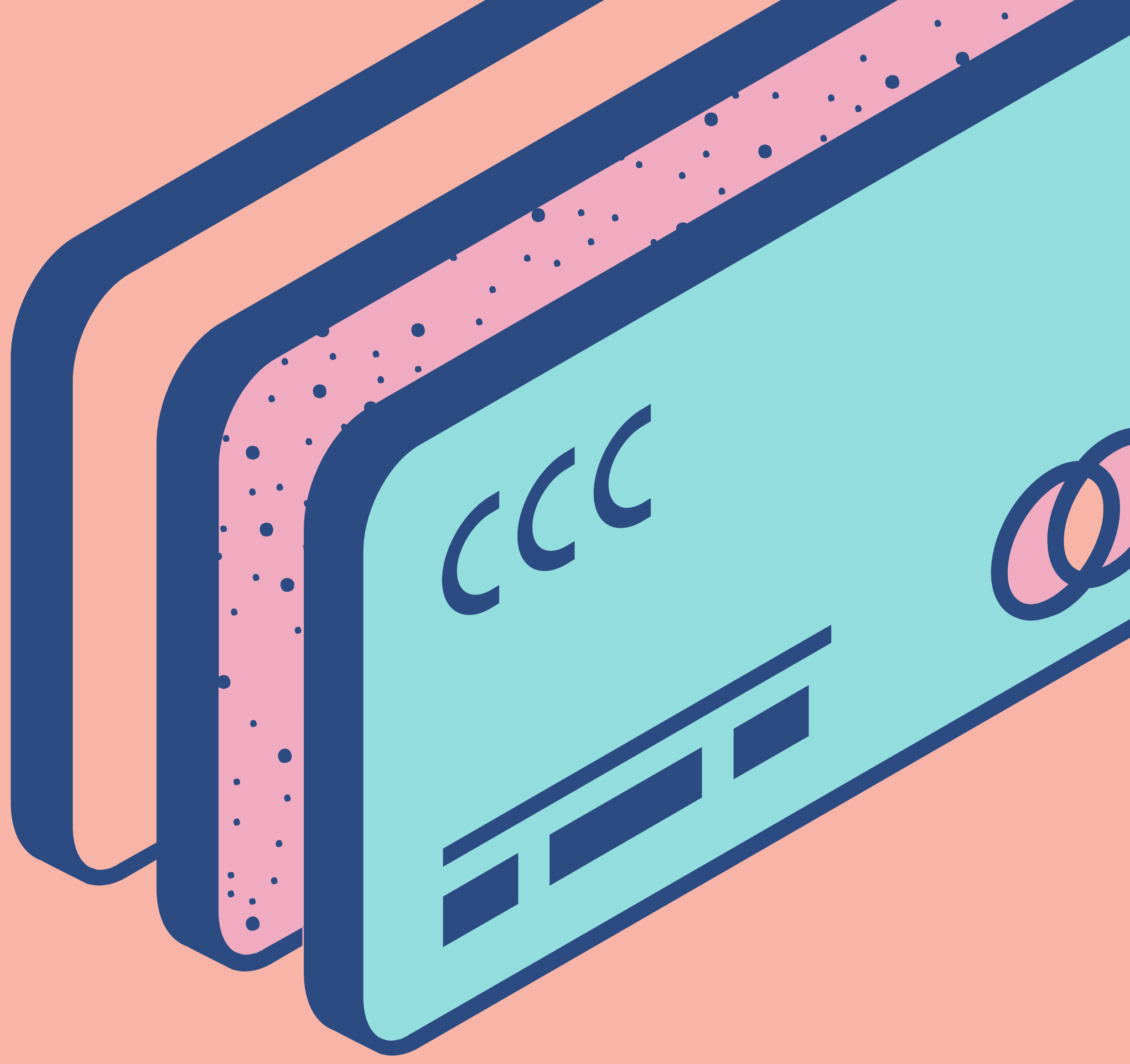
Permasalahan *scaling* :

Attributes perlu di-*scale*

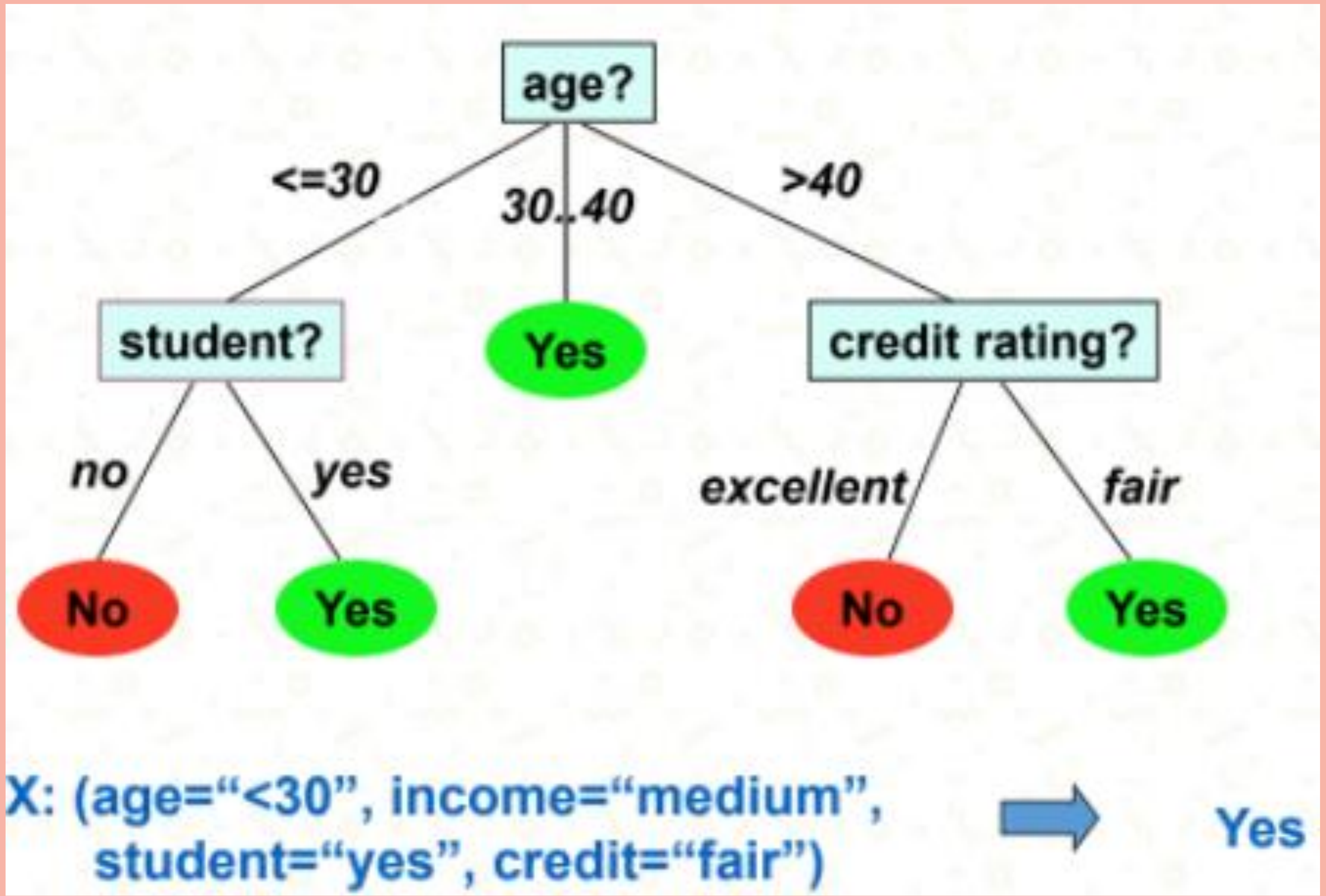
Untuk mencegah jarak *(distance) di dominasi oleh salah satu atribut. Contoh : Tinggi seseorang dapat bervariasi dari 1,4m – 1,8m*

Training Dataset

No.	age	income	student	credit_rating	buys_computer
1	<=30	high	no	fair	no
2	<=30	high	no	excellent	no
3	31…40	high	no	fair	yes
4	>40	medium	no	fair	yes
5	>40	low	yes	fair	yes
6	>40	low	yes	excellent	no
7	31…40	low	yes	excellent	yes
8	<=30	medium	no	fair	no
9	<=30	low	yes	fair	yes
10	>40	medium	yes	fair	yes
11	<=30	medium	yes	excellent	yes
12	31…40	medium	no	excellent	yes
13	31…40	high	yes	fair	yes
14	>40	medium	no	excellent	no



Output : A Decision Tree for buys_computer



Algorithm for DT Induction

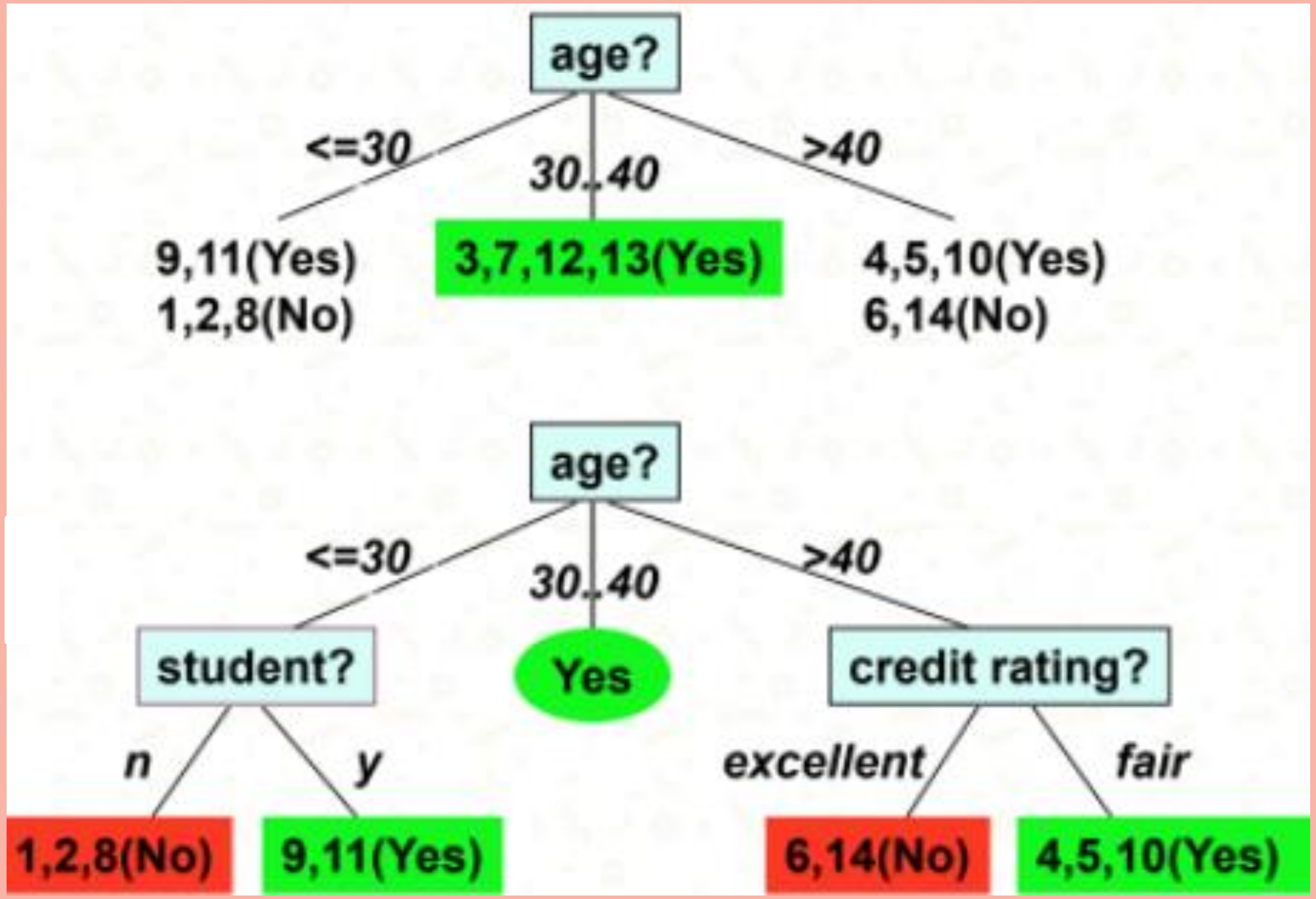
Top-down recursive divide-and-conquer manner

