MACHINE LEARNING

MEMBANGUN MODEL -1

(Naïve Bayes, Support Vector Machine, Boosting dan Decision Tree)

Magister Teknik Informatika

Dr. Chairani, S.Kom., M.Eng

IIB DARMAJAYA, 2023/2024

Subject

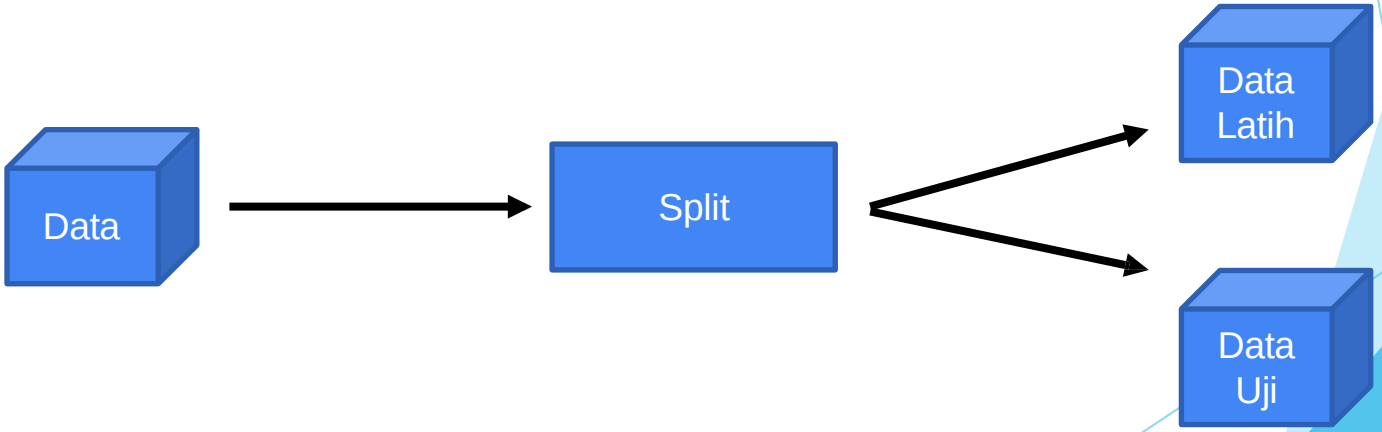
- Modul ini adalah bagian pertama dari Membangun Model
- Membangun Model yang dibahas adalah :
  - Merancang Skenario Model
  - Membangun Model Klasifikasi
- Terdapat beberapa algoritma yang akan dibahas yaitu k-Nearest Neighbor, Naive Bayes, Support Vector Maching, Boosting dan Decision Tree
- Pembangunan model menggunakan library
- Modul ini akan dilanjutkan dengan pembahasan pembangunan model regresi pada modul 12.

Outline

- Membangun Skenario Pemodelan :
  - Pembagian data : data latih-uji, *k*-fold cross validation
  - Menentukan Langkah Eksperimen
  - Parameter Evaluasi
- Membangun Model Klasifikasi :
  - Algoritma klasifikasi yang diimplementasi menggunakan library, yaitu : k-NN, Decision Tree, Naïve Bayes, Support Vector Machine dan Boosting
  - Matriks Performansi Klasifikasi

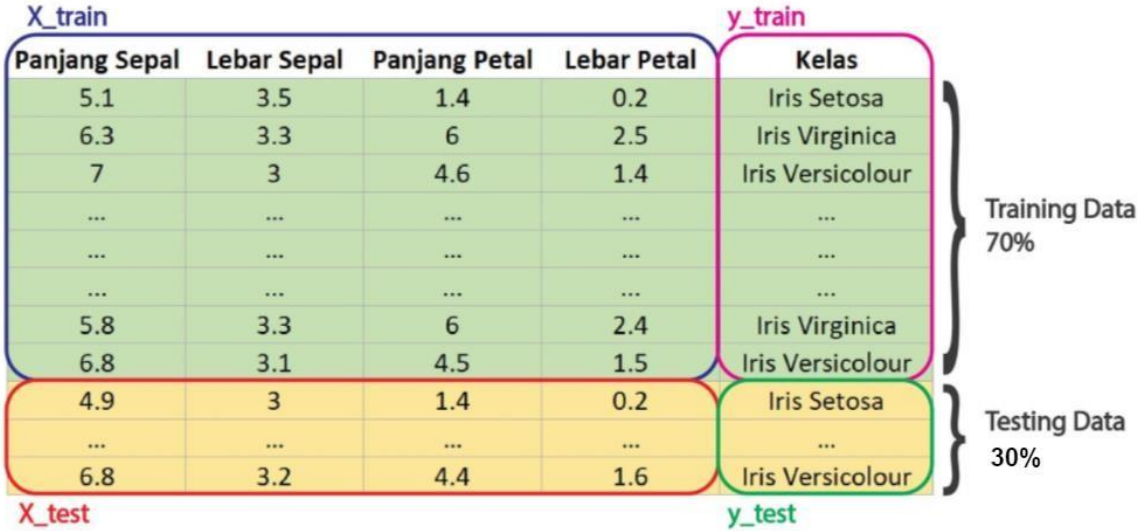
Pembagian Data

- Data dibagi menjadi 2 bagian :
  - Data Latih (*Training Data*) : untuk mengembangkan model
  - Data Uji (*Testing Data*) : untuk Mengukur performansi model



Pembagian Data

- Dataset Iris (https://archive.ics.uci.edu/ml/datasets/iris):
  - Data Latih (*Training Data*) : 70%
  - Data Uji (*Testing Data*) : 30%



Hands On

Data Latih : 70%
Data Uji : 30%

```
[5]  from sklearn.model_selection import train_test_split

     X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7)
```

```
[6]  print("Banyak data latih setelah dilakukan Train-Test Split: ", len(X_train))
     print("Banyak data uji setelah dilakukan Train-Test Split: ", len(X_test))

     Banyak data latih setelah dilakukan Train-Test Split:  105
     Banyak data uji setelah dilakukan Train-Test Split:  45
```

Output, jumlah data latih dan data uji

k-Fold Cross Validation

- *k*-Fold Cross Validation digunakan pada dataset dengan jumlah data yang relatif sedikit
- *k*-Fold Cross Validation dilakukan pada data latih
- Data latih dibagi menjadi *k* bagian kemudian secara iteratif, 1 bagian menjadi data validasi

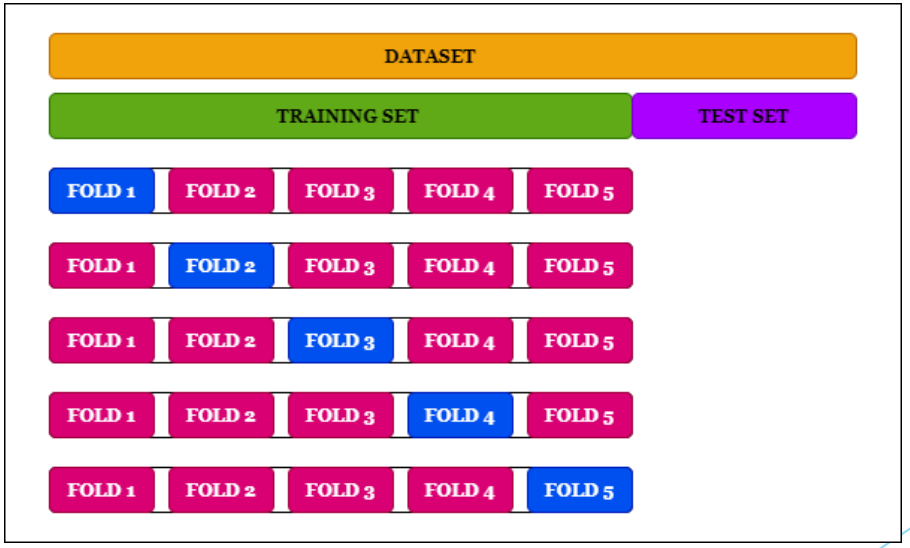


Image Source : https://medium.com/the-owl/k-fold-cross-validation-in-keras-3ec4a3a00538

Hands On

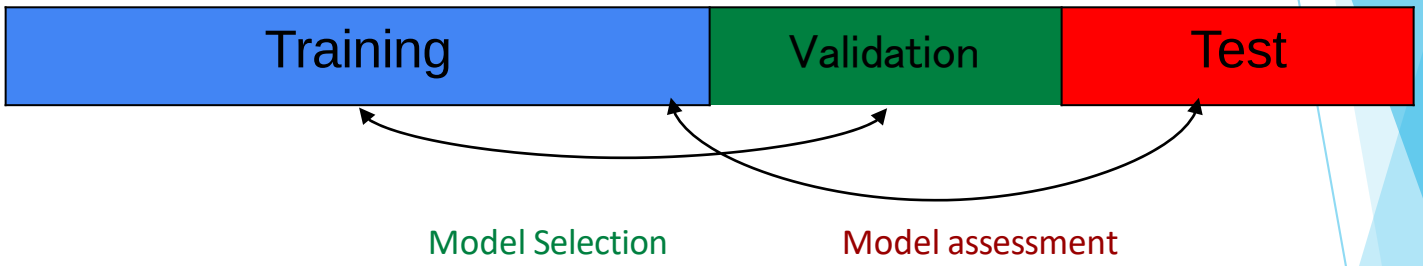
```
[ ]  from sklearn.model_selection import cross_val_score
     from sklearn.svm import SVC

     model = SVC(kernel = 'linear', C = 1)
     scores = cross_val_score(model, X, y, cv = 5)
     print("Akurasi model SVM untuk tiap fold: ", scores)
     print("Akurasi model SVM dengan 5-Fold Cross Validation: ", scores.mean())
```

```
Akurasi model SVM untuk tiap fold:  [0.96666667 1.         0.96666667 0.96666667 1.        ]
Akurasi model SVM dengan 5-Fold Cross Validation:  0.9800000000000001
```

5 Cross Validation

Output akurasi dari setiap fold
Akurasi rata- rata dari seluruh fold

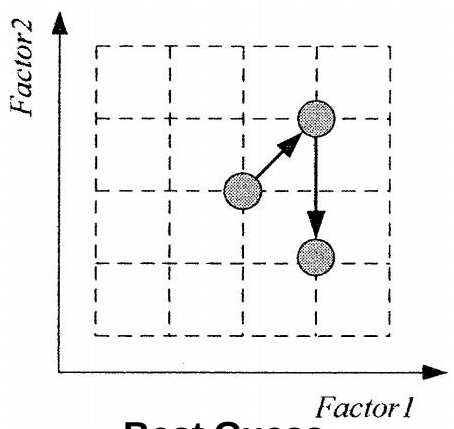
Training – Validation – Testing Data



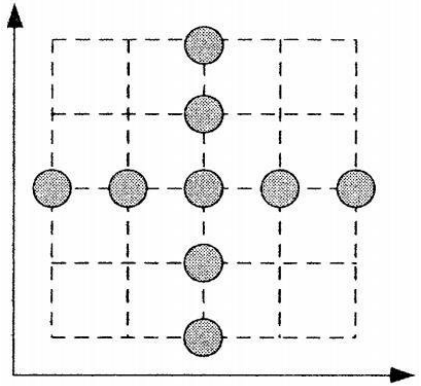
- Model Selection : Mengestimasi performa model - model yang berbeda untuk memilih model yang terbaik, yaitu model dengan minimum error
- Model Assessment : Dari model yang terpilih, mengestimasi error untuk data baru (data uji)

Menentukan Langkah Eksperimen

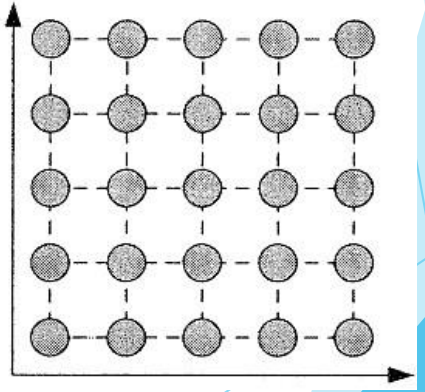
- Setiap metode memiliki parameter tertentu
- Dilakukan eksperimen dengan beberapa variasi parameter
- Parameter yang menghasilkan model performa terbaik akan digunakan selanjutnya
- Beberapa strategi pencarian parameter untuk menghasilkan model terbaik



Best Guess



One Factor at A Time



Grid Search

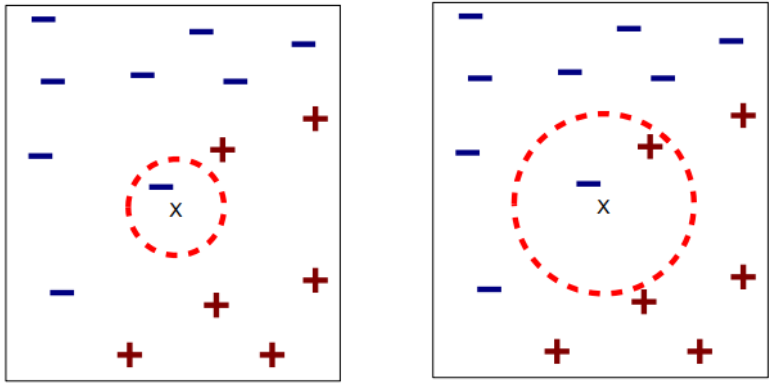
Parameter Evaluasi

- Klasifikasi
  - Akurasi
  - Presisi
  - Recall/Sensitivity
  - Specificity
  - F1-measure
  - …
- Regresi
  - MSE (Mean Squared Error)
  - MAPE (Mean Absolute Percentage Error)
  - …
- Klastering
  - Silhouette Score
  - Davies-Bouldin Index
  - …
- Parameter Evaluasi akan dijelaskan secara detail pada modul – modul berikutnya

Outline

- Membangun Skenario Pemodelan :
  - Pembagian data : data latih-uji, *k*-fold cross validation
  - Menentukan Langkah Eksperimen
  - Parameter Evaluasi
- **Membangun Model Klasifikasi :**
  - Algoritma klasifikasi yang diimplementasi menggunakan library, yaitu : k-NN, Decision Tree, Naïve Bayes, Support Vector Machine dan Boosting
  - Matriks Performansi Klasifikasi

k – Nearest Neighbor (k-NN) Classifier



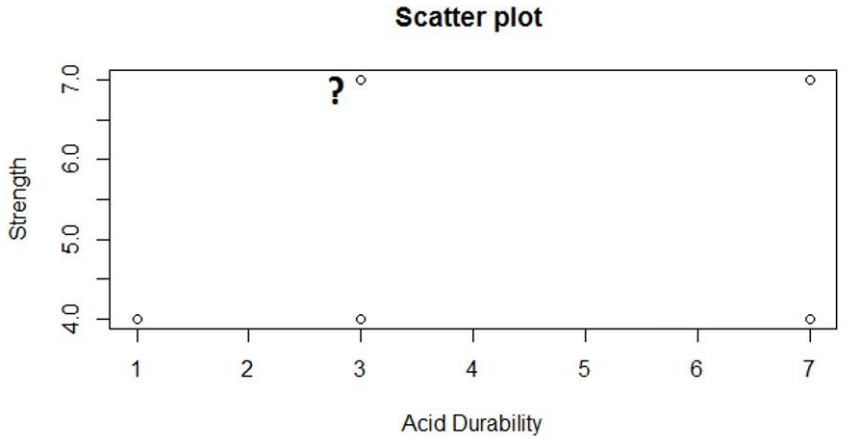
(a) 1-nearest neighbor (b) 2-nearest neighbor

Nearest Neighbor terhadap data baru (x)

- Algoritma :
  - Menentukan nilai k
  - Menghitung jarak antara data baru terhadap semua training data
  - Mengidentifikasi k nearest neighbor
  - Menentukan label/kelas data baru berdasarkan kelas k-nearest neighbor (dapat menggunakan voting)

Contoh kasus

- Terdapat 4 data latih (P1, P2, P3, P4). Data memiliki dua atribut (x1 dan x2)
- Data latih terdiri dari dua kelas (kelas BAD dan kelas GOOD)
- Diperlukan klasifikasi untuk menentukan kelas dari data uji (P5).



Points	X1(Acid Durability)	X2(Strength)	Y(Classification)
P1	7	7	BAD
P2	7	4	BAD
P3	3	4	GOOD
P4	1	4	GOOD
P5	3	7	?

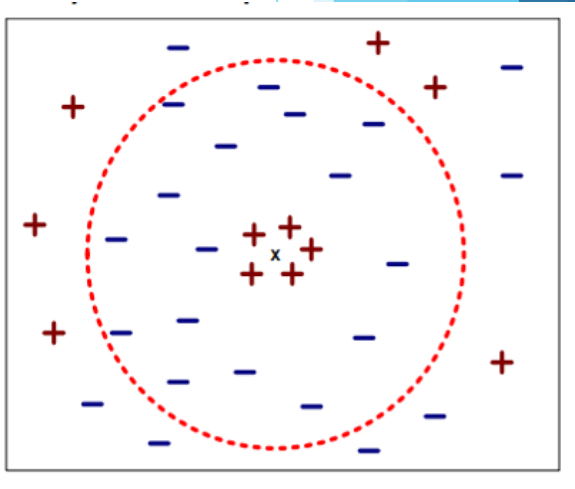
Contoh kasus (lanjutan)

Euclidean Distance of P5(3,7) from	P1	P2	P3	P4
	(7,7)	(7,4)	(3,4)	(1,4)
	Sqrt((7-3) ^{2} + (7-7)^2) = $\sqrt{16}$ = 4	Sqrt((7-3) ^{2} + (4-7)^2) = $\sqrt{25}$ = 5	Sqrt((3-3) ^{2} + (4-7)^2) = $\sqrt{9}$ = 3	Sqrt((1-3) ^{2} + (4-7)^2) = $\sqrt{13}$ = 3.60
Class	BAD	BAD	GOOD	GOOD

Points	X1(Durability)	X2(Strength)	Y(Classification)
P1	7	7	BAD
P2	7	4	BAD
P3	3	4	GOOD
P4	1	4	GOOD
P5	3	7	GOOD

Kelebihan dan Kekurangan *k*-NN Classifier

- Cocok untuk data numerik
- Mudah dipahami dan diimplementasikan
- *k*-NN merupakan *lazy learner* (tidak membangun model secara eksplisit)
- Penentuan label/kelas data baru membutuhkan *computational cost* yang cukup tinggi
- Perlu menentukan nilai k yang sesuai
  - Jika k terlalu kecil, sensitif terhadap noise
  - Jika k terlalu besar, nearest neigbor mungkin mencakup data dari kelas lain



Hands On

```
[ ]  from sklearn.neighbors import KNeighborsClassifier
     from sklearn import metrics

     knn = KNeighborsClassifier()

     knn.fit(X_train, y_train)
     y_pred = knn.predict(X_test)
     score = metrics.accuracy_score(y_test, y_pred)
     print("Akurasi dengan menggunakan Nearest Neighbor: ", score)

     Akurasi dengan menggunakan Nearest Neighbor:  0.9777777777777777
```

k-NN Classifier

Proses training data latih, prediksi data uji, perhitungan akurasi

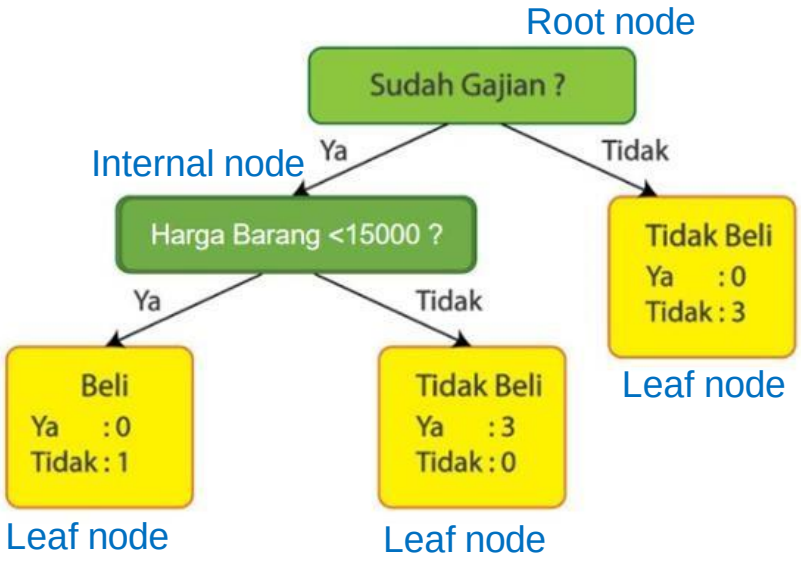
Output akurasi yang dihasilkan *k-NN classifier*

Hands On (lanjutan)

Parameter	Keterangan	Contoh Nilai
n_neighbors	Jumlah Neighbor	- Bilangan Integer (1,2,3,4,….) - Nilai default : 4

```
# Import kNN Classifier dari sklearn
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=3) # memilih 3 sebagai banyaknya neighbors
```

Decision Tree Classifier



- Membangun *decision tree* (pohon keputusan) dari data lati*h*
- *Output* dari pohon keputusan => *rule*. *Rule* tersebut digunakan untuk klasifikasi
- Pohon keputusan, terdiri dari :
  - Root Node
  - Internal Node
  - Leaf Node

Decision Tree Classifier

Kebutuhan Primer	Sudah Gajian	Harga Barang	Beli
Ya	Ya	7000	Tidak
Ya	Tidak	12000	Tidak
Tidak	Ya	18000	Ya
Tidak	Ya	35000	Ya
Ya	Ya	38000	Ya
Ya	Tidak	50000	Tidak
Tidak	Tidak	83000	Tidak

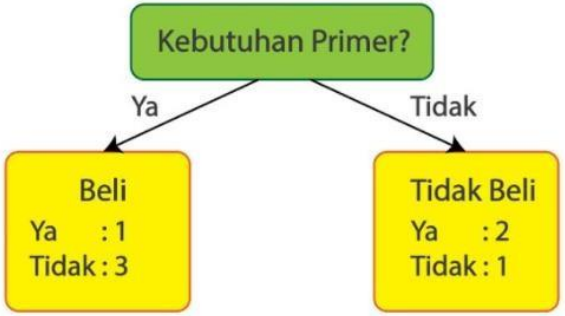
- Data latih terdiri dari 7 data. Data terdiri dari 3 atribut.
- Terdapat 2 kelas : Beli dan Tidak Beli

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

- Menghitung Gini Index (GI) untuk setiap atribut
- Menentukan root berdasarkan nilai GI. Root adalah atribut dengan nilai GI terkecil.
- Ulangi langkah 1 dan 2 untuk level berikutnya pada tree hingga nilai GI = 0.

Gini Index/Impurity untuk Atribut "Kebutuhan Primer"

Kebutuhan Primer	Sudah Gajian	Harga Barang	Beli
Ya	Ya	7000	Tidak
Ya	Tidak	12000	Tidak
Tidak	Ya	18000	Ya
Tidak	Ya	35000	Ya
Ya	Ya	38000	Ya
Ya	Tidak	50000	Tidak
Tidak	Tidak	83000	Tidak



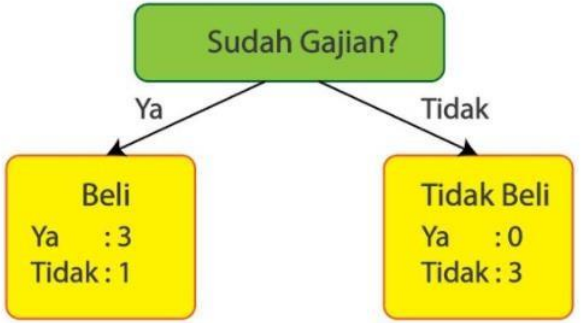
Gini Impurity
$= 1 - (1 / 4)^2 - (3 / 4)^2$
$= 0.38$

Gini Impurity
$= 1 - (2 / 3)^2 - (1 / 3)^2$
$= 0.44$

Total Gini Impurity
$= (4/7) * 0.38 + (3/7) * 0.44$
$= 0.4$

Gini Index/Impurity untuk Atribut "Sudah Gajian"

Kebutuhan Primer	Sudah Gajian	Harga Barang	Beli
Ya	Ya	7000	Tidak
Ya	Tidak	12000	Tidak
Tidak	Ya	18000	Ya
Tidak	Ya	35000	Ya
Ya	Ya	38000	Ya
Ya	Tidak	50000	Tidak
Tidak	Tidak	83000	Tidak

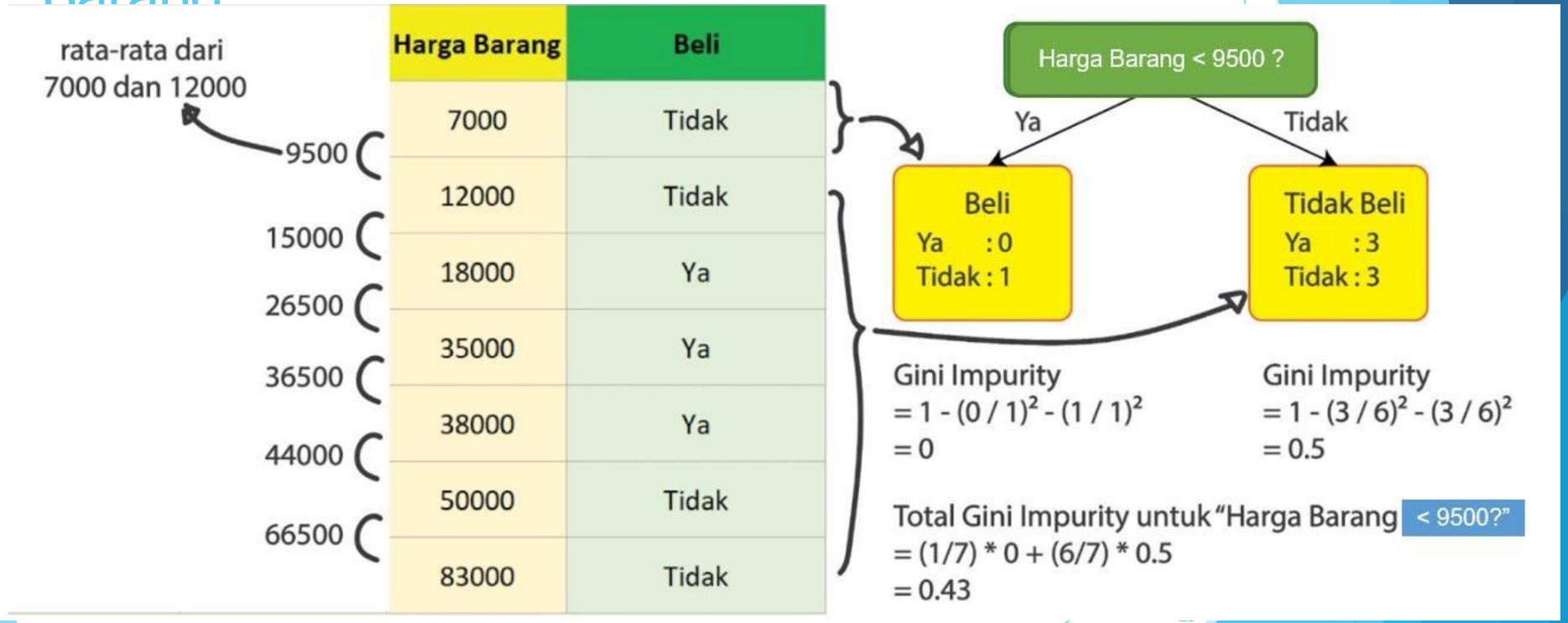


Gini Impurity
$= 1 - (3 / 4)^2 - (1 / 4)^2$
$= 0.38$

Gini Impurity
$= 1 - (0 / 3)^2 - (3 / 3)^2$
$= 0$

Total Gini Impurity
$= (4/7) * 0.38 + (3/7) * 0$
$= 0.22$

Gini Index/Impurity untuk Atribut "Harga Barang"

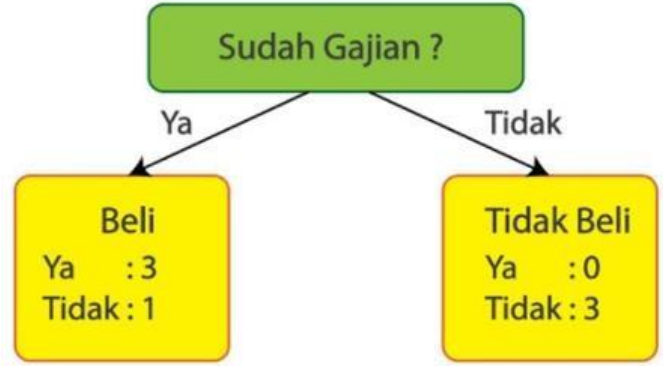


Gini Index/Impurity untuk Atribut "Harga Barang"

Harga Barang	Beli		Gini Impurities
7000	Tidak	9500	0.43
12000	Tidak	15000	0.34
18000	Ya	26500	0.48
35000	Ya	36500	0.48
38000	Ya	44000	0.34
50000	Tidak	66500	0.43
83000	Tidak		

Penentuan Root

- Gini Index "Kebutuhan Primer" : 0.44
- **Gini Index "Sudah gajian" : 0.22**
- Gini Index "Harga barang < 15000" : 0.34
- Attribut yang terpilih sebagai root adalah "Sudah Gajian"
- Dilakukan proses perhitungan Gini Index bagi left node untuk attribut "Kebutuhan Primer" dan "Sudah Gajian"

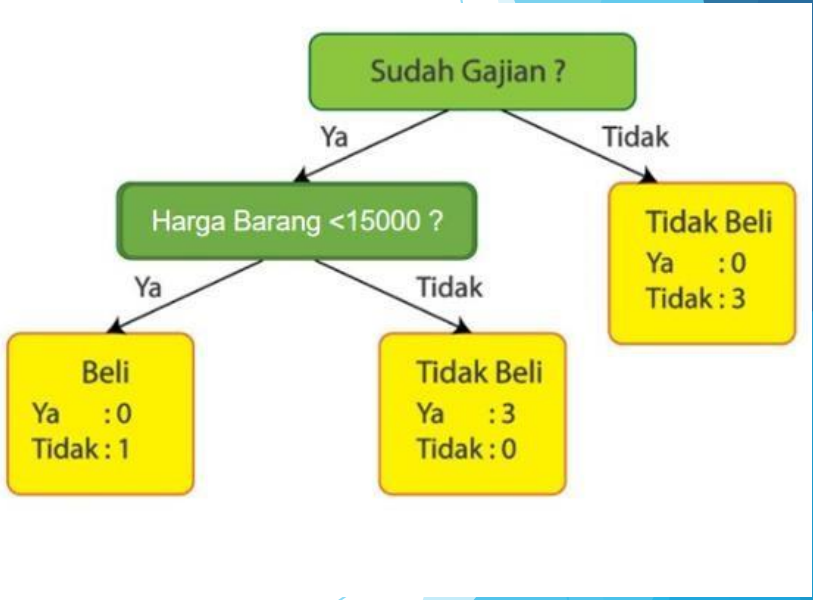


Leaf node

Merupakan *internal node* sehingga memerlukan perhitungan *Gini Index*

Decision Tree Classifier final untuk data latih

Kebutuhan Primer	Sudah Gajian	Harga Barang	Beli
Ya	Ya	7000	Tidak
Ya	Tidak	12000	Tidak
Tidak	Ya	18000	Ya
Tidak	Ya	35000	Ya
Ya	Ya	38000	Ya
Ya	Tidak	50000	Tidak
Tidak	Tidak	83000	Tidak



Hands On

```
[ ] from sklearn.tree import DecisionTreeClassifier
    from sklearn import metrics

    dt = DecisionTreeClassifier(
      max_depth = None,
      min_samples_split = 2
    )

    dt.fit(X_train, y_train)
    y_pred = dt.predict(X_test)
    score = metrics.accuracy_score(y_test, y_pred)
    print("Akurasi dengan menggunakan Decision Tree: ", score)

    Akurasi dengan menggunakan Decision Tree:  0.9555555555555556
```

→ Decision tree classifier

→ Proses training data latih, prediksi data uji, perhitungan akurasi

Output akurasi yang dihasilkan oleh *Decision Tree classifier*

Hands On (lanjutan)

Parameter	Keterangan	Contoh Nilai
max_depth	Maksimum kedalaman tree yang dibentuk	- Bilangan Integer (1,2,3,4,….) - Nilai default : None (jika None maka tree akan terus mendalam sampai seluruh Gini Impurity = 0)
min_samples_split	Minimum sampel yang diperlukan agar node dapat ‘split’	- Bilangan Integer (1,2,3,4,….) - Nilai default : 2

```
# Import Decision Tree Classifier dari sklearn
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(
      max_depth=4, # menentukan kedalaman maksimum tree = 4
      min_samples_split = 2 # menentukan 2 minimal sampel agar node bisa split
    )
```

Kelebihan dan Kekurangan Decision Tree Classifier

- Pohon keputusan dapat memberikan penjelasan dari proses klasifikasi yang dilakukan berupa rule yang dihasilkan
- Proses pembangunan pohon keputusan pada data numerik menjadi lebih rumit dan memungkinkan terdapat informasi yang hilang
- Pohon keputusan dapat tumbuh menjadi sangat kompleks pada data yang rumit.

Naïve Bayes Classifier

- Membangun probabilistik model untuk klasifikasi
  - Discriminative Model

$$P(C \mid \mathbf{X}) \quad C = c_1, \cdots, c_L, \mathbf{X} = (X_1, \cdots, X_n)$$

  - Generative Model

$$P(\mathbf{X} \mid C) \quad C = c_1, \cdots, c_L, \mathbf{X} = (X_1, \cdots, X_n)$$

- MAP Classification Rule
  - MAP : Maximum A Posterior
  - Assign x to c^* if $P(C = c^* \mid \mathbf{X} = \mathbf{x}) > P(C = c \mid \mathbf{X} = \mathbf{x}) \quad c \neq c^*, \; c = c_1, \cdots, c_L$
- Generative classification dengan MAP rule
  - Menerapkan Bayesian Rule

$$P(C \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid C) P(C)}{P(\mathbf{X})} \propto P(\mathbf{X} \mid C) P(C)$$

Contoh Kasus

- Terdapat 14 data latih. Data memiliki 4 atribut (Outlook, Temperature, Humidity dan Wind)
- Data latih terdiri dari dua kelas (kelas YES dan kelas NO untuk bermain tenis)
- Diperlukan klasifikasi untuk menentukan kelas dari data uji, apakah bermain tenis (YES) atau tidak (NO).

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Proses Pelatihan

Outlook	Play=*Yes*	Play=*No*
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=*Yes*	Play=*No*
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=*Yes*	Play=*No*
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=*Yes*	Play=*No*
Strong	3/9	3/5
Weak	6/9	2/5

$P(\text{Play}=Yes) = 9/14$ $P(\text{Play}=No) = 5/14$

Proses Pelatihan

Outlook	Play=*Yes*	Play=*No*
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=*Yes*	Play=*No*
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=*Yes*	Play=*No*
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=*Yes*	Play=*No*
Strong	3/9	3/5
Weak	6/9	2/5

$P(\text{Play}=Yes) = 9/14$ $P(\text{Play}=No) = 5/14$

- Data baru,

  $\mathbf{x}'$=(Outlook=*Sunny*, Temperature=*Cool*, Humidity=*High*, Wind=*Strong*)

- Look up tabel

  P(Outlook=*Sunny* | Play=*Yes*) = 2/9
  P(Temperature=*Cool* | Play=*Yes*) = 3/9
  P(Huminity=*High* | Play=*Yes*) = 3/9
  P(Wind=*Strong* | Play=*Yes*) = 3/9
  P(Play=*Yes*) = 9/14

  P(Outlook=*Sunny* | Play=*No*) = 3/5
  P(Temperature=*Cool* | Play==*No*) = 1/5
  P(Huminity=*High* | Play=*No*) = 4/5
  P(Wind=*Strong* | Play=*No*) = 3/5
  P(Play=*No*) = 5/14

- MAP rule

  $P(Yes|\mathbf{x}')$: [P(*Sunny*|Yes)P(*Cool*|*Yes*)P(*High*|Yes)P(*Strong*|*Yes*)]P(Play=*Yes*) = 0.0053
  $P(No|\mathbf{x}')$: [P(*Sunny*|No) P(*Cool*|No)P(*High*|*No*)P(*Strong*|*No*)]P(Play=*No*) = 0.0206

  Diketahui $P(Yes|\mathbf{x}') < P(No|\mathbf{x}')$, maka label $\mathbf{x}'$ adalah "*No*".

Hands On

```
[ ]  from sklearn import naive_bayes
     from sklearn import metrics

     nb = naive_bayes.BernoulliNB()
```

Naïve Bayes Classifier

```
     nb.fit(X_train, y_train)
     y_pred = nb.predict(X_test)
     score = metrics.accuracy_score(y_test, y_pred)
     print("Akurasi dengan menggunakan Naive Bayes: ", score)
```

Proses training data latih, prediksi data uji, perhitungan akurasi

```
Akurasi dengan menggunakan Naive Bayes:  0.2222222222222222
```

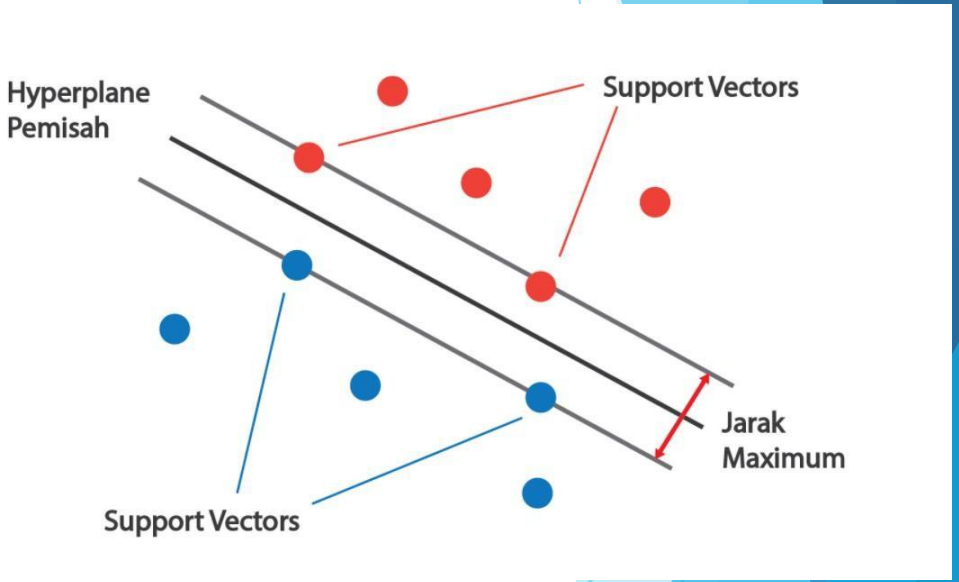
Output akurasi yang dihasilkan oleh *Naïve Bayes Classifier*

Kelebihan dan Kekurangan Naïve Bayes Classifier

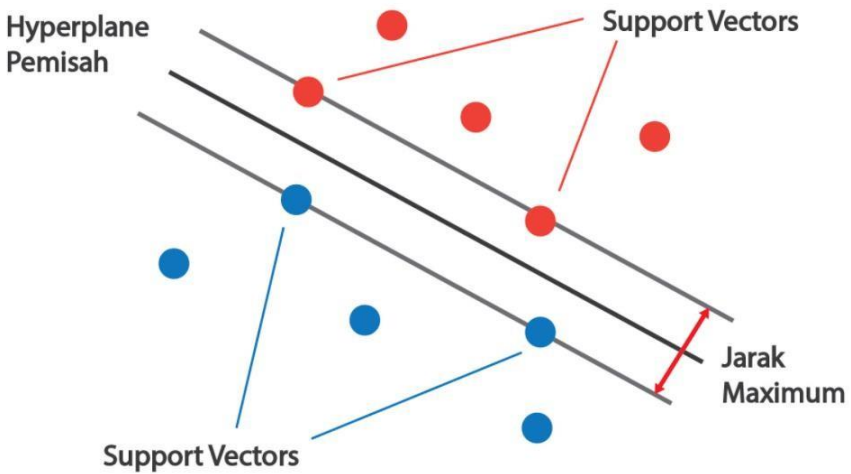
- Naïve Bayes Classifier bekerja lebih baik pada training data yang kecil
- Proses pembangunan Naïve Bayes Classifier pada data numerik menjadi lebih rumit dan memungkinkan terdapat informasi yang hilang
- Pada Naïve Bayes Classifier, diasumsikan bahwa satu fitur dengan fitur yang lain saling independen, hal ini mungkin tidak selalu terjadi pada kasus nyata

Support Vector Machine

- Ide utama dari SVM adalah untuk menemukan hyperplane terbaik yang memisahkan 2 daerah keputusan dengan baik
- Hyperplane adalah sebuah fungsi yang dapat digunakan sebagai pemisah antar kelas



Support Vector Machine



Algoritma SVM :

- Ide Fitur-fitur dari data dipetakan ke dalam ruang dimensi lebih tinggi menggunakan fungsi kernel (linear, polynomial, radial basis function, sigmoid).
- Mencari hyperplane terbaik yang memisahkan data yang telah dipetakan dalam ruang dimensi tinggi.
- Hyperplane terbaik dapat diperoleh dengan memaksimalkan jarak hyperplane dengan titik data terdekat (Support Vectors)

Kernel pada Support Vector Machine

- Fitur-fitur dari data dipetakan ke dalam ruang dimensi tinggi menggunakan fungsi kernel (linear, polynomial, sigmoid, atau radial basis function)
- Dengan menggunakan fungsi kernel, akan dihasilkan fitur-fitur baru yang akan digunakan untuk mengklasifikasi (tidak lagi menggunakan fitur-fitur lama)

Kernel *Linear*

$$K(x_i, x_j) = x_i^T x_j$$

Kernel *Polynomial*

$$K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \quad \gamma > 0$$

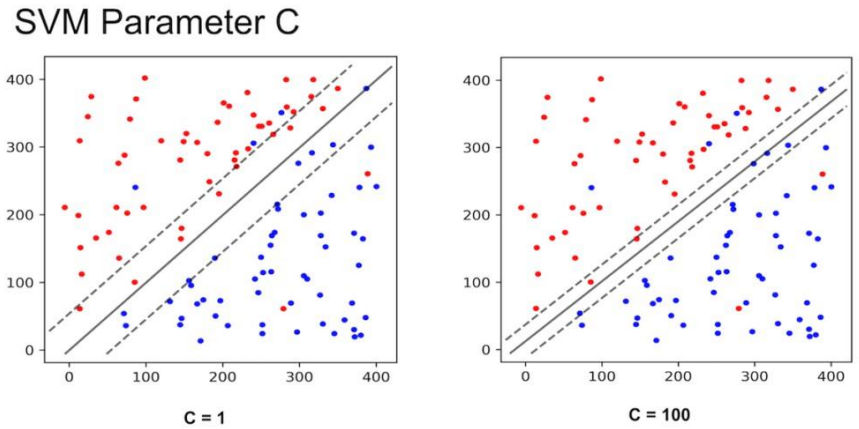
Kernel *Sigmoid*

$$K(x_i, x_j) = \tanh(x_i^T x_j + r)$$

Kernel *Radial Basis Function*

$$K(x_i, x_j) = \exp\left(-\gamma \left|\left|x_i^T - x_j\right|\right|^2\right), \quad \gamma > 0$$

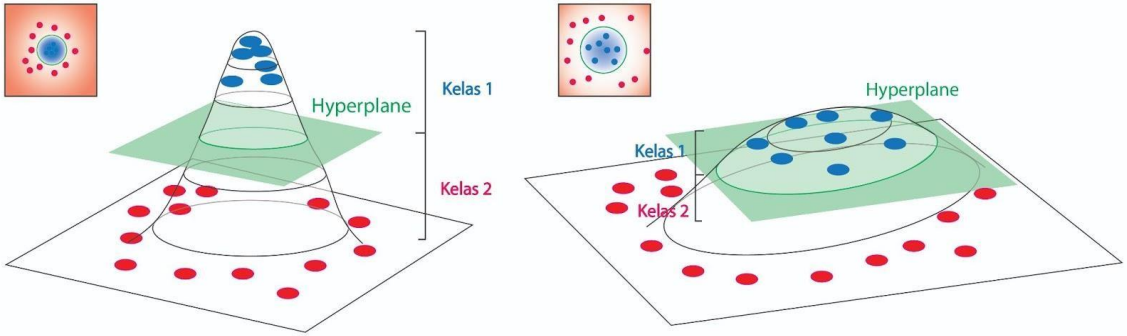
Parameter C pada Support Vector Machine

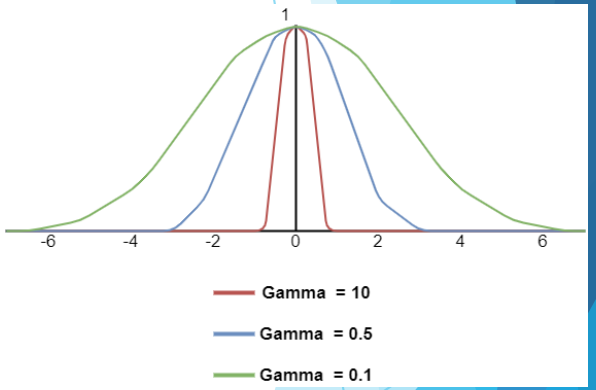


Sumber: https://learnopencv.com/svm-using-scikit-learn-in-python/

- Nilai C yang terbaik harus dicari agar SVM terjadi keseimbangan antara jarak semaksimal dan kesalahan klasifikasi seminimal mungkin.
- Nilai C yang tinggi = model akan menerima lebih banyak penalti ketika model gagal mengklasifikasikan data dengan tepat.
- Nilai C yang rendah = model akan mentolerir data yang salah diklasifikasikan.
- Nilai C yang rendah biasanya baik digunakan pada data yang memiliki banyak noise, sebaliknya nilai C yang tinggi biasanya baik digunakan pada data yang memiliki sedikit noise.

Parameter γ pada Support Vector Machine





Nilai γ yang besar akan menghasilkan kelengkungan yang tajam pada ruang dimensi tinggi. Sebaliknya, nilai γ yang kecil akan menghasilkan ruang dimensi tinggi yang lebih landai.

Hands On

Parameter	Keterangan	Contoh Nilai
kernel	Memilih kernel untuk SVM	- 'linear' untuk linear kernel - 'poly' untuk polynomial kernel - 'rbf' untuk radial basis function kernel - Nilai default : rbf
C	Parameter regularisasi SVM untuk semua kernel	- Bilangan Float positif (…, 0.1, 0.5, 1.0, 1.5 , …) - Nilai default = 1.0
gamma	Parameter koefisien kernel untuk 'poly' dan 'rbf'	- Bilangan Float positif (…, 0.001, 0.01, 0.1, 1, …)
degree	Derajat polynomial untuk kernel 'poly')	- Bilangan Integer (2,3,4,….) - Nilai default : 3

```
# Import Support Vector Machine Classifier dari sklearn
from sklearn.svm import SVC

svmLinear = SVC(
        kernel = 'linear',
        C = 1
      )

svmPoly = SVC(
        kernel = 'poly',
        C = 1,
        gamma = 0.01,
        degree = 2
      )

svmRBF = SVC(
        kernel = 'rbf',
        C = 1,
        gamma = 0.01
      )
```

Hands On (lanjutan)

```
[ ]  from sklearn.svm import SVC
     from sklearn import metrics

     svm = SVC(
       kernel = 'rbf',
       C = 1,
       gamma = 0.01
     )

     svm.fit(X_train, y_train)
     y_pred = svm.predict(X_test)
     score = metrics.accuracy_score(y_test, y_pred)
     print("Akurasi dengan menggunakan Support Vector Machine: ", score)

     Akurasi dengan menggunakan Support Vector Machine:  0.9333333333333333
```

→ Support Vector Machine

→ Proses training data latih, prediksi data uji, perhitungan akurasi

Output akurasi yang dihasilkan oleh *Naïve Bayes Classifier*

Kelebihan dan Kekurangan Support Vector Machine

- SVM menggunakan strategi kernel sehingga dapat mengatasi data yang kompleks.
- Outlier dapat diatasi menggunakan soft margin constant C.
- Hyper parameter dan kernel perlu di-tuning untuk mendapatkan akurasi yang diharapkan
- Waktu training yang lebih lama untuk dataset yang lebih besar

Boosting Algorithm

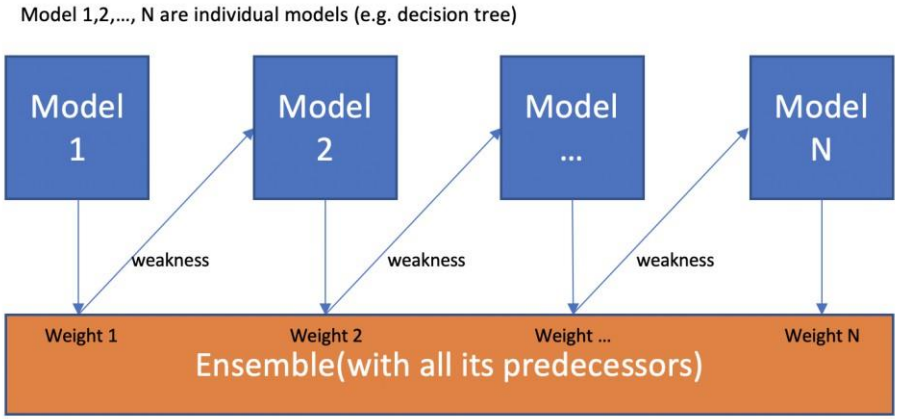


Image Source : https://towardsdatascience.com/boosting-algorithms-explained-d38f56ef3f30

- Weak Model menjadi Strong Model
- Mengkombinasikan hasil prediksi dari setiap weak model dengan metode
  - Average/Weighted Average
  - Higher voting
- Tipe Boosting
  - **AdaBoost (Adaptive Boosting)**
  - Gradient Tree Boosting
  - XGBoost

AdaBoost (Adaptive Boosting)

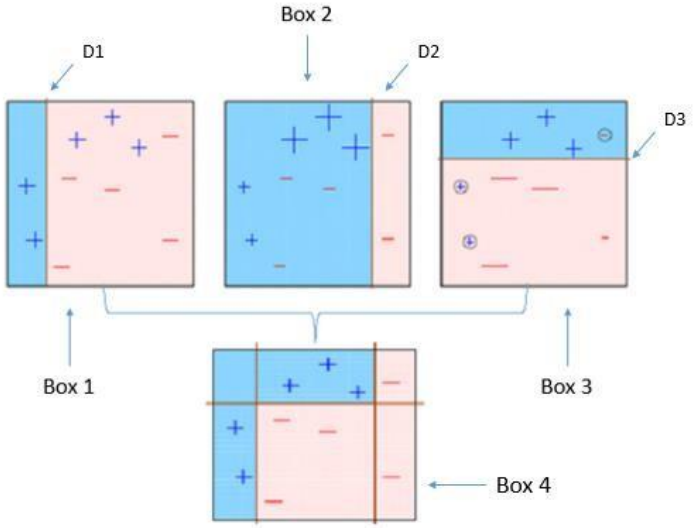


Image Source :
https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning

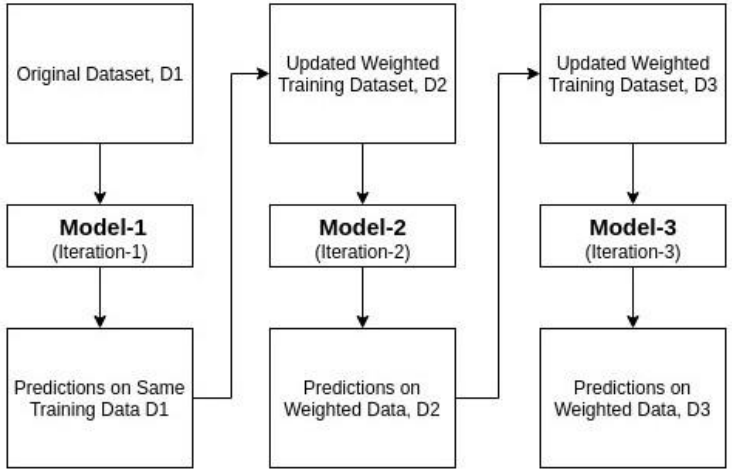


Image source :
https://www.datacamp.com/community/tutorials/adaboost-classifier-python

- AdaBoost mengidentifikasi miss-classified data
- Memberikan bobot lebih terhadap miss-classified data, sehingga classifier berikutnya lebih concern terhadap hal tersebut

Hands On

```
# Import AdaBoost Classifier dari sklearn
from sklearn.ensemble import AdaBoostClassifier

# Import metric untuk memeriksa akurasi
from sklearn import metrics

ab = AdaBoostClassifier(
        n_estimators = 50,
        learning_rate = 1
      )

ab.fit(X_train, y_train)
y_pred = ab.predict(X_test)
score = metrics.accuracy_score(y_test, y_pred)
print(score)
```

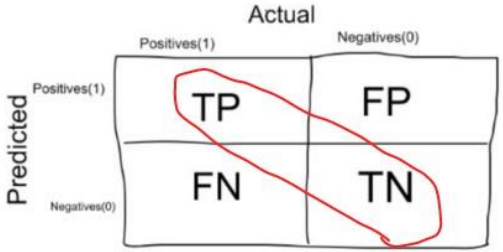
AdaBoostClassifier
Secara default,
classifier yang di boosting adalah Decision Tree

Hands On (lanjutan)

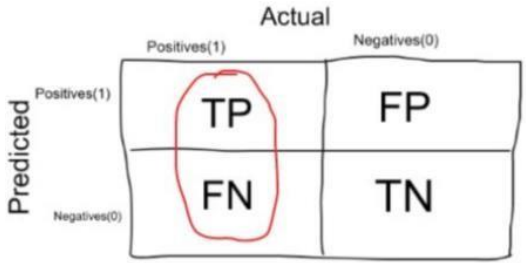
Parameter	Keterangan	Contoh Nilai
n_estimators	Jumlah weak learner yang digunakan	- Bilangan Integer (…,10,11,12,…) - Nilai default = 50
learning_rate	Bobot setiap classifier untuk setiap iterasi boosting	- Bilangan Float positif (…, 0.1, 0.5, 1.0, 1.5 , …) - Nilai default = 1.0

```
# Import AdaBoost Classifier dari sklearn
from sklearn.ensemble import AdaBoostClassifier

ab = AdaBoostClassifier(
        n_estimators = 24, # menentukan banyaknya weak learner sebanyak 24
        learning_rate = 1 # menentukan learning rate sebesar 1
      )
```

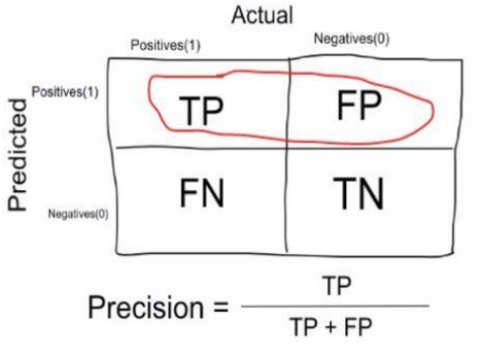
Matriks Performansi Klasifikasi



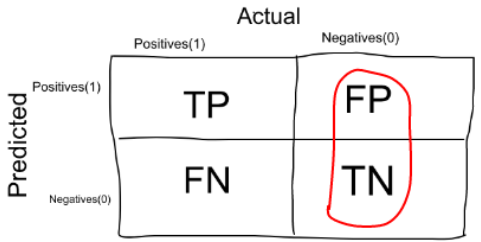
$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$



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



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



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

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Referensi

- CM Bishop, 2006, Pattern Recognition and Machine Learning, Springer
- https://medium.com/the-owl/k-fold-cross-validation-in-keras-3ec4a3a00538
- https://learnopencv.com/svm-using-scikit-learn-in-python/
- https://towardsdatascience.com/boosting-algorithms-explained-d38f56ef3f30
- https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning
- https://www.datacamp.com/community/tutorials/adaboost-classifier-python
- https://medium.com/@MohammedS/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d432082b

Tools /Lab Online

- Jupyter Notebook
- Google Collabs

Summary

- Perancangan skenario eksperimen merupakan langkah yang dilakukan sebelum membangun model
- Perancangan scenario eksperimen terdiri dari beberapa item :
  - Pembagian Data
  - Strategi langkah eksperimen
  - Parameter evaluasi yang digunakan
- Model klasifikasi dibangun dengan beberapa pilihan algoritma
- Setiap algoritma klasifikasi memiliki kelebihan dan kekurangan
- Perlu dilakukan eksperimen yang komprehensif untuk mendapatkan model yang terbaik bagi suatu kasus/dataset

Tugas Harian

- Gunakan dataset Iris (https://archive.ics.uci.edu/ml/datasets/iris)
- Data dibagi dengan proporsi :
  - Data Latih (70%)
  - Data Uji (30%)
- Gunakan metode klasifikasi dengan *parameter tuning* untuk mendapatkan
- akurasi terbaik
- Terapkan Adaboost algorithm dan lakukan analisa perbandingan terhadap
- performa yang dihasilkan dari metode klasifikasi sebelumnya

TERIMA KASIH