Python Programming for Machine Learning

PERTEMUAN - 3

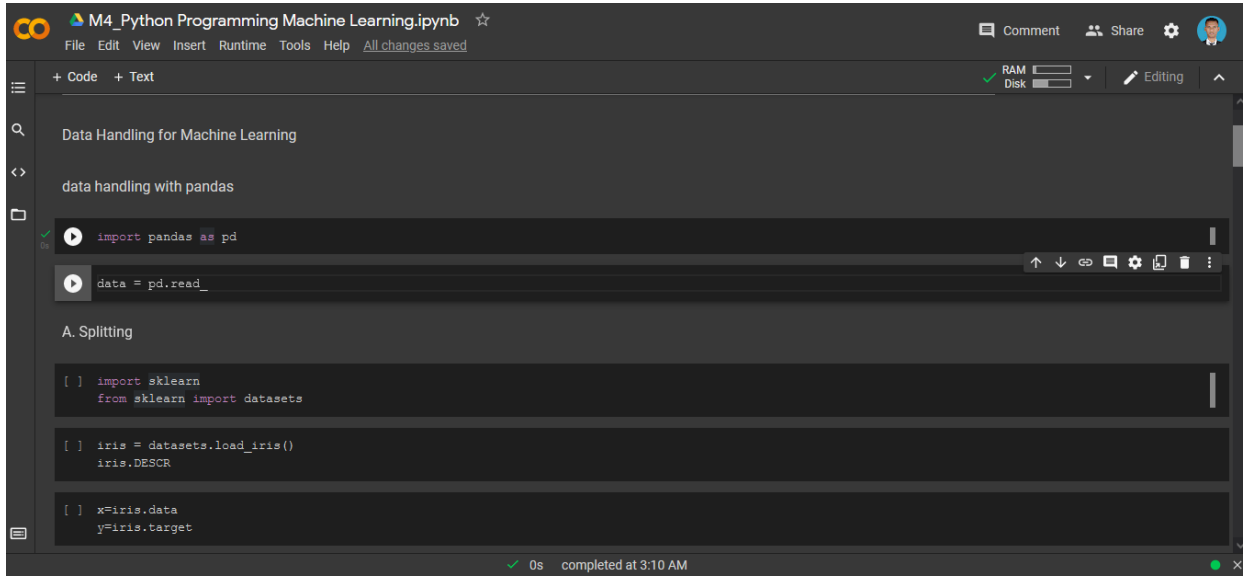
AI ACADEMY 2021

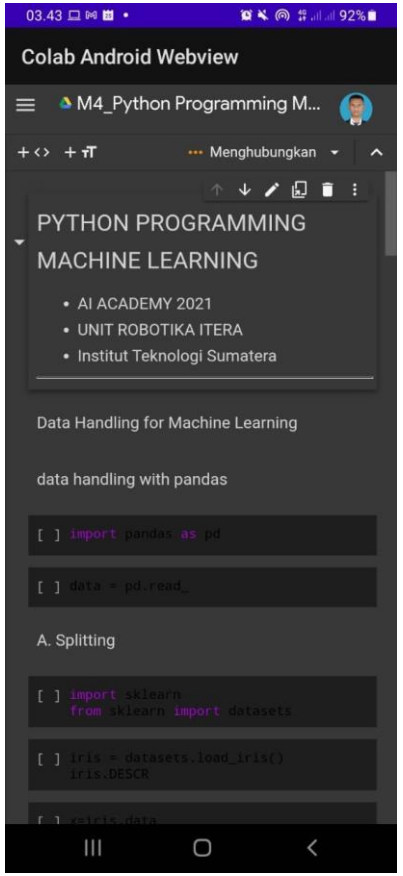
Bahasan

1. Data Retrieval
2. Data Preparation Featuring Engineering
3. Visualisasi Data
4. Modelling
5. Metode Evaluasi

Alat dan Bahan

1. Laptop atau HP Android





2. Google Colaboratory

https://colab.research.google.com

Library

numpy - is used for its N-dimensional array objects

pandas – is a data analysis library that includes dataframes

matplotlib – is 2D plotting library for creating graphs and plots

scikit-learn - the algorithms used for data analysis and data mining tasks

seaborn – a data visualization library based on matplotlib

Keras – a deep learning library

Tensorflow – a deep learning library

Types of Learning

Supervised (inductive) learning

- Training data includes desired outputs

Unsupervised learning

- Training data does not include desired outputs

Semi-supervised learning

- Training data includes a few desired outputs

Reinforcement learning

- Rewards from sequence of actions

Data Retrieval

Pandas DataFrame

```
matrix_data = np.random.randint(1,20,size=20).reshape(5,4)
row_labels = ['A','B','C','D','E']
column_headings = ['W','X','Y','Z']

df = pd.DataFrame(data=matrix_data, index=row_labels, columns=column_headings)
print("\nThe data frame looks like\n",'-'*45, sep='')
print(df)
```

```
The data frame looks like
---------------------------------------------
    W   X   Y   Z
A   3   1   7   1
B  14   1   2  12
C  14   4  16  11
D   4  11  18  15
E   1  11  11   3
```

```
d={'a':[10,20],'b':[30,40],'c':[50,60]}
df2=pd.DataFrame(data=d,index=['X','Y'])
print(df2)
```

```
    a   b   c
X  10  30  50
Y  20  40  60
```

Data Handling dengan Pandas

```
>>> import pandas as pd
>>> df = pd.read_csv('https://archive.ics.uci.edu/ml/'
...                  'machine-learning-databases/iris/iris.data',
...                  header=None)
>>> df.tail()
```

	0	1	2	3	4
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

Membaca data dengan pandas

1. df = pd.read_csv("./Data/wine.data.csv")
2. df = pd.read_excel("./Data/Height_Weight.xlsx")
3. df = pd.read_json("./Data/Height_Weight.json")

Data Handling dengan Pandas

Quick checking DataFrames

- `.head()`
- `.tail()`
- `.sample()`
- `.info()`
- `.describe()`

Sintaks

data.head()

Data Handling dengan Pandas

Basic descriptive statistics on a DataFrame

- `mean()`
- `std()`
- `var()`
- `min()` and `max()`

Sintaks

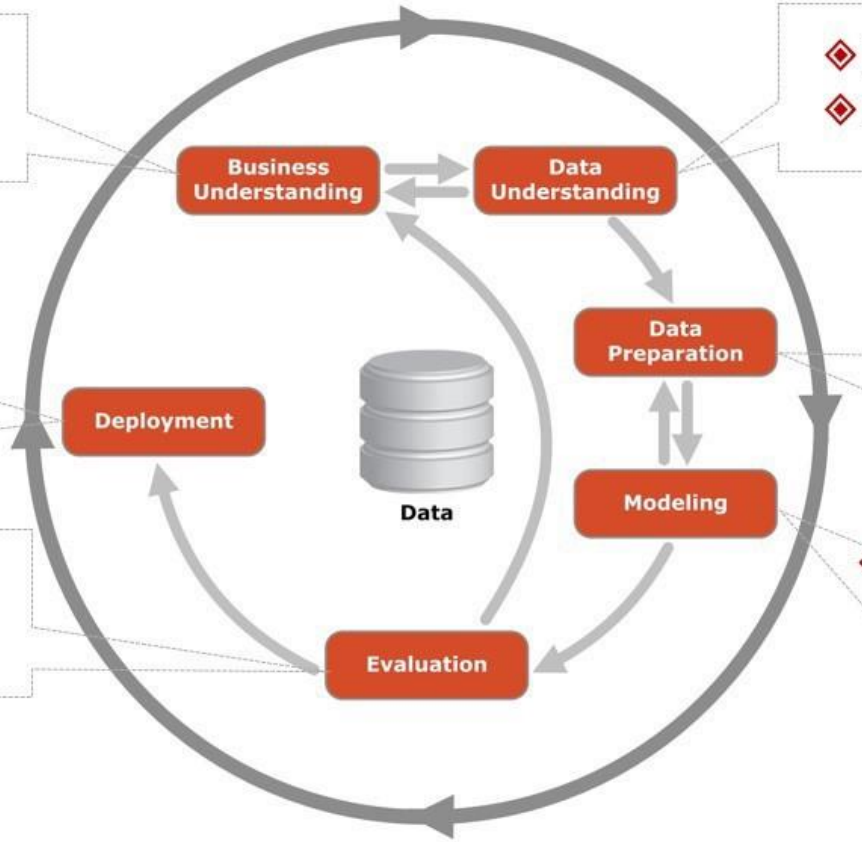
data.mean()

FEATURE ENGINEERING

CRISP-DM Framework

- Project Objective
- Business Objective

- Data Exploration
- Data Selection



- UAT
- Automation

- Feature Engineering

- Blind Test
- Evaluation Metrics

- Predicting Potential Customers using Machine Learning algorithms

a. Feature Extraction and Feature Engineering

Example:

- Texts(ngrams, word2vec, tf-idf etc)
- Images(CNN'S, texts, q&a)
- Geospatial data(lat, long etc)
- Date and time(day, month, week, year, rolling based)
- Time series, web, etc
- Dimensional Reduction Techniques (PCA, SVD, Eigen-Faces etc)
- Maybe we can use Clustering as well (DBSCAN etc)
- .....(And Many Others)

b. Feature Transformations

Example:

- Standardization
- Normalization and changing distribution(Scaling)
- Interactions
- Filling in the missing values(median filling etc)
- .....(And Many Others)

c. Feature Selection

Example:

- Statistical approaches
- Selection by modeling
- Grid search
- Cross Validation
- .....(And Many Others)

Example:

- Imputation
- Handling Outliers
- Binning
- Log Transform
- One-Hot Encoding
- Grouping Operations
- Feature Split
- Scaling
- Extracting Date

Example

Imputation

```
import numpy as np
from numpy import nan

X = np.array([[ nan, 0,   3  ],
              [ 3,   7,   9  ],
              [ 4,   5,   2  ],
              [ 5,   nan, 6  ]])
```

```
from sklearn.impute import SimpleImputer
imp = SimpleImputer(strategy='mean')
X2 = imp.fit_transform(X)
X2
```

```
array([[4., 0., 3.],
       [3., 7., 9.],
       [4., 5., 2.],
       [5., 4., 6.]])
```

Example

CountVectorizer

```
sample = ['Teknik Fisika',
          'Institut Teknologi Sumatera',
          'Lampung Selatan']
```

```
from sklearn.feature_extraction.text import CountVectorizer

vec = CountVectorizer()
HitungHuruf = vec.fit_transform(sample)
HitungHuruf
```

```
<3x7 sparse matrix of type '<class 'numpy.int64'>'
        with 7 stored elements in Compressed Sparse Row format>
```

Example

DictVectorizer

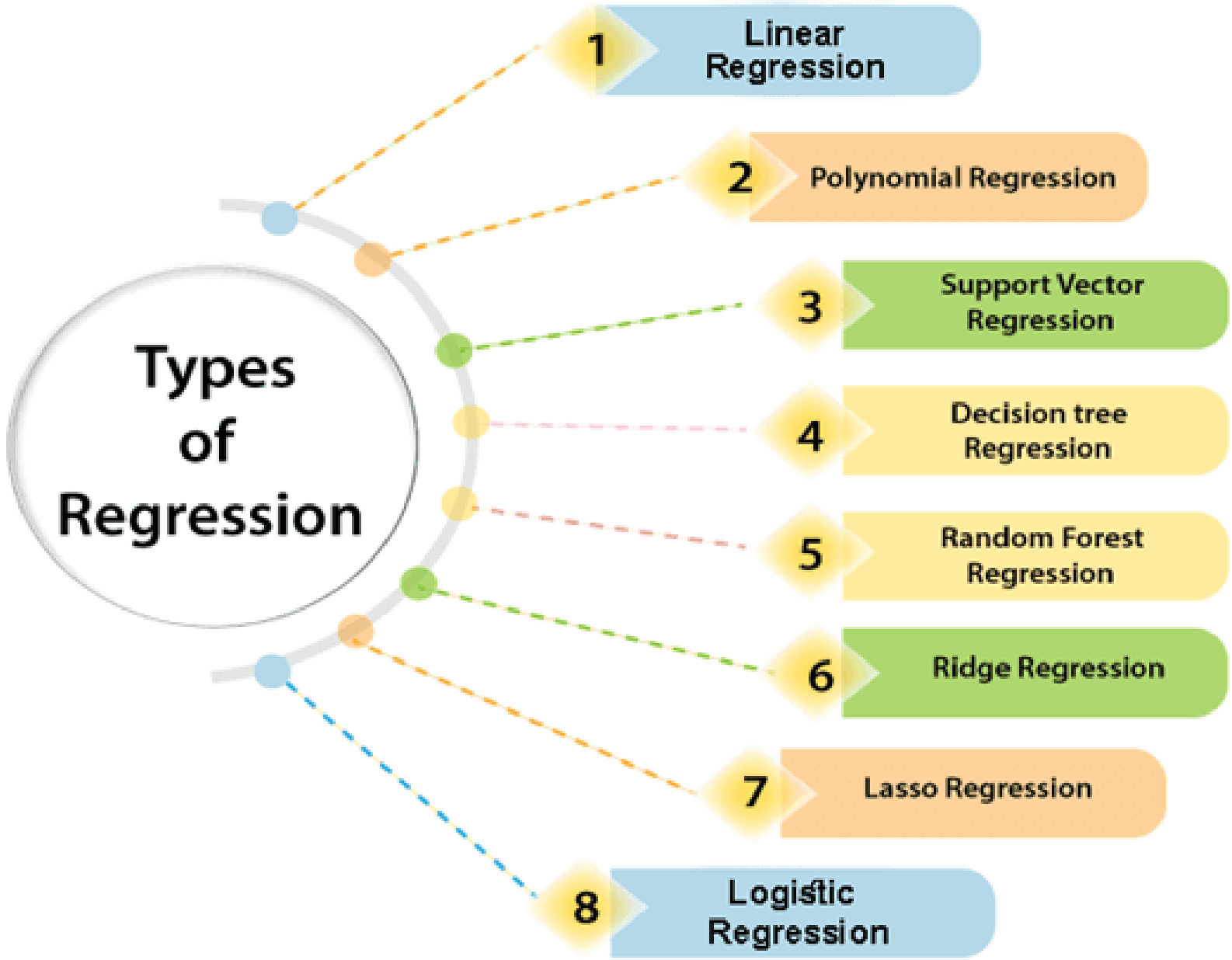
```
from sklearn.feature_extraction import DictVectorizer

Monitoring = [
    {'kota': 'Jakarta', 'temperature': 33},
    {'kota': 'Bandung', 'temperature': 20},
    {'kota': 'Bandar Lampung', 'temperature': 40},
]
vec = DictVectorizer()
```

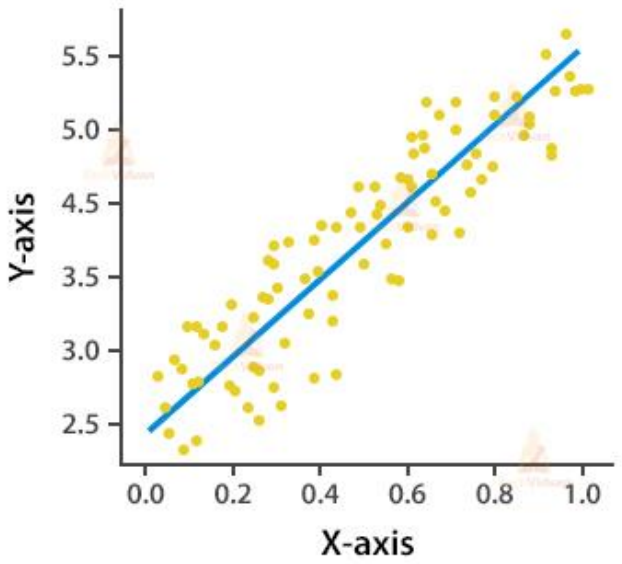
```
vec.fit_transform(Monitoring).toarray()
vec.get_feature_names()
```

```
['kota=Bandar Lampung', 'kota=Bandung', 'kota=Jakarta', 'temperature']
```

Linear Regression & Logistic Regression

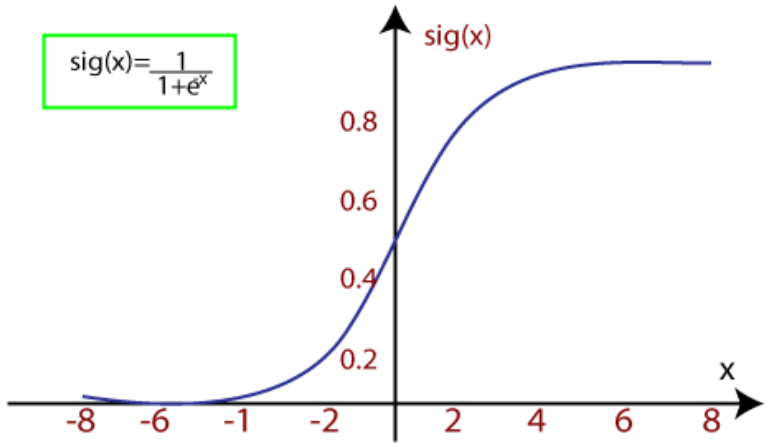

Types of Regression
1
Linear Regression
2
Polynomial Regression
3
Support Vector Regression
4
Decision tree Regression
5
Random Forest Regression
6
Ridge Regression
7
Lasso Regression
8
Logistic Regression

Linear Regression

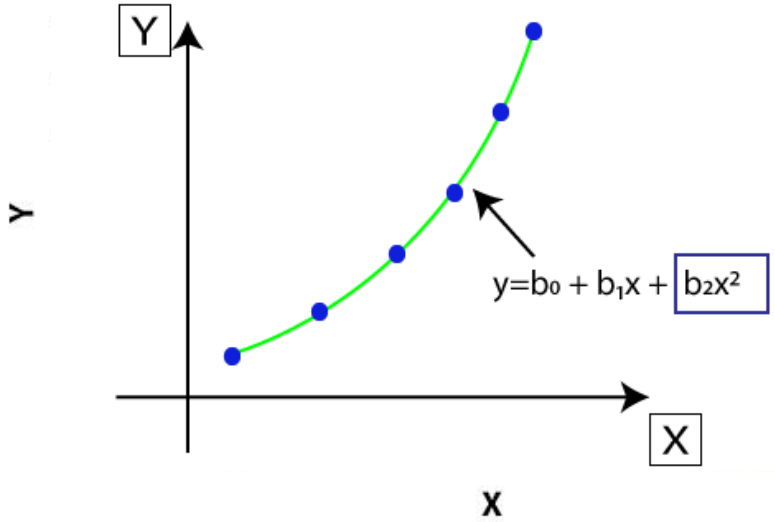


Logistic Regression

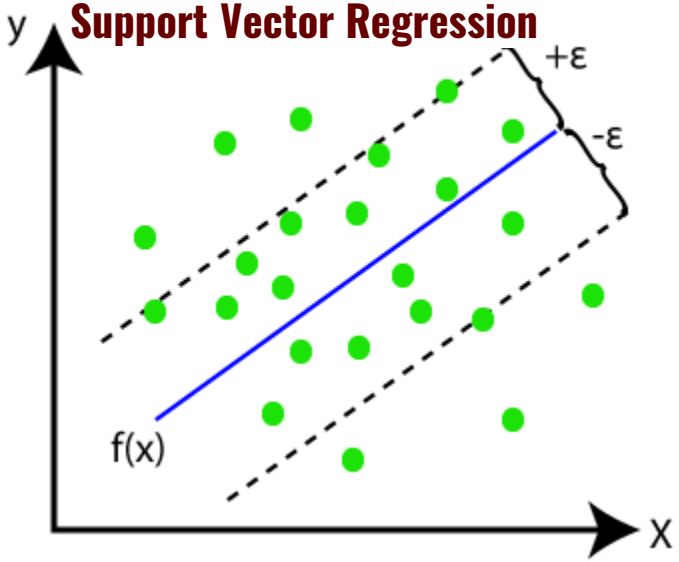
$$sig(x) = \frac{1}{1+e^{x}}$$



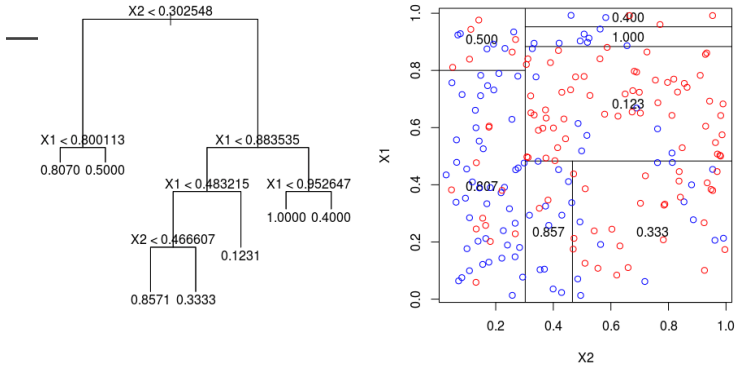
Polynomial Regression



Support Vector Regression



Decision Tree Regression

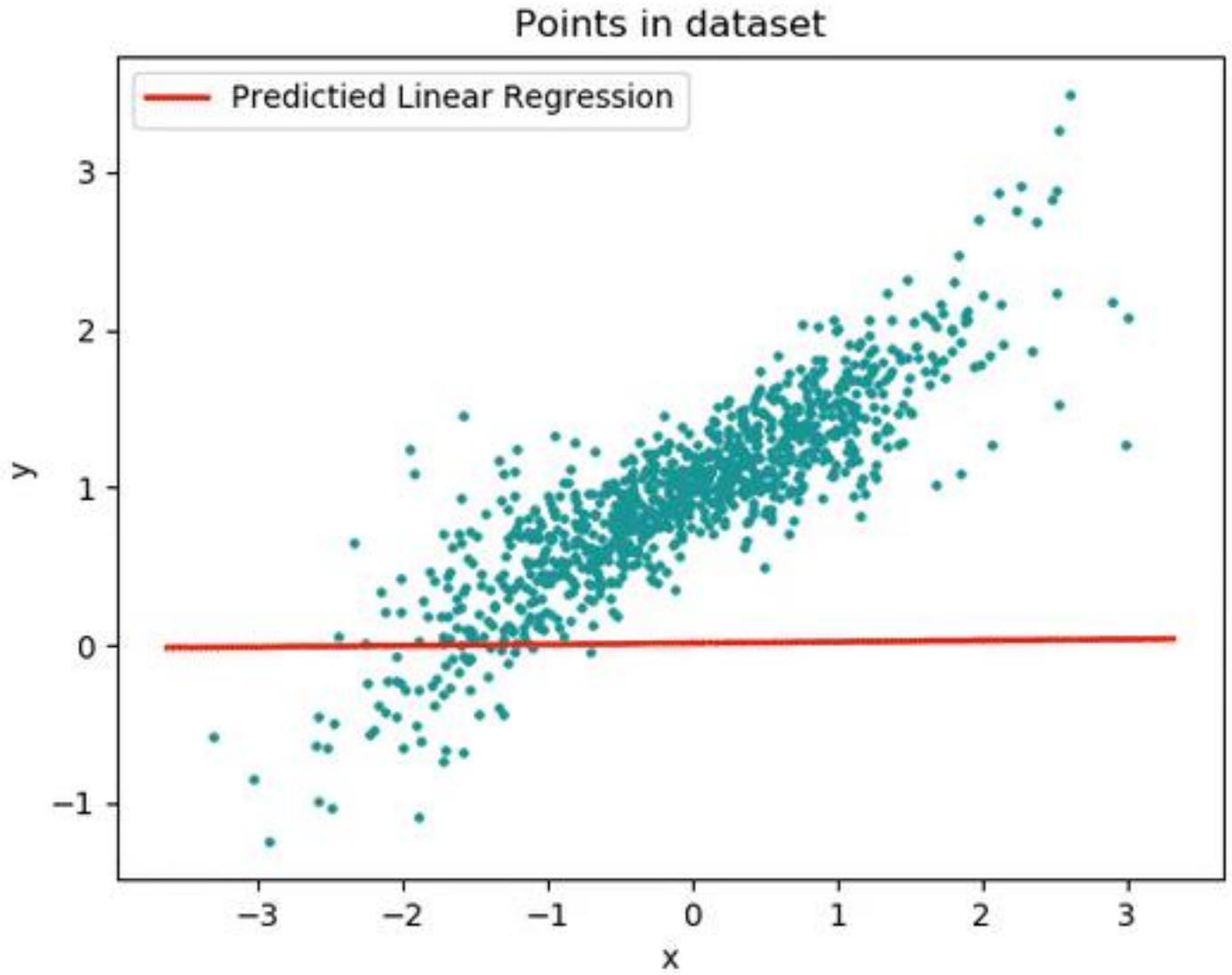


Lasso Regression

$$= \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \underbrace{\|y - X\beta\|_2^2}_{\text{Loss}} + \lambda \underbrace{\|\beta\|_1}_{\text{Penalty}}$$

ElasticNet Regression

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}}(\|y - X\beta\|^2 + \lambda_2\|\beta\|^2 + \lambda_1\|\beta\|_1).$$


Points in dataset
Predictied Linear Regression
3
2
1
0
−1
y
−3
−2
−1
0
1
2
3
x

Support Vector Machine

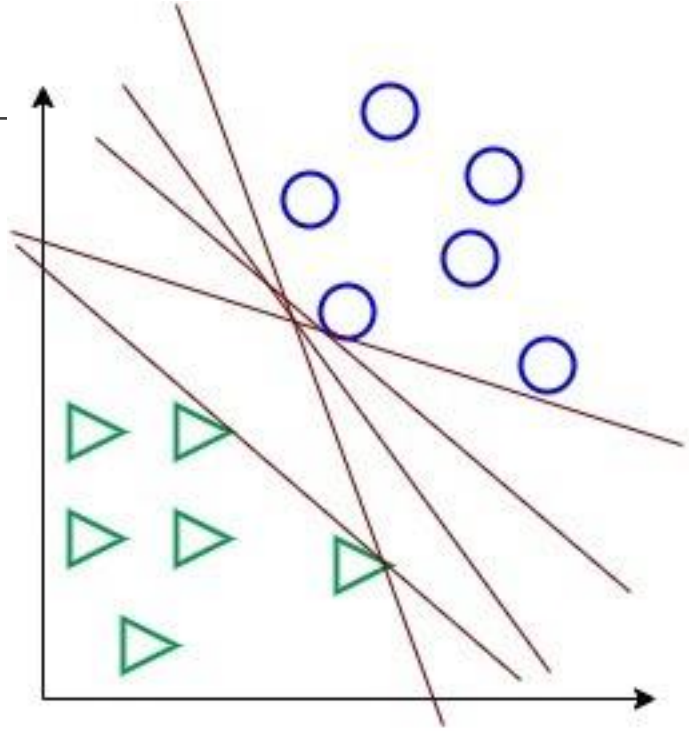
$$K\left(\overline{x}\right) = \begin{matrix} 1 & \text{if} \left\|\overline{x}\right\| \le 1 \\ 0 & \text{otherwise} \end{matrix}$$

SVM

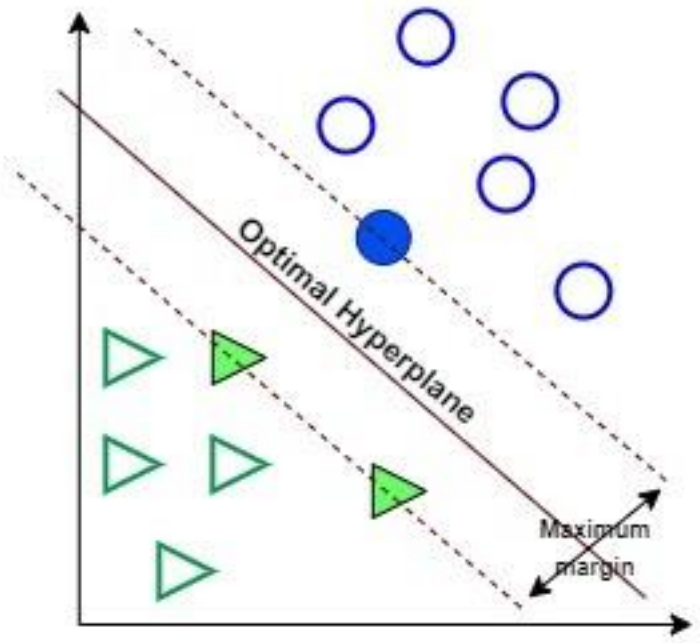
Support Vector Machine adalah model ML multifungsi yang dapat digunakan untuk menyelesaikan permasalahan klasifikasi, regresi, dan pendeteksian outlier. Termasuk ke dalam kategori supervised learning, SVM adalah salah satu metode yang paling populer dalam machine learning.

Tujuan dari algoritma SVM adalah untuk menemukan hyperplane terbaik dalam ruang berdimensi-N (ruang dengan N-jumlah fitur) yang berfungsi sebagai pemisah yang jelas bagi titik-titik data input.

Perhatikan gambar berikut



Before SVM

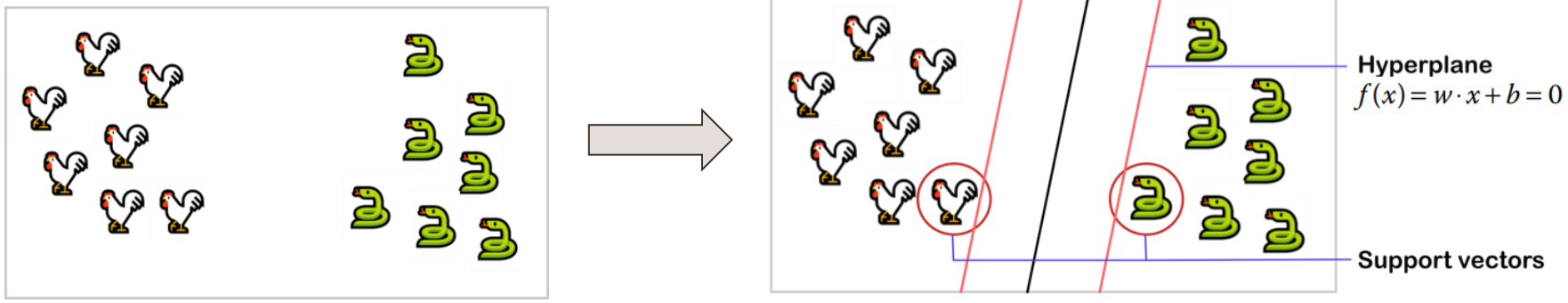


After SVM

Technique	Advantages	Disadvantages
SVM	It is highly accurate	Its speed is low
	It can handle many features	It requires more time to process

SVM Classifier