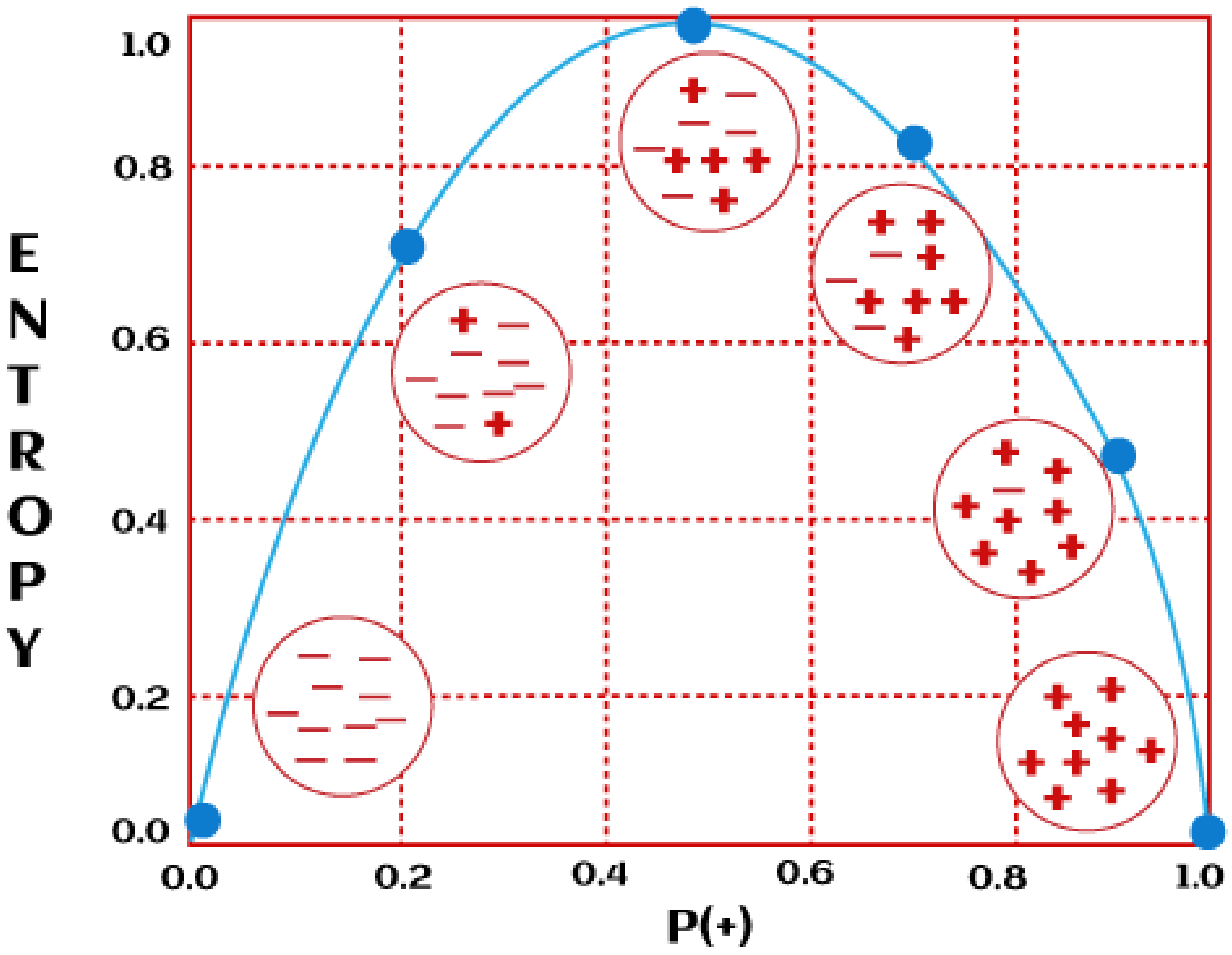
- Awalnya, semua *training data* ada di *root*.
- Atribut yang di *node* dipilih *berdasarkan* ***heuristic*** *atau* ***statistik*** ***(misal : information gain***)
- *Training data* di partisi secara rekursif berdasarkan atribut yang dipilih

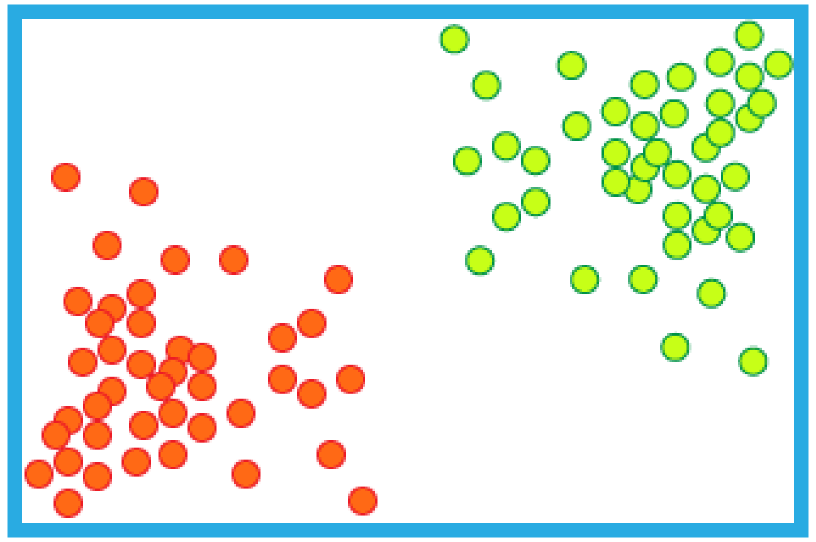
Kondisi untuk menghentikan partisi

- Semua *sample* termasuk dalam kelas yang sama
- Tidak ada *sample* yang tersisa
- Tidak ada atribut yang tersisa

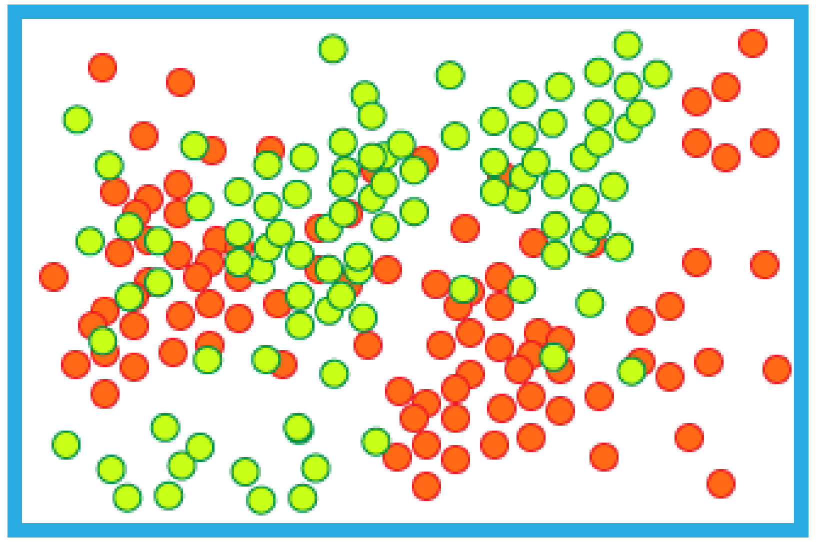
Algorithm for DT Induction







Low Entropy



High Entropy

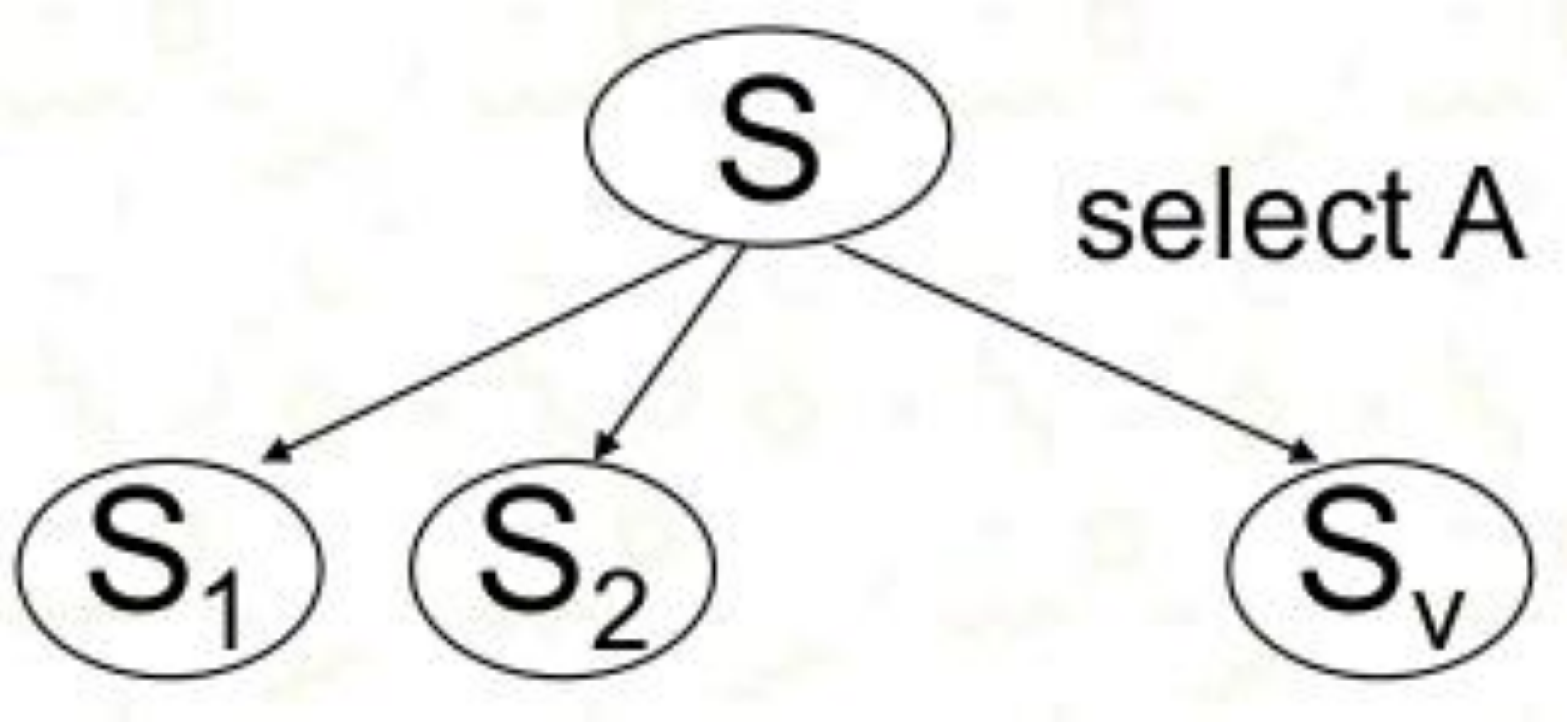
Entropy

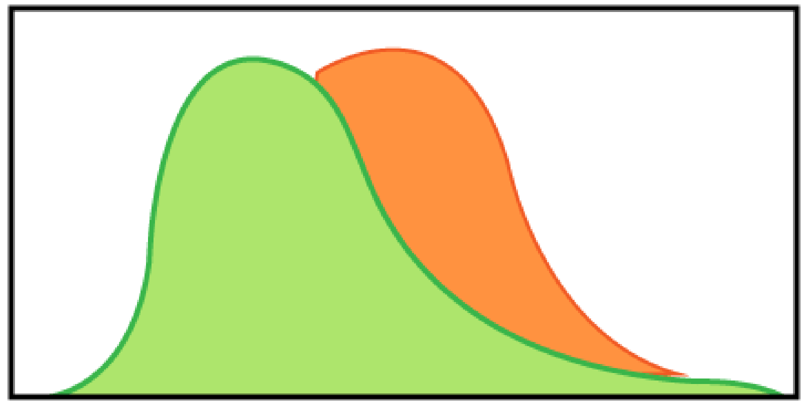
ENTROPI DAPAT DIDEFINISIKAN SEBAGAI UKURAN KEMURNIAN SUB SPLIT.

