AI VIETNAM
All-in-One Course

Final Project: Tree and Its Variant

Vinh Dinh Nguyen
PhD in Computer Science

Year 2023

Outline

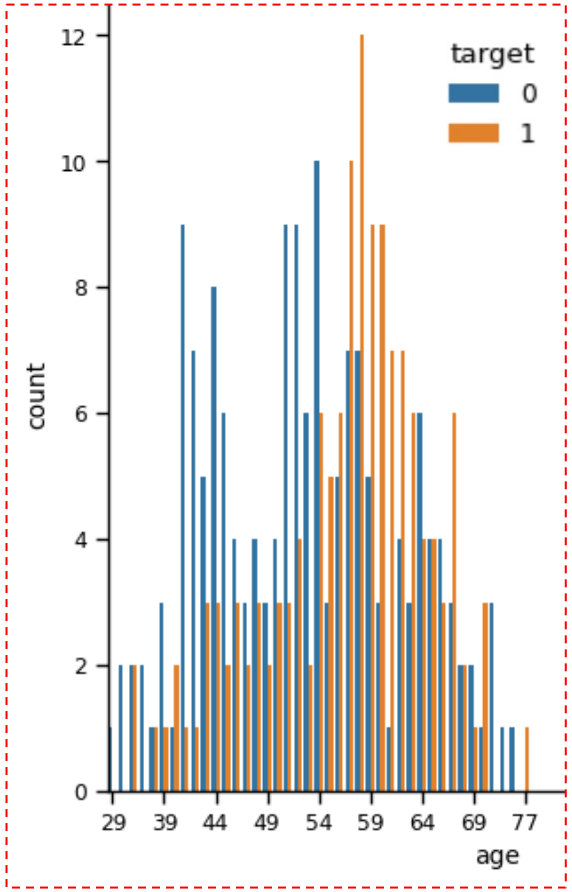
- Introduction to Dataset and Problem
- Solving Problem by using KNN
- Solving Problem by Naïve Bayes Classifier
- Solving Problem by Decision Tree
- Solving Problem by Random Forest
- Solving Problem by AdaBoost
- Solving Problem by Gradient Boosting
- Solving Problem by XGBoost
- Solving Problem by SVM
- Solving Problem by Stacking
- XGBoost with Time Series Data
- Feature Important in DT

Heart Disease Dataset

13 feature, 1 label, 303 samples

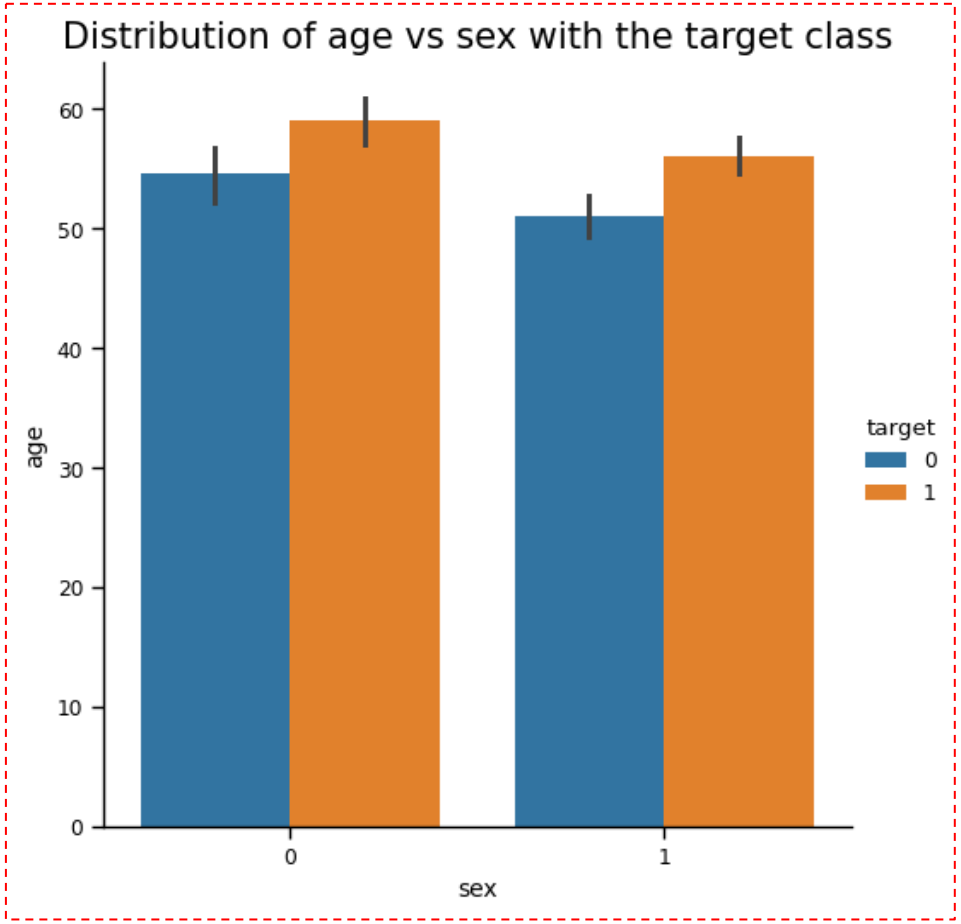
Age	Sex	CP	Restbps	Chol	Fbs	restecg	Thalach	Exang	Oldpeak	Slope	Ca	Thal	Target
63	**1**	**1**	**145**	**233**	**1**	**2**	**150**	**0**	**2.3**	**3**	**0**	**6**	**0**
67	1	4	160	286	0	2	108	1	1.5	2	3	3	2
67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
37	1	3	130	250	0	0	187	0	3.5	3	0	3	0
41	0	2	130	204	0	2	172	0	1.4	1	0	3	0
56	1	2	120	236	0	0	178	0	0.8	1	0	3	0
62	0	4	140	268	0	2	160	0	3.6	3	2	3	3
57	0	4	120	354	0	0	163	1	0.6	1	0	3	0
63	1	4	130	254	0	2	147	0	1.4	2	1	7	2
53	1	4	140	203	1	2	155	1	3.1	3	0	7	1
57	1	4	140	192	0	0	148	0	0.4	2	0	6	0
56	0	2	140	294	0	2	153	0	1.3	2	0	3	0
56	1	3	130	256	1	2	142	1	0.6	2	1	6	2
44	1	2	120	263	0	0	173	0	0	1	0	7	0

Distribution of Target vs Age




```python
# Bài tập 1
df = pd.read_csv('cleveland.csv', header = None)
df.columns = ['age', 'sex', 'cp', 'trestbps', 'chol',
              'fbs', 'restecg', 'thalach', 'exang',
              'oldpeak', 'slope', 'ca', 'thal', 'target']
df['target'] = df.target.map({0: 0, 1: 1, 2: 1, 3: 1, 4: 1})
df['thal'] = df.thal.fillna(df.thal.mean())
df['ca'] = df.ca.fillna(df.ca.mean())

# distribution of target vs age
sns.set_context("paper", font_scale = 1, rc = {"font.size": 3,"axes.titlesize": 15,"axes.labelsize": 10})
ax = sns.catplot(kind = 'count', data = df, x = 'age', hue = 'target', order = df['age'].sort_values().unique())
# ax.set(xticklabels=[])
ax.ax.set_xticks(np.arange(0, 80, 5))
plt.title('Variation of Age for each target class')
plt.show()
```

Distribution of age vs sex with the target class

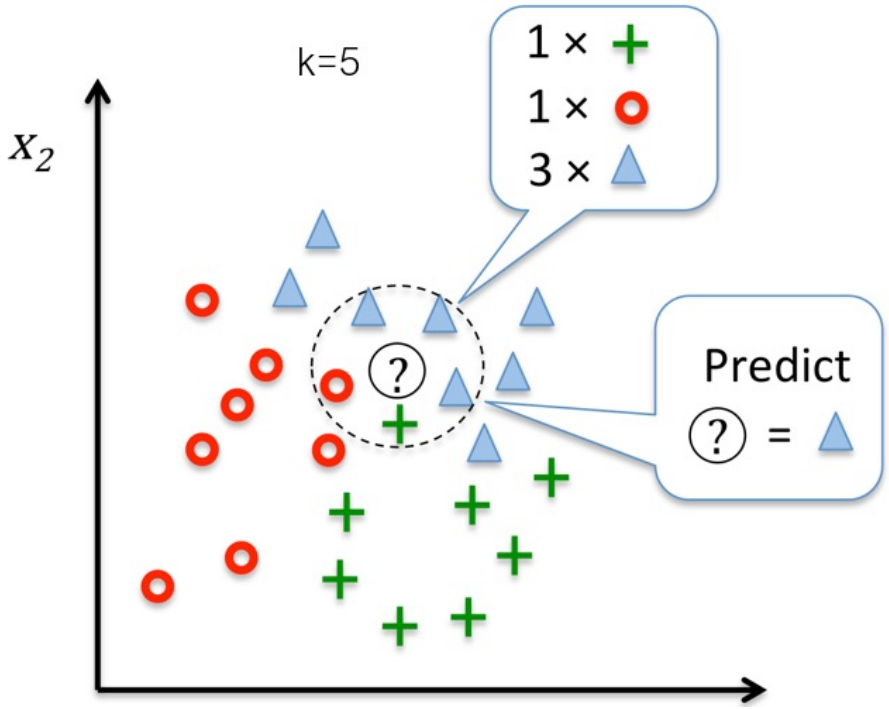


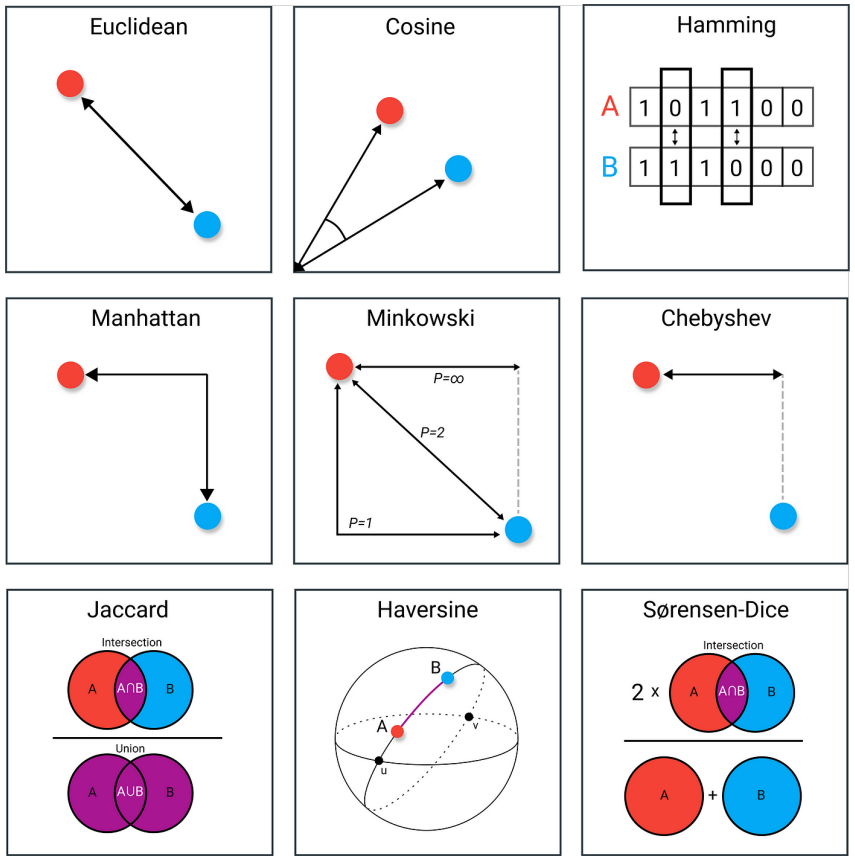

```
# bai tap 2
# barplot of age vs sex with hue = target
sns.catplot(kind = 'bar', data = df, y = 'age', x = 'sex', hue = 'target')
plt.title('Distribution of age vs sex with the target class')
plt.show()
```

Outline

- Introduction to Dataset and Problem
- Solving Problem by using KNN
- Solving Problem by Naïve Bayes Classifier
- Solving Problem by Decision Tree
- Solving Problem by Random Forest
- Solving Problem by AdaBoost
- Solving Problem by Gradient Boosting
- Solving Problem by XGBoost
- Solving Problem by SVM
- Solving Problem by Stacking
- Time Series Data
- Feature Important in DT

K-NN





Why should K be an odd number?
Can we set K with an even number?
KNN Limitations?

https://towardsdatascience.com/9-distance-measures-in-data-science-918109d069fa

K-NN

a) accuracy for train = 0.76 and accuracy for test = 0.69
b) accuracy for train = 1.76 and accuracy for test = 0.69
c) accuracy for train = 2.76 and accuracy for test = 0.69
d) accuracy for train = 3.76 and accuracy for test = 0.69

```
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)

from sklearn.neighbors import KNeighborsClassifier
# define the model
classifier = KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski')

classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix
cm_test = confusion_matrix(y_pred, y_test)

y_pred_train = classifier.predict(X_train)
cm_train = confusion_matrix(y_pred_train, y_train)

print()
accuracy_for_train = np.round((cm_train[0][0] + cm_train[1][1])/len(y_train),2)
accuracy_for_test = np.round((cm_test[0][0] + cm_test[1][1])/len(y_test),2)
print('Accuracy for training set for KNeighborsClassifier = {}'.format(accuracy_for_train))
print('Accuracy for test set for KNeighborsClassifier = {}'.format(accuracy_for_test))
```

Outline

- Introduction to Dataset and Problem
- Solving Problem by using KNN
- Solving Problem by Naïve Bayes Classifier
- Solving Problem by Decision Tree
- Solving Problem by Random Forest
- Solving Problem by AdaBoost
- Solving Problem by Gradient Boosting
- Solving Problem by XGBoost
- Solving Problem by SVM
- Solving Problem by Stacking
- Time Series Data
- Feature Important in DT

Bayes' Rule

For any two events A and B, where $P(A) \neq 0$:

LIKELIHOOD
The probability of "A" being True, given "B" True

PRIOR
The probability of "B" being True. This is the knowledge

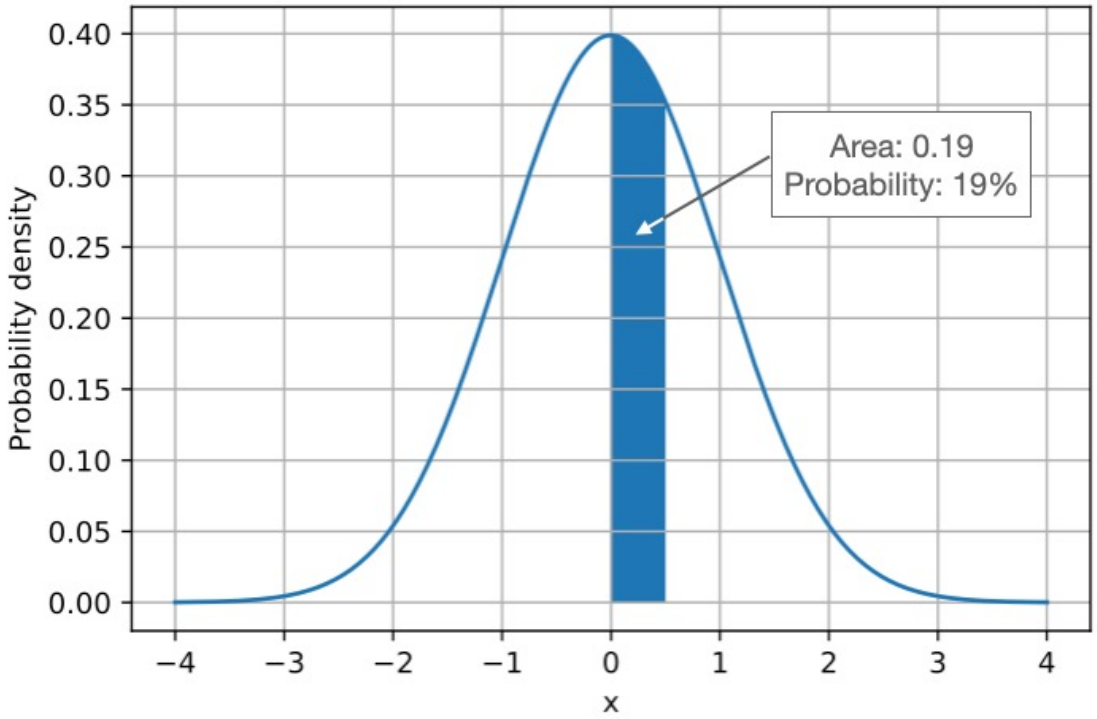
$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

POSTERIOR
The probability of "B" being True. Given "A" True

MARGINALIZATION
The probability of "A" being True.

Probabilities Vs Likelihood

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$



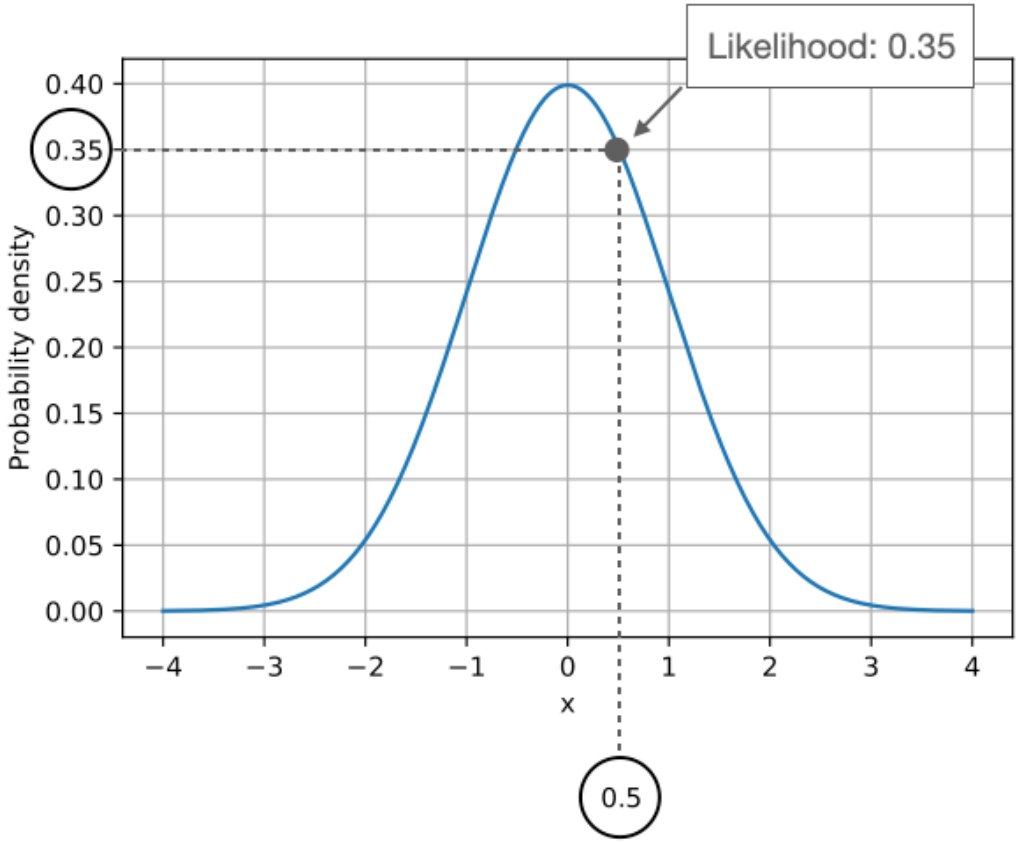
The probability of drawing a sample x with a value between 0 and 0.5 for a standard normal distribution (mean 0 and standard deviation 1)

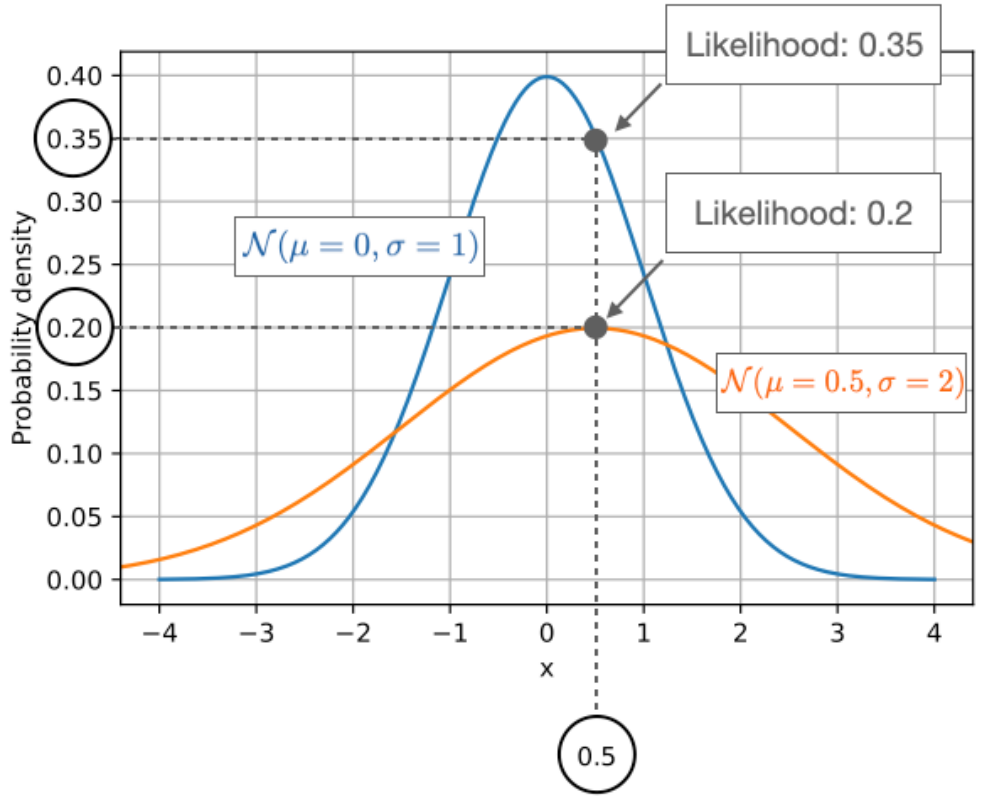
➡

p(0< x < 0.5 | mean = 0 and standard deviation = 1)

$$\Pr(0 < x < 0.5|\mathbf{w}) = \int_0^{0.5} p(x|\mathbf{w})dx$$

Probabilities Vs Likelihood

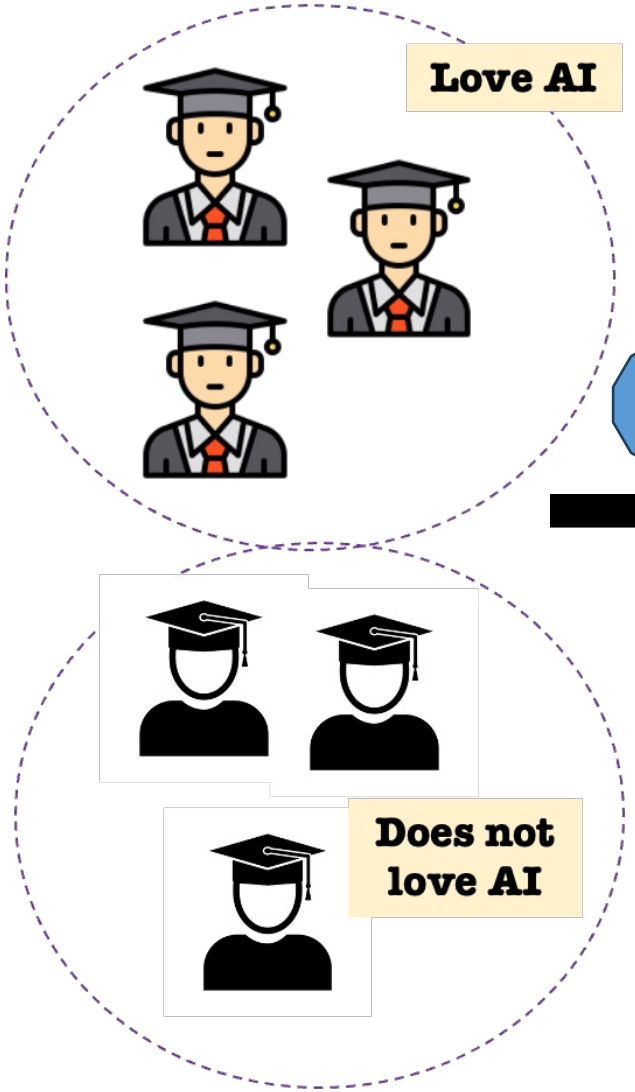




L(mean = 0 and standard deviation = 1 | x = 0.5)

L(mean = 0.5 and standard deviation = 2 | x = 0.5)

Gaussian Naive Bayes



1

2

Given a student information (math, art, and english scores), what is the probability that he/she loves AI or not.

