DigitalSkola
Uncover The World of Digital Skills with Us

Classification I

Classification II

Regression

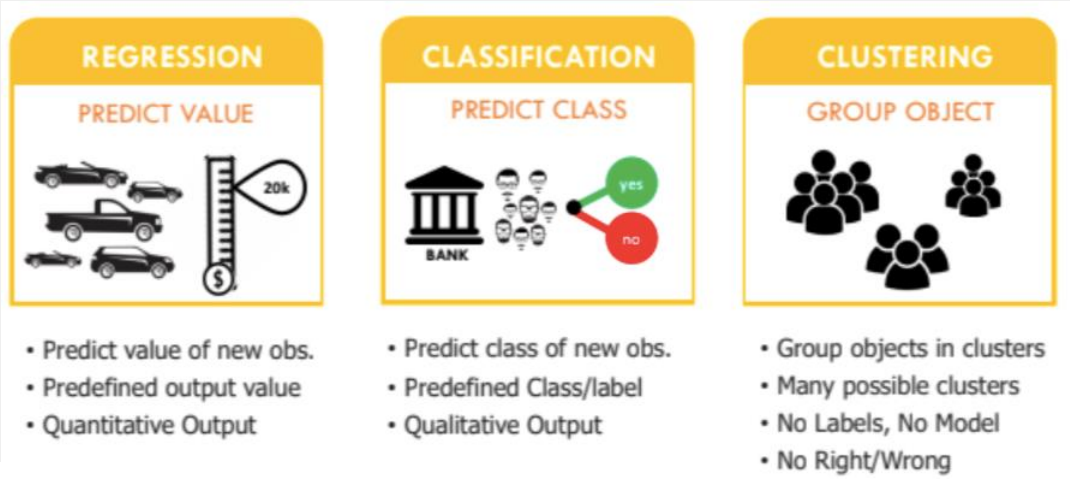
Learning Progress Review Week 13

By:
MARVEL TEAM
Fikrie | Natalia | Satria

1. *Classification 1*

Pengenalan Klasifikasi

Classification dapat didefinisikan sebagai proses memprediksi kelas atau kategori dari nilai yang sudah di amati atau titik data yang diberikan. Contoh kasus klasifikasi yang sering kita jumpai adalah deteksi *spam di email.*

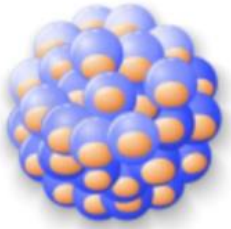


Pengenalan Klasifikasi

DigitalSkola
Uncover The World of Digital Skills with Us

Classification Case

Fraud Detection



Cancer Cell Classification

Credit Scoring

Spam Detector

Churn Analysis

K-Nearest Neighbour

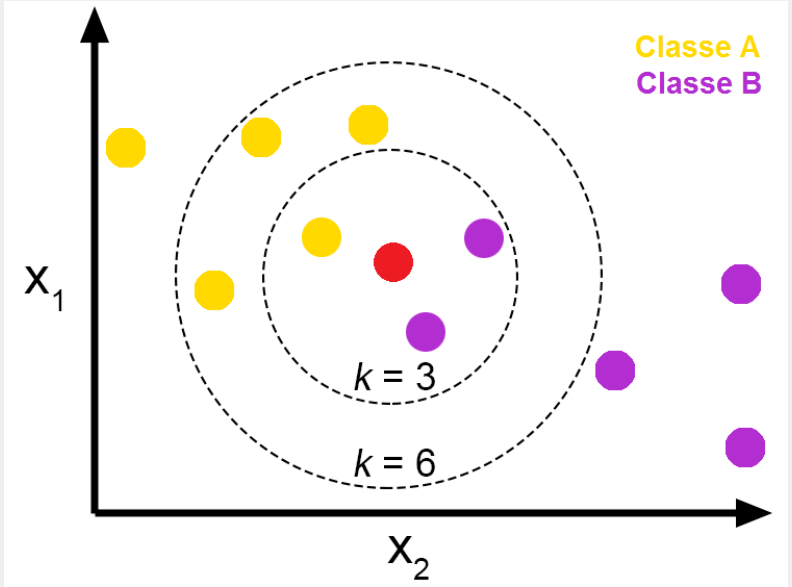
K-nearest neighbors atau **KNN** adalah algoritma yang berfungsi untuk melakukan klasifikasi suatu data berdasarkan data pembelajaran (*train data sets*), yang diambil dari k tetangga terdekatnya (*nearest neighbors*). Dengan k merupakan banyaknya tetangga terdekat.

Teknik pencarian tetangga terdekat yang umum dilakukan dengan menggunakan formula jarak *Euclidean*. Untuk menggunakan algoritma k nearest neighbors, perlu ditentukan banyaknya k tetangga terdekat yang digunakan untuk melakukan klasifikasi data baru.

DigitalSkola
Uncover The World of Digital Skills with Us

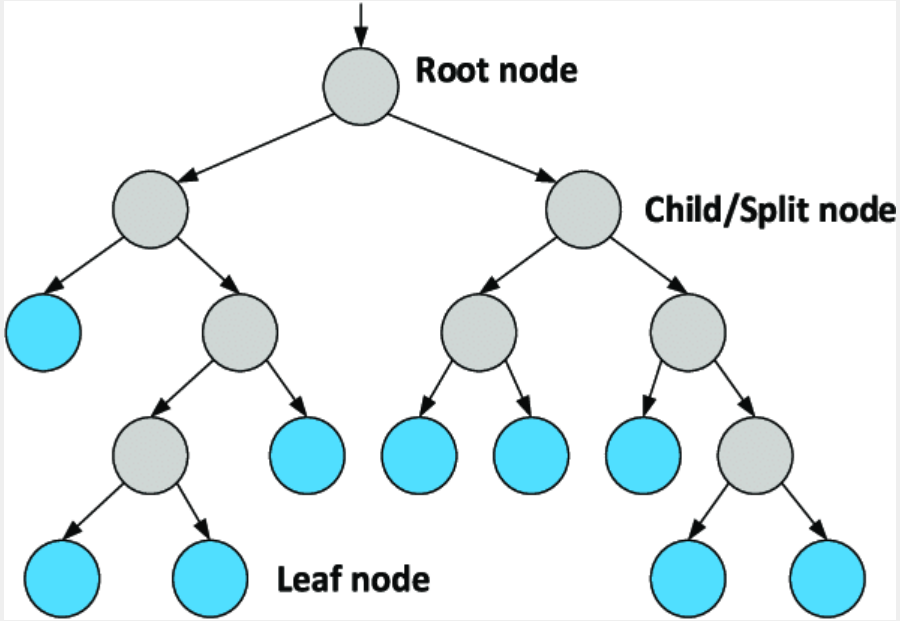
Algoritma KNN

- Tentukan K bilangan bulat positif berdasarkan ketersediaan data pembelajaran.
- Pilih tetangga terdekat dari data baru sebanyak k.
- Tentukan klasifikasi paling umum pada Langkah (2), dengan menggunakan frekuensi terbanyak.
- Keluaran klasifikasi dari data sampel baru.



Decision Tree

Decision Tree adalah sebuah diagram alir yang berbentuk seperti struktur pohon yang mana setiap internal node menyatakan pengujian terhadap suatu attribute, setiap cabang menyatakan output dari pengujian tersebut dan leaf node menyatakan kelas-kelas atau distribusi kelas.



DigitalSkola
Uncover The World of Digital Skills with Us

Formula *Information Gain*

$$Gain(A) = I(s_1, s_2, .., s_m) - E(A)$$ .....(Rumus 2.1.1)

$I(s_1, s_2, ..., s_m)$ adalah informasi harapan (split info) yang memiliki rumus :

$$I(s_1, s_2, ..., s_m) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$ .....(Rumus 2.1.2)

Dengan

m : Banyaknya nilai yang berbeda atribut label kelas yang akan mendefinisikan kelas yang berbeda, C_i $(i = 1,2,...,m)$

s_i : Jumlah sampel dalam himpunan sampel S (berisi s sampel) yang masuk kelas C_i

p_i : Peluang bahwa suatu sampel akan masuk ke kelas C_i dan diestimasi dengan $\frac{s}{si}$

DigitalSkola
Uncover The World of Digital Skills with Us

Formula *Entropy*

E(*A*) : *Entropy* A

Secara statistik, *entropy* menyatakan ukuran ketidakpastian secara probabilistik.

Entropy A memiliki rumusan :

$$E(A) = \sum_{j=1}^{v} \frac{s_{1j} + s_{2j} + \ldots + s_{mj}}{s} I(s_{1j}, s_{2j}, \ldots, s_{mj})$$

.....*(Rumus 2.1.3)*

v : Banyaknya nilai/kategori yang berbeda yang dimiliki atribut A

s_{ij} : Banyaknya sampel pada atribut A yang masuk kategori ke j dan kelas C_i

$\frac{s_{1j} + s_{2j} + \ldots + s_{mj}}{s}$ menyatakan proporsi jumlah sampel atribut A kategori j terhadap jumlah sampel total

Logistic Regression

Sebelum kita mempelajari ***Logistic Regression***, alangkah baiknya kita mengetahui ***Linear Regression*** terlebih dahulu.

Apasih *Linear Regression*?. *Linear Regression* adalah suatu cara permodelan masalah keterhubungan antara suatu variabel *independen*t terhadap variabel *dependen*.

Logistic Regression adalah sebuah algoritma klasifikasi untuk mencari hubungan antara fitur(input) diskrit/kontinu dengan probabilitas hasil output diskrit tertentu.

Tipe-tipe *Logistic Regression*:

- ***Binary Logistic Regression*** : adalah *Logistic Regression* yang hanya memiliki 2 output saja.
- ***Multinasional Logistic Regression*** : adalah *Logistic* yang memiliki 2 output atau lebih.
- ***Ordinal Logistic Regression*** : adalah *Logistic Regression* yang memiliki 2 output atau lebih dengan memperhatikan urutan.

Logistic Regression

DigitalSkola
Uncover The World of Digital Skills with Us

Rumus
Logistic Regression

LINEAR FUNCTION

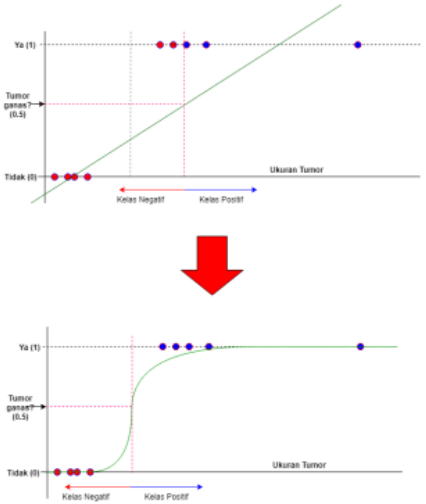
$$Y = b_0 + b_1 * X$$

SIGMOID FUNCTION

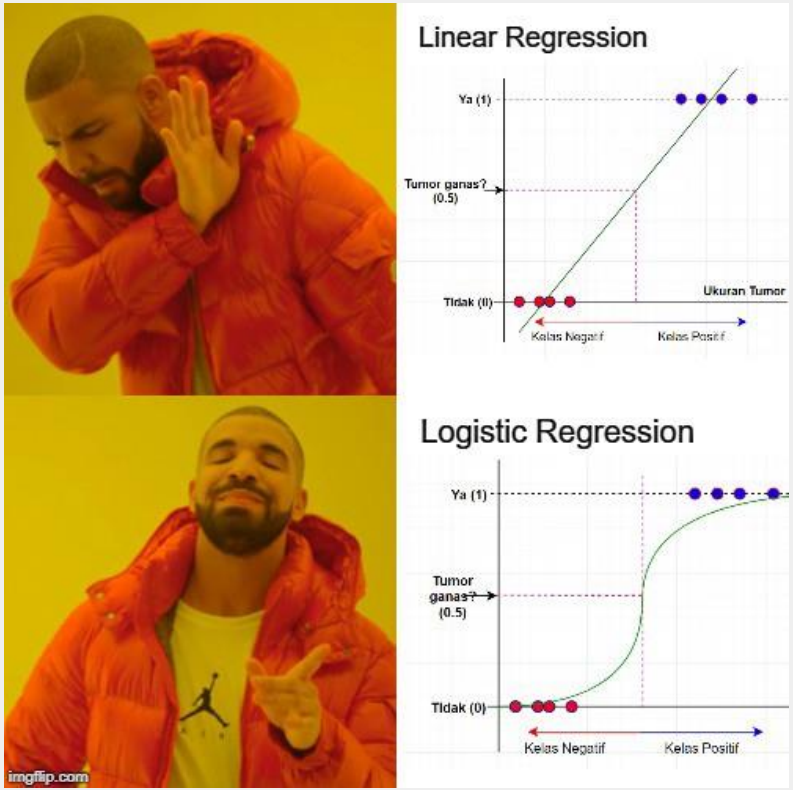
$$P = \frac{1}{1 + e^{-Y}}$$

LOGISTIC FUNCTION

$$P = \frac{1}{1 + e^{-(b0 + b1 * X)}}$$

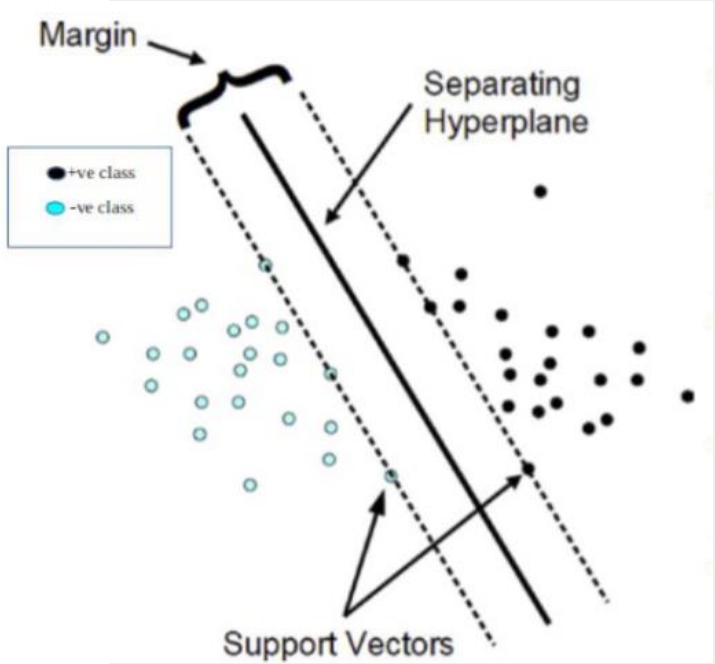


Logistic Regression



2. Classification II

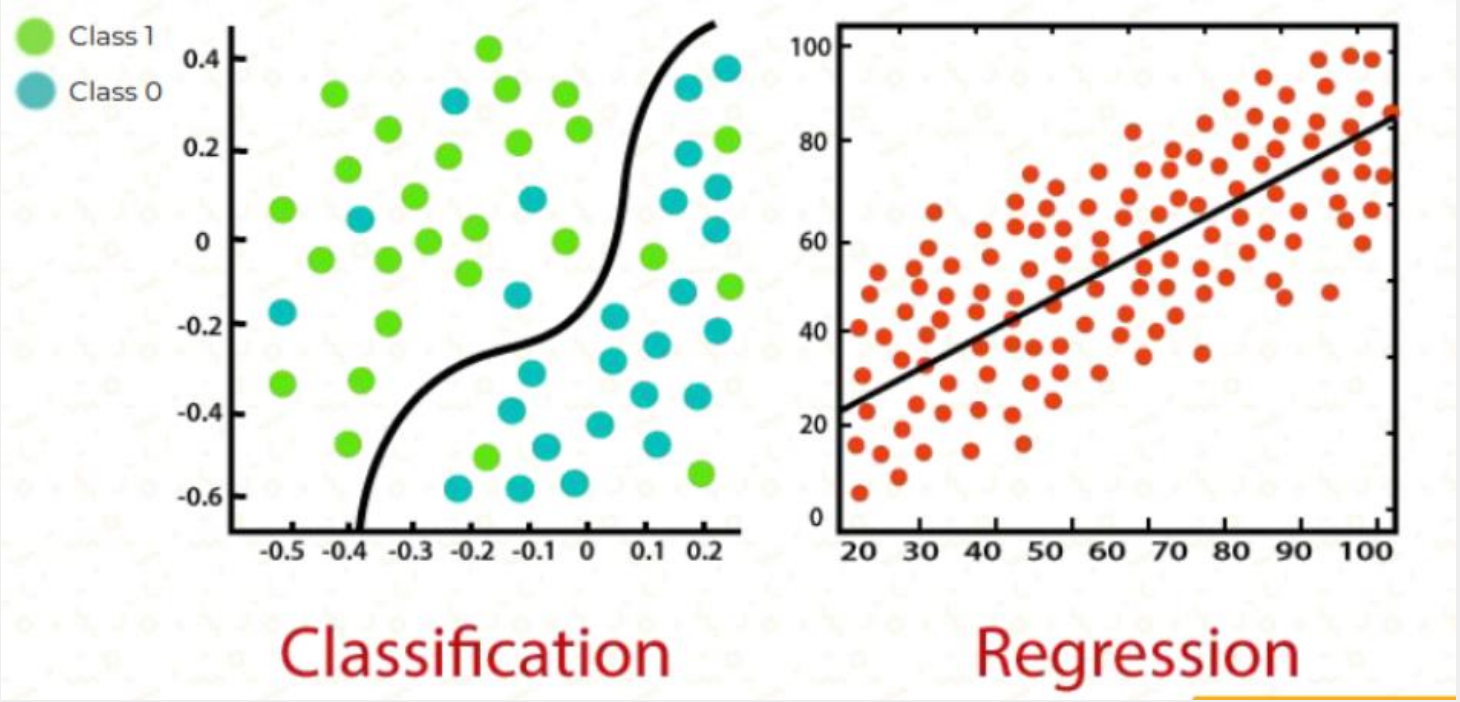
SVM : Support Vector Machine



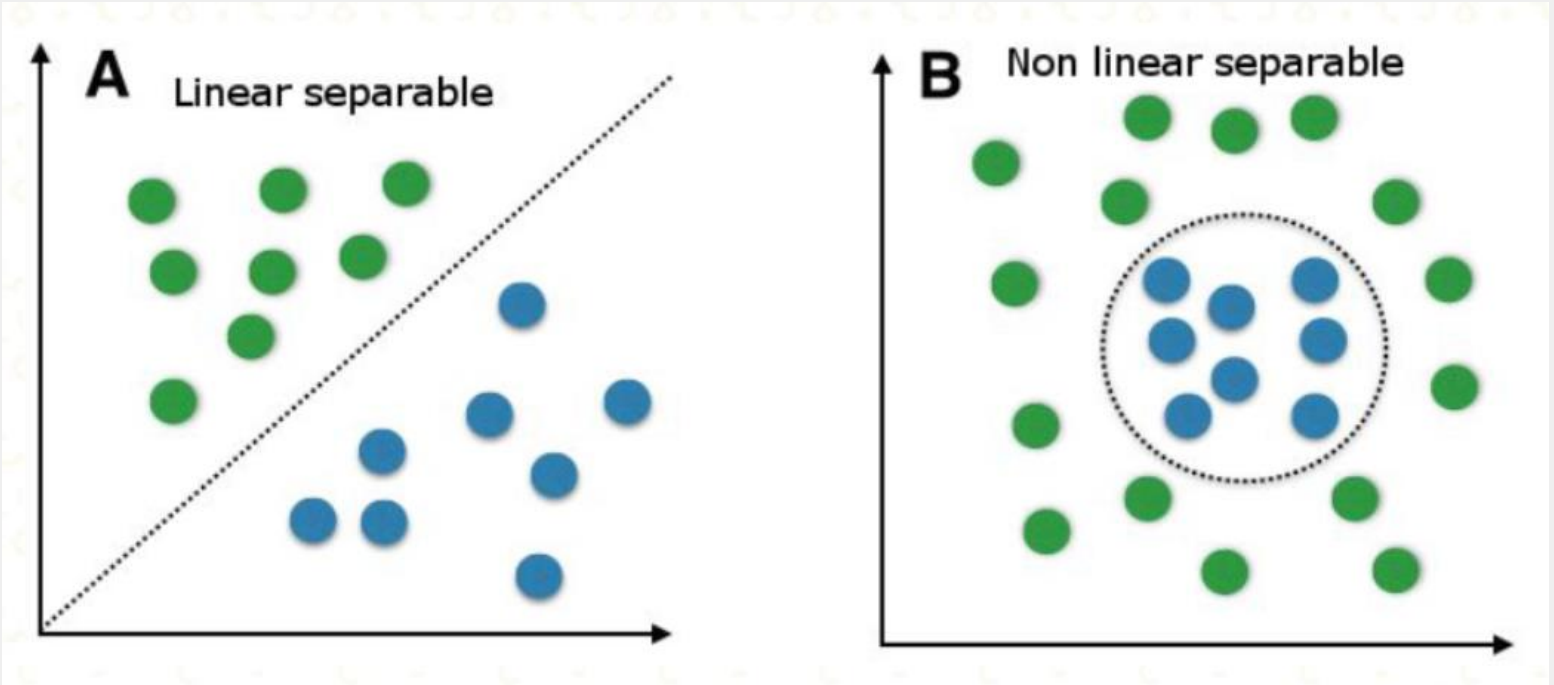
SVM adalah suatu teknik untuk melakukan prediksi, baik dalam kasus regresi maupun klasifikasi.

Penggunaan SVM untuk mendapatkan fungsi pemisah *(hyperplane)* untuk memisahkan observasi yang memiliki nilai variabel yang berbeda.

SVM di Klasifikasi & Regresi



SVM (Linear & Non-Linear)



Pros & Cons SVM

Keunggulan :

- Teknik ini sangat baik digunakan pada data yang memiliki margin pemisah antar kelas sangat jelas.
- Efektif untuk dimensi data yang sangat tinggi (*feature* yang cukup banyak).
- Efektif pada data yang memiliki dimensi lebih banyak daripada jumlah sampel.

Kelemahan :

- Kurang baik pada dataset yang terlalu besar dan banyak.
- Kurang baik pada dataset yang memiliki banyak *noise*.
- SVM tidak menyediakan hasil probabilitas.

Kegunaan SVM

- ***Handwriting Recognition***
- ***Intrusion Detection***
- ***Face Detection***
- ***Email Classification***
- ***Gene Classification***

Ensemble

- Metode penerapan *Machine Learning* yang mampu menggabungkan beberapa model.
- Mudah untuk melakukan *tuning hyperparameter.*
- Banyak memenangkan kompetisi *Machine Learning* di Kaggle.
- Mempunyai kemampuan memprediksi data sangat tepat.

Pendekatan *Ensemble*

- ***Bagging***
- ***Random Forest***
- ***Boosting (Adaboost, Gradient Boosting, XGBoost)***
- ***Stacking***

Metode *Ensemble*

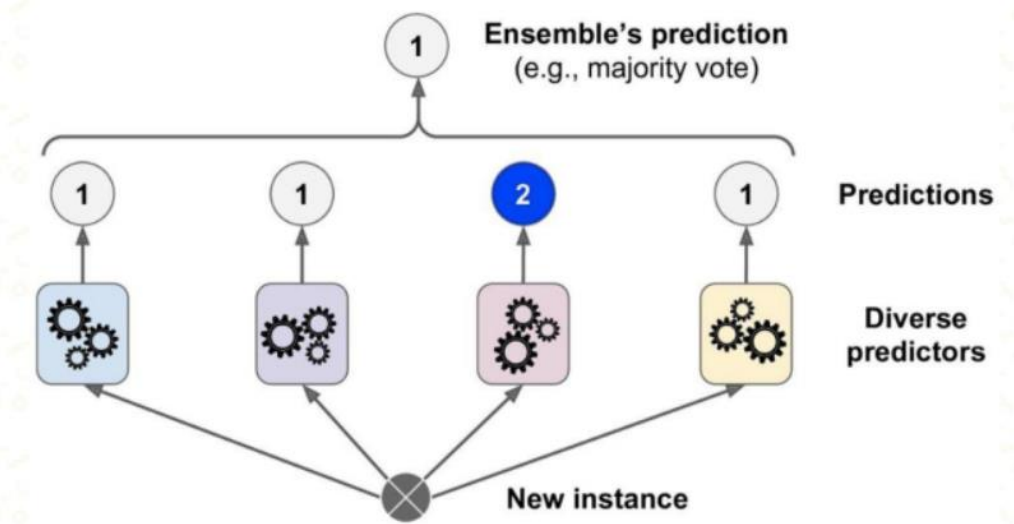
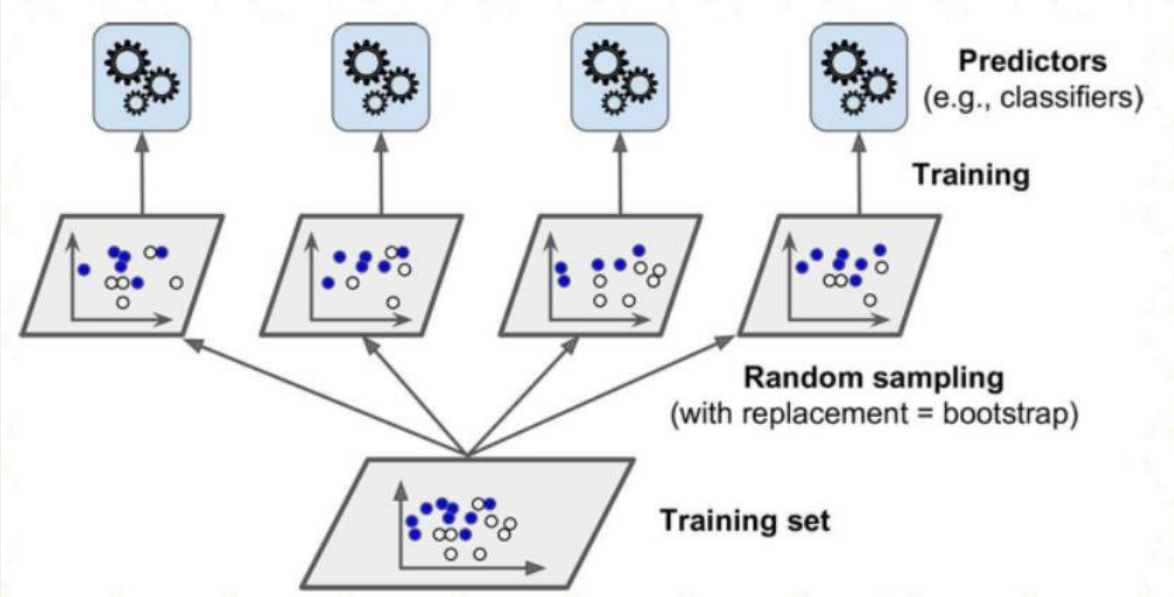


Figure 7-2. Hard voting classifier predictions

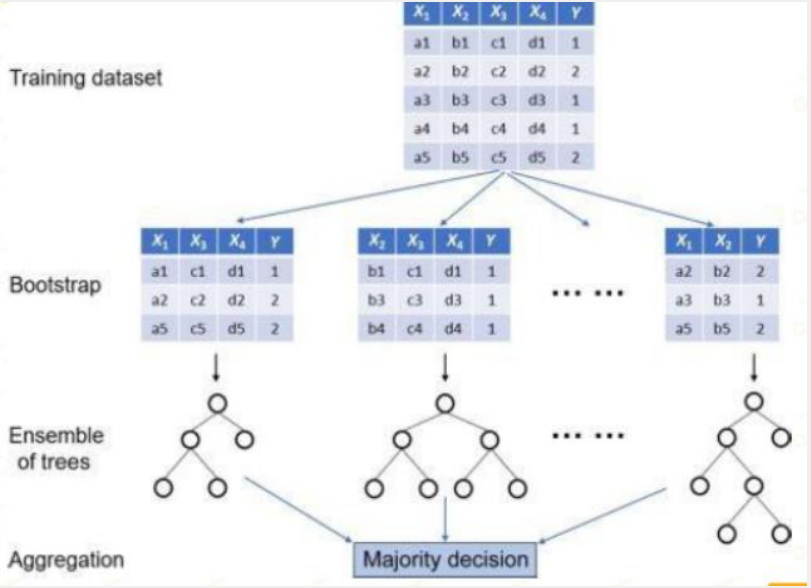
Dataset ditraining dengan beberapa model. Jika ada dataset baru, akan dimasukkan ke masing – masing model. Hasil prediksi akan dihitung tingkat akurasinya. Model dengan akurasi tertinggi itulah yang akan digunakan untuk memprediksi data.

Bagging – Traditional Method



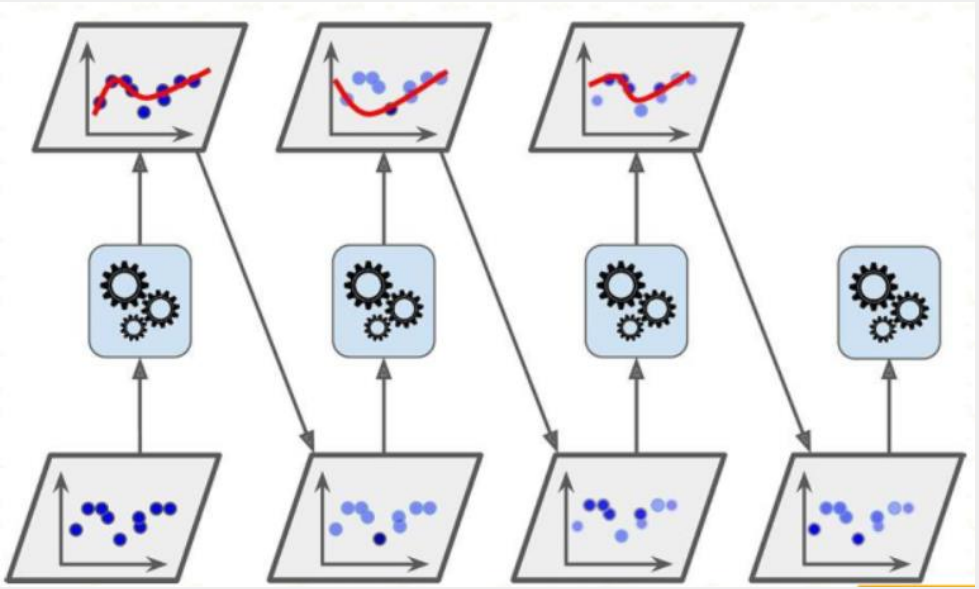
Dataset ditraining dengan satu model yang sama. Model ini akan diberikan beberapa dataset yang telah *dirandom sampling* dengan pendekatan *boostrap*. Dari dataset yang diolah akan diukur akurasinya. Model dengan akurasi tertinggi yang akan dipakai.

Bagging – Random Forest



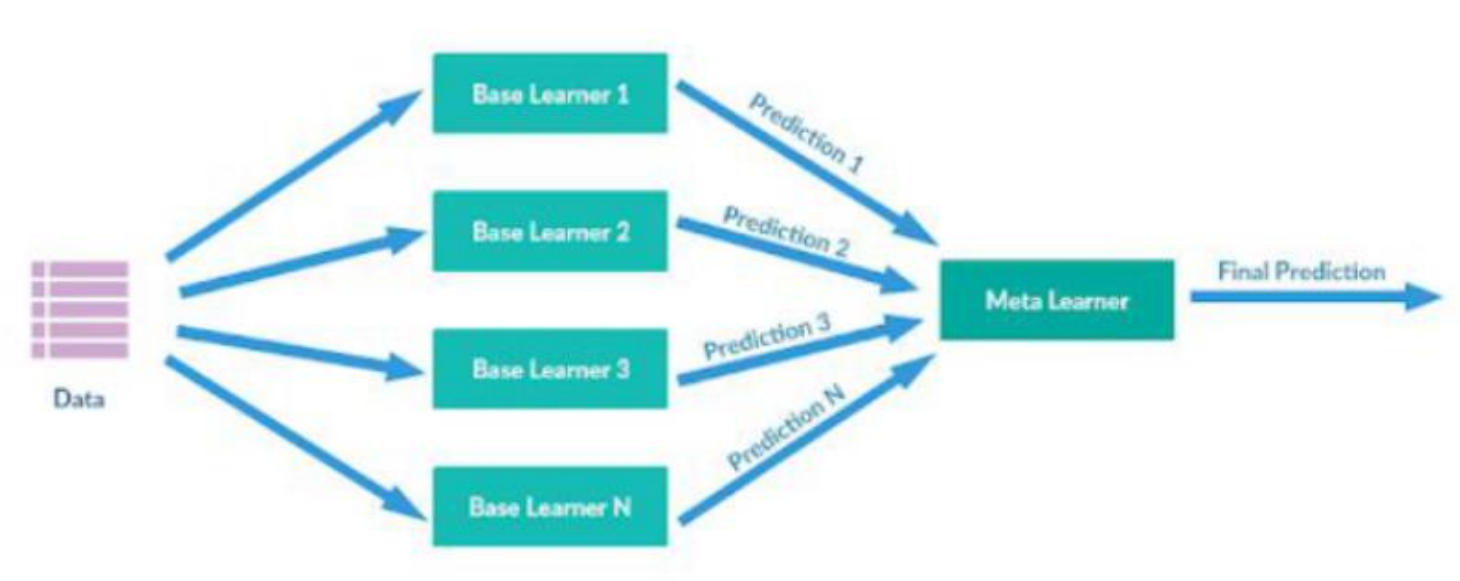
Dataset ditraining dengan satu model yang sama. Model ini akan diberikan beberapa dataset yang telah *dirandom sampling* dengan pendekatan *boostrap (feature selection)*. Dari dataset yang diolah akan diukur akurasinya. Model dengan akurasi tertinggi itulah yang akan dipakai.

Boosting



Model seolah-olah belajar dari training data yang belum optimal. Model terus belajar dari data training sebelumnya sampai diperoleh model yang paling optimal.

Stacking



Data memiliki beberapa model yang akan dipakai untuk memprediksi dan menghasilkan data target awal. Data target awal ini yang akan digunakan sebagai *data training* untuk menghasilkan data klasifikasi akhir.

3. *Regression*

Linear Regression

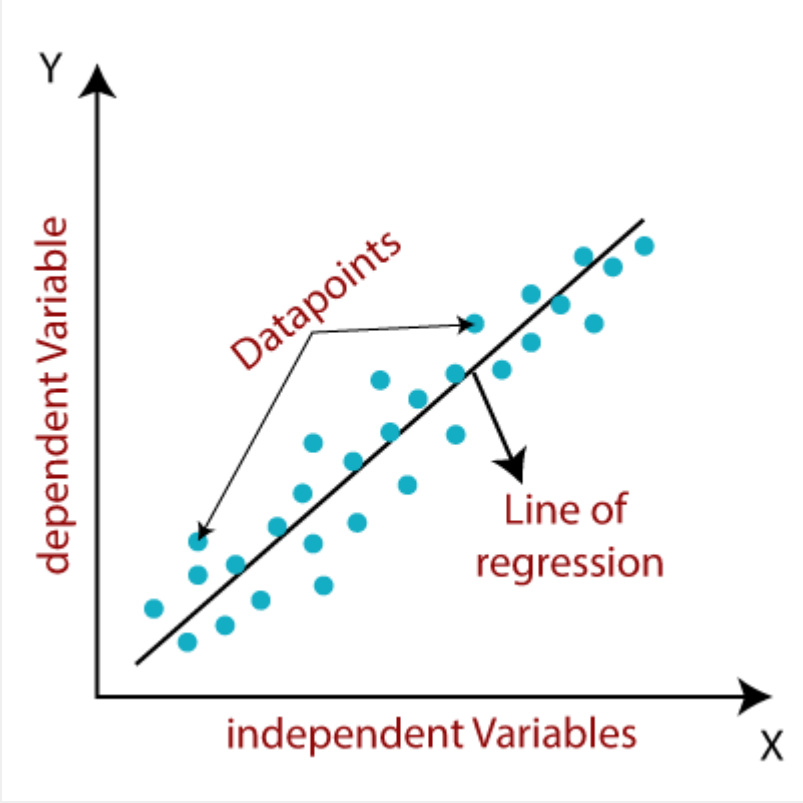
Secara matematis, **persamaan dari *Linear Regression*** adalah sebagai berikut :

$$y = mx + b + e$$

y = dependent variable
m = slope dari garis (persamaan diatas merupakan sebuah garis)
x = independent variable
b = intercept
e = error

Secara sederhana, **tujuan dari *Linear Regression*** adalah untuk memprediksi nilai dari y dengan mengetahui nilai x dan menemukan nilai m dan b yang errornya paling minimal. Karena ini merupakan sebuah prediksi, maka persamaan ditambahkan nilai error.

Linear Regression



Linear Regression di Python

```
>>> import numpy as np
>>> from sklearn.linear_model import LinearRegression
>>> X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
>>> # y = 1 * x_0 + 2 * x_1 + 3
>>> y = np.dot(X, np.array([1, 2])) + 3
>>> reg = LinearRegression().fit(X, y)
>>> reg.score(X, y)
1.0
>>> reg.coef_
array([1., 2.])
>>> reg.intercept_
3.0...
>>> reg.predict(np.array([[3, 5]]))
array([16.])
```

Lasso Regression

Lasso Regression adalah jenis analisis regresi dimana pemilihan variabel dan regulasi terjadi secara bersamaan. Metode ini menggunakan penalti yang memengaruhi nilai koefisien regresi.

Karakteristik *Lasso Regression* adalah:

- Menggunakan teknik *L1 Regularization*
- Umumnya digunakan ketika kita mempunyai jumlah fitur yang banyak, karena *Lasso Regression* melakukan *feature selection* secara otomatis.

Lasso Regression di Python

DigitalSkola
Uncover The World of Digital Skills with Us

```
>>> from sklearn import linear_model
>>> clf = linear_model.Lasso(alpha=0.1)
>>> clf.fit([[0,0], [1, 1], [2, 2]], [0, 1, 2])
Lasso(alpha=0.1)
>>> print(clf.coef_)
[0.85 0.  ]
>>> print(clf.intercept_)
0.15...
```

Ridge Regression

Ridge Regression adalah metode *tuning model* yang digunakan untuk menganalisis data yang mengalami multikolinieritas (suatu kondisi adanya hubungan linier di antara variabel-variabel independen dalam model regresi). Metode ini melakukan *L2 Regularization*.

Tujuan utama dari *Ridge Regression* adalah untuk menemukan koefisien yang meminimalkan jumlah kuadrat kesalahan dengan menerapkan penalti pada koefisien tersebut.

Ridge Regression di Python

DigitalSkola
Uncover The World of Digital Skills with Us

```
>>> from sklearn.linear_model import Ridge
>>> import numpy as np
>>> n_samples, n_features = 10, 5
>>> rng = np.random.RandomState(0)
>>> y = rng.randn(n_samples)
>>> X = rng.randn(n_samples, n_features)
>>> clf = Ridge(alpha=1.0)
>>> clf.fit(X, y)
Ridge()
```

Elastic Net Regression

Elastic Net Regression adalah kompromi antara *Lasso* dan *Ridge*. Ini menggabukan penalti *Lasso* dan *Ridge* dengan menerapkan rasio.

Elastic Net Regression harus digunakan jika ingin menggabungkan fungsi *Lasso* dan *Ridge*, yang biasanya menghasilkan kinerja yang lebih baik ketika disetel dengan benar.

Elastic Net Regression di Python

DigitalSkola
Uncover The World of Digital Skills with Us

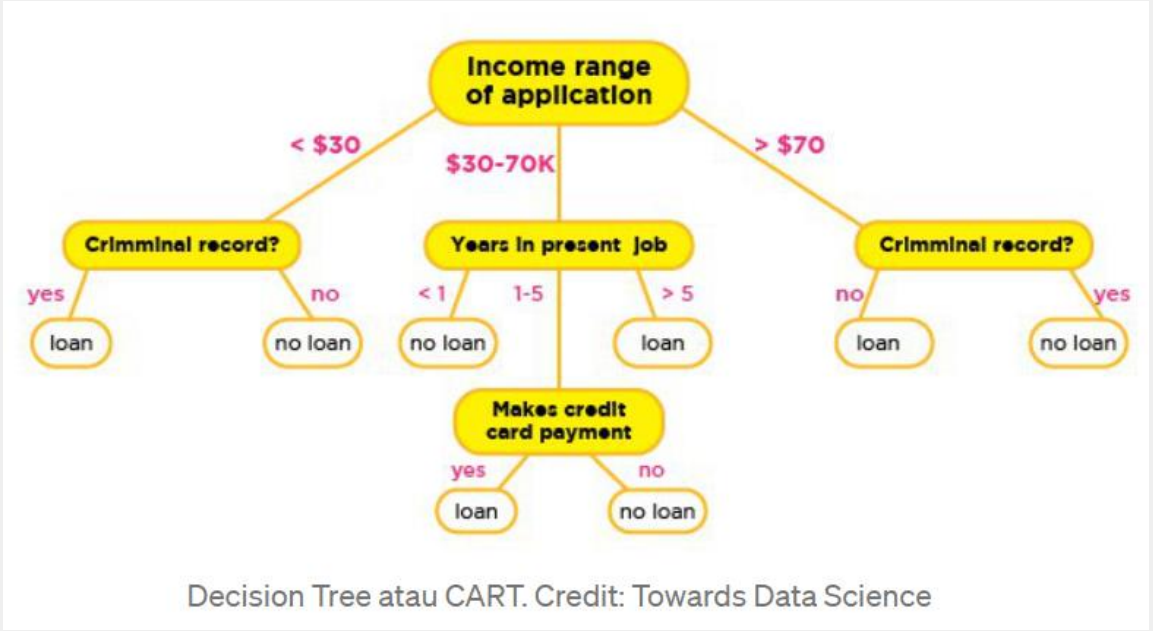
```
>>> from sklearn.linear_model import ElasticNet
>>> from sklearn.datasets import make_regression

>>> X, y = make_regression(n_features=2, random_state=0)
>>> regr = ElasticNet(random_state=0)
>>> regr.fit(X, y)
ElasticNet(random_state=0)
>>> print(regr.coef_)
[18.83816048 64.55968825]
>>> print(regr.intercept_)
1.451...
>>> print(regr.predict([[0, 0]]))
[1.451...]
```

Decision Tree Regressor

Decision Tree adalah metode non parametrik yang digunakan untuk klasifikasi dan regresi.

Tujuan dari *decision tree* adalah membuat model yang memprediksi nilai variabel target dengan mengikuti aturan keputusan sederhana dari fitur data yang tersedia.

Contoh Aplikasi *Decision Tree*



Decision Tree atau CART. Credit: Towards Data Science

Decision Tree Regressor di Python

DigitalSkola
Uncover The World of Digital Skills with Us

```
print(__doc__)

# Import the necessary modules and libraries
import numpy as np
from sklearn.tree import DecisionTreeRegressor
import matplotlib.pyplot as plt

# Create a random dataset
rng = np.random.RandomState(1)
X = np.sort(5 * rng.rand(80, 1), axis=0)
y = np.sin(X).ravel()
y[::5] += 3 * (0.5 - rng.rand(16))

# Fit regression model
regr_1 = DecisionTreeRegressor(max_depth=2)
regr_2 = DecisionTreeRegressor(max_depth=5)
regr_1.fit(X, y)
regr_2.fit(X, y)
```

```
# Predict
X_test = np.arange(0.0, 5.0, 0.01)[:, np.newaxis]
y_1 = regr_1.predict(X_test)
y_2 = regr_2.predict(X_test)

# Plot the results
plt.figure()
plt.scatter(X, y, s=20, edgecolor="black",
            c="darkorange", label="data")
plt.plot(X_test, y_1, color="cornflowerblue",
         label="max_depth=2", linewidth=2)
plt.plot(X_test, y_2, color="yellowgreen", label="max_depth=5", linewidth=2)
plt.xlabel("data")
plt.ylabel("target")
plt.title("Decision Tree Regression")
plt.legend()
plt.show()
```

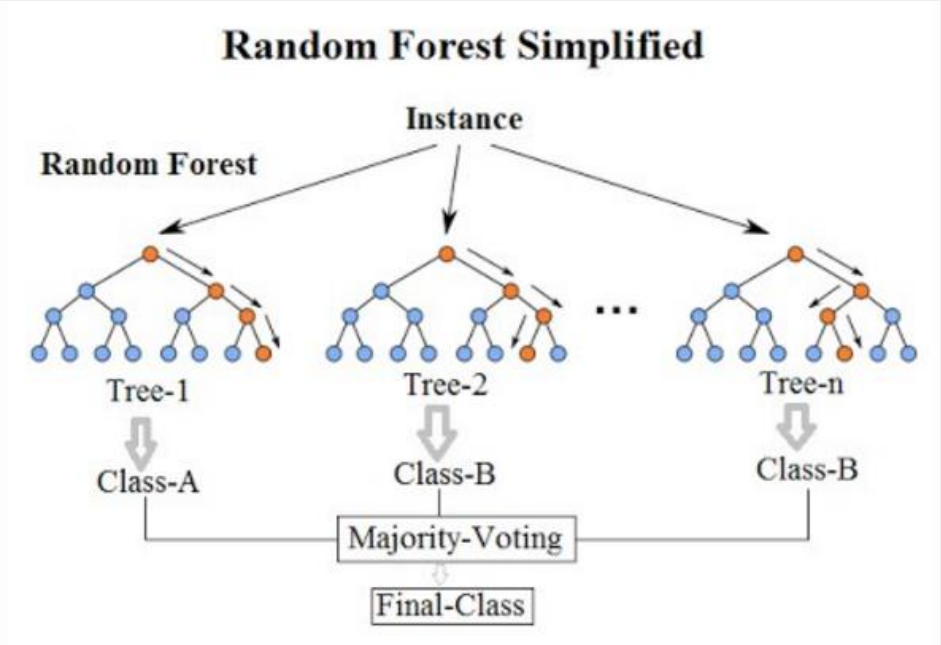
Random Forest Regressor

Random Forest adalah suatu algoritma yang digunakan pada klasifikasi data dalam jumlah yang besar.

Random forest merupakan kombinasi dari masing – masing pohon (*tree*) dari model *Decision Tree* yang baik, dan kemudian dikombinasikan ke dalam satu model.

Random Forest Regressor

Penggunaan *tree* yang semakin banyak akan mempengaruhi akurasi yang akan didapatkan menjadi lebih baik. Penentuan klasifikasi dengan random forest diambil berdasarkan hasil voting dari *tree* yang terbentuk.



Random Forest Regressor di Python

```
>>> from sklearn.ensemble import RandomForestRegressor
>>> from sklearn.datasets import make_regression
>>> X, y = make_regression(n_features=4, n_informative=2,
...                        random_state=0, shuffle=False)
>>> regr = RandomForestRegressor(max_depth=2, random_state=0)
>>> regr.fit(X, y)
RandomForestRegressor(...)
>>> print(regr.predict([[0, 0, 0, 0]]))
[-8.32987858]
```

Special Thanks to :

Slide template by SlideCarnival