$$E(S) = \sum_{i=1}^{m} p_i(-\log_2 p_i) = -p_1 \log_2 p_1 - p_2 \log_2 p_2$$

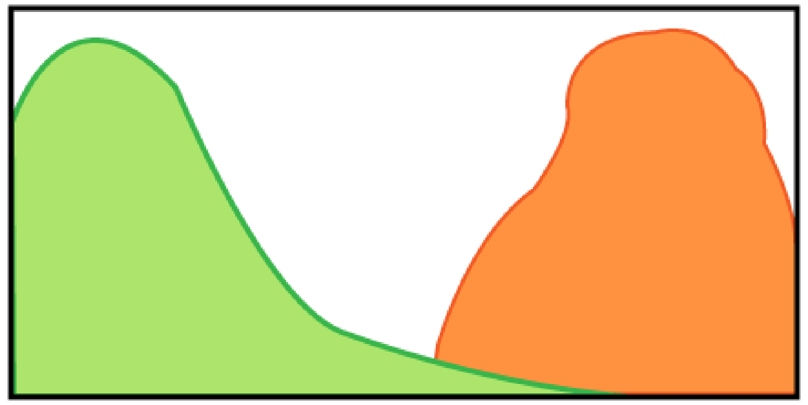
Examples

- S: {Y(1,2), N(3,4)} for instances 1,2,3,4
  - E(S) = (-1/2)*log(1/2) + (-1/2)*log(1/2) = 1
- S: {Y(1,3,4), N(2)} for instances 1,2,3,4
  - E(S) = (-3/4)*log(3/4) + (-1/4)*log(1/4) = 0.81
- S: {Y(1,2,3,4), N()} for instances 1,2,3,4
  - E(S) = (-4/4)*log(4/4) + (-0/4)*log(0/4) = 0





Low information gain High entropy



High information gain Low entropy

Information Gain

DIDEFINISIKAN SEBAGAI POLA YANG DI AMATI DALAM DATASET DAN PENGURANGAN ENTROPI.

$$E(A) = \sum_{j=1}^{v} \frac{s_j}{s} E(s_j) = \frac{s_1}{s} E(s_1) + \frac{s_2}{s} E(s_2) + \ldots + \frac{s_v}{s} E(s_v)$$

- Hitung information gain dari attribute A

$$Gain(A) = E(S) - E(A)$$

- Pilih atribut dengan Information gain yang terbesar

Attribute Selection by Information Gain - Example

- C_1: buys_computer = "yes", C_2: buys_computer = "no"
- E(D) = E(9,5) = $-\frac{9}{14}\log\frac{9}{14}-\frac{5}{14}\log\frac{5}{14}=0.94$

$$\left(-\frac{2}{5}\log_2\frac{2}{5}-\frac{3}{5}\log_2\frac{3}{5}\right)$$

age	C1	C2	$E(S_i)$
<=30	2	3	0.971
30…40	4	0	0
>40	3	2	0.971

- E(age) = $\frac{5}{14}E("<=30")+\frac{4}{14}E("30..40")+\frac{5}{14}E(">40")=0.69$
- Gain(age) = $\boxed{E(D)-E(age)=0.25}$

Gain(income) = 0.03, Gain(student) = 0.15

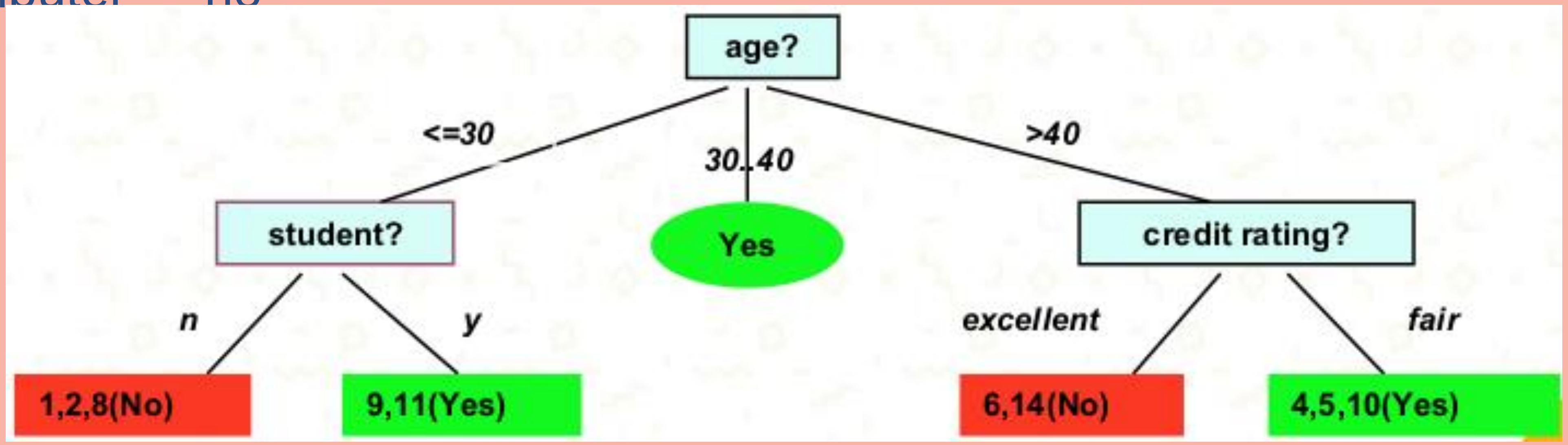
PILIH ATRIBUT YANG TERBESAR

Gain(age) merupakan atribut terbesar yang telah di hitung daripada Gain(income) dan Gain(student)

Extracting Classification Rules

- Merepresentasikan pengetahuan dalam bentuk IF-THEN rules
  - Satu aturan (rule) dibuat untuk setiap jalur dari akar ke daun
  - Node daun menyimpan prediksi kelas
- Rules mudah dipahami manusia
- Example

IF age = “<=30” AND student = “no” THEN buys_computer = “no”
IF age = “<=30” AND student = “yes” THEN buys_computer = “yes”
IF age = “31...40” THEN buys_computer = “yes”
IF age = “>40” AND credit = “excellent” THEN buys_computer = “yes”
IF age = “>40” AND credit = “fair” THEN buys_computer = “no”



Avoid Overfitting

- Tree yang dibuat mungkin akan overfit terhadap training data
  - Terlalu banyak cabang, diakibatkan oleh ouliers data
  - Hasilnya akurasi yang rendah terhadap data testing

PRUNING

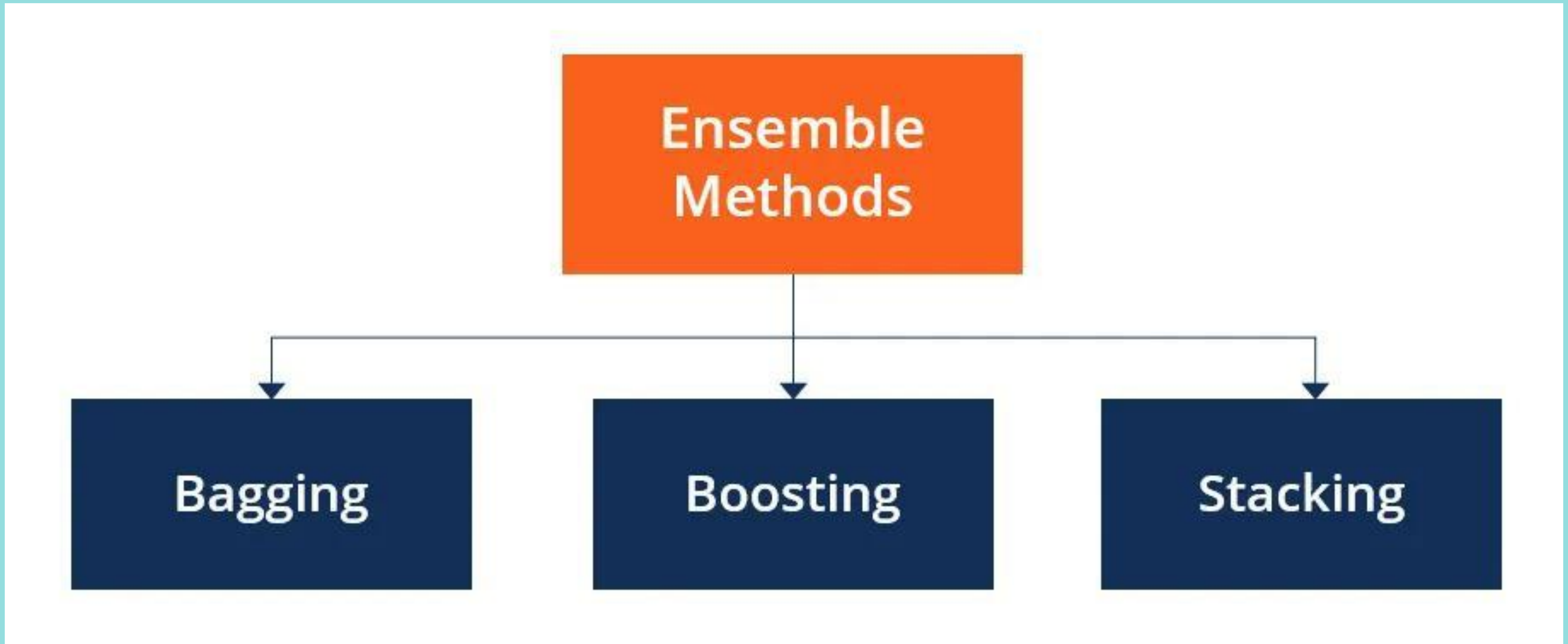
- Prepruning
  - Hentikan konstruksi pohon lebih awal—jangan membagi simpul jika ini akan mengakibatkan akurasi jatuh di bawah ambang batas
- Postpruning
  - Hapus cabang dari Tree / pohon yang telalu lebat.
  - Jika memangkas / pruning sebuah node menghasilkan tingkat error yang lebih kecil (terhadap test set), silahkan di pruning

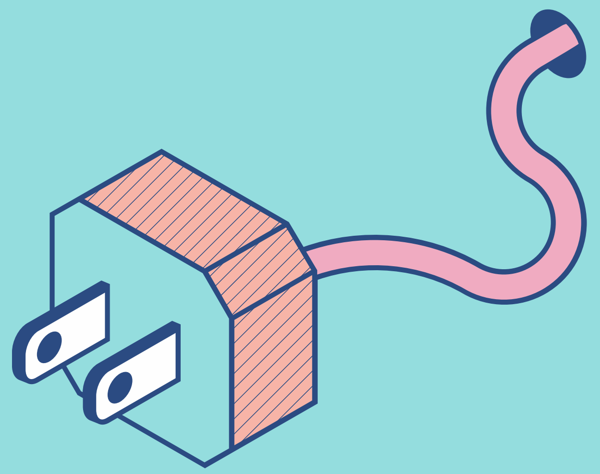
Discussion on Decision Tree

- Kelebihan
  - Aturan klasifikasi yang dapat dipahami oleh manusia
  - Kecepatan belajar/klasifikasi yang relatif lebih cepat
- Kekurangan
  - Sensitive (not robust/ tidak kuat) terhadap noises
  - Atribut bernilai kontinu - secara dinamis mempartisi nilai atribut kontinu ke dalam set interval diskrit

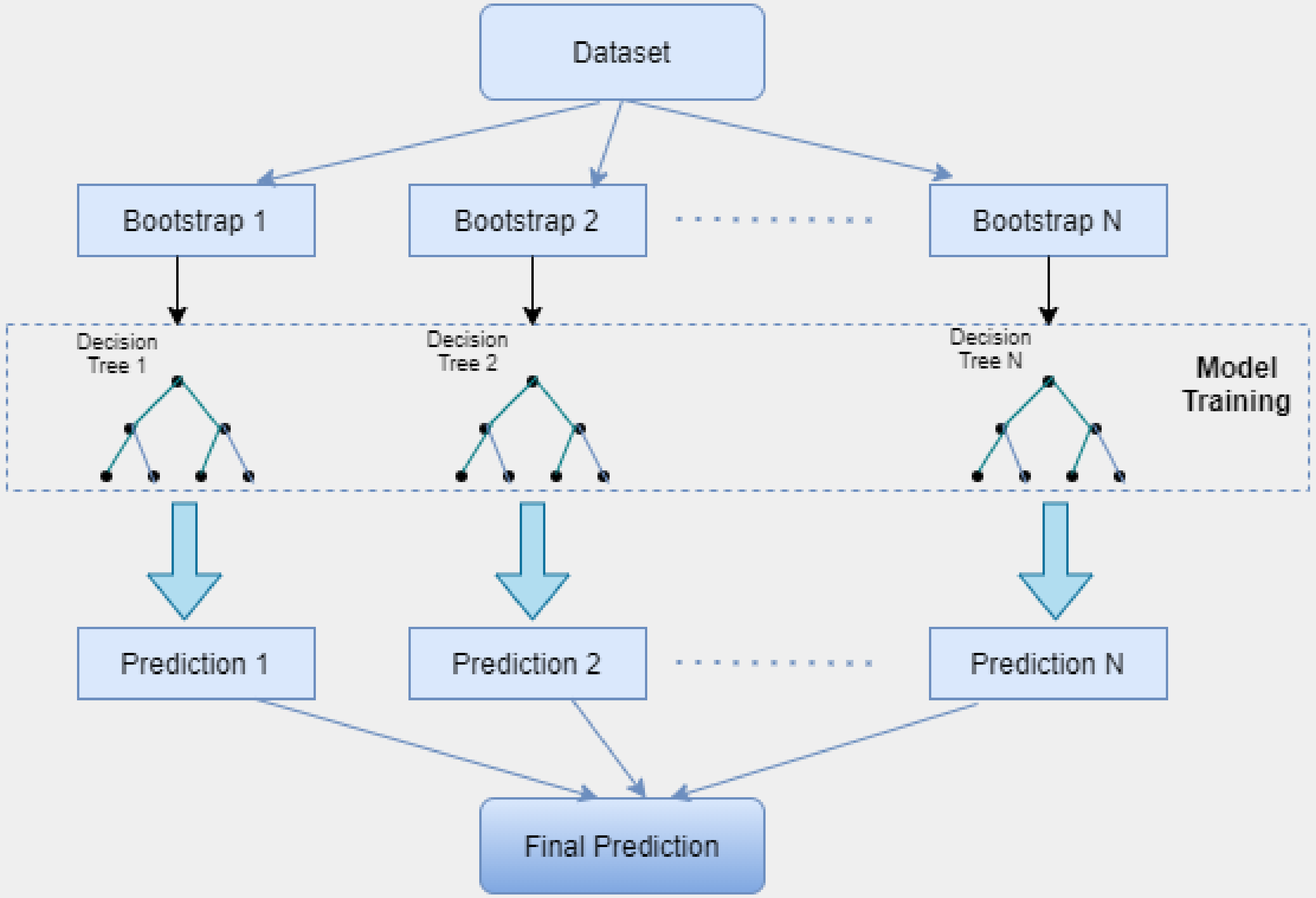
Ensemble Methods

Ensemble methods is a machine learning technique that **combines several base models** in order to produce one optimal predictive model.

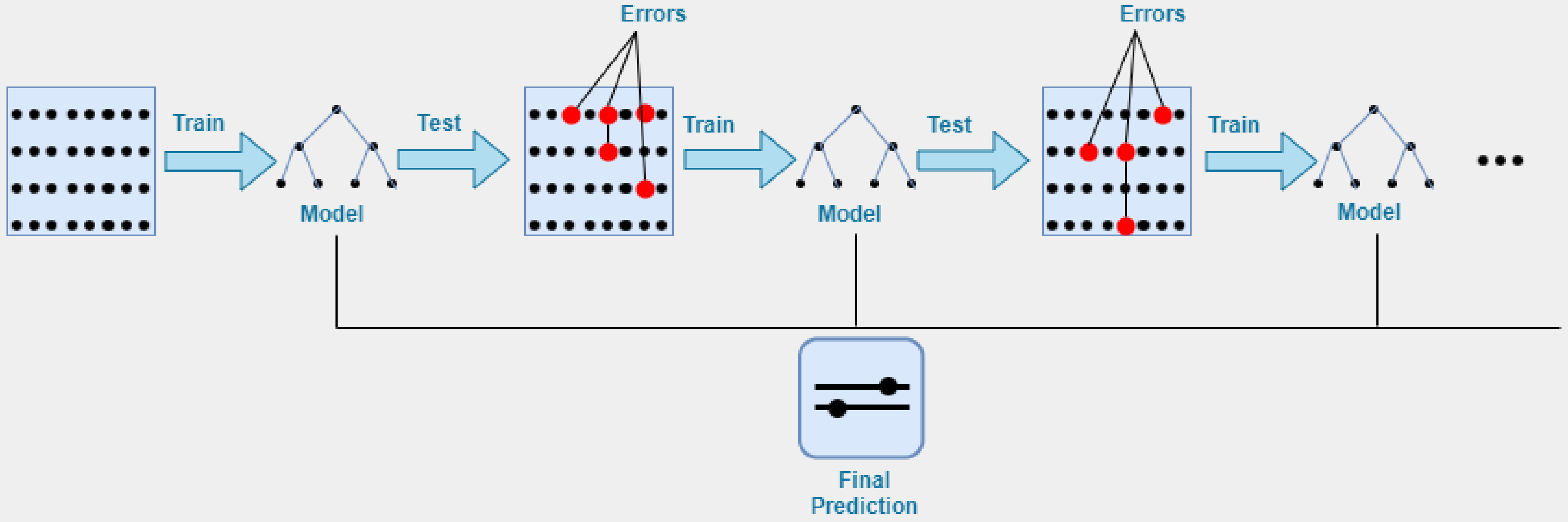




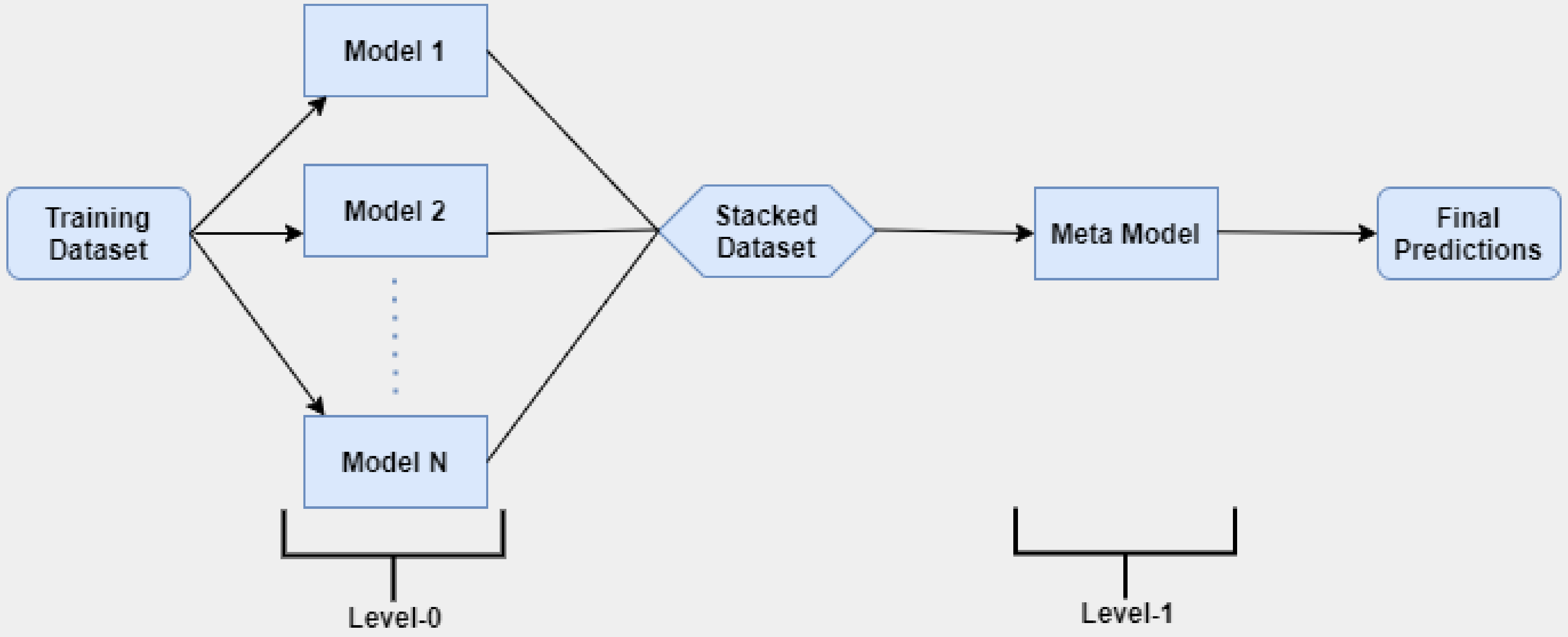
Bagging



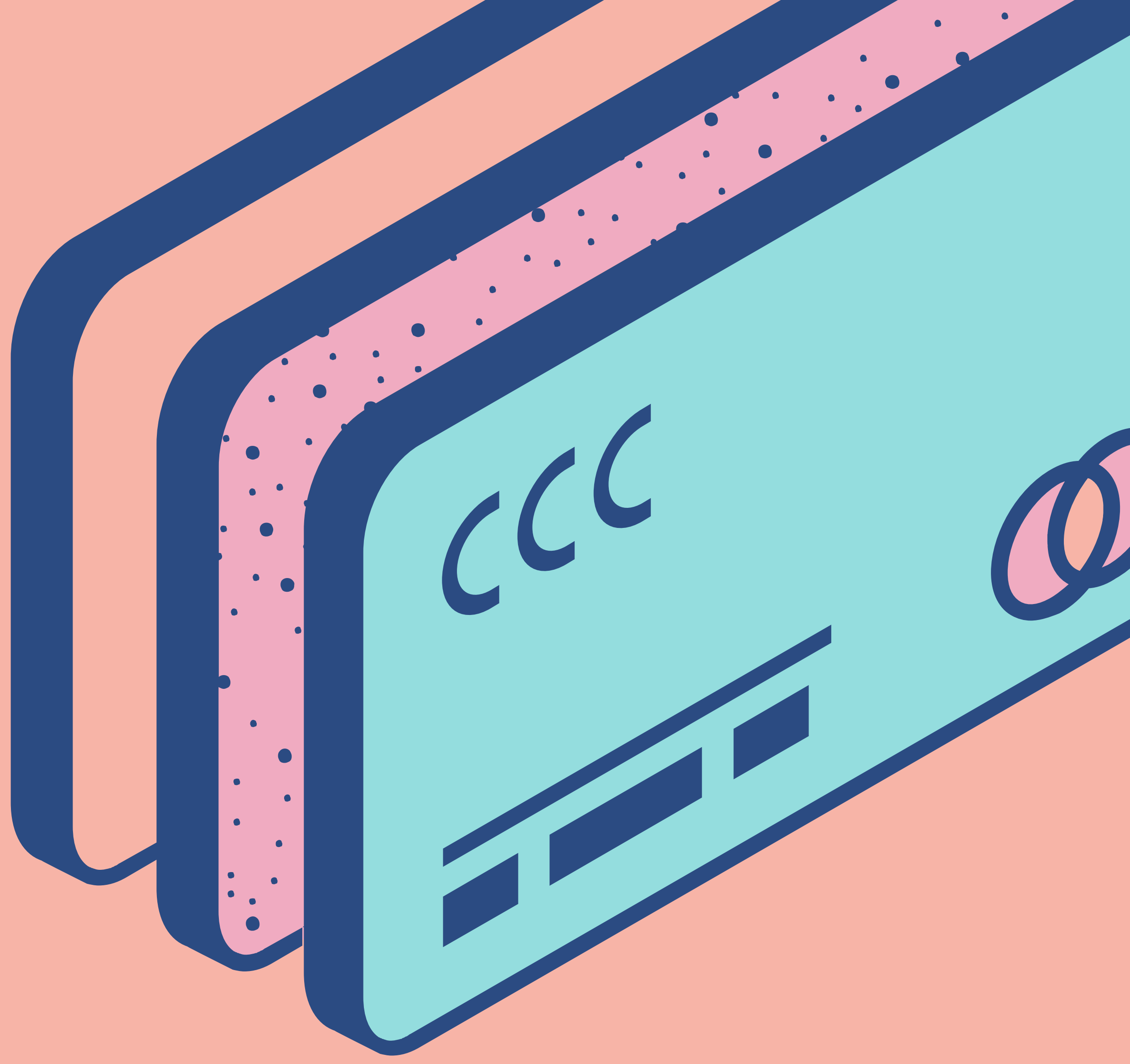
Boosting



Stacking



Ensemble Method Libraries



BAGGING

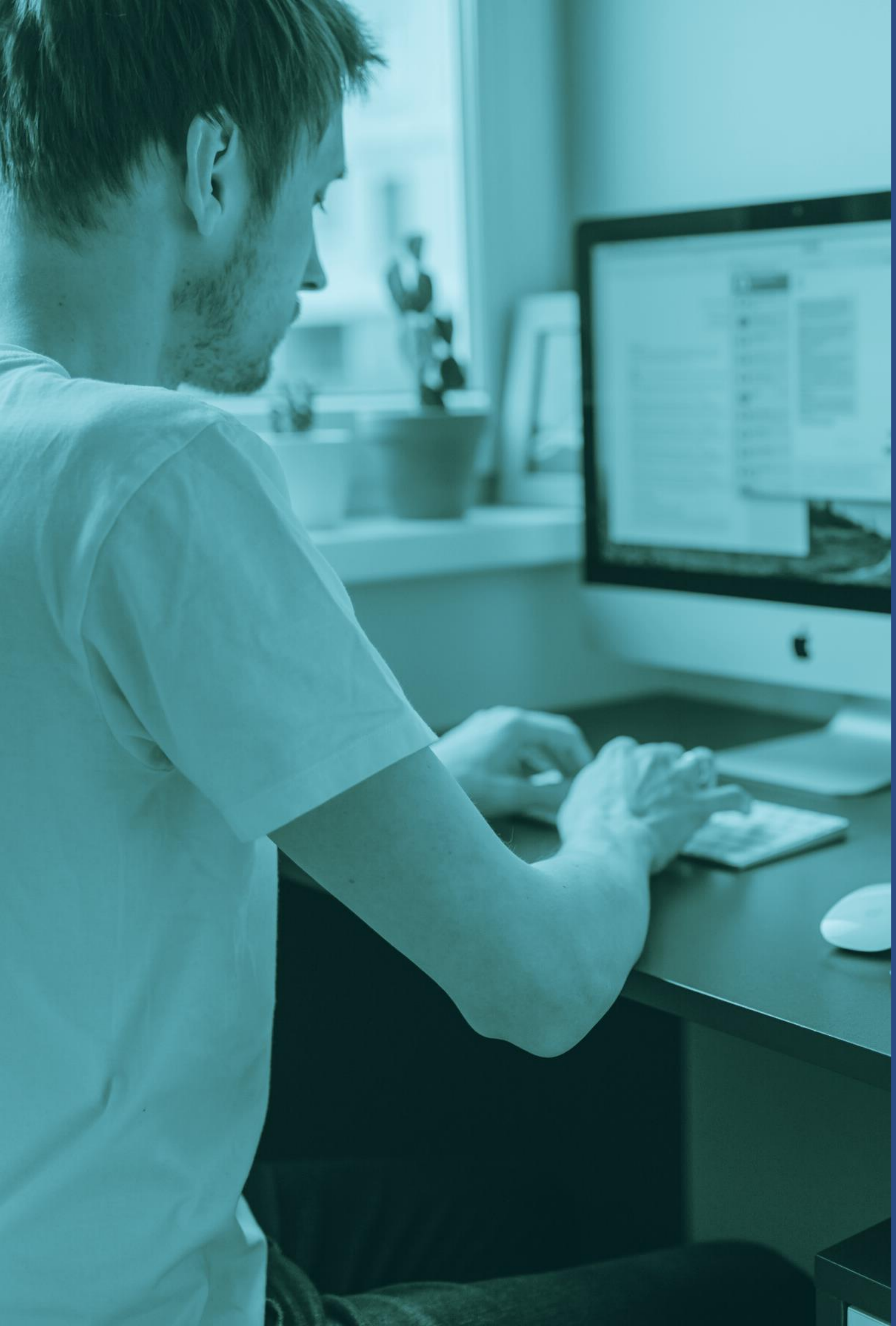
- Random Forest
- Bagged Decision Trees
- Extra Trees.
- sklearn library also provides:
  - BaggingClassifier
  - BaggingRegressor

BOOSTING

- AdaBoost
- Gradient Boosting Machine (GBM)
- XGBoost
- LightGBM
- CatBoost

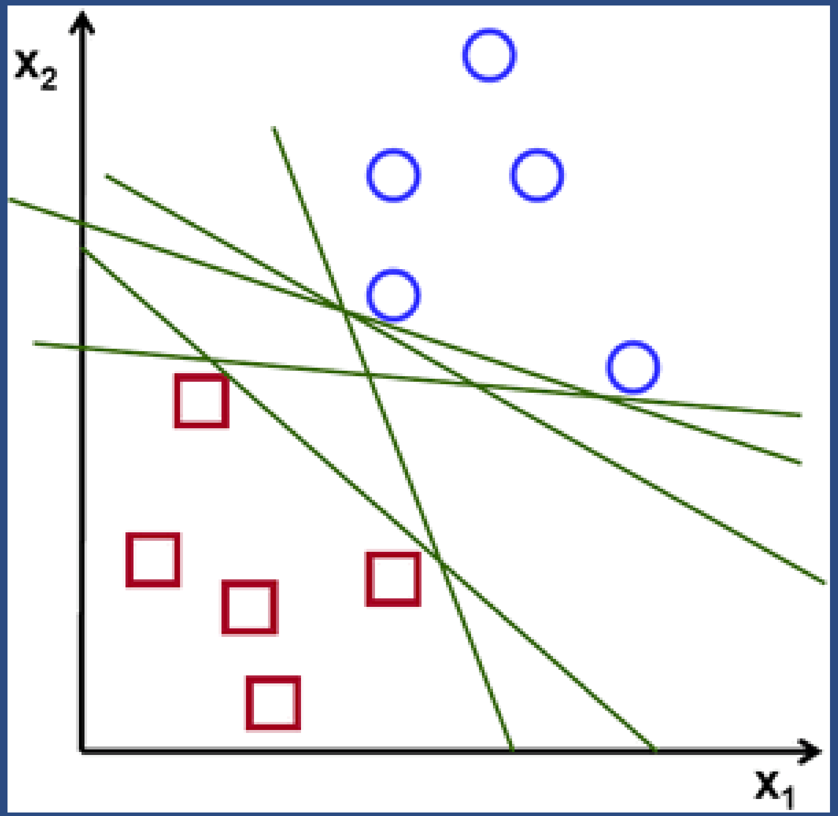
STACKING

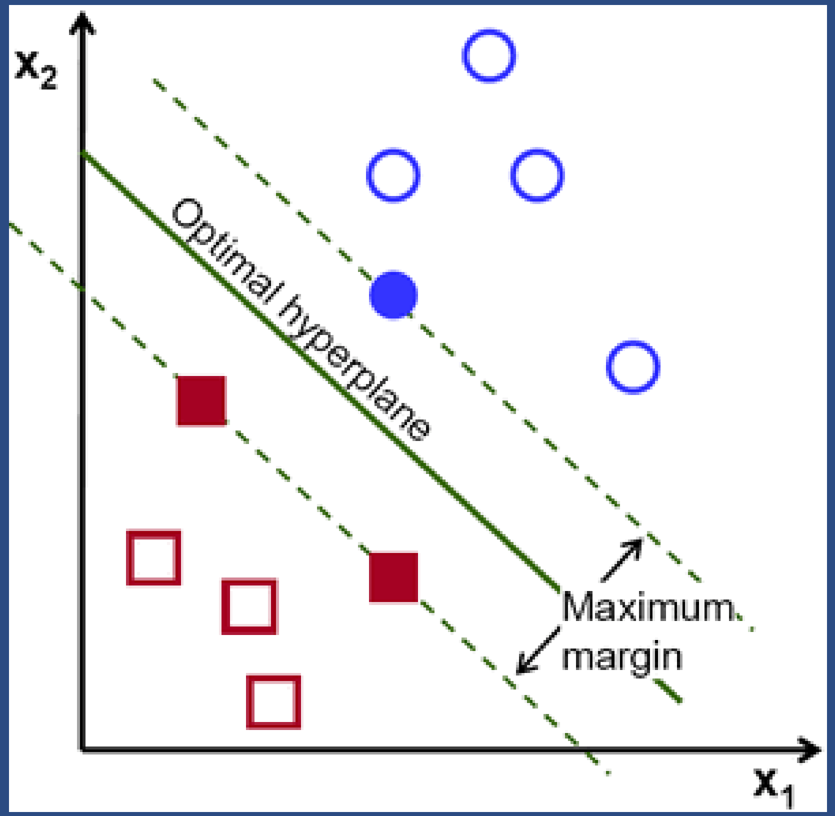
- StackingClassifier
- StackingRegressor
- make_classification
- make_regression
- ML Ensemble
- H20



CLASSIFICATION II

Support Vector Machines (SVM)





Apa itu Support Vector Machines (SVM)?

Seperangkat metode supervised learning yang digunakan untuk klasifikasi, regresi, dan deteksi outlier.

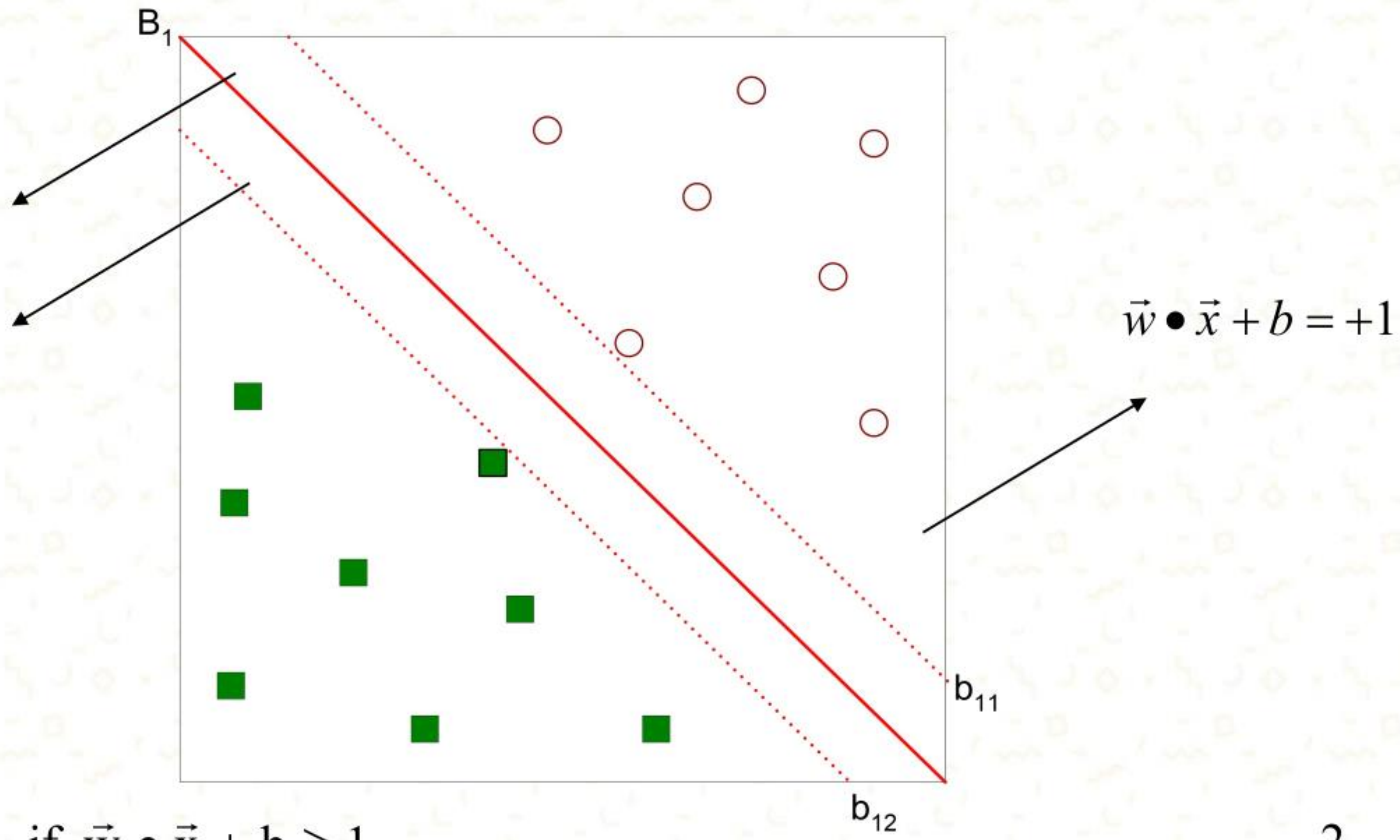
Tujuan Support Vector Machines (SVM)

Untuk menemukan hyperplane dalam ruang N-dimensi (N – jumlah fitur) yang secara jelas mengklasifikasikan data poins.

Terdapat banyak kemungkinan hyperplane yang dapat dipilih, hyperline mana yang harus dipakai?

Tenemukan bidang yang memiliki **margin maksimum**, yaitu jarak maksimum antara titik data dari kedua kelas.

Support Vector Machines (SVM)



$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\|\vec{w}\|^2}$$

Bagaimana jika tidak bisa dipisahkan secara linear (not linearly separable)?

- *Introduce slack variables*
  - *Need to minimize:*

$$L(w) = \frac{\|\vec{w}\|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i^k\right)$$

  - *Subject to:*

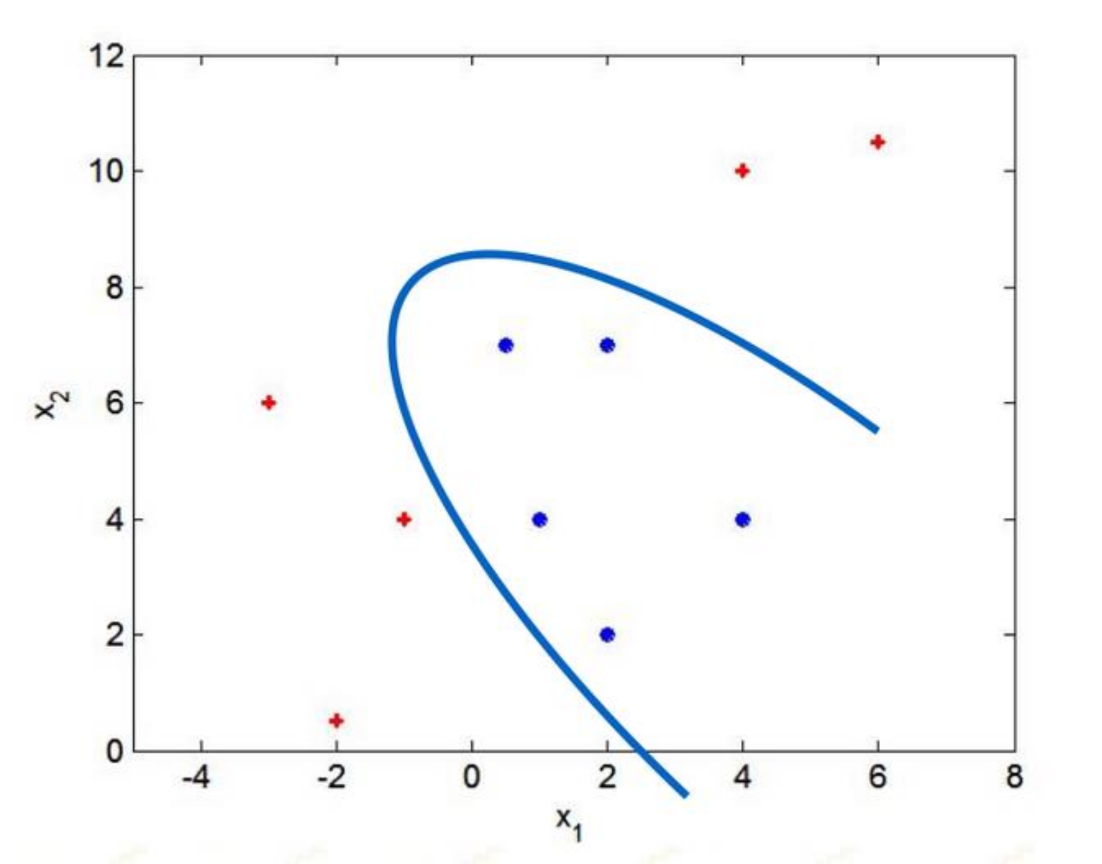
$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_{\mathrm{i}} + \mathrm{b} \geq 1 - \xi_{\mathrm{i}} \\ -1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_{\mathrm{i}} + \mathrm{b} \leq -1 + \xi_{\mathrm{i}} \end{cases}$$

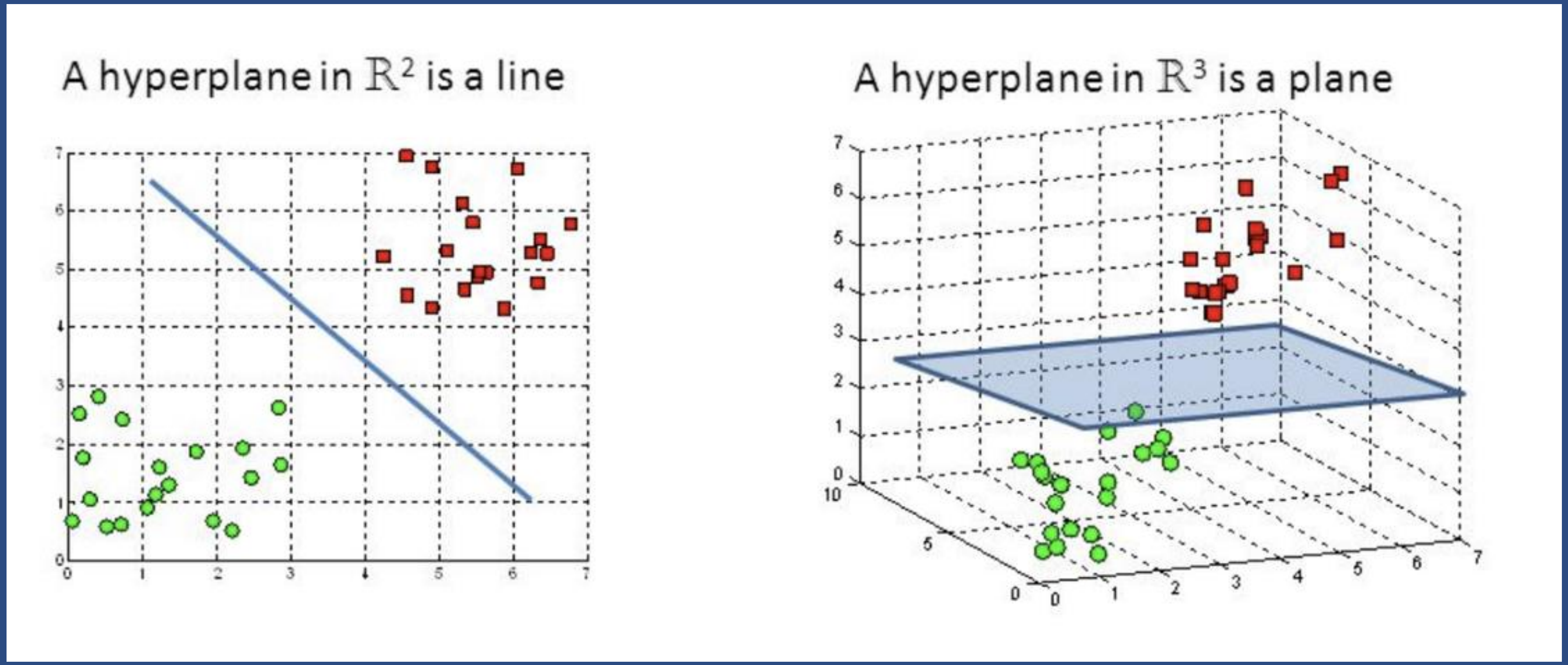
DigitalSkola

Nonlinear Support Vector Machines

BAGAIMANA JIKA BATAS KEPUTUSAN TIDAK LINIER (DECISION BOUNDARY IS NOT LINEAR)?

Ubah data ke ruang dimensi yang lebih tinggi (higher dimensional space)

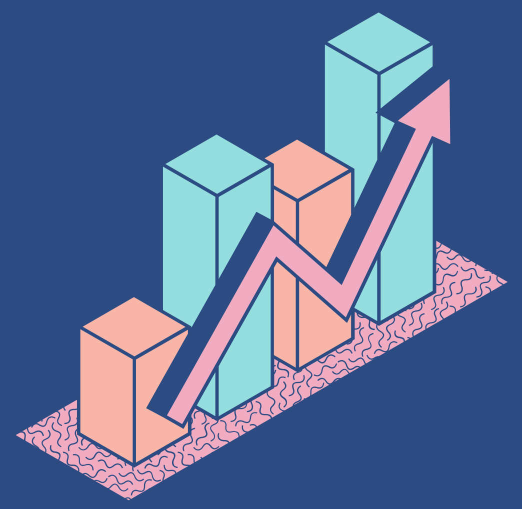




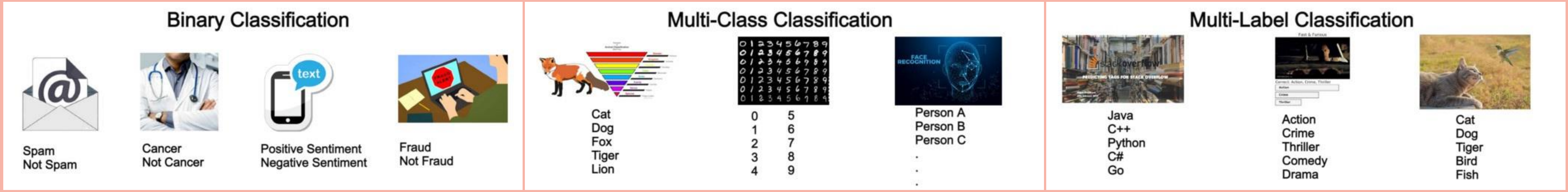
Contoh Script Model SVM

```python
#import library
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from imblearn.metrics import sensitivity_specificity_support
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OrdinalEncoder
from sklearn.svm import SVC

df = pd.read_csv('https://raw.githubusercontent.com/ganjar87/data_science_practice/main/BankChurners.csv', delimiter=',')

# get X and y
df_X = df.drop(['CLIENTNUM',  'Attrition_Flag'],axis=1)
df_y = df[['Attrition_Flag']]

#label encoding for y
le = LabelEncoder()
df_y= le.fit_transform(df_y['Attrition_Flag'])

#categorical encoding
cats = df_X.select_dtypes(include=['object', 'bool']).columns
cat_features = list(cats.values)
le = LabelEncoder()
for i in cat_features:
  df_X[i] = le.fit_transform(df_X[i])

#menyimpan X dan y menjadi numpy arrays
X = df_X.astype(float).values
y = df_y.astype(float)

#define dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

#scaling
scaler = StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

```python
#fit model
model=SVC()
model.fit(X_train, y_train)

#make final predictions
y_pred = model.predict(X_test)

print('Accuracy ',accuracy_score(y_test, y_pred))
print('Precision ',precision_score(y_test, y_pred, average='macro'))
print('Recall ',recall_score(y_test, y_pred, average='macro'))
print('Confusion matrix ', confusion_matrix(y_test, y_pred))
plot_confusion_matrix(model, X_test, y_test, cmap=plt.cm.Blues)
plt.show()
```



Three Type of Classification Tasks



Binary Classification

The target class label has **TWO CLASSES** and the task is to **predict one of the classes.**

Multi-Class Classification

The number of target class labels is **more than two,** and **ONLY** one class can be predicted as output.

Multi-Label Classification

The number of target class labels is **more than two,** and **MORE THAN** one class can be predicted as output.

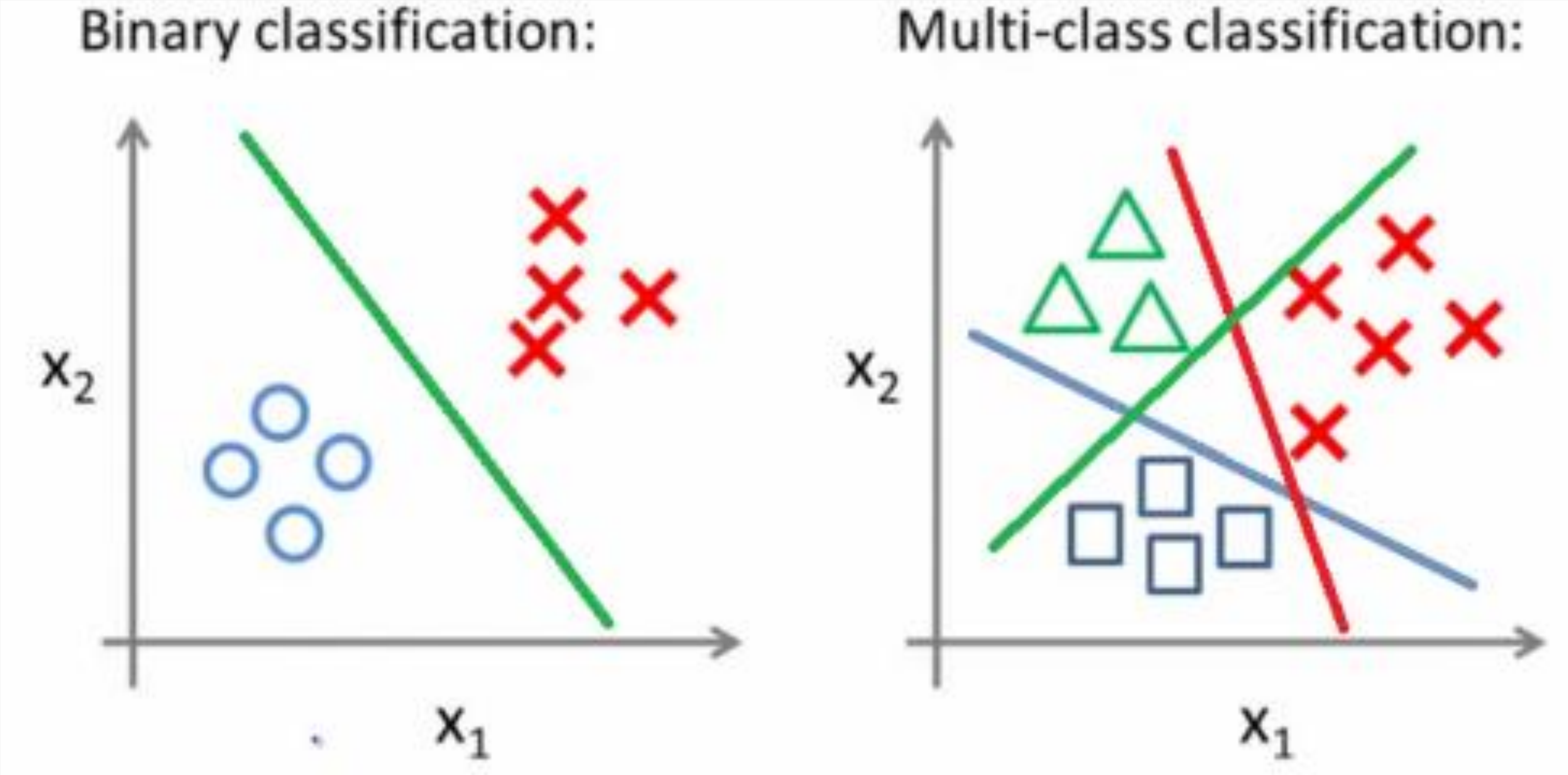
Binary to Multi-Class

BISAKAH KITA MENGGUNAKAN BINARY CLASSIFIER UNTUK MEMBANGUN MULTICLASS classifier?

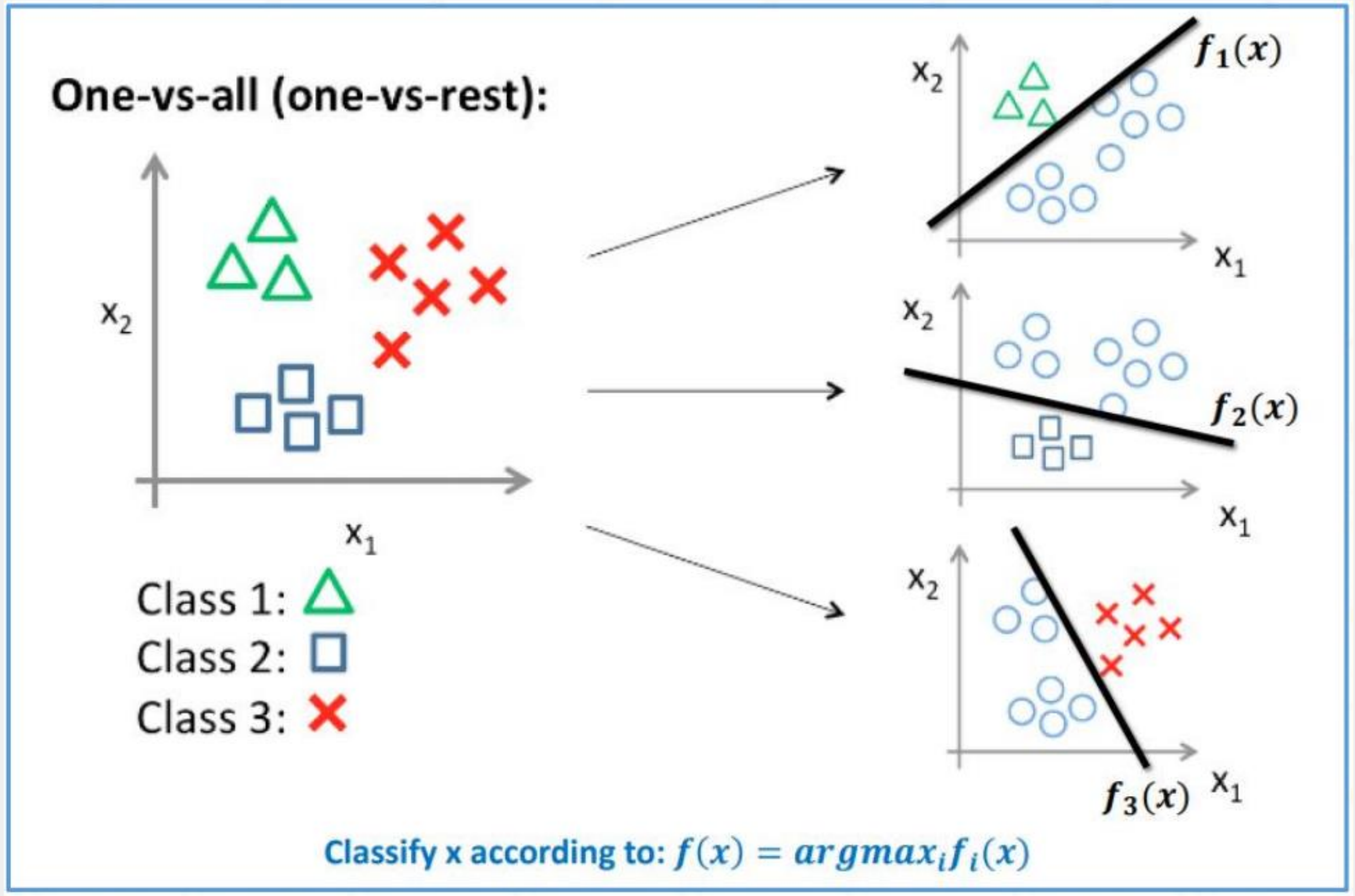
Mengurai prediksi menjadi multiple binary decisions

CARA YANG BISA DIPAKAI?

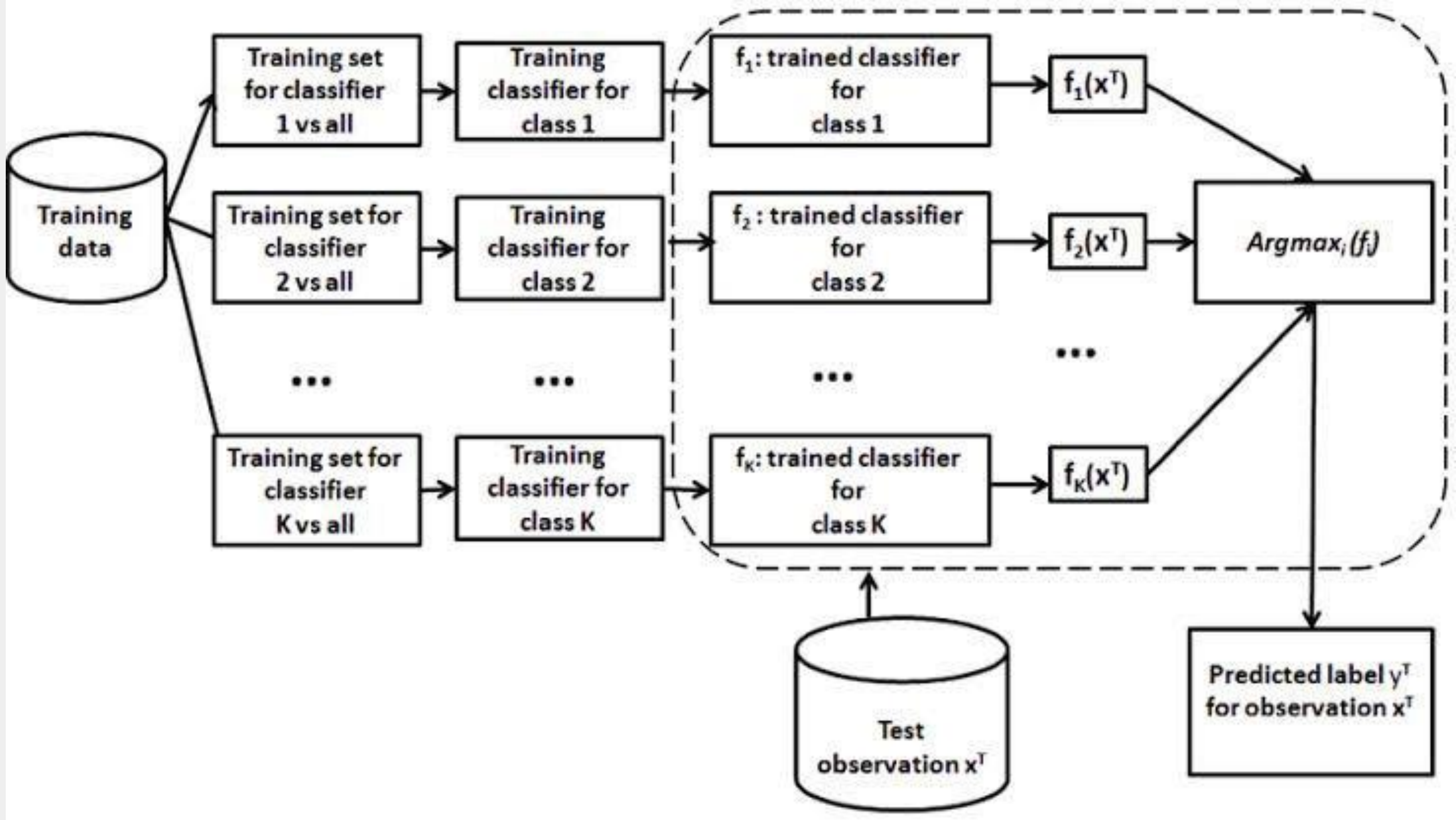
- One-vs-All (One vs Rest / OvR)
- One vs One (OvO)



Visualizing One vs Rest (OvR)



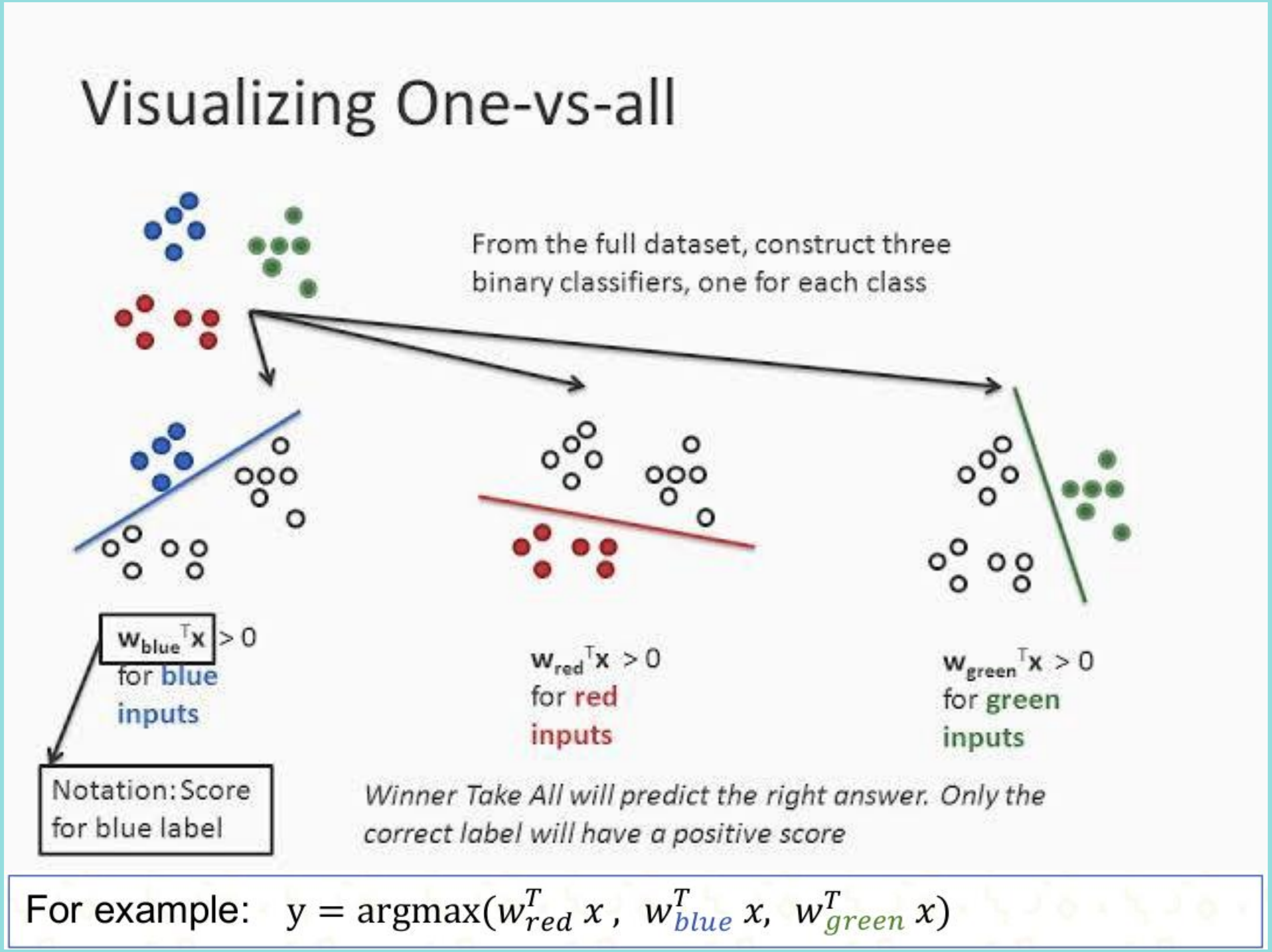
One vs Rest (OvR) Learning Algorithm

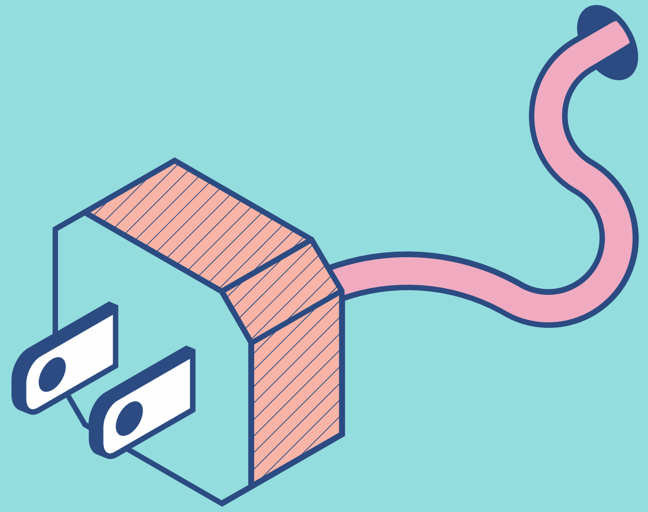


One vs Rest (OvR) Inference Algorithm

Inference: "Winner takes all"

$$\hat{y} = \text{argmax}_{y \in \{1,2,\dots,K\}} \, w_y^T x$$

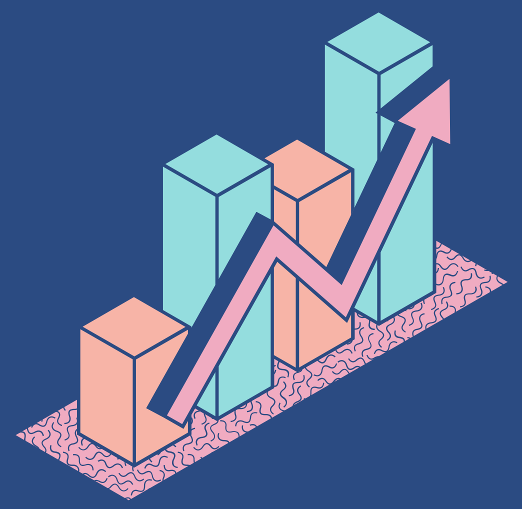




Contoh Script Model OvR

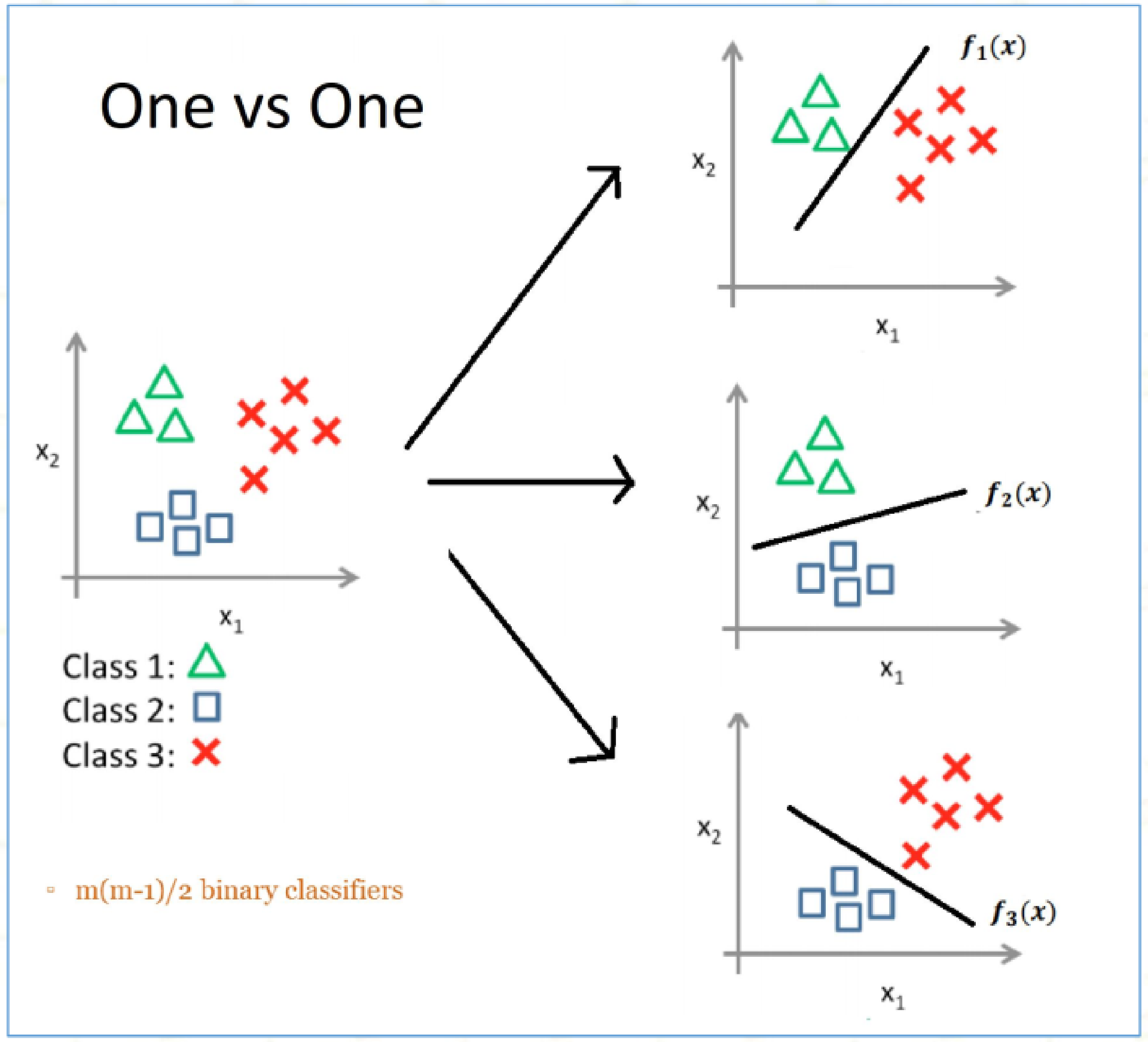
```python
#import library
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from imblearn.metrics import sensitivity_specificity_support
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OrdinalEncoder
from sklearn.svm import SVC

df = pd.read_csv('https://raw.githubusercontent.com/ganjar87/data_science_practice/main/BankChurners.csv', delimiter=',')

# get X and y
df_X = df.drop(['CLIENTNUM',  'Attrition_Flag'],axis=1)
df_y = df[['Attrition_Flag']]

#label encoding for y
le = LabelEncoder()
df_y= le.fit_transform(df_y['Attrition_Flag'])

#categorical encoding
cats = df_X.select_dtypes(include=['object', 'bool']).columns
cat_features = list(cats.values)
le = LabelEncoder()
for i in cat_features:
  df_X[i] = le.fit_transform(df_X[i])

#menyimpan X dan y menjadi numpy arrays
X = df_X.astype(float).values
y = df_y.astype(float)

#define dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

#scaling
scaler = StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

```python
#fit model
model=SVC(decision_function_shape='ovr')
model.fit(X_train, y_train)

#make final predictions
y_pred = model.predict(X_test)

print('Accuracy ',accuracy_score(y_test, y_pred))
print('Precision ',precision_score(y_test, y_pred, average='macro'))
print('Recall ',recall_score(y_test, y_pred, average='macro'))
print('Confusion matrix ', confusion_matrix(y_test, y_pred))
plot_confusion_matrix(model, X_test, y_test, cmap=plt.cm.Blues)
plt.show()
```



Visualizing One vs One (OvO)

- Buat sebuah classifier untuk setiap pasangan classes
- Diberikan m classes, buat binary classifiers berjumlah m(m-1)/2
- Setiap classifier mempelajari data dari 2 classes (2 pasangan yang berbeda)
- Untuk memprediksi data baru X, setiap classifier melakukan voting.
- Data baru X diberikan class dengan voting suara terbanyak



Classify x according to majority voting

DigitalSkola

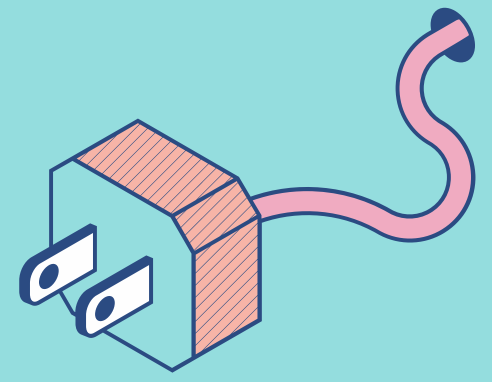
One vs One (OvO) Learning Algorithm

- Learning: Diberikan dataset $D = \{(x_i, y_i)\}$
  $x_i \in R^n,\ y_i \in \{1,2,3, \ldots K\}$
- Memecah menjadi binary classification dengan jumlah K(K-1)/2. (K=jumlah class)
  - Biarkan model dengan jumlah K(K-1)/2 belajar dari training set:
    $w_1,\ w_2,\ w_3, \ldots w_{K*(K-1)/2}$
  - Untuk setiap pasangan class (i,j), buat sebuah binary classification
  - dengan tugas mempelajari:
    - Data Positive : Data dari dataset D dengan label i
    - Data Negative: Data dari dataset D dengan label j
  - Binary classification dapat diselesaikan dengan semua algoritma

One vs One (OvO) Inference Algorithm

Prediction:

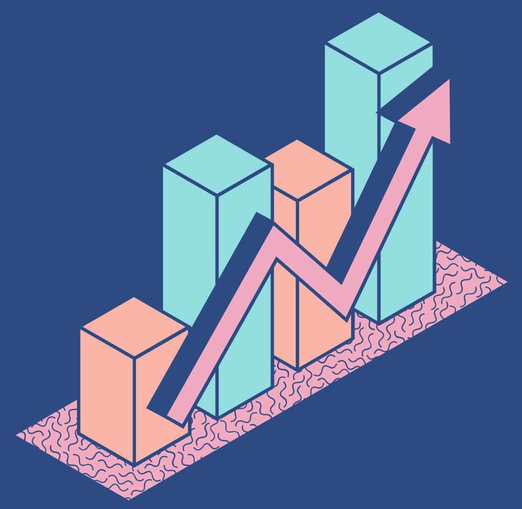
- Majority (mayoritas): Pick the label with maximum votes (pilih label dengan jumlah voting suara terbanyak)



Contoh Script Model OvO

```
#import library
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from imblearn.metrics import sensitivity_specificity_support
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OrdinalEncoder
from sklearn.svm import SVC

df = pd.read_csv('https://raw.githubusercontent.com/ganjar87/data_science_practice/main/BankChurners.csv', delimiter=',')

# get X and y
df_X = df.drop(['CLIENTNUM',  'Attrition_Flag'],axis=1)
df_y = df[['Attrition_Flag']]

#label encoding for y
le = LabelEncoder()
df_y= le.fit_transform(df_y['Attrition_Flag'])

#categorical encoding
cats = df_X.select_dtypes(include=['object', 'bool']).columns
cat_features = list(cats.values)
le = LabelEncoder()
for i in cat_features:
  df_X[i] = le.fit_transform(df_X[i])

#menyimpan X dan y menjadi numpy arrays
X = df_X.astype(float).values
y = df_y.astype(float)

#define dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

#scaling
scaler = StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

```
#fit model
model=SVC(decision_function_shape='ovo')
model.fit(X_train, y_train)

#make final predictions
y_pred = model.predict(X_test)

print('Accuracy ',accuracy_score(y_test, y_pred))
print('Precision ',precision_score(y_test, y_pred, average='macro'))
print('Recall ',recall_score(y_test, y_pred, average='macro'))
print('Confusion matrix ', confusion_matrix(y_test, y_pred))
plot_confusion_matrix(model, X_test, y_test, cmap=plt.cm.Blues)
plt.show()
```



DigitalSkola

THANK YOU!

By Omicron