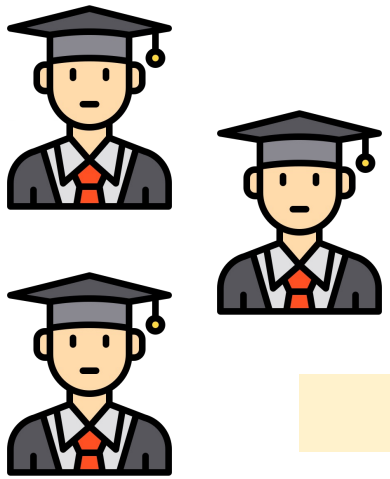
Gaussian Naive Bayes Classifier

2

Does she love AI or Not

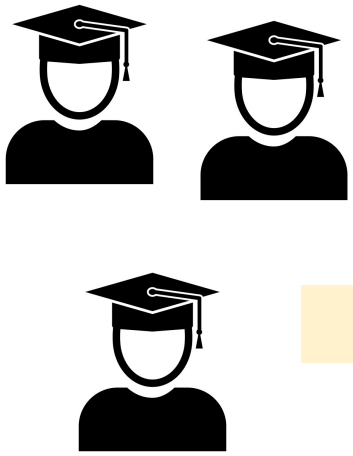
Gaussian Naive Bayes

Math	Art	English
9.5	7.5	5.5
8.2	8.0	6.5
7.0	9.0	7.0
…	…	…



Love AI

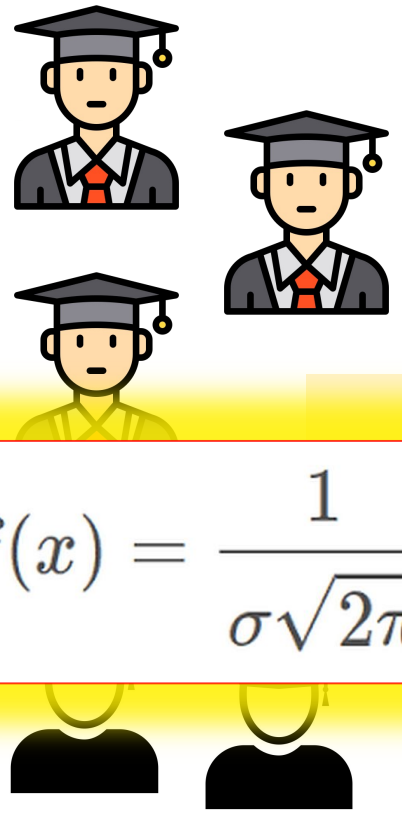
Math	Art	English
1.5	6.5	8.5
5.0	8.5	8.5
9.0	8.0	8.0
…	…	…



Does not love AI

Gaussian Naive Bayes

	Math	Art	English
	9.5	7.5	5.5
	8.2	8.0	6.5
	7.0	9.0	7.0
Mean	…	…	…
Std	…	…	…



Love AI

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

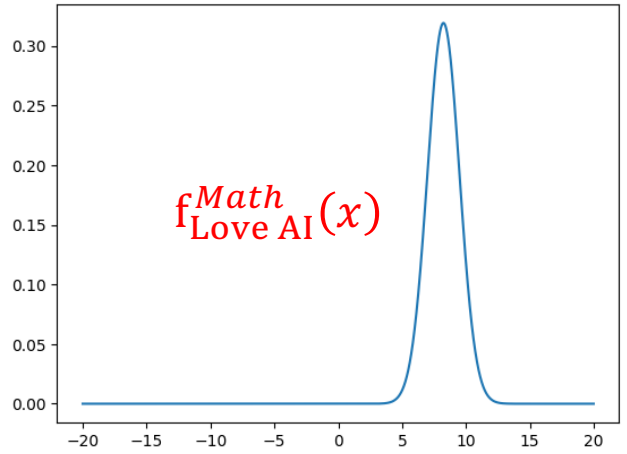
	Math	Art	English
	1.5	6.5	8.5
	5.0	8.5	8.5
	9.0	8.0	8.0
Mean	…	…	…
Std	…	…	…

Does not love AI

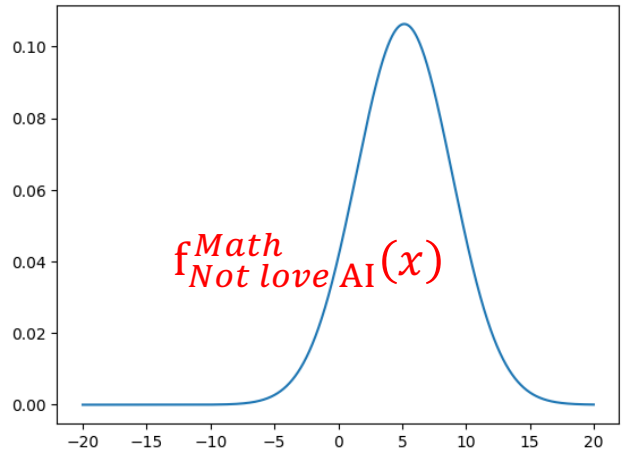
Gaussian Naive Bayes

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

Love AI	Math	Art	English
	9.5	7.5	5.5
	8.2	8.0	6.5
	7.0	9.0	7.0
μ	…	…	…
σ	…	…	…



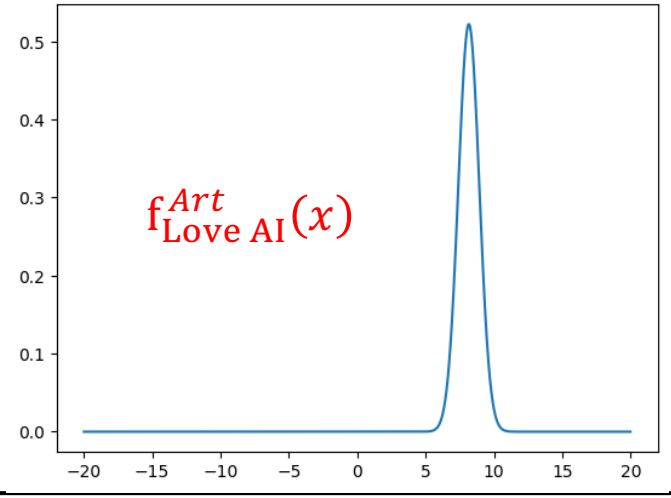
Not Love AI	Math	Art	English
	1.5	6.5	8.5
	5.0	8.5	8.5
	9.0	8.0	8.0
μ	…	…	…
σ	…	…	…



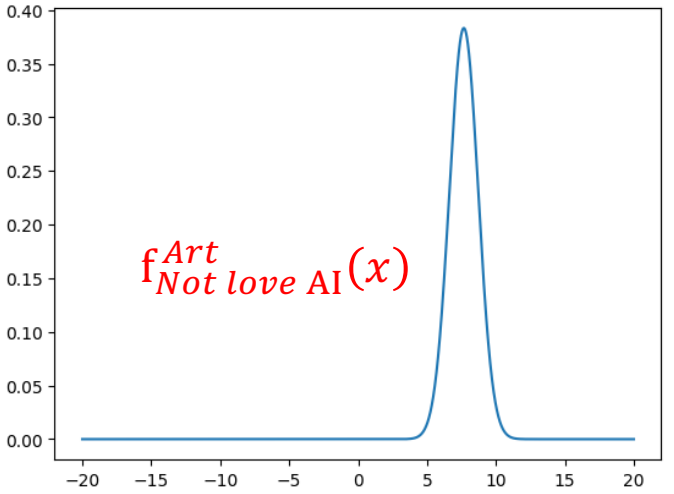
Gaussian Naive Bayes

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

Love AI	Math	Art	English
	9.5	7.5	5.5
	8.2	8.0	6.5
	7.0	9.0	7.0
μ	…	…	…
σ	…	…	…



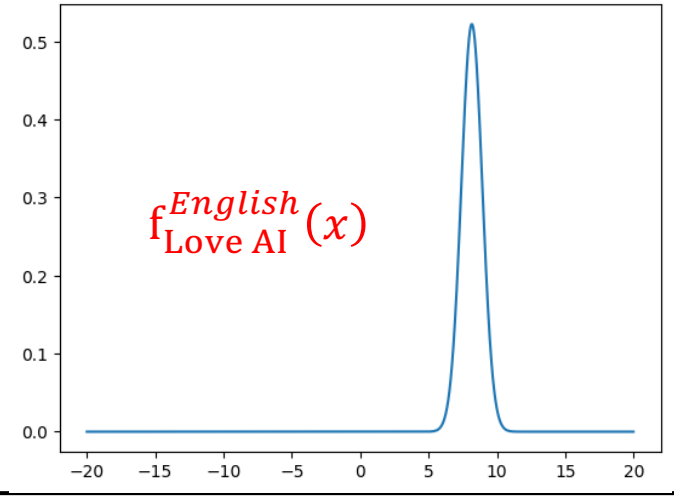
Not Love AI	Math	Art	English
	1.5	6.5	8.5
	5.0	8.5	8.5
	9.0	8.0	8.0
μ	…	…	…
σ	…	…	…



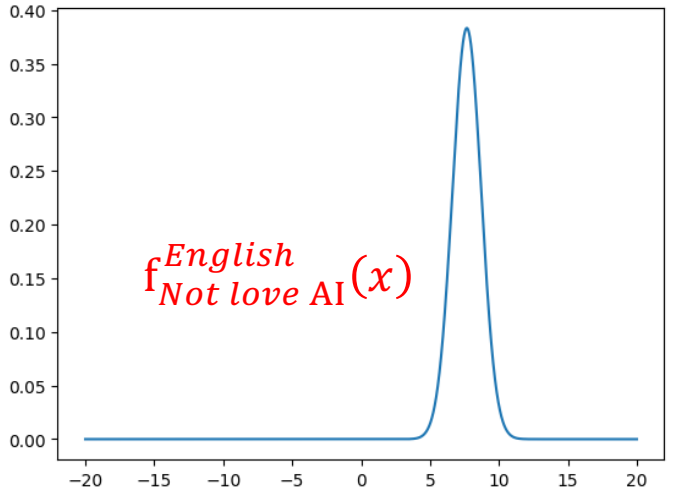
Gaussian Naive Bayes

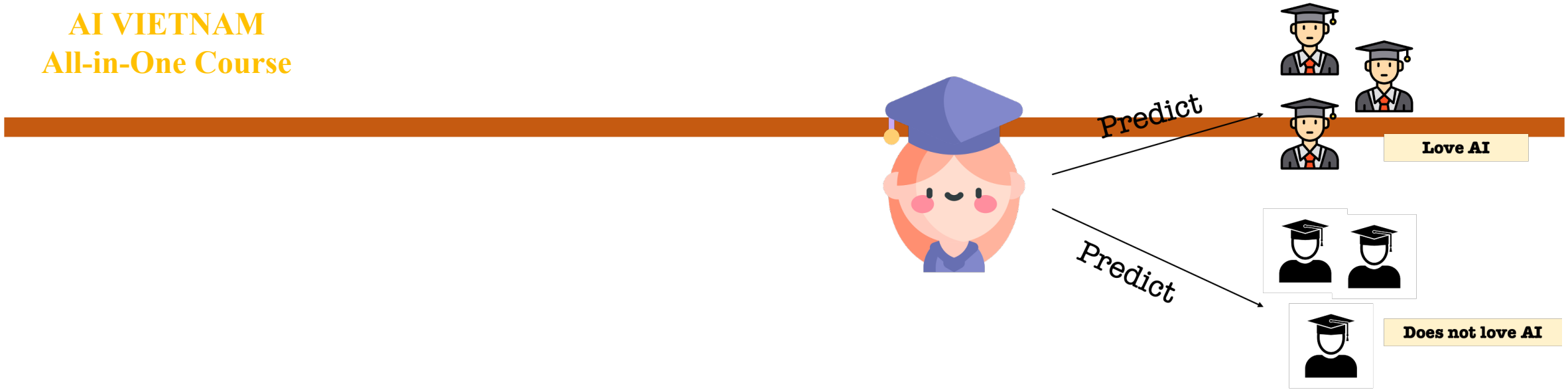
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

Love AI	Math	Art	English
	9.5	7.5	5.5
	8.2	8.0	6.5
	7.0	9.0	7.0
μ	…	…	…
σ	…	…	…



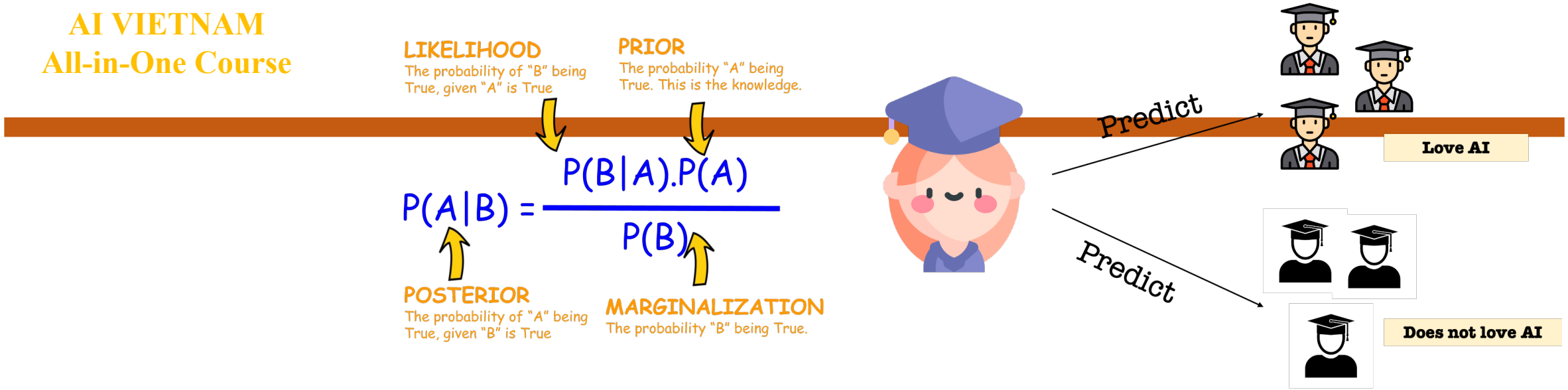
Not Love AI	Math	Art	English
	1.5	6.5	8.5
	5.0	8.5	8.5
	9.0	8.0	8.0
μ	…	…	…
σ	…	…	…





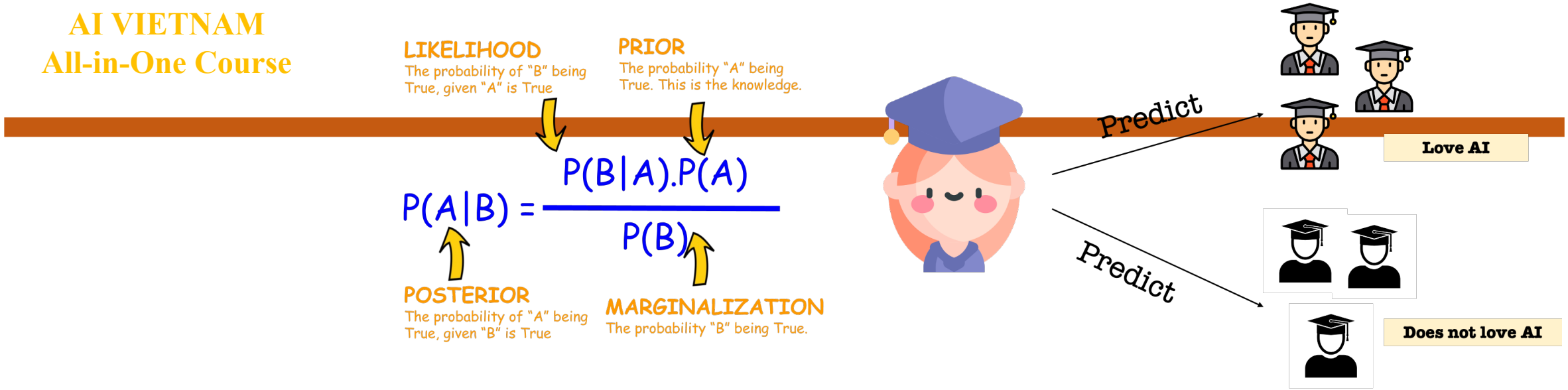
P (love AI) = Initial Guesses = Prior probability = 3/6 = 0.5

P (Not love AI) = Initial Guesses = Prior probability = 3/6 = 0.5



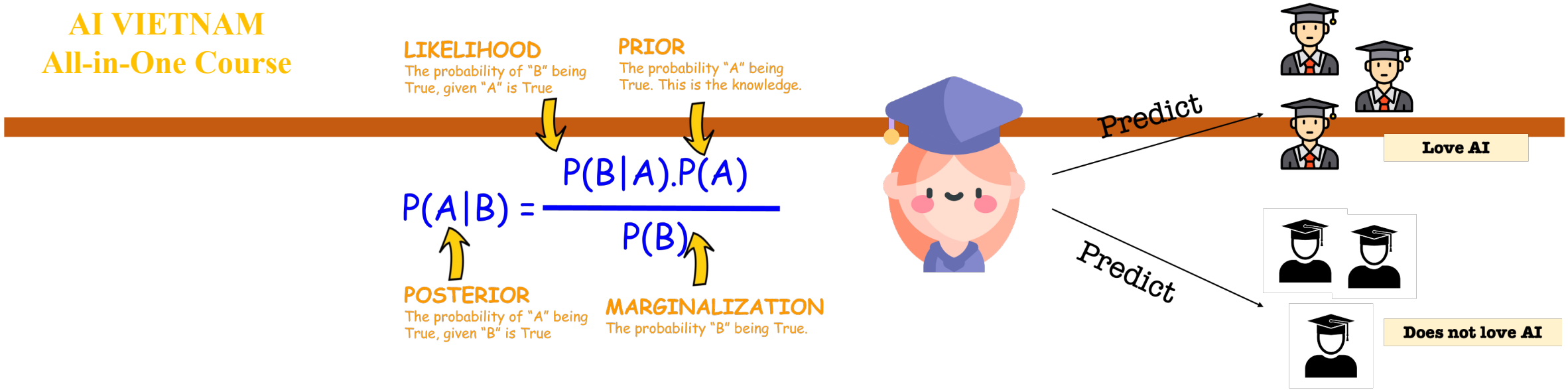
$$P(\text{Love AI} \mid \text{Math} = 5 \,\&\, \text{Art} = 6 \,\&\, \text{English} = 7) = \frac{P(\text{Math} = 5 \,\&\, \text{Art} = 6 \,\&\, \text{English} = 7 \mid \text{Love AI}) \cdot P(\text{Love AI})}{P(\text{Math} = 5 \,\&\, \text{Art} = 6 \,\&\, \text{English} = 7)}$$

$$P(\text{Not Love AI} \mid \text{Math} = 5 \,\&\, \text{Art} = 6 \,\&\, \text{English} = 7) = \frac{P(\text{Math} = 5 \,\&\, \text{Art} = 6 \,\&\, \text{English} = 7 \mid \text{Not Love AI}) \cdot P(\text{Not Love AI})}{P(\text{Math} = 5 \,\&\, \text{Art} = 6 \,\&\, \text{English} = 7)}$$



$$P(\text{Love AI} \mid \text{Math} = 5 \,\&\, \text{Art} = 6 \,\&\, \text{English} = 7) = \frac{P(\text{Math} = 5 \,\&\, \text{Art} = 6 \,\&\, \text{English} = 7 \mid \text{Love AI}) \cdot P(\text{Love AI})}{P(\text{Math} = 5 \,\&\, \text{Art} = 6 \,\&\, \text{English} = 7)}$$

$$P(\text{Not Love AI} \mid \text{Math} = 5 \,\&\, \text{Art} = 6 \,\&\, \text{English} = 7) = \frac{P(\text{Math} = 5 \,\&\, \text{Art} = 6 \,\&\, \text{English} = 7 \mid \text{Not Love AI}) \cdot P(\text{Not Love AI})}{P(\text{Math} = 5 \,\&\, \text{Art} = 6 \,\&\, \text{English} = 7)}$$



P (Love AI |Math = 5 & Art = 6 & English = 7) = P (Math = 5 & Art = 6 & English = 7 | Love AI) . P(Love AI)

P (Math = 5| Love AI) . P(Art = 6| Love AI) . P(English = 7 | Love AI) . P(Love AI)

$$f^{Math}_{\text{Love AI}}(x = 5) \qquad f^{Art}_{love\ \text{AI}}(x = 6) \qquad f^{English}_{love\ \text{AI}}(x) = 7$$

P (Not Love AI |Math = 5 & Art = 6 & English = 7) = P (Math = 5 & Art = 6 & English = 7 | Not Love AI) . P(Not Love AI)

P (Math = 5| Not Love AI) . P(Art = 6| Not Love AI) . P(English = 7 | Not Love AI) . P(Not Love AI)

$$f^{Math}_{not\ \text{Love AI}}(x = 5) \qquad f^{Art}_{not\ love\ \text{AI}}(x = 6) \qquad f^{English}_{not\ love\ \text{AI}}(x) = 7$$

Estimating probabilities

- **Bayes Theorem**

$$P(c|X) = \frac{P(X|c).P(c)}{P(X)}$$

  - $P(X)$ is constant for all classes
  - $P(c)$: relative freq of class c samples
  - c such that $P(c|X)$ is maximum = c such that $P(X|c).P(c)$ is maximum

Naïve Bayes Classifier

Bayes Classification

$$\mathbf{P(c|X) \propto P(X|c).P(c) = P(x_1, x_2, \dots, x_N|c).P(c)}$$

- Learning the join probability:

$$\overbrace{\mathbf{P(x_1, x_2, \dots, x_N|c)}}^{\text{likelihood}}$$

Naïve Bayes Classification

- Maximum Likelihood Estimation (MLE)
- Assumption: all input feature are conditionally independent!

$$\mathbf{P(x_1, x_2, \dots, x_N|c) = P(x_1|c)P(x_2|c) \dots P(x_N|c)}$$

Naïve Bayes Classifier

Naïve Bayes Classification

- Maximum Likelihood Estimation (MLE)
- Assumption: all input feature are conditionally independent!

$$\mathbf{P(x_1, x_2, \dots, x_N|c) = P(x_1|c)P(x_2|c) \dots P(x_N|c)}$$

$$\mathbf{P(c|X) \propto P(X|c).P(c)}$$
$$\mathbf{= P(x_1, x_2, \dots, x_N|c).P(c)}$$
$$\mathbf{= P(x_1|c)P(x_2|c) \dots P(x_N|c).P(c)}$$

Naïve Bayes Classifier

a) accuracy for train = 0.76 and accuracy for test = 0.69
b) accuracy for train = 0.66 and accuracy for test = 0.67
c) accuracy for train = 0.85 and accuracy for test = 0.84
d) accuracy for train = 3.76 and accuracy for test = 0.69

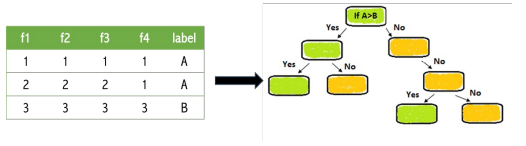
```
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)

from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)


# Predicting the Test set results
y_pred = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix
cm_test = confusion_matrix(y_pred, y_test)

y_pred_train = classifier.predict(X_train)
cm_train = confusion_matrix(y_pred_train, y_train)

print()
accuracy_for_train = np.round((cm_train[0][0] + cm_train[1][1])/len(y_train),2)
accuracy_for_test = np.round((cm_test[0][0] + cm_test[1][1])/len(y_test),2)
print('Accuracy for training set for GaussianNB = {}'.format(accuracy_for_train))
print('Accuracy for test set for GaussianNB = {}'.format(accuracy_for_test))
```

Outline

- Introduction to Dataset and Problem
- Solving Problem by using KNN
- Solving Problem by Naïve Bayes Classifier
- Solving Problem by Decision Tree
- Solving Problem by Random Forest
- Solving Problem by AdaBoost
- Solving Problem by Gradient Boosting
- Solving Problem by XGBoost
- Solving Problem by SVM
- Solving Problem by Stacking
- Time Series Data
- Feature Important in DT

Evolution of Tree and Its Variant

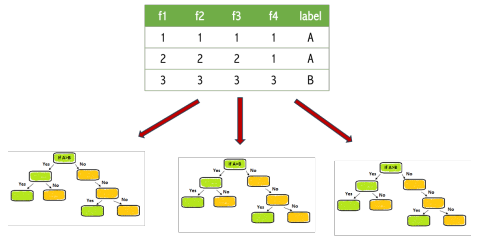
Decision Tree



$$Gini = 1 - \sum_j p_j^2 \qquad H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

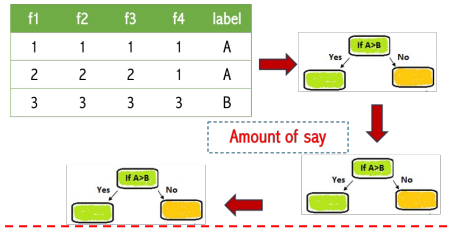
Tree is created by using GNI or Entropy metrics

Random Forest



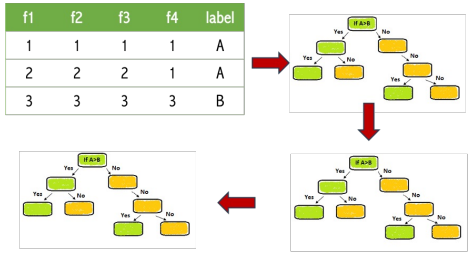
Tree are created independently
Same contribution of each tree to the final prediction

Adaboost



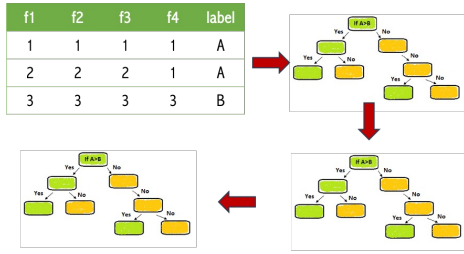
Stumps are created sequentially.
Different contribution of each stump to the final prediction

Gradient Boosting



Made up of Gradient descent and Boosting.
Minimize the cost function of the ensemble

XGBoost



Regularization
Approximate Greed Algorithm
Parallel Learning
Weighted Quantile Sketching
Sparsity-Aware Finding
Cache-Aware Access

Decision Tree for Classification

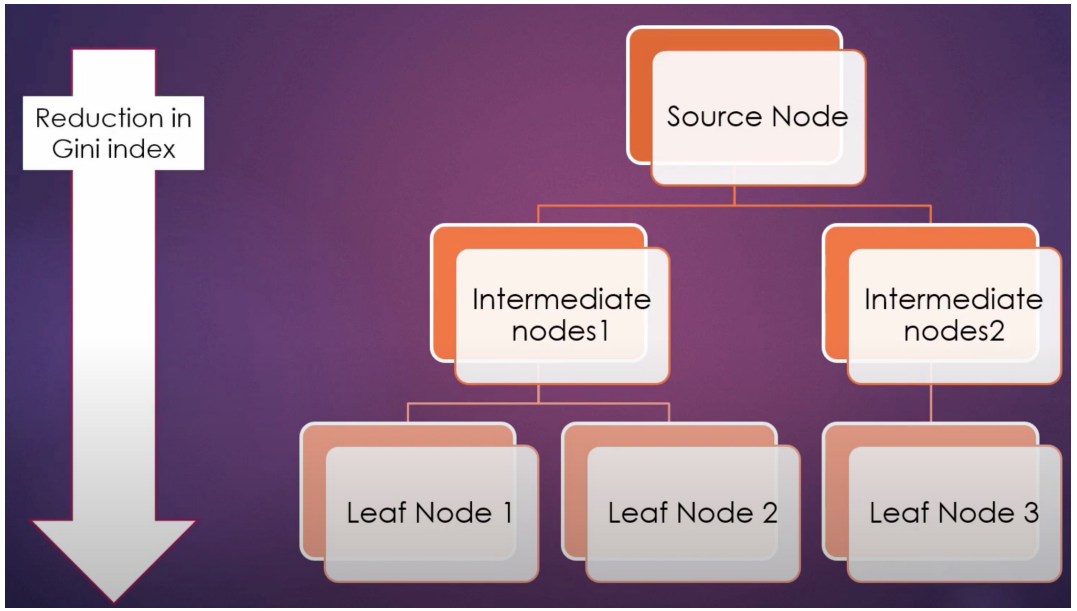
No.	Love Math	Love Art	Age	Love AI
1	Yes	Yes	7	No
2	Yes	No	12	No
3	No	Yes	18	Yes
4	No	Yes	35	Yes
5	Yes	Yes	38	Yes
6	Yes	No	50	No
7	No	No	83	No

Love Math

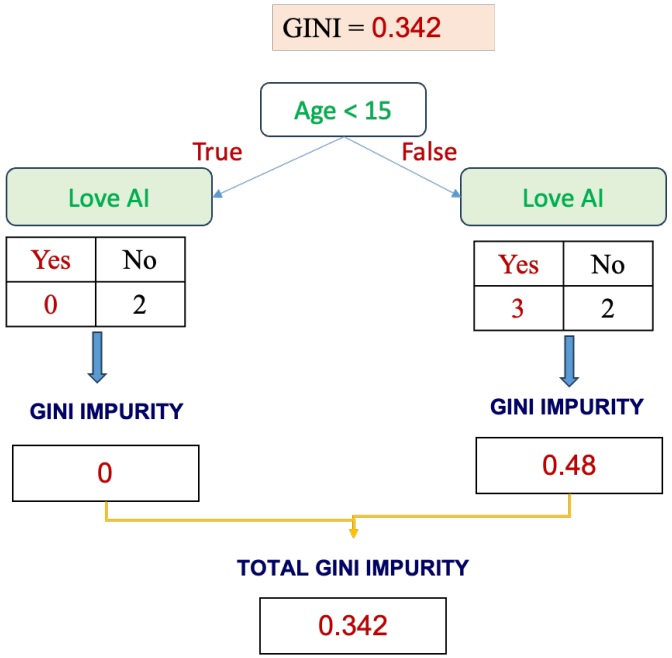
Love Art

Age

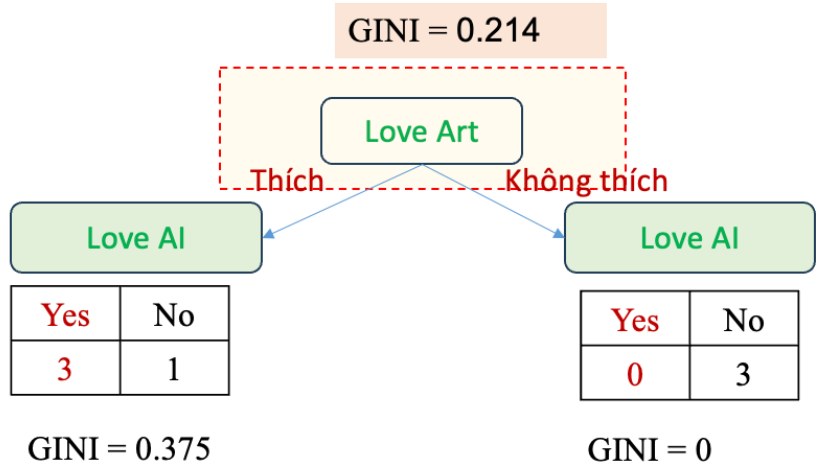
Which one is the root node?

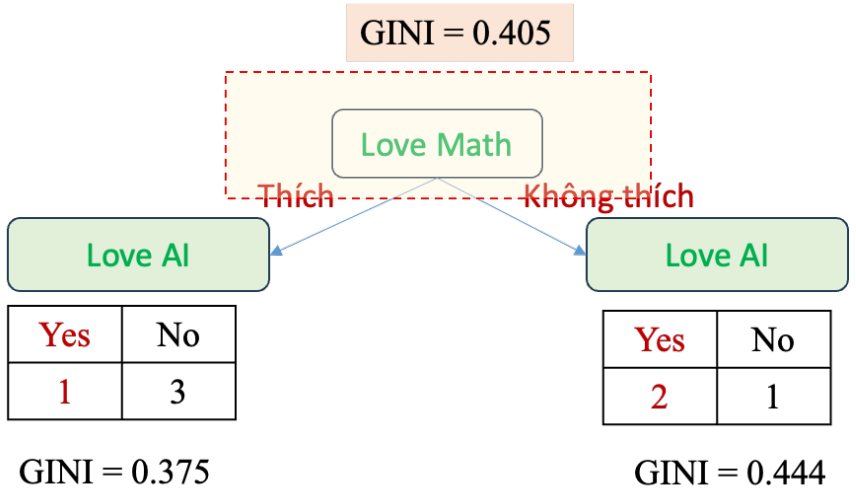


GINI Example



Which attribute is in the first node?

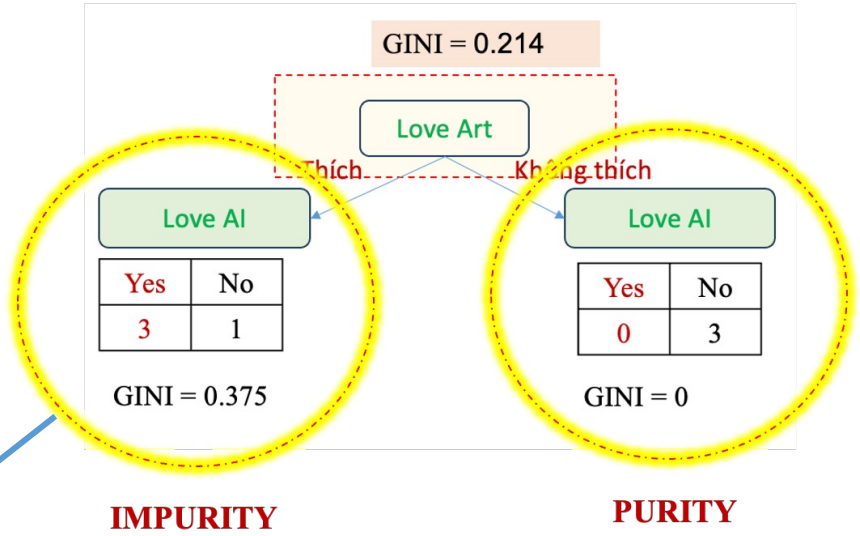




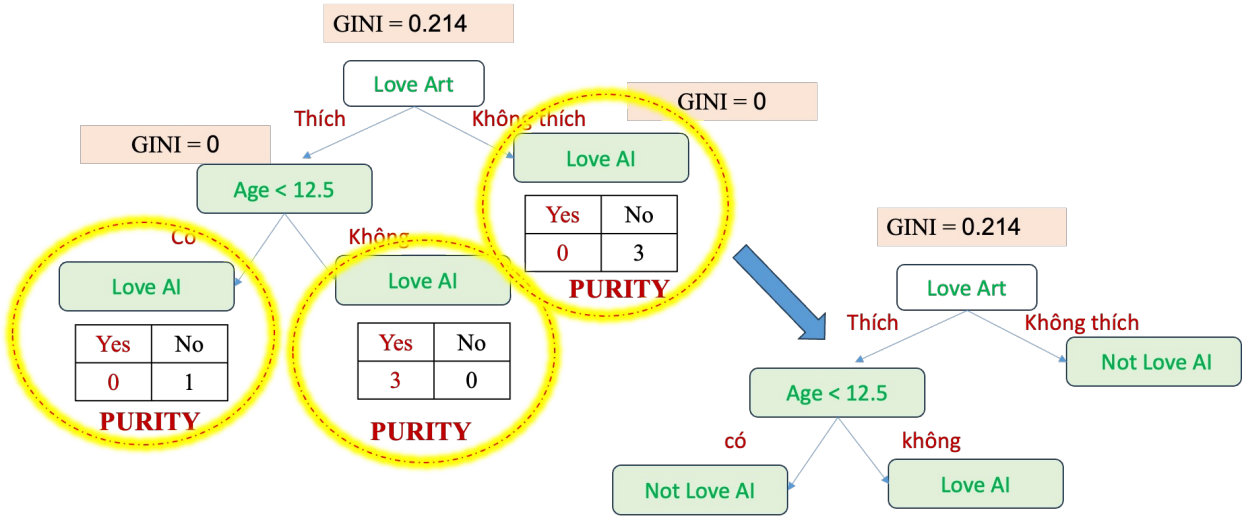
Love Art is the best one

GINI Solution

No.	Love Math	Love Art	Age	Love AI
1	Yes	Yes	7	No
2	Yes	No	12	No
3	No	Yes	18	Yes
4	No	Yes	35	Yes
5	Yes	Yes	38	Yes
6	Yes	No	50	No
7	No	No	83	No



No.	Love Math	Love Art	Age	Love AI
1	Yes	Yes	7	No
3	No	Yes	18	Yes
4	No	Yes	35	Yes
5	Yes	Yes	38	Yes



Decision Tree

a) accuracy for train = 0.76 and accuracy for test = 0.69
b) accuracy for train = 0.66 and accuracy for test = 0.67
c) accuracy for train = 0.85 and accuracy for test = 0.84
d) accuracy for train = 1.0 and accuracy for test = 0.75

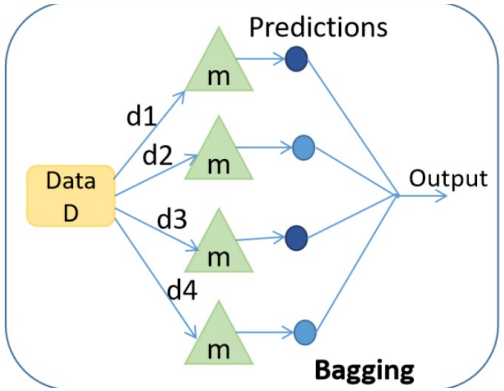
```
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)

from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion='gini', max_depth=10, min_samples_split=2, random_state=42)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix
cm_test = confusion_matrix(y_pred, y_test)

y_pred_train = classifier.predict(X_train)
cm_train = confusion_matrix(y_pred_train, y_train)

print()
accuracy_for_train = np.round((cm_train[0][0] + cm_train[1][1])/len(y_train),2)
accuracy_for_test = np.round((cm_test[0][0] + cm_test[1][1])/len(y_test),2)
print('Accuracy for training set for DecisionTreeClassifier = {}'.format(accuracy_for_train))
print('Accuracy for test set for DecisionTreeClassifier = {}'.format(accuracy_for_test))
```

Ensemple Learning Techniques

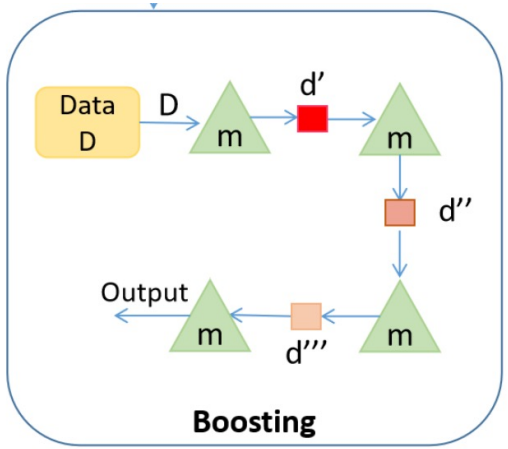
Ensemple Learning

Thông dụng ở các cuộc thi về AI

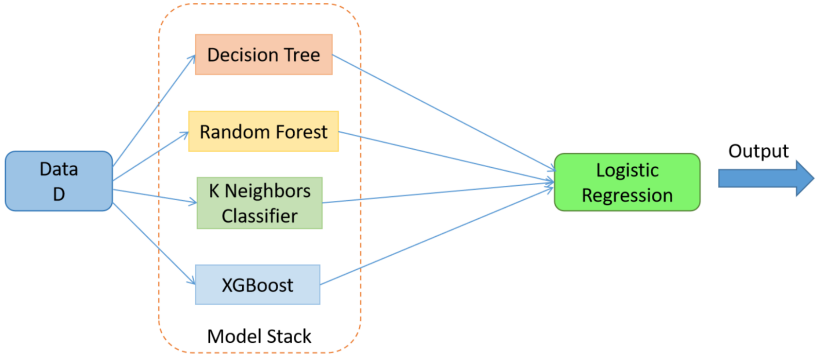
Bagging
homogeneous weak learners



Boosting
homogeneous weak learners



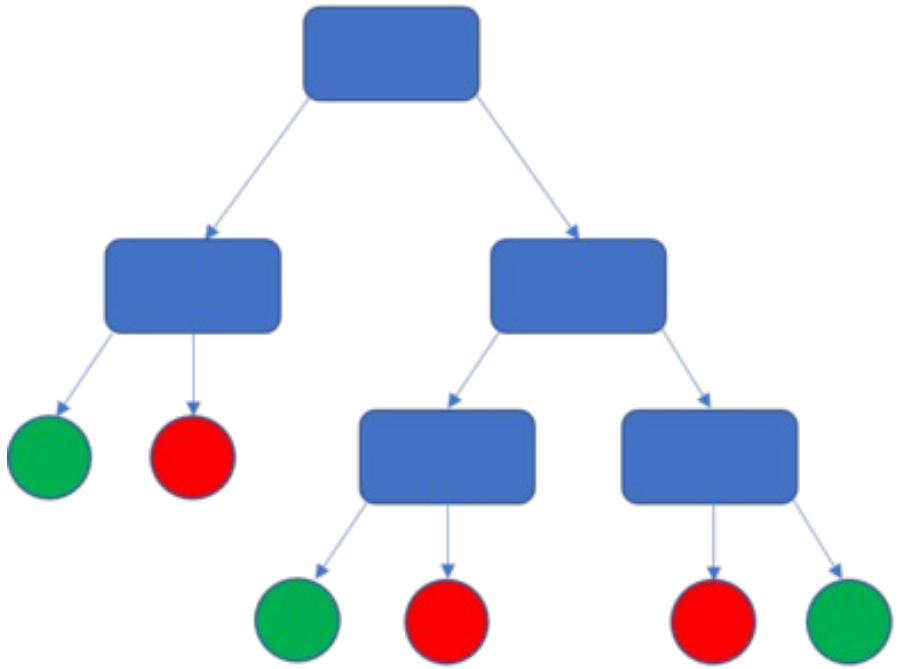
Stacking
Heterogeneous weak learners



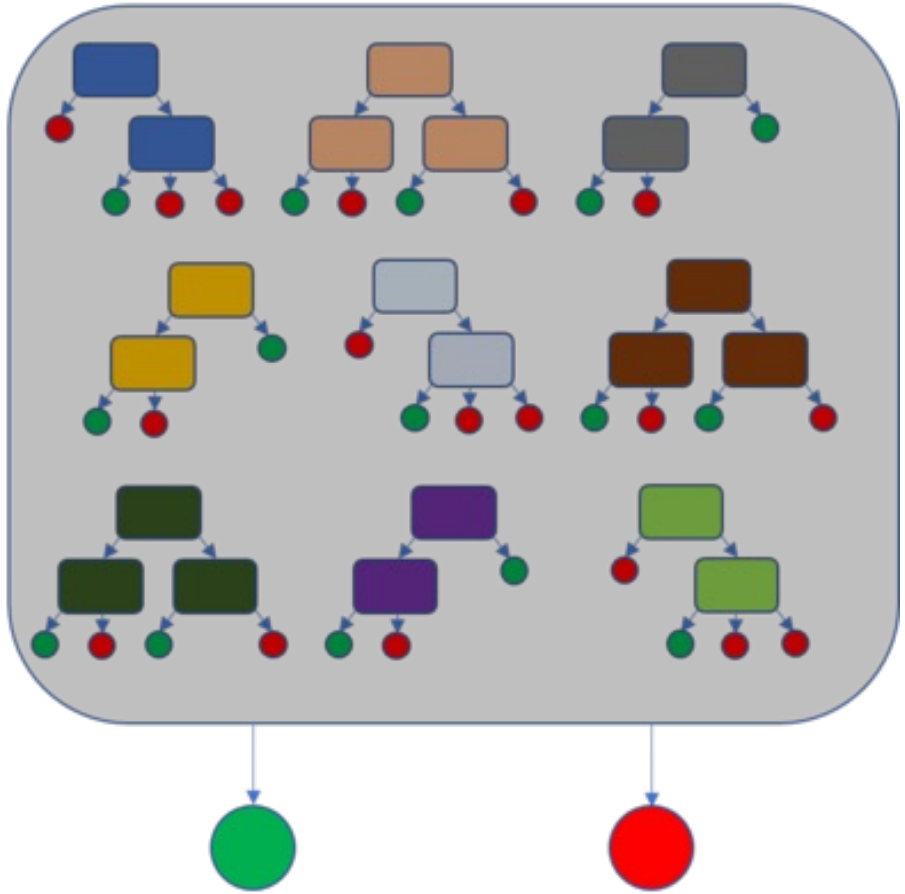
Outline

- Introduction to Dataset and Problem
- Solving Problem by using KNN
- Solving Problem by Naïve Bayes Classifier
- Solving Problem by Decision Tree
- Solving Problem by Random Forest
- Solving Problem by AdaBoost
- Solving Problem by Gradient Boosting
- Solving Problem by XGBoost
- Solving Problem by SVM
- Solving Problem by Stacking
- Time Series Data
- Feature Important in DT

Random Forest



Decision Tree

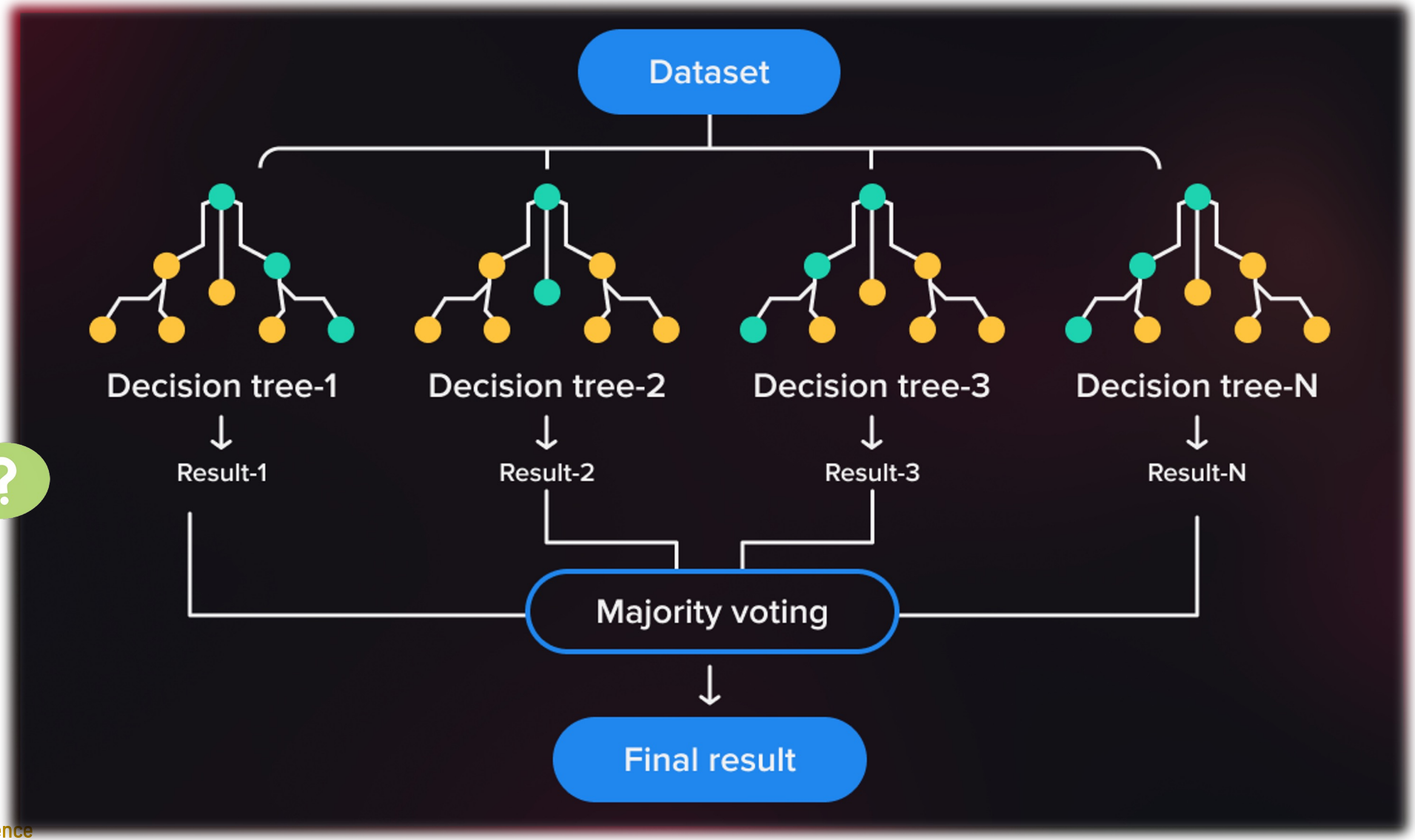


Random Forest

Bagging-based Method

Random Forest

Last Week

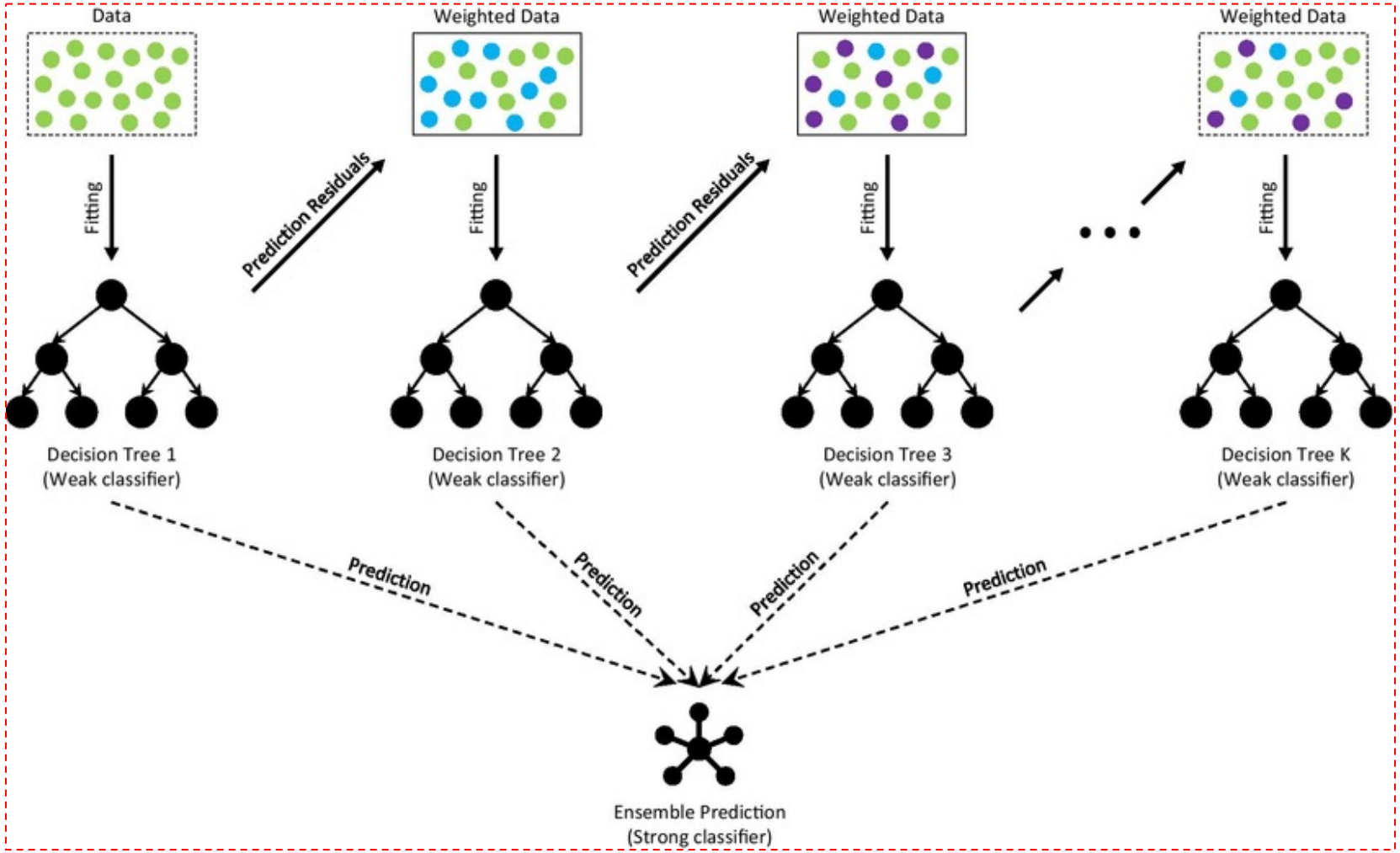


Random Forest

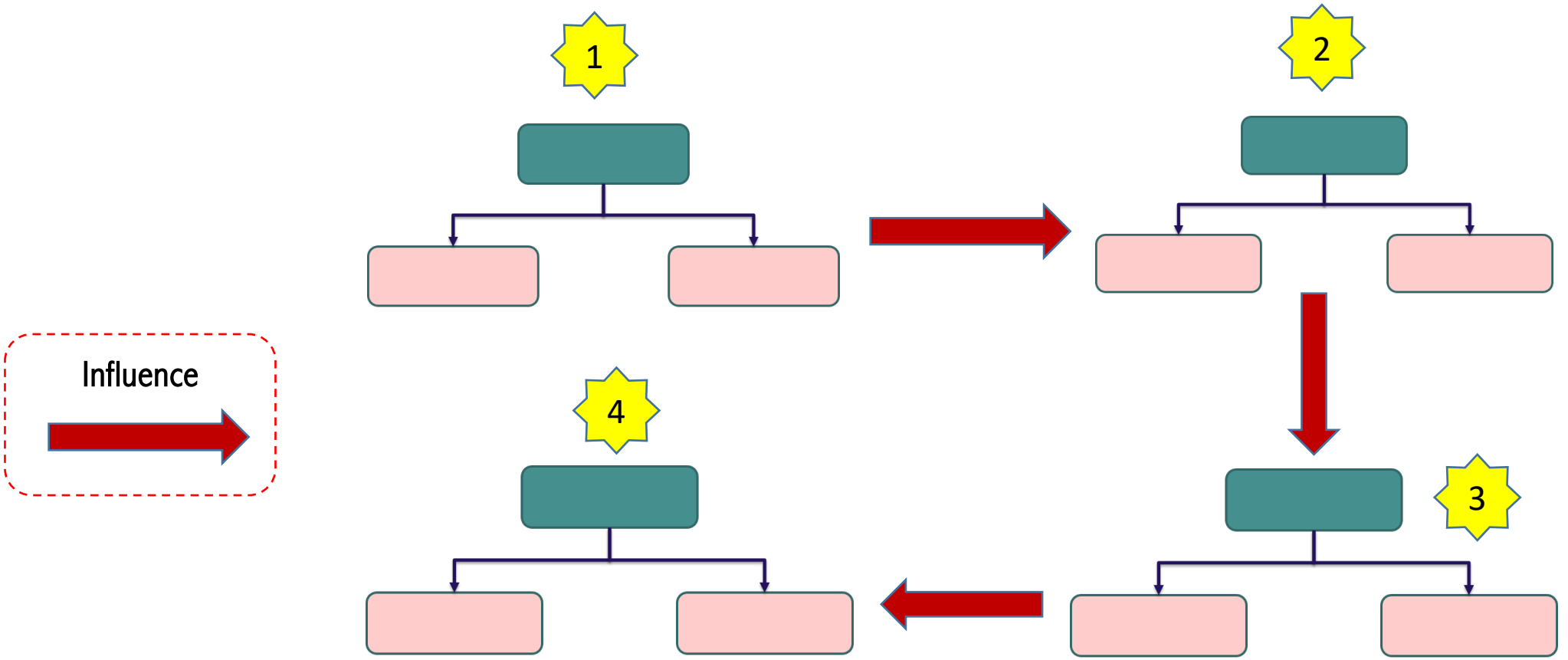
a) accuracy for train = 0.98 and acccuracy for test = 0.8
b) accuracy for train = 0.66 and acccuracy for test = 0.67
c) accuracy for train = 0.85 and acccuracy for test = 0.84
d) accuracy for train = 1.0 and acccuracy for test = 0.75

```
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)

from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(criterion='gini', max_depth=10, min_samples_split=2, n_estimators = 10, random_state=42)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix
cm_test = confusion_matrix(y_pred, y_test)

y_pred_train = classifier.predict(X_train)
cm_train = confusion_matrix(y_pred_train, y_train)

print()
accuracy_for_train = np.round((cm_train[0][0] + cm_train[1][1])/len(y_train),2)
accuracy_for_test = np.round((cm_test[0][0] + cm_test[1][1])/len(y_test),2)
print('Accuracy for training set for RandomForestClassifier = {}'.format(accuracy_for_train))
print('Accuracy for test set for RandomForestClassifier = {}'.format(accuracy_for_test))
```

Boosting-Based Method



AdaBoost: FOREST OF STUMP



Adaboost builds a stump based on the the error made by previous stumps

Outline

- Introduction to Dataset and Problem
- Solving Problem by using KNN
- Solving Problem by Naïve Bayes Classifier
- Solving Problem by Decision Tree
- Solving Problem by Random Forest
- Solving Problem by AdaBoost
- Solving Problem by Gradient Boosting
- Solving Problem by XGBoost
- Solving Problem by SVM
- Solving Problem by Stacking
- Time Series Data
- Feature Important in DT

AdaBoost: FOREST OF STUMP

Algorithm 1 *AdaBoost (Freund & Schapire 1997)*

1. *Initialize the observation weights* $w_i = 1/n,\ i = 1, 2, \ldots, n.$
2. *For* $m = 1$ *to* M*:*
   (a) *Fit a classifier* $T^{(m)}(\boldsymbol{x})$ *to the training data using weights* w_i.
   (b) *Compute*
   $$err^{(m)} = \sum_{i=1}^{n} w_i \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right) / \sum_{i=1}^{n} w_i.$$
   (c) *Compute*
   $$\alpha^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}}.$$
   (d) *Set*
   $$w_i \leftarrow w_i \cdot \exp\left(\alpha^{(m)} \cdot \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right)\right),\ i = 1, 2, \ldots, n.$$
   (e) *Re-normalize* w_i.
3. *Output*
   $$C(\boldsymbol{x}) = \arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \cdot \mathbb{I}(T^{(m)}(\boldsymbol{x}) = k).$$

AdaBoost: FOREST OF STUMP

a) accuracy for train = 0.98 and accuracy for test = 0.8
b) accuracy for train = 0.91 and accuracy for test = 0.84
c) accuracy for train = 0.85 and accuracy for test = 0.84
d) accuracy for train = 1.0 and accuracy for test = 0.75

```
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)

from sklearn.ensemble import AdaBoostClassifier
# define the model
classifier = AdaBoostClassifier(n_estimators=50, learning_rate=1.0, random_state=42)

classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix
cm_test = confusion_matrix(y_pred, y_test)

y_pred_train = classifier.predict(X_train)
cm_train = confusion_matrix(y_pred_train, y_train)

print()
accuracy_for_train = np.round((cm_train[0][0] + cm_train[1][1])/len(y_train),2)
accuracy_for_test = np.round((cm_test[0][0] + cm_test[1][1])/len(y_test),2)
print('Accuracy for training set for AdaBoostClassifier = {}'.format(accuracy_for_train))
print('Accuracy for test set for AdaBoostClassifier = {}'.format(accuracy_for_test))
```

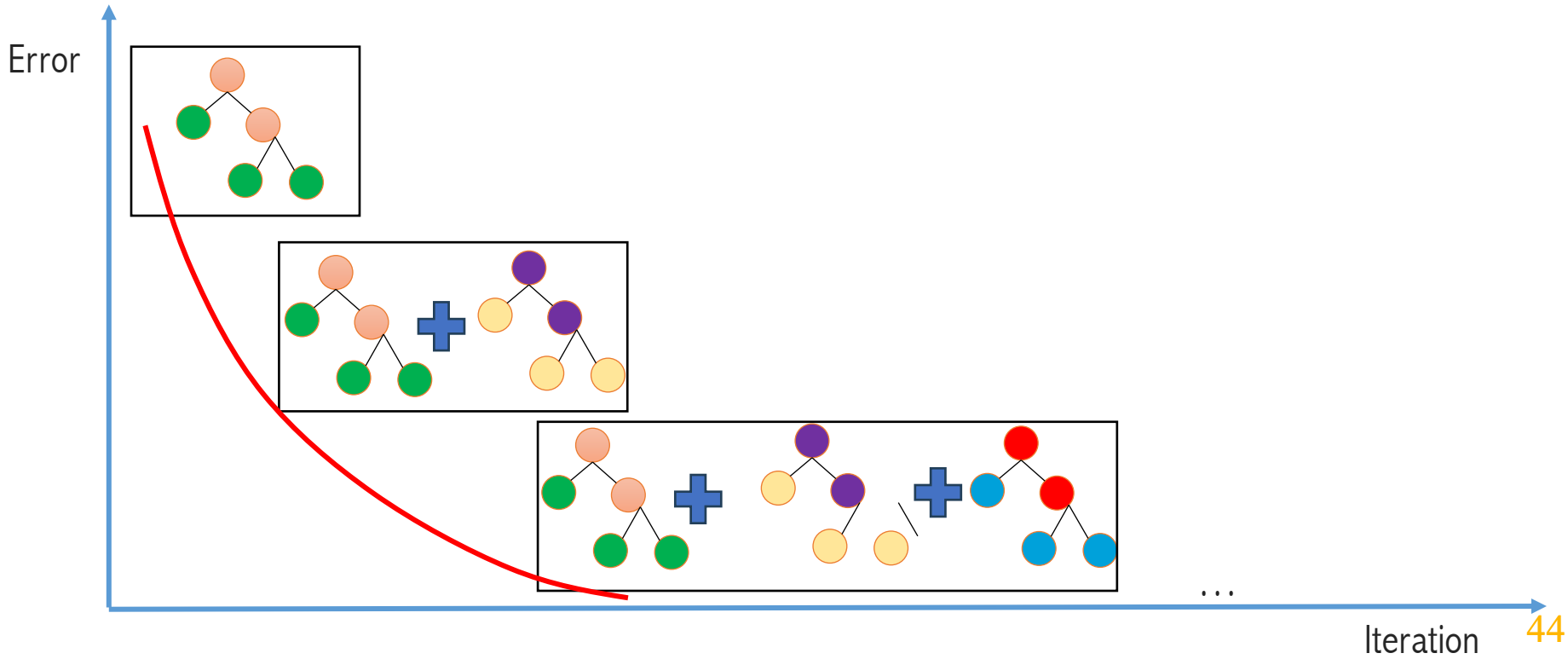
Outline

- Introduction to Dataset and Problem
- Solving Problem by using KNN
- Solving Problem by Naïve Bayes Classifier
- Solving Problem by Decision Tree
- Solving Problem by Random Forest
- Solving Problem by AdaBoost
- Solving Problem by Gradient Boosting
- Solving Problem by XGBoost
- Solving Problem by SVM
- Solving Problem by Stacking
- Time Series Data
- Feature Important in DT

Gradient Boosting

Gradient + Boosting

Applies the concepts of **logistic regression**. It uses **log-odds** to make a prediction, converts log-odds to **probabilities** through logistic function, then make a classification based on self-defined **threshold**.



Gradient Boosting

a) accuracy for train = 0.98 and accuracy for test = 0.8
b) accuracy for train = 0.91 and accuracy for test = 0.84
c) accuracy for train = 1.0 and accuracy for test = 0.85
d) accuracy for train = 1.0 and accuracy for test = 0.75

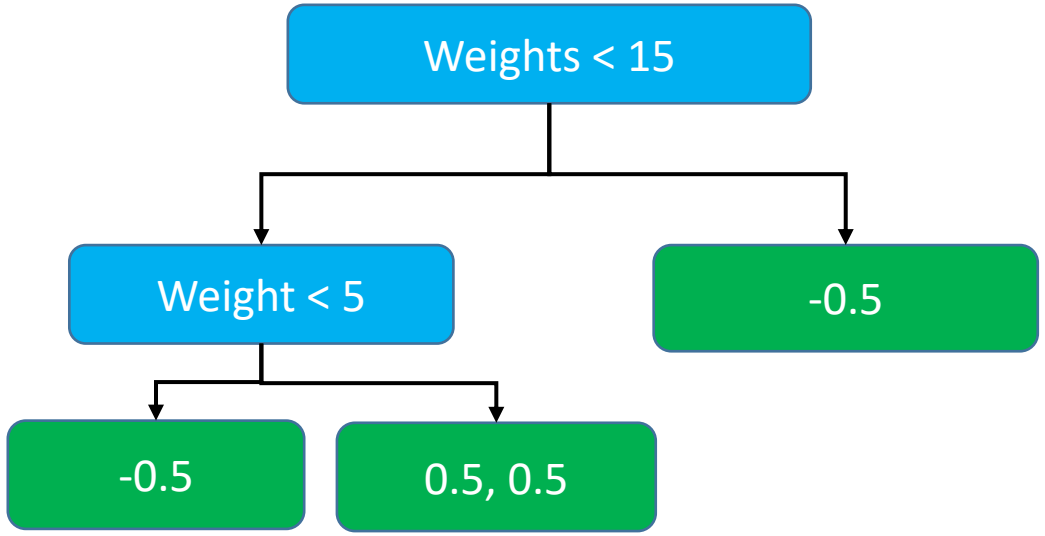
```
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)

from sklearn.ensemble import GradientBoostingClassifier
# define the model


classifier = GradientBoostingClassifier(learning_rate=0.1, n_estimators=100, subsample=1.0, min_samples_split=2, max_depth=3, random_state=42)

classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix
cm_test = confusion_matrix(y_pred, y_test)

y_pred_train = classifier.predict(X_train)
cm_train = confusion_matrix(y_pred_train, y_train)

print()
accuracy_for_train = np.round((cm_train[0][0] + cm_train[1][1])/len(y_train),2)
accuracy_for_test = np.round((cm_test[0][0] + cm_test[1][1])/len(y_test),2)
print('Accuracy for training set for GradientBoostingClassifier = {}'.format(accuracy_for_train))
print('Accuracy for test set for GradientBoostingClassifier = {}'.format(accuracy_for_test))
```

Outline

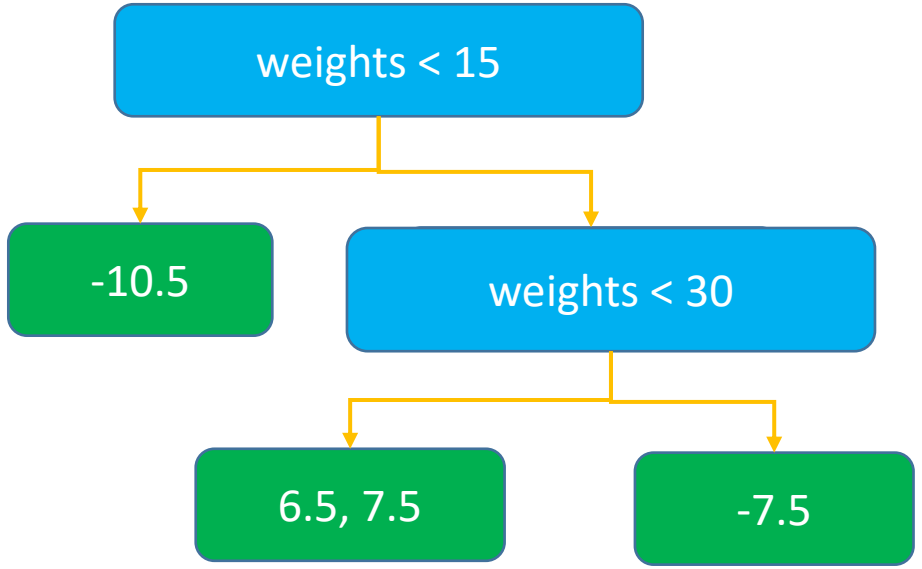
- Introduction to Dataset and Problem
- Solving Problem by using KNN
- Solving Problem by Naïve Bayes Classifier
- Solving Problem by Decision Tree
- Solving Problem by Random Forest
- Solving Problem by AdaBoost
- Solving Problem by Gradient Boosting
- Solving Problem by XGBoost
- Solving Problem by SVM
- Solving Problem by Stacking
- Time Series Data
- Feature Important in DT

XGBoost

Classification



Regression



$$Similarity\ Score\ = \frac{(\sum \text{Residual})^2}{\sum \bar{y}_i \times (1-\bar{y}_i) + \lambda}$$

$$Ouput\ value\ = \frac{(\sum \text{Residual})}{\sum \bar{y}_i \times (1-\bar{y}_i) + \lambda}$$

$$Similarity\ Score\ = \frac{(\sum \text{Residual})^2}{\text{Number of Residual} + \lambda}$$

$$\text{Output Value} = \frac{(\sum \text{Residual})}{\text{Number of Residual} + \lambda}$$

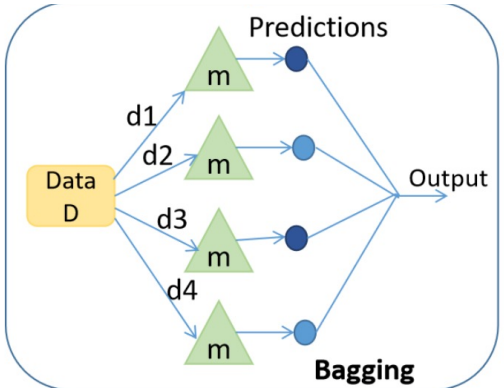
Ensemple Learning Techniques

Ensemple Learning

Thông dụng ở các cuộc thi về AI

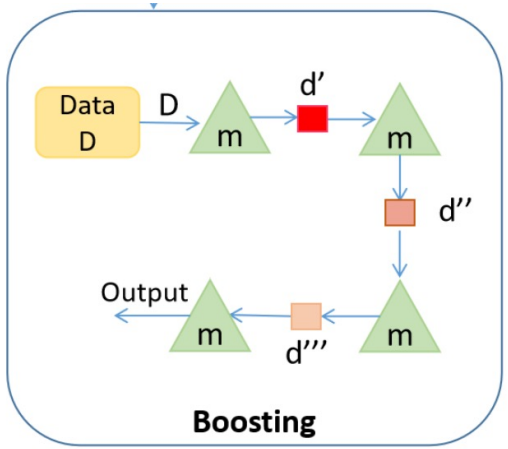
Bagging

homogeneous weak learners



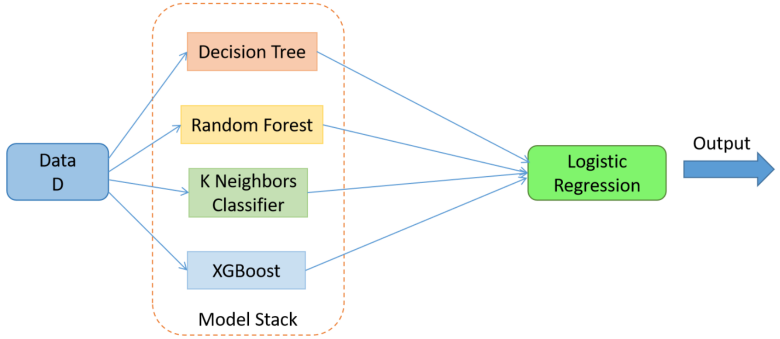
Boosting

homogeneous weak learners



Stacking

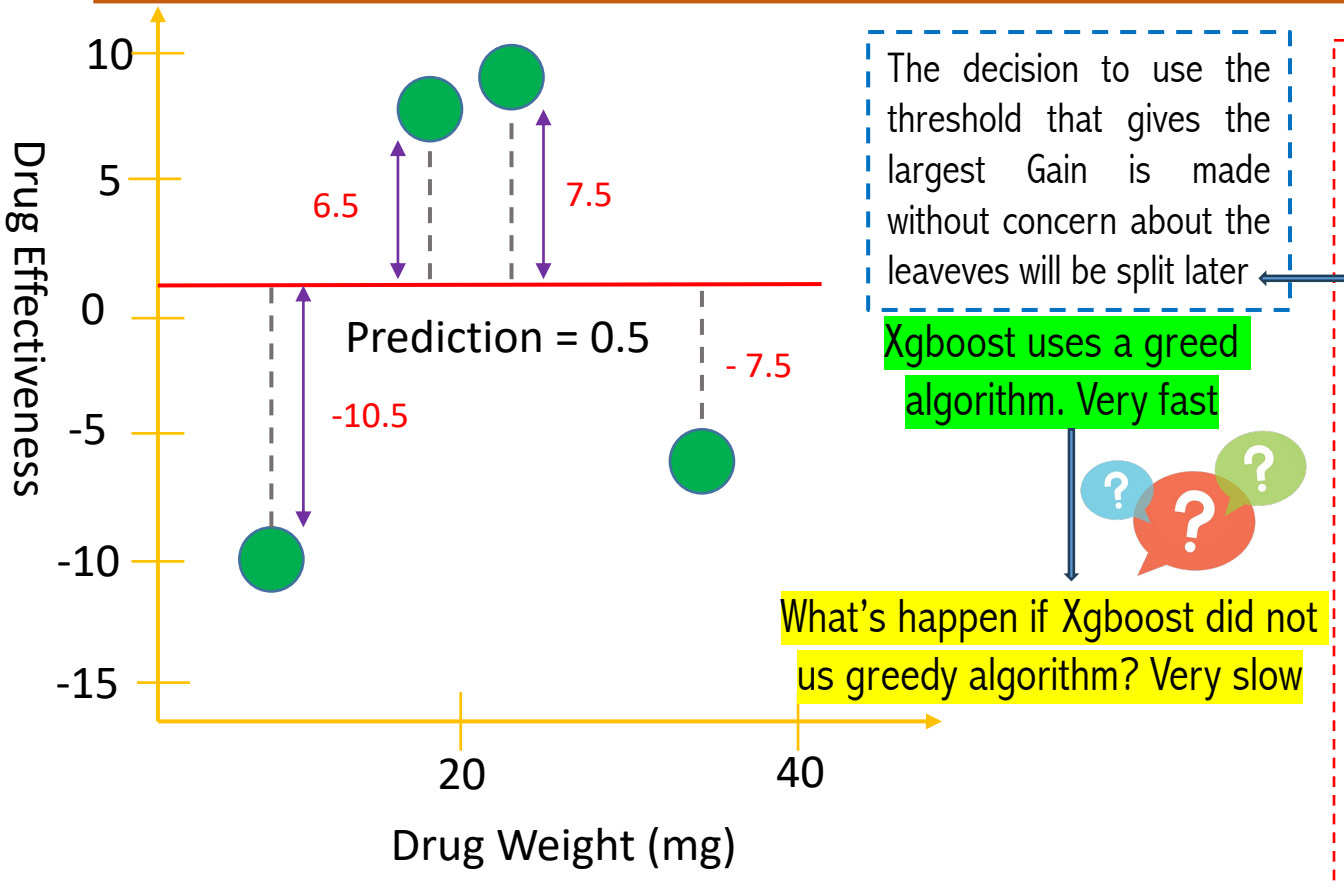
Heterogeneous weak learners



XGBoost Review

- Gradient Boost
- Regularization
- Approximate Greedy Algorithm
- Parallel Learning
- Weighted Quantile Sketch
- Sparsity-Aware Split Finding
- Cache-Aware Access

Greedy Algorithm



The decision to use the threshold that gives the largest Gain is made without concern about the leaveves will be split later

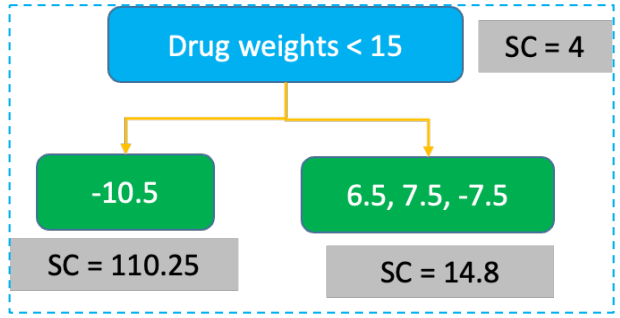
Xgboost uses a greed algorithm. Very fast

What's happen if Xgboost did not us greedy algorithm? Very slow

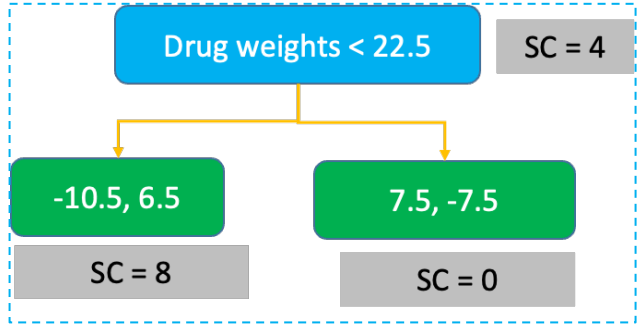
A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment. It doesn't worry whether the current best result will bring the overall optimal result.

Maximize

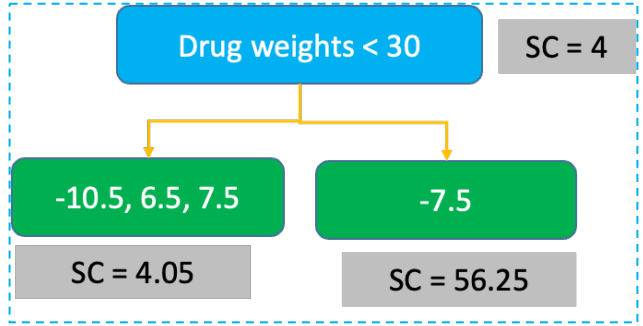
Compute Information Gain



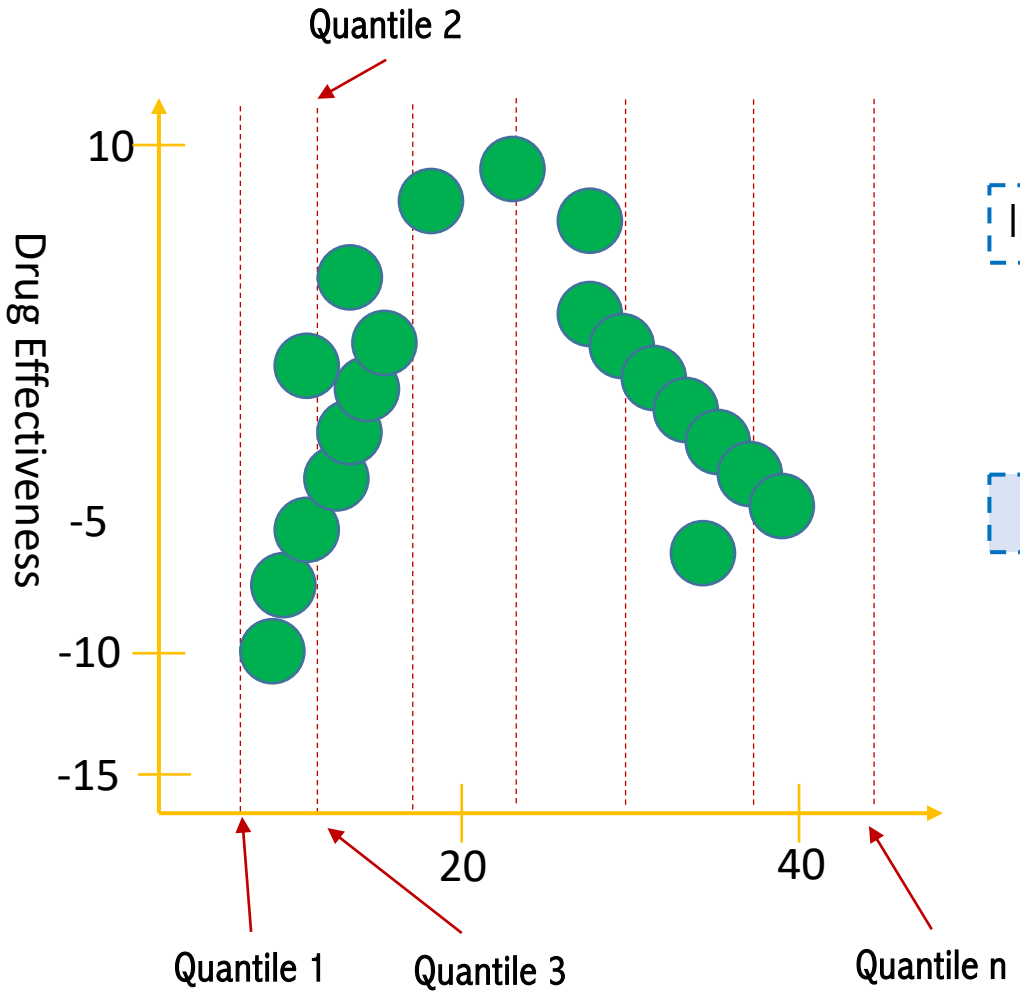
Compute Information Gain



Compute Information Gain



Approximate Greedy Algorithm



Instead of testing every single threshold, we could divide the data into Quantiles

By default, the Approximate Greed algorithm uses ~ 33 quantiles

Parallel Learning

Weighted Quantile Sketch

f1	f2	f3	…	label
1.1	3.3	3.6	4.5	A
3.4	4.2	2.5	4.7	B
4.5	6.7	8.7	9.7	A
7.6	8.4	9.2	4.9	A
…	…	…	…	…

Parallel Learning & Quantile Sketch Algorithm

f1	f2	label
1.1	3.3	A
3.4	4.2	B
4.5	6.7	A
7.6	8.4	A
…	…	…

XGBoost used weighted approximate quantiles

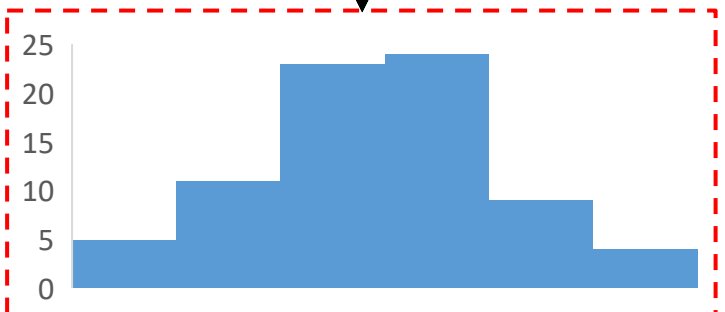
f1	f2	label
1.1	3.3	A
3.4	4.2	B

f1	f2	label
4.5	6.7	A
7.6	8.4	A

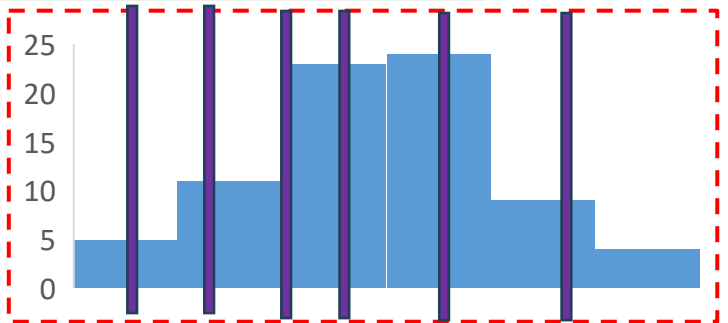
f1	f2	label
5.5	7.7	A
6.6	8.8	A

f1	f2	label
…	…	…
…	…	…

Build histogram



Approximate Quantiles



Sparsity-Aware Split Finding

Sparse-Aware Split Finding

Initial Prediction for Drug Effectiveness

0.5

Dosage	Drug Effectiveness
10	-7
???	-3
21	7
25	8
5	-5
???	-2

Dosage	Drug Effectiveness	Residual
10	-7	-7.5
???	-3	-3.5
21	7	6.5
25	8	7.5
5	-5	-5.5
???	-2	-2.5

Sparsity-Aware Split Finding

Table without missing values

Dosage	Drug Effectiveness	Residual
5	-5	-5.5
10	-7	-7.5
21	7	6.5
25	8	7.5

Build tree

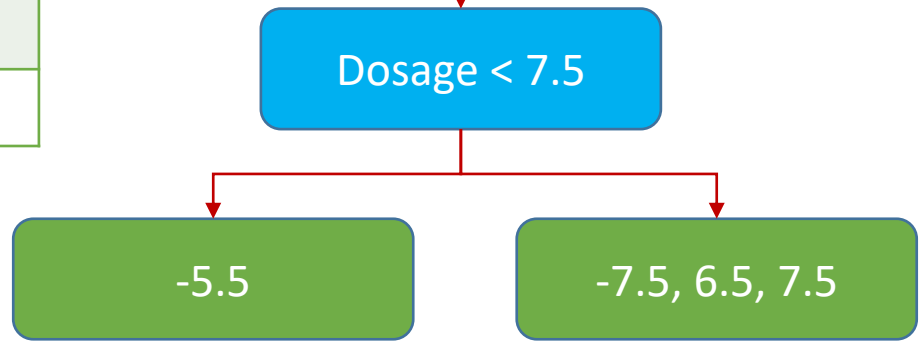


Table with missing values

Dosage	Drug Effectiveness	Residual
???	-3	-3.5
???	-2	-2.5

Sparsity-Aware Split Finding

Table without missing values

Dosage	Drug Effectiveness	Residual
5	-5	-5.5
10	-7	-7.5
21	7	6.5
25	8	7.5

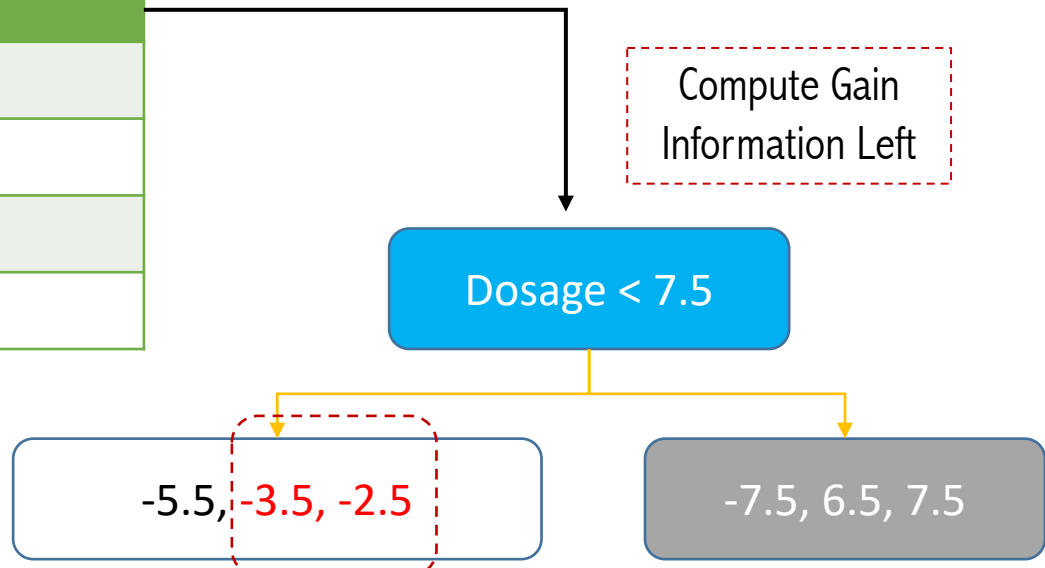
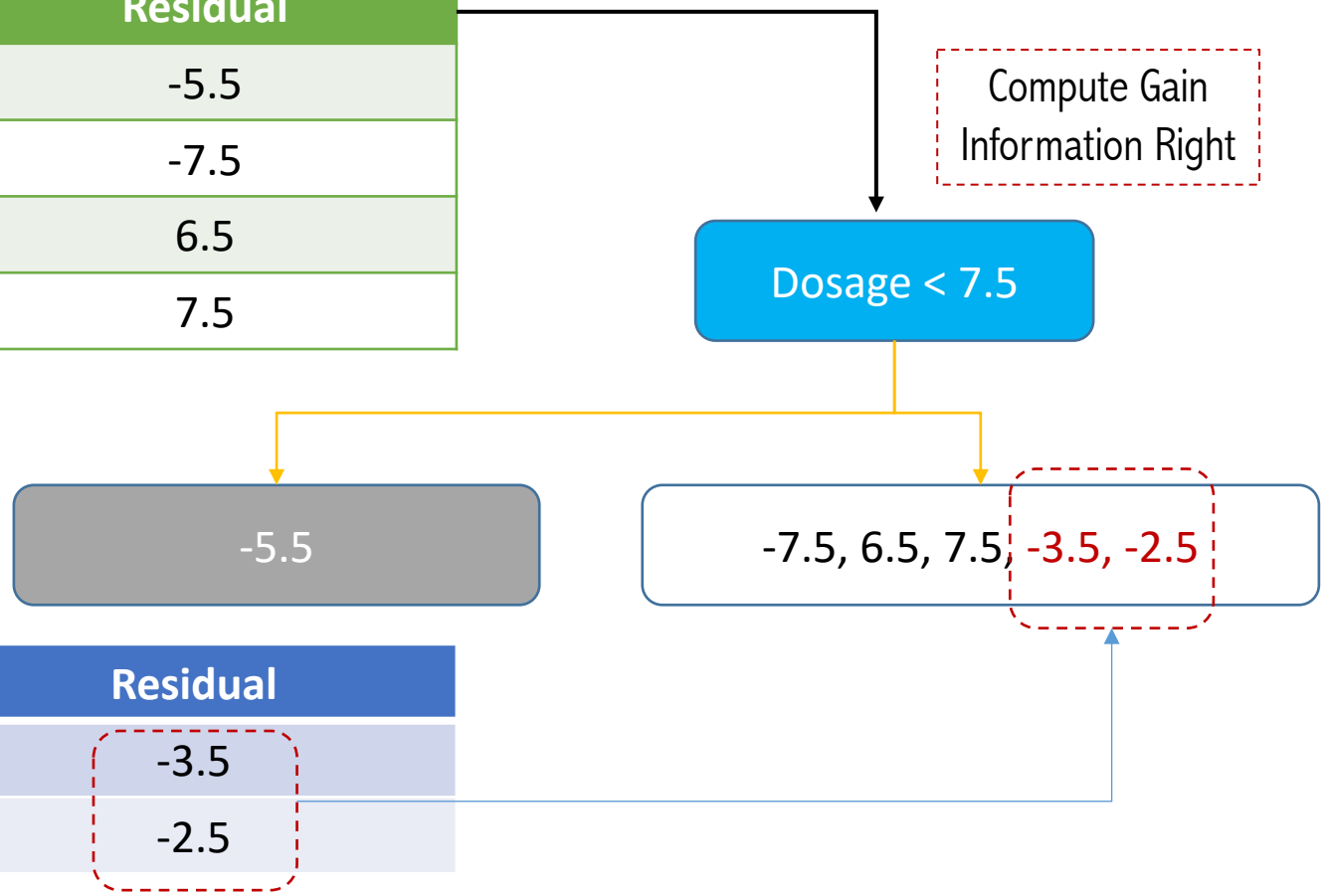


Table with missing values

Dosage	Drug Effectiveness	Residual
???	-3	-3.5
???	-2	-2.5

Sparsity-Aware Split Finding

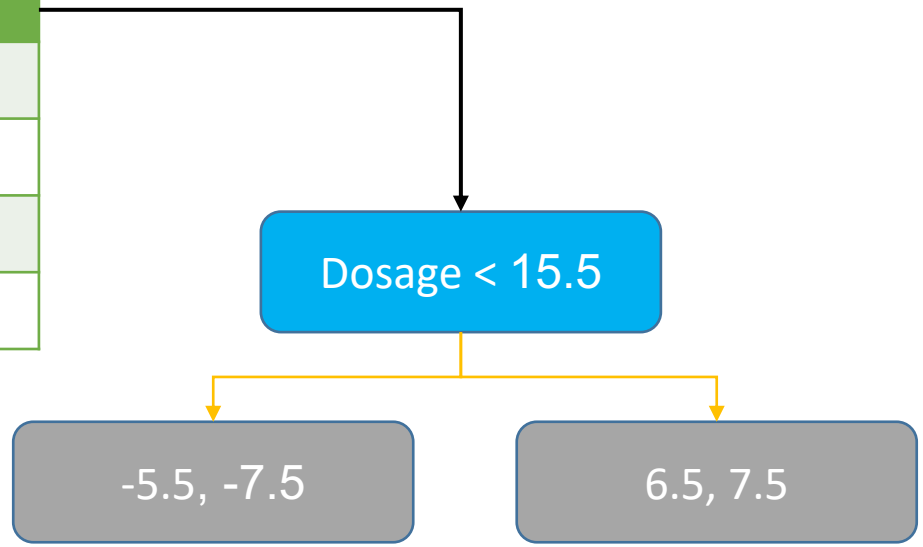
Dosage	Drug Effectiveness	Residual
5	-5	-5.5
10	-7	-7.5
21	7	6.5
25	8	7.5



Dosage	Drug Effectiveness	Residual
???	-3	-3.5
???	-2	-2.5

Sparsity-Aware Split Finding

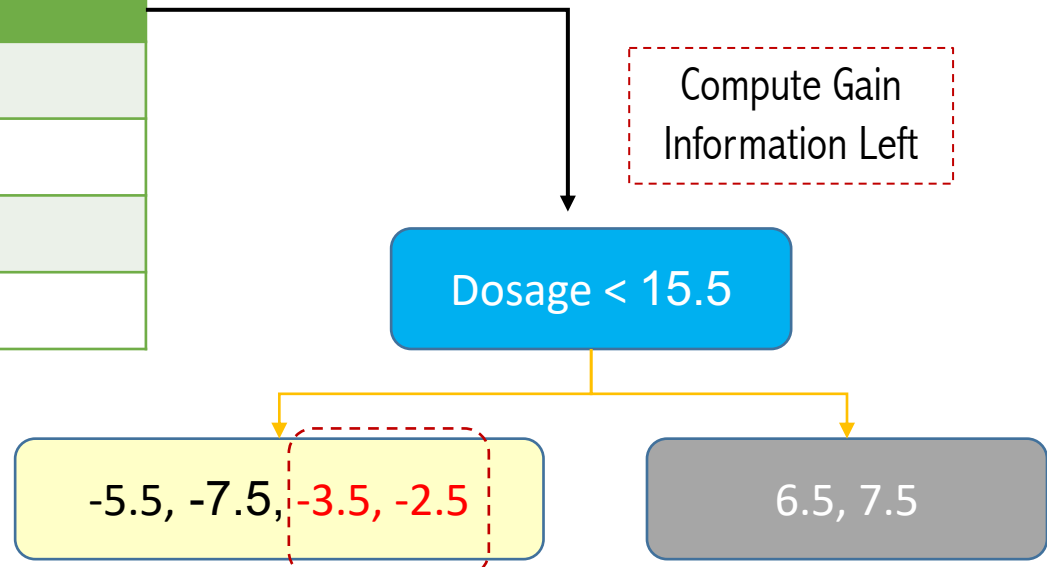
Dosage	Drug Effectiveness	Residual
5	-5	-5.5
10	-7	-7.5
21	7	6.5
25	8	7.5



Dosage	Drug Effectiveness	Residual
???	-3	-3.5
???	-2	-2.5

Sparsity-Aware Split Finding

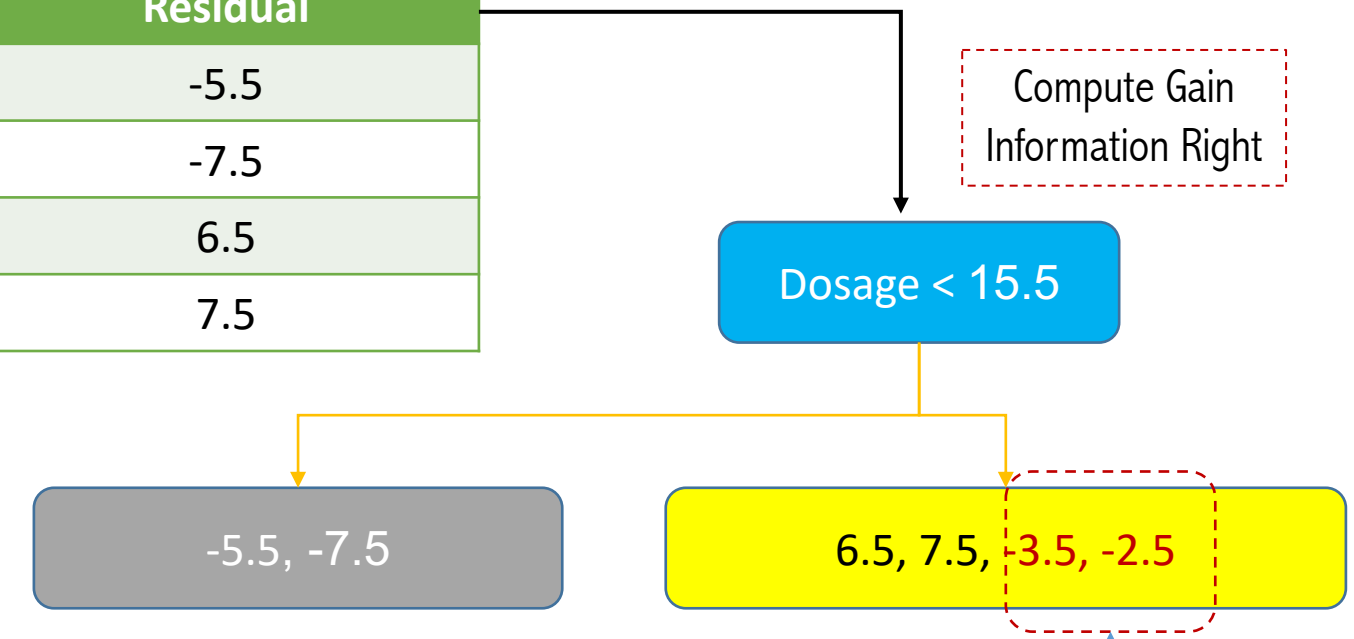
Dosage	Drug Effectiveness	Residual
5	-5	-5.5
10	-7	-7.5
21	7	6.5
25	8	7.5



Dosage	Drug Effectiveness	Residual
???	-3	-3.5
???	-2	-2.5

Sparsity-Aware Split Finding

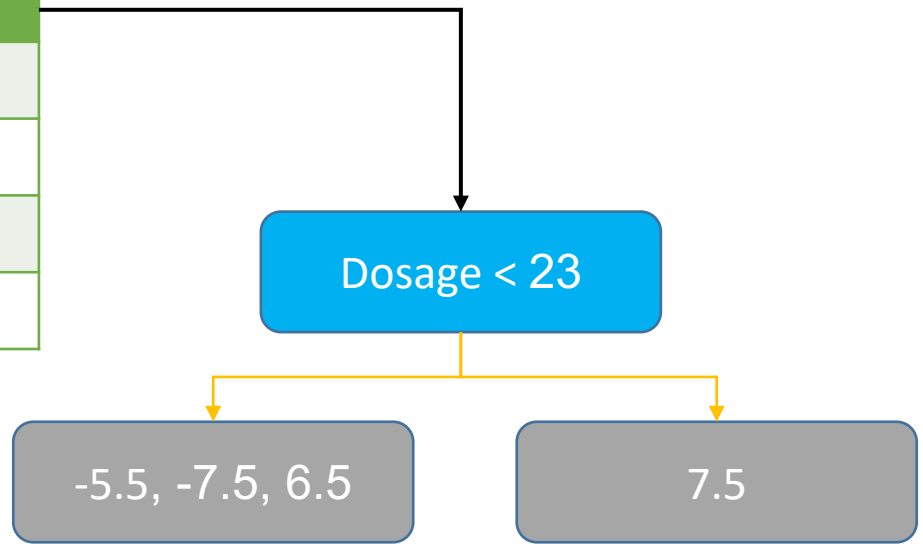
Dosage	Drug Effectiveness	Residual
5	-5	-5.5
10	-7	-7.5
21	7	6.5
25	8	7.5



Dosage	Drug Effectiveness	Residual
???	-3	-3.5
???	-2	-2.5

Sparsity-Aware Split Finding

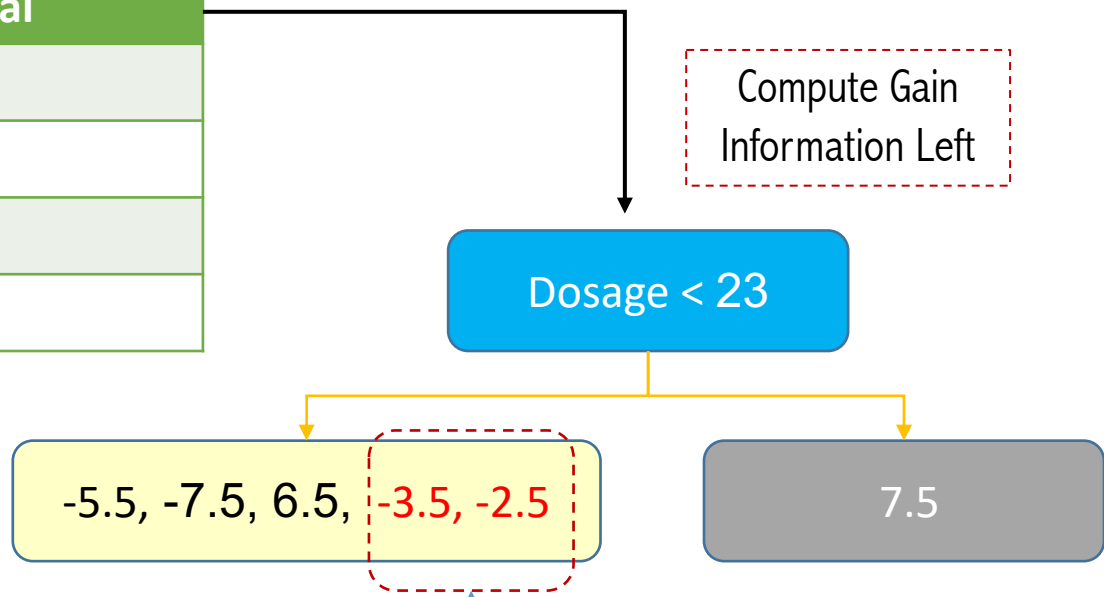
Dosage	Drug Effectiveness	Residual
5	-5	-5.5
10	-7	-7.5
21	7	6.5
25	8	7.5



Dosage	Drug Effectiveness	Residual
???	-3	-3.5
???	-2	-2.5

Sparsity-Aware Split Finding

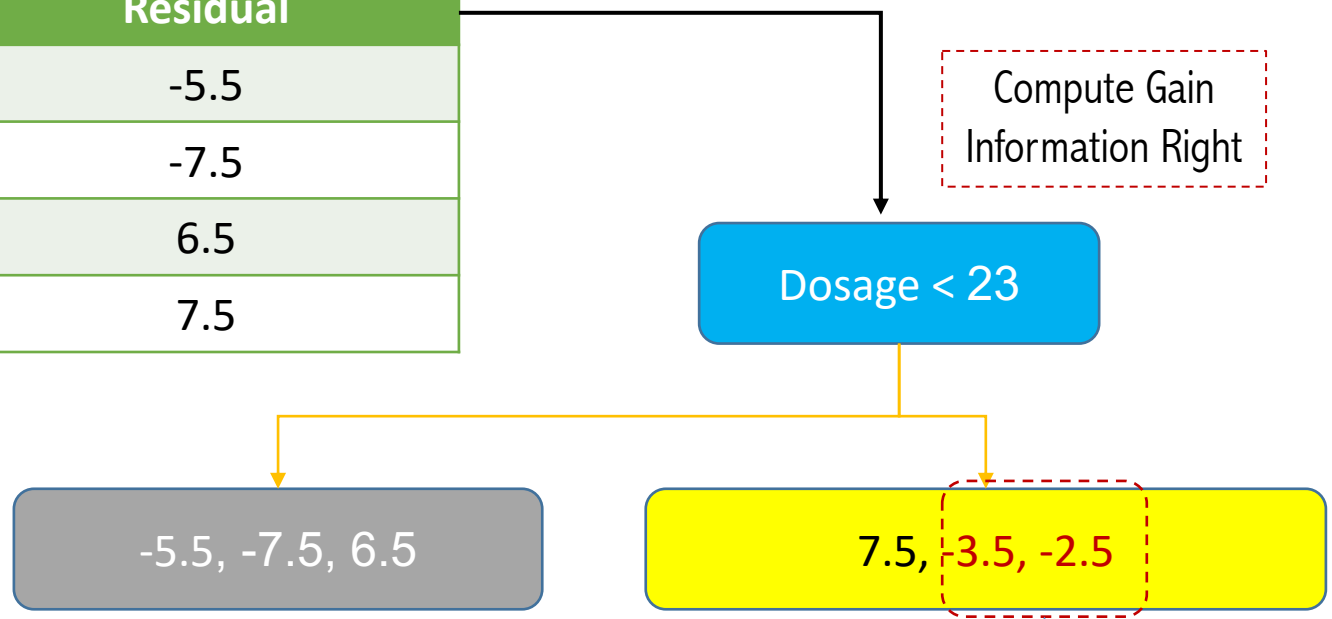
Dosage	Drug Effectiveness	Residual
5	-5	-5.5
10	-7	-7.5
21	7	6.5
25	8	7.5



Dosage	Drug Effectiveness	Residual
???	-3	-3.5
???	-2	-2.5

Sparsity-Aware Split Finding

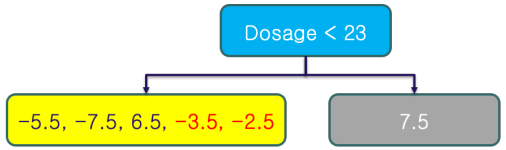
Dosage	Drug Effectiveness	Residual
5	-5	-5.5
10	-7	-7.5
21	7	6.5
25	8	7.5



Dosage	Drug Effectiveness	Residual
???	-3	-3.5
???	-2	-2.5

Dosage	Drug Effectiveness	Residual
5	−5	−5.5
10	−7	−7.5
21	7	6.5
25	8	7.5

Compute Gain value

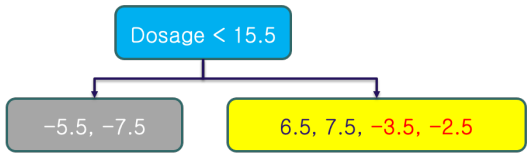


Dosage	Drug Effectiveness	Residual
5	−5	−5.5
10	−7	−7.5
21	7	6.5
25	8	7.5

Compute Gain value

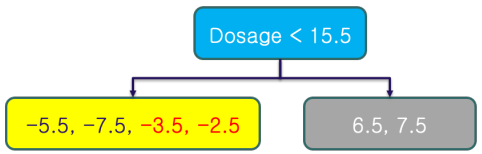
Dosage	Drug Effectiveness	Residual
5	−5	−5.5
10	−7	−7.5
21	7	6.5
25	8	7.5

Compute Gain value



Dosage	Drug Effectiveness	Residual
5	−5	−5.5
10	−7	−7.5
21	7	6.5
25	8	7.5

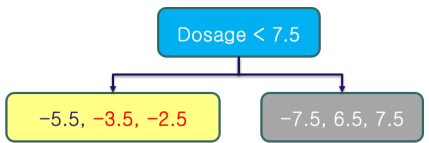
Compute Gain value



MAXIUM GAIN VALUE

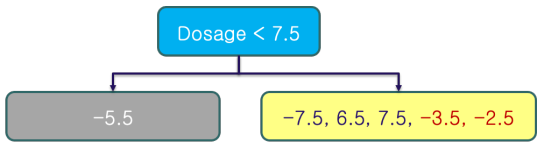
Dosage	Drug Effectiveness	Residual
10	−7	−7.5
21	7	6.5
25	8	7.5
5	−5	−5.5

Compute Gain value



Dosage	Drug Effectiveness	Residual
10	−7	−7.5
21	7	6.5
25	8	7.5
5	−5	−5.5

Compute Gain value

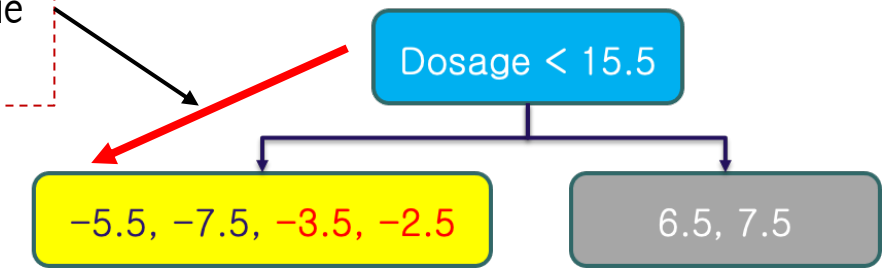


How to Handle Missing Value

Dosage	Drug Effectiveness	Residual
5	−5	−5.5
10	−7	−7.5
21	7	6.5
25	8	7.5

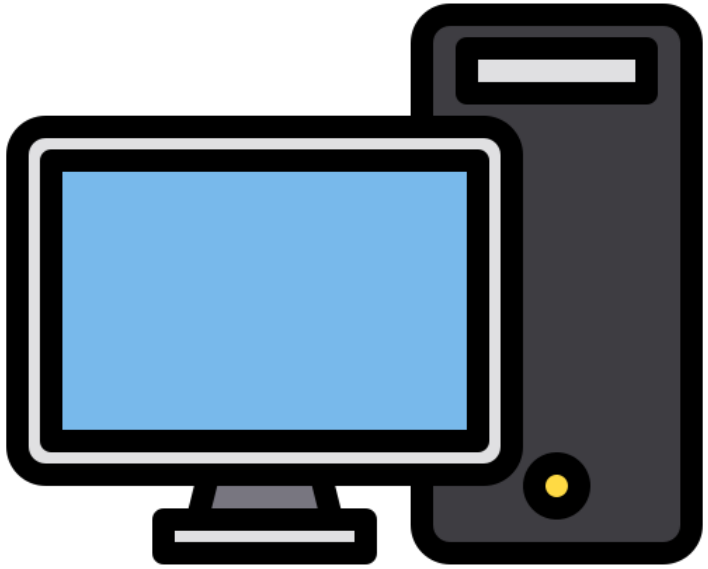
Compute Gain value

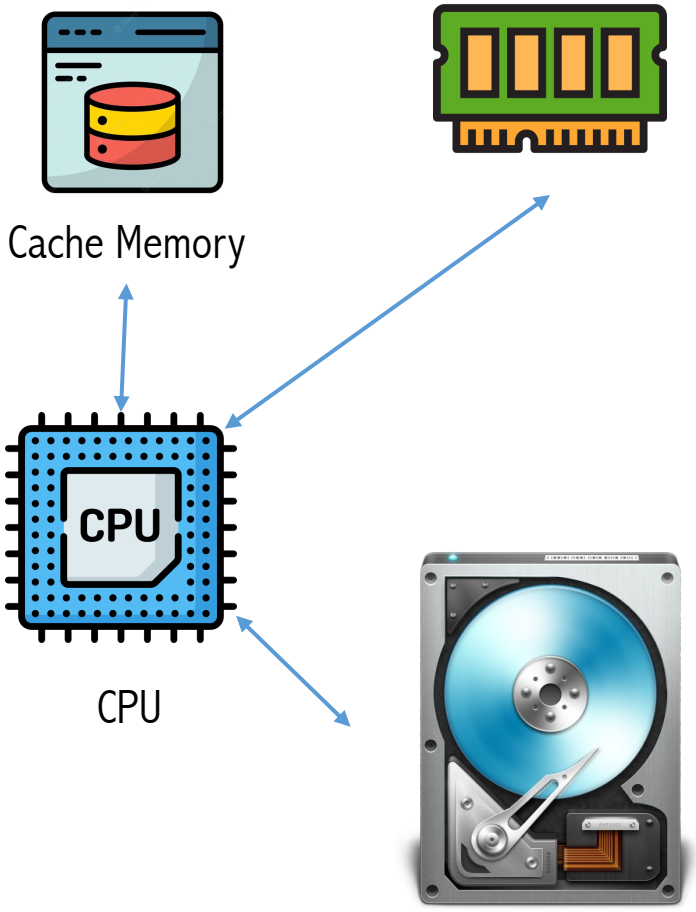
Rẻ nhánh mặc định cho tất cả các missing value trong Dosage



Dosage	Drug Effectiveness	Residual
???	−3	−3.5
???	−2	−2.5

Cache-Aware Access





XGBoost put the gradient and hessians in the Cache so that it can quickly calculate Similarity Score and Output Values

XGBoost

a) accuracy for train = 0.98 and accuracy for test = 0.8
b) accuracy for train = 0.91 and accuracy for test = 0.84
c) accuracy for train = 1.0 and accuracy for test = 0.85
d) accuracy for train = 0.92 and accuracy for test = 0.84

```
from xgboost import XGBClassifier
xg = XGBClassifier(objective="binary:logistic", random_state=42, n_estimators = 100)
xg.fit(X_train, y_train)
y_pred = xg.predict(X_test)

from sklearn.metrics import confusion_matrix
cm_test = confusion_matrix(y_pred, y_test)

y_pred_train = xg.predict(X_train)

print()
accuracy_for_train = np.round((cm_train[0][0] + cm_train[1][1])/len(y_train),2)
accuracy_for_test = np.round((cm_test[0][0] + cm_test[1][1])/len(y_test),2)
print('Accuracy for training set for XGBClassifier = {}'.format(accuracy_for_train))
print('Accuracy for test set for XGBClassifier = {}'.format(accuracy_for_test))
```

XGBoost

A time series is a sequence of information that attaches a time period to each value.

Do We Really Need Deep Learning Models for Time Series Forecasting?

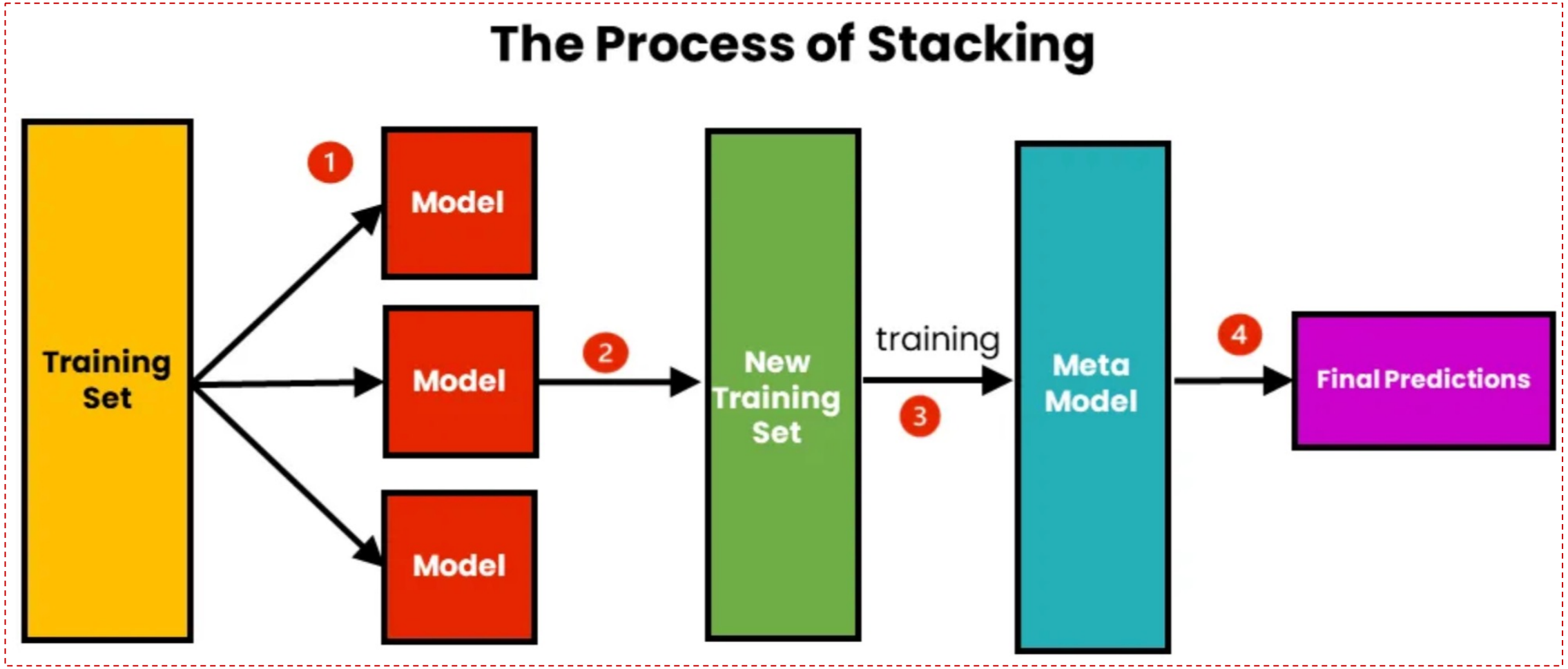
Shereen Elsayed*, Daniela Thyssens*, Ahmed Rashed, Hadi Samer Jomaa, and Lars Schmidt-Thieme

Department of Computer Science
University of Hildesheim
31141 Hildesheim, Germany

Abstract. Time series forecasting is a crucial task in machine learning, as it has a wide range of applications including but not limited to forecasting electricity consumption, traffic, and air quality. Traditional forecasting models rely on rolling averages, vector auto-regression and auto-regressive integrated moving averages. On the other hand, deep learning and matrix factorization models have been recently proposed to tackle the same problem with more competitive performance. However, one major drawback of such models is that they tend to be overly complex in comparison to traditional techniques. In this paper, we report the results of prominent deep learning models with respect to a well-known machine learning baseline, a Gradient Boosting Regression Tree (GBRT) model. Similar to the deep neural network (DNN) models, we transform the time series forecasting task into a window-based regression problem. Furthermore, we feature-engineered the input and output structure of the GBRT model, such that, for each training window, the target values are concatenated with external features, and then flattened to form one input instance for a multi-output GBRT model. We conducted a comparative study on nine datasets for eight state-of-the-art deep-learning

https://arxiv.org/pdf/2101.02118.pdf

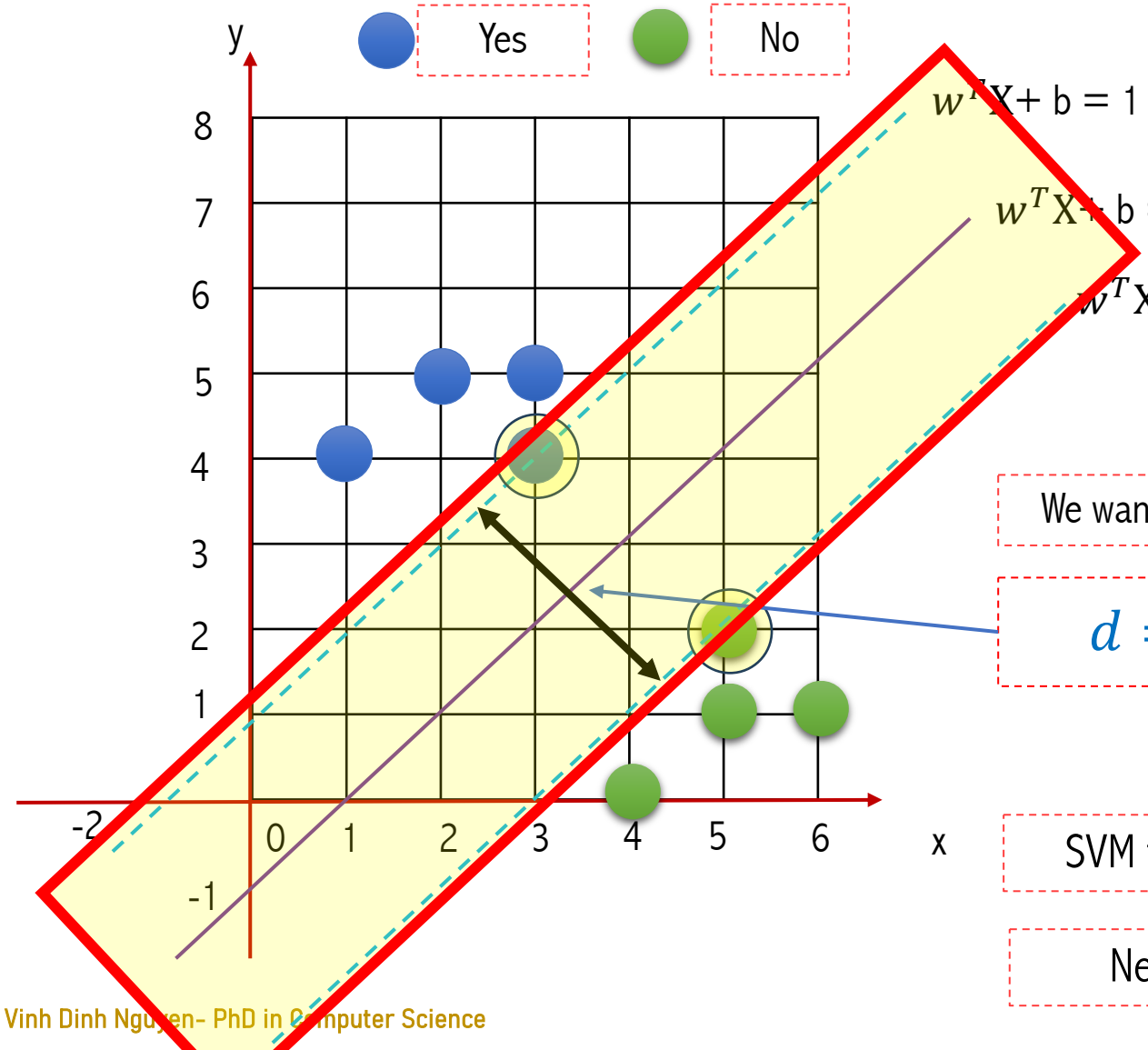
Stacking-Based Method



Outline

- Introduction to Dataset and Problem
- Solving Problem by using KNN
- Solving Problem by Naïve Bayes Classifier
- Solving Problem by Decision Tree
- Solving Problem by Random Forest
- Solving Problem by AdaBoost
- Solving Problem by Gradient Boosting
- Solving Problem by XGBoost
- Solving Problem by SVM
- Solving Problem by Stacking
- Time Series Data
- Feature Important in DT

SVM



$$Y = \begin{cases} +1 \; if w^T X+ b \geq 0 \\ -1 \; if w^T X+ b < 0 \end{cases}$$

We want to maximize the distance d

$$d = \frac{|2|}{\sqrt{A^2+B^2}} = \frac{2}{\|W\|_2}$$

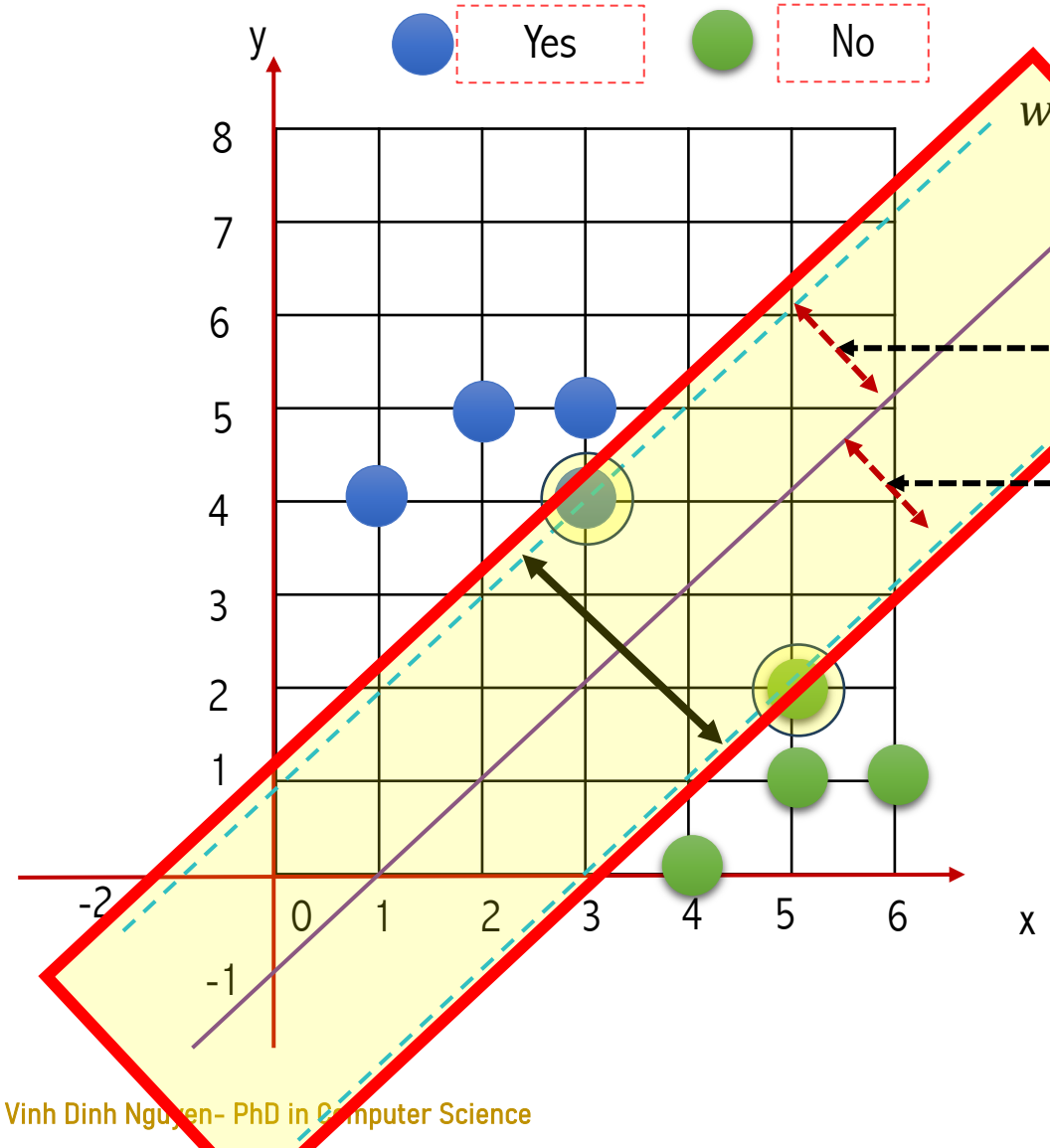
We want to minimize $\|W\|_2$

$$d = \frac{|2|}{\sqrt{A^2+B^2}} = \frac{2}{\|W\|_2}$$

SVM tries to find a hyperplane that maximizes the width of this margin

Need some constraints because the margin can be infinitely large

SVM



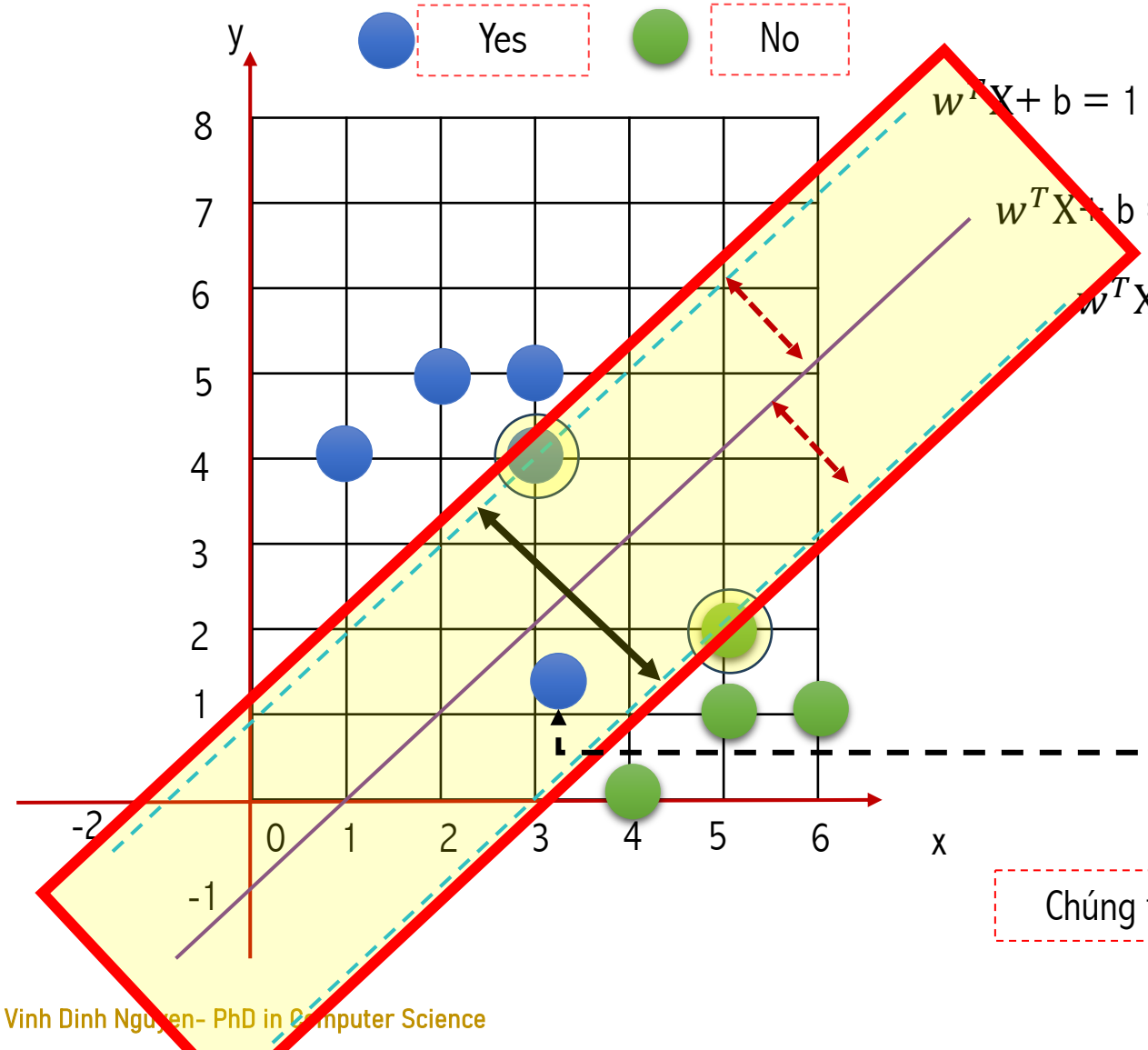
$$\text{Max } \frac{2}{\|w\|} \text{ such that } \begin{cases} w^T X + b \geq 1 \text{ if } Y = +1 \\ w^T X + b \leq -1 \text{ if } Y = -1 \end{cases}$$

$$\text{Max } \frac{2}{\|w\|_2} \text{ such that } Y * (w^T X + b) \geq 1$$

SVM tries to find a hyperplane that maximizes the width of this margin

Constraints: the margin should not span beyond the support vectors

SVM

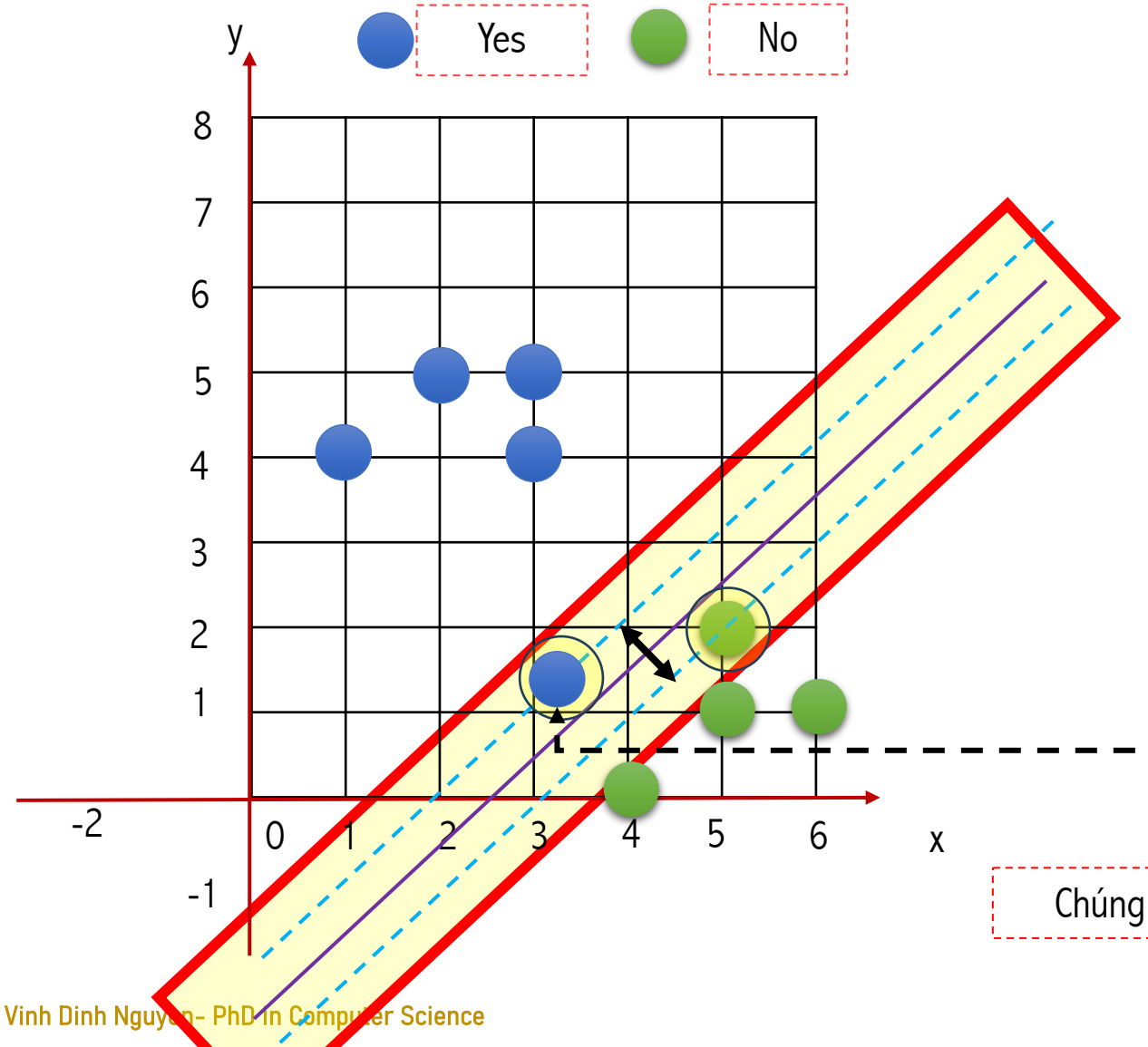


$$\text{Max } \frac{2}{\|w\|_2} \text{ such that } Y * (w^T X + b) \geq 1$$

What should we do if a green data point is noise?

Chúng ta nên chấp nhận như là miss-classification hay là thay đổi hyperplane

SVM

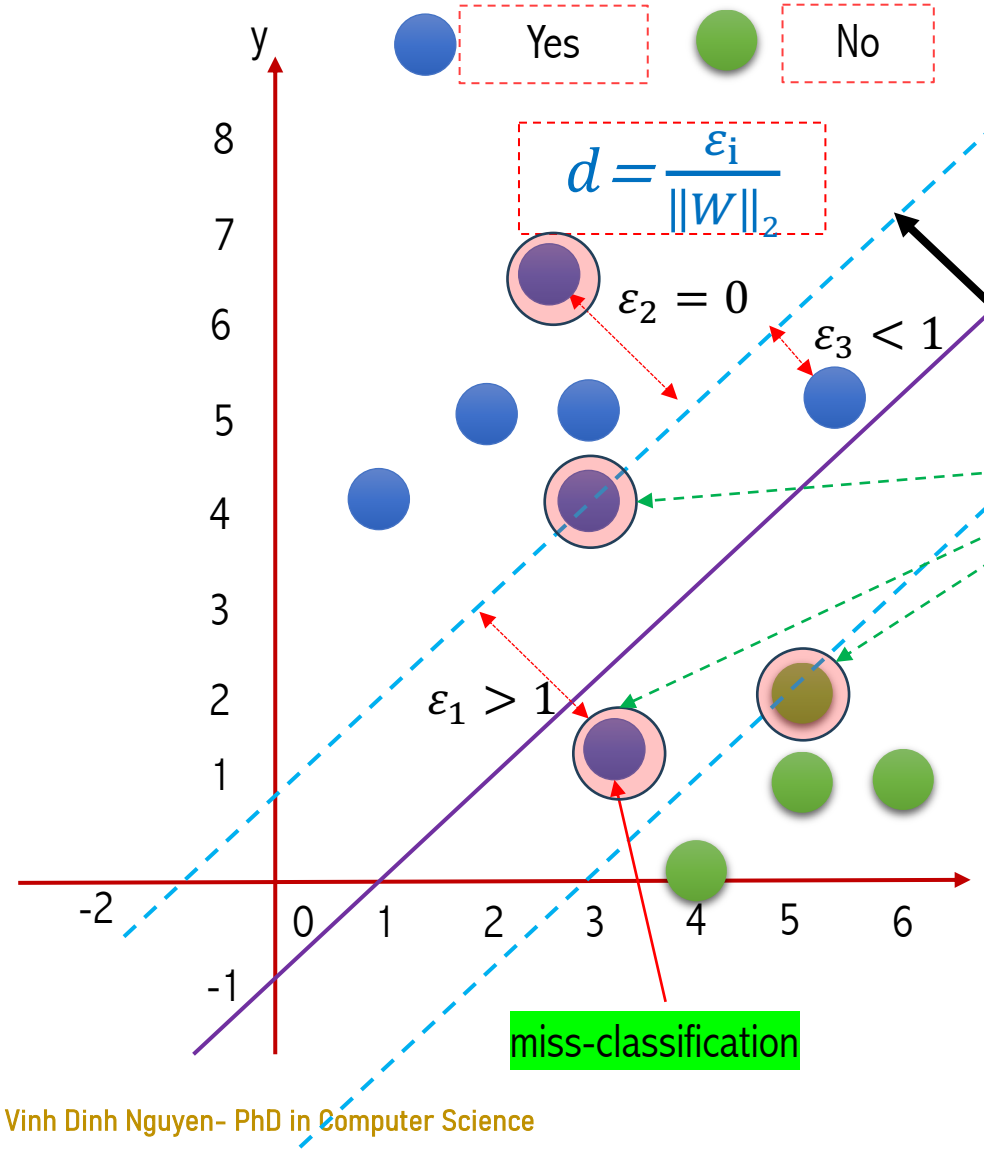


$$\text{Max } \frac{2}{\|w\|_2} \text{ such that } Y * (w^T X + b) \geq 1$$

What should we do if a green data point is noise?

Chúng ta nên chấp này như là miss-classification hay là thay đổi hyperplane

SVM



$w^T X + b = 1$

$w^T X + b = 0$

$w^T X + b = -1$

$$d = \frac{|2|}{\sqrt{A^2+B^2}} = \frac{2}{\|W\|_2}$$

Support Vector

Slack variable in SVM

$$\text{Max } \frac{2}{\|W\|_2} \text{ such that } Y * (w^T X + b) \geq 1$$

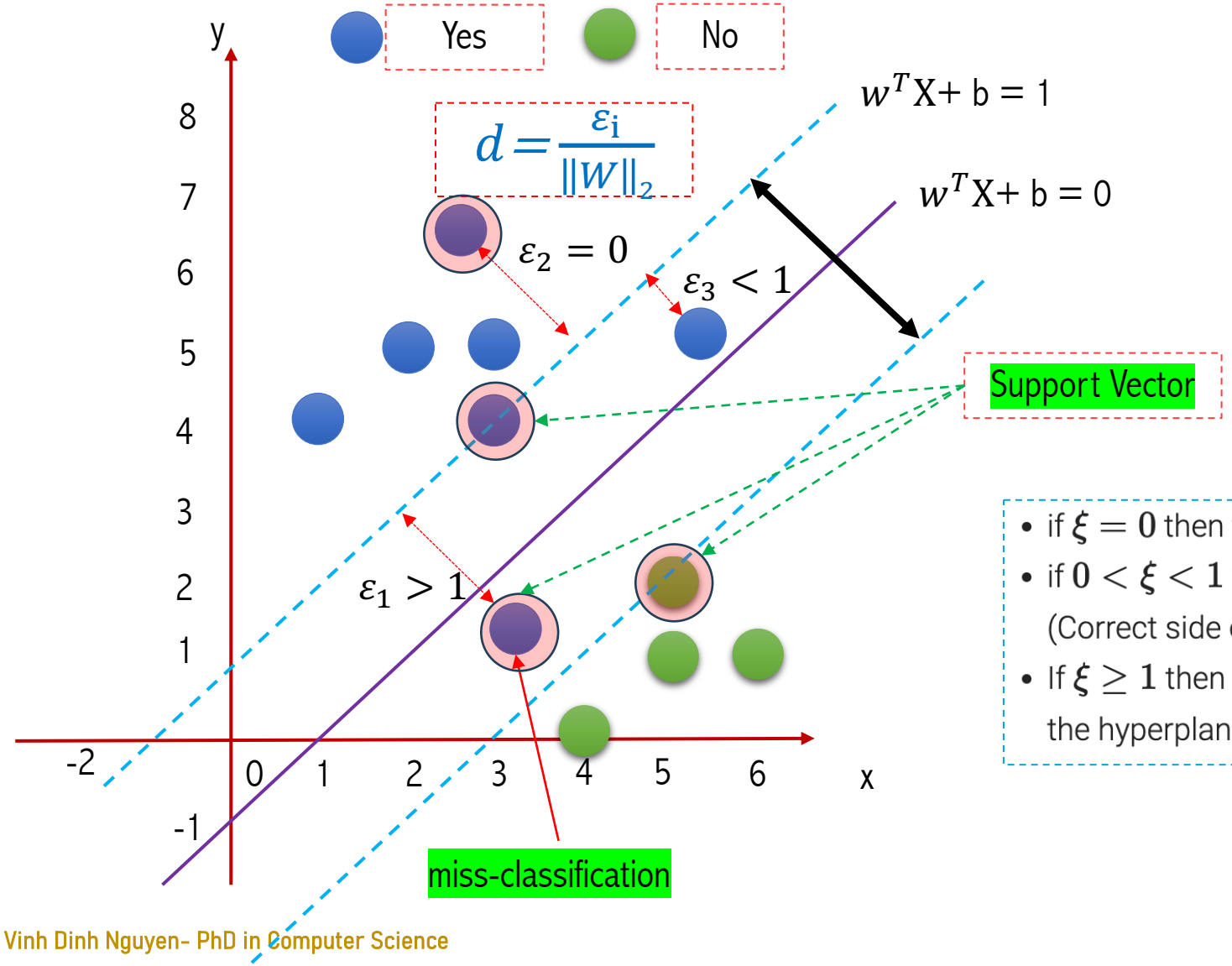
$$\min \frac{1}{2} \|W\|_2^2 + C \sum_i^N \varepsilon_i \text{ such that } Y * (w^T X + b) \geq 1 - \varepsilon_i$$

Allow for miss classification

ε_i is a distance measure of the data points from their corresponding blue line.

C controls how much weight should set on the misclassification data

SVM



- if $\xi = 0$ then the corresponding point ξ is on the margin or further away.
- if $0 < \xi < 1$ then the point ξ is within the margin and classified correctly (Correct side of the hyperplane).
- If $\xi \geq 1$ then the point is misclassified and present at the wrong side of the hyperplane.

SVM

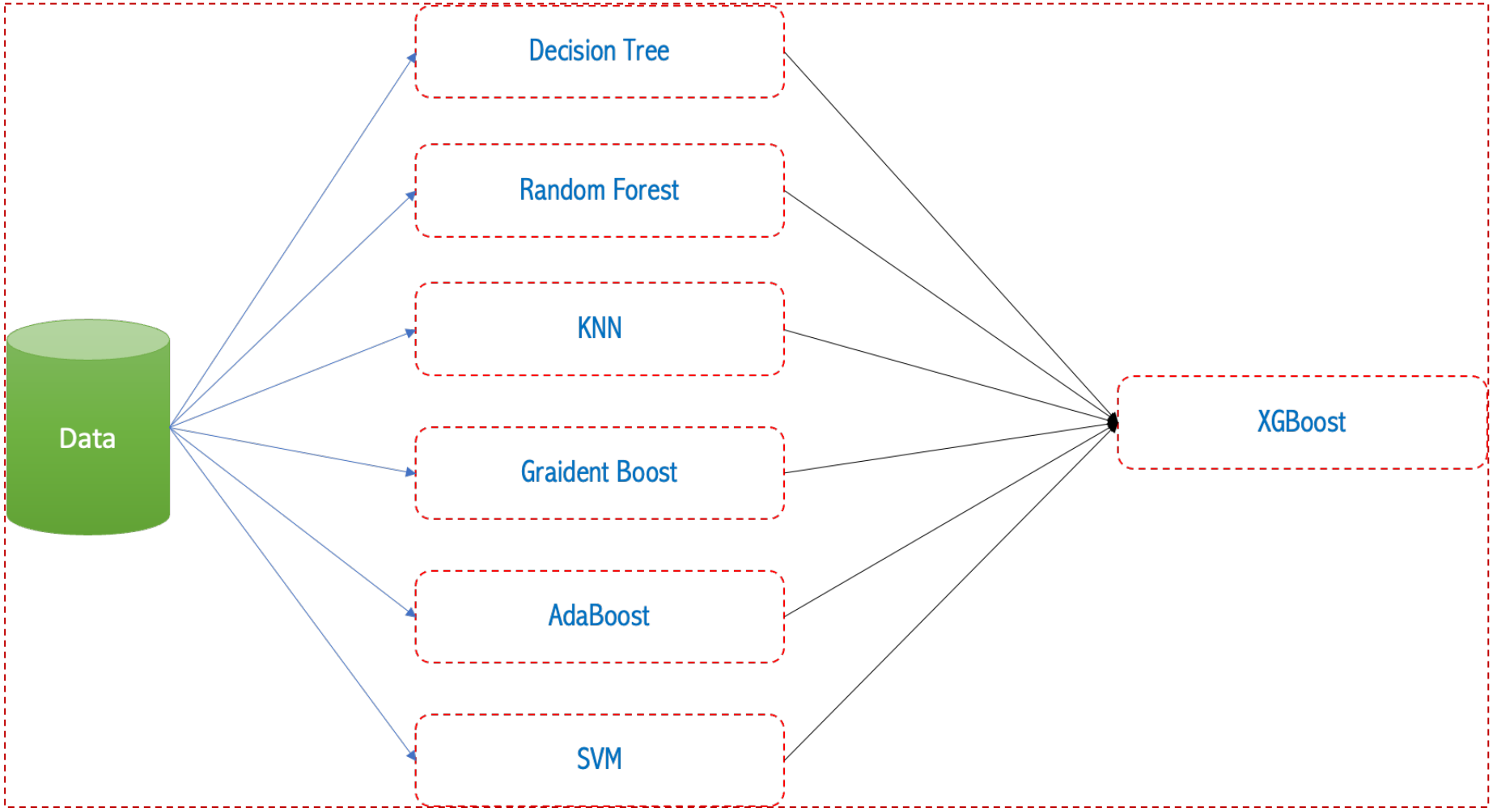
a) accuracy for train = 0.76 and acccuracy for test = 0.69
b) accuracy for train = 0.66 and acccuracy for test = 0.67
c) accuracy for train = 2.76 and acccuracy for test = 0.69
d) accuracy for train = 3.76 and acccuracy for test = 0.69

```
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state=42)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix
cm_test = confusion_matrix(y_pred, y_test)

y_pred_train = classifier.predict(X_train)
cm_train = confusion_matrix(y_pred_train, y_train)

print()
accuracy_for_train = np.round((cm_train[0][0] + cm_train[1][1])/len(y_train),2)
accuracy_for_test = np.round((cm_test[0][0] + cm_test[1][1])/len(y_test),2)
print('Accuracy for training set for SVC = {}'.format(accuracy_for_train))
print('Accuracy for test set for SVC = {}'.format(accuracy_for_test))
```

Outline

- Introduction to Dataset and Problem
- Solving Problem by using KNN
- Solving Problem by Naïve Bayes Classifier
- Solving Problem by Decision Tree
- Solving Problem by Random Forest
- Solving Problem by AdaBoost
- Solving Problem by Gradient Boosting
- Solving Problem by XGBoost
- Solving Problem by SVM
- Solving Problem by Stacking
- Time Series Data
- Feature Important in DT

Stacking-Based Method



Stacking-Based Method

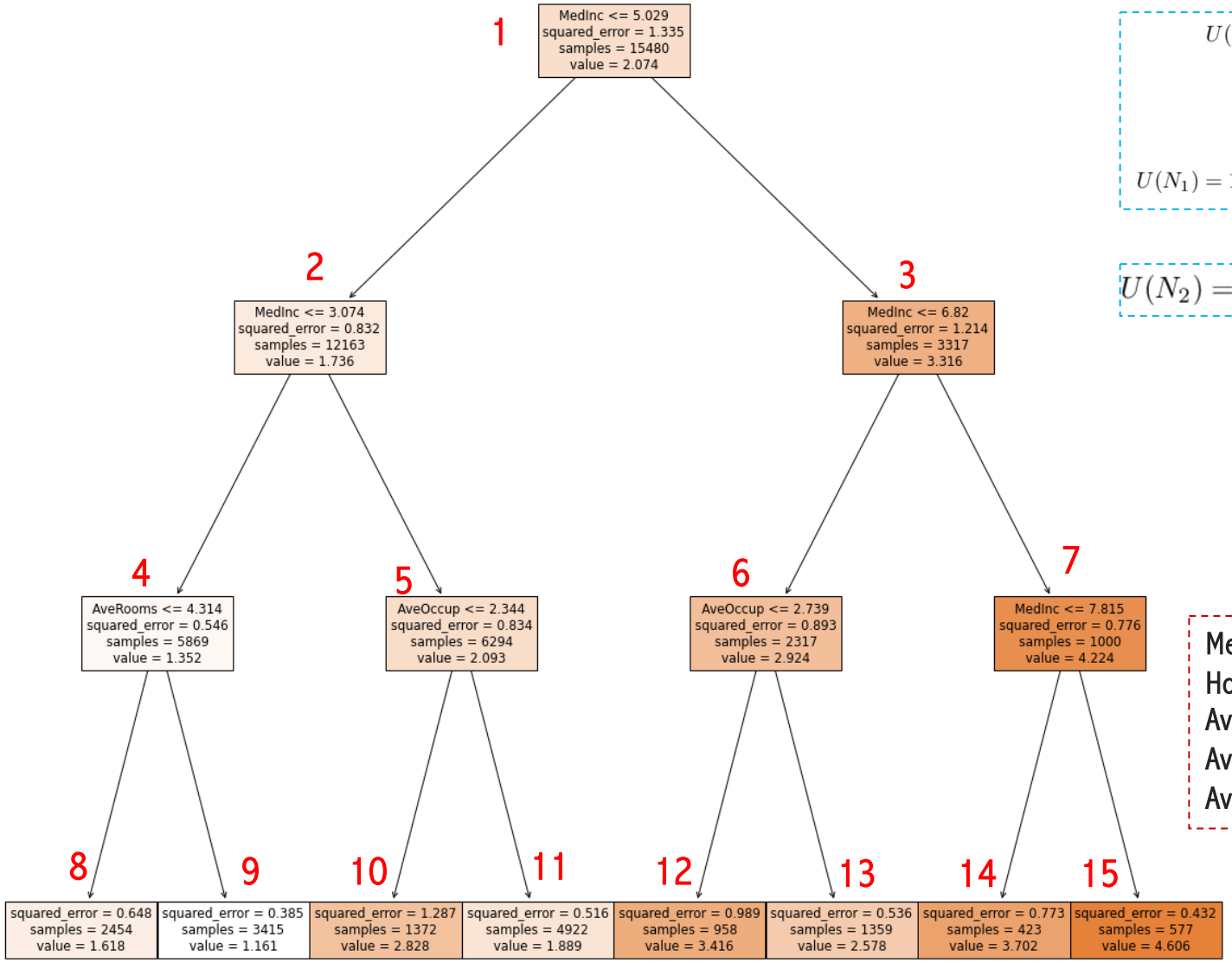
a) accuracy for train = 0.92 and acccuracy for test = 0.9
b) accuracy for train = 0.91 and acccuracy for test = 0.84
c) accuracy for train = 1.0 and acccuracy for test = 0.85
d) accuracy for train = 1.0 and acccuracy for test = 0.84

```
dtc =  DecisionTreeClassifier(random_state=42)
rfc = RandomForestClassifier(random_state=42)
knn =  KNeighborsClassifier()
xgb = XGBClassifier(XGBClassifier)
gc = GradientBoostingClassifier(random_state=42)
svc = SVC(kernel = 'rbf', random_state=42)
ad = AdaBoostClassifier(random_state=42)

clf = [('dtc',dtc),('rfc',rfc),('knn',knn), ('gc',gc), ('ad',ad), ('svc', svc)] #list of (str, estimator)
from sklearn.ensemble import StackingClassifier
xg = XGBClassifier()
classifier = StackingClassifier( estimators = clf,final_estimator = xg)
classifier.fit(X_train,y_train)

classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix
cm_test = confusion_matrix(y_pred, y_test)

y_pred_train = classifier.predict(X_train)
cm_train = confusion_matrix(y_pred_train, y_train)
```

Outline

- Introduction to Dataset and Problem
- Solving Problem by using KNN
- Solving Problem by Naïve Bayes Classifier
- Solving Problem by Decision Tree
- Solving Problem by Random Forest
- Solving Problem by AdaBoost
- Solving Problem by Gradient Boosting
- Solving Problem by XGBoost
- Solving Problem by SVM
- Solving Problem by Stacking
- Time Series Data
- Feature Important in DT

Feature Importance in DT: Regression



$$U(N_1) = w_1 i_1^2 - W(N_1^l) I^2(N_1^l) - W(N_1^r) I^2(N_1^r)$$

$$U(N_1) = w_1 i_1^2 - w_2 i_2^2 - w_3 i_3^2$$

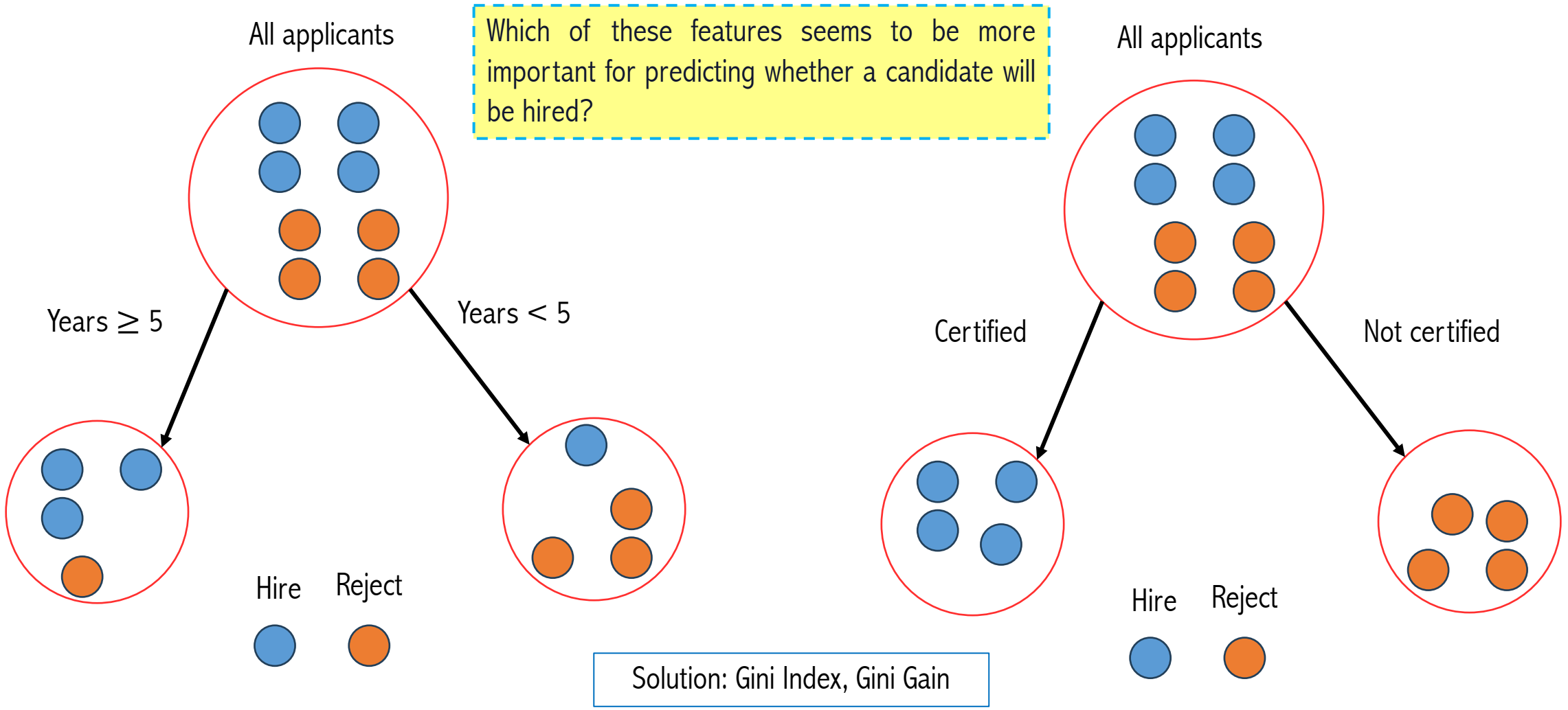
$$U(N_1) = 1.0 * 1.335 - 0.786 * 0.832 - 0.214 * 1.214 = 0.421252$$

$$U(N_2) = 0.832 * 0.786 - 0.546 * 0.379 - 0.834 * 0.407 = 0.10758$$

```
{
  "MedInc": 0.421 + 0.10758
  "HouseAge": 0,
  "AveRooms": 0,
  "AveBedrms": 0,
  "AveOccup": 0

}
```

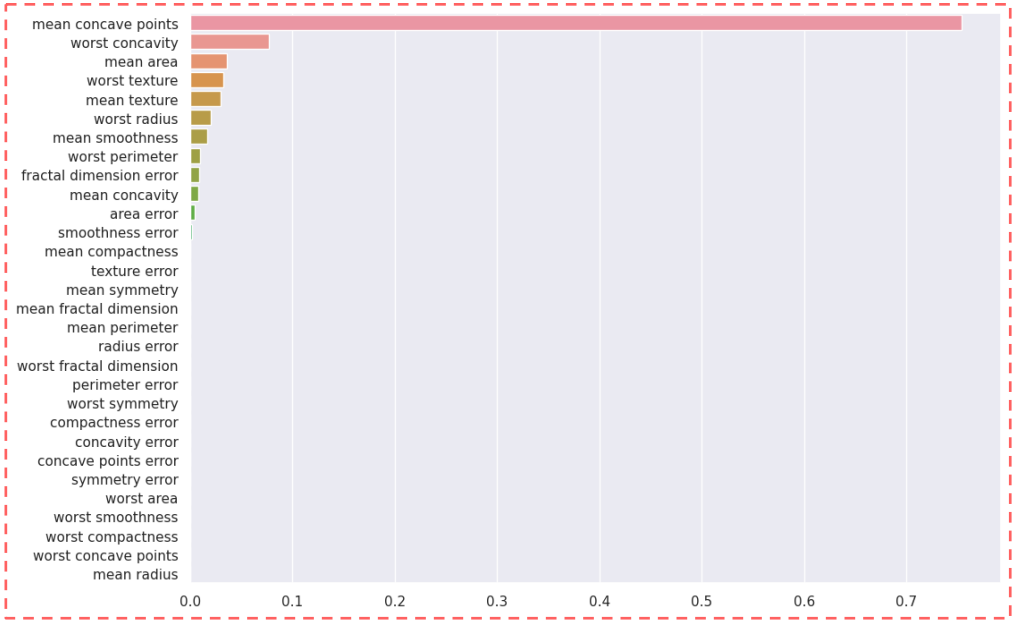
MedInc — Median household income in the past 12 months (hundreds of thousands)
HouseAge — Age of the house (years)
AveRooms — Average number of rooms per dwelling
AveBedrms — Average number of bedrooms per dwelling
AveOccup — Average number of household members

Feature Importance in DT: Classification



Feature Importance in DT: Classification

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
import seaborn as sns

dataset = datasets.load_breast_cancer()
X = pd.DataFrame(dataset.data, columns=dataset.feature_names)
y = dataset.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)


clf = DecisionTreeClassifier(criterion='gini')
# Fit the decision tree classifier
clf = clf.fit(X_train, y_train)
```




```
# Print the feature importances
feature_importances = clf.feature_importances_

# Sort the feature importances from greatest to least using the sorted indices
sorted_indices = feature_importances.argsort()[::-1]
sorted_feature_names = dataset.feature_names[sorted_indices]
sorted_importances = feature_importances[sorted_indices]

# Create a bar plot of the feature importances
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.barplot(x = sorted_importances, y = sorted_feature_names)
```


Thank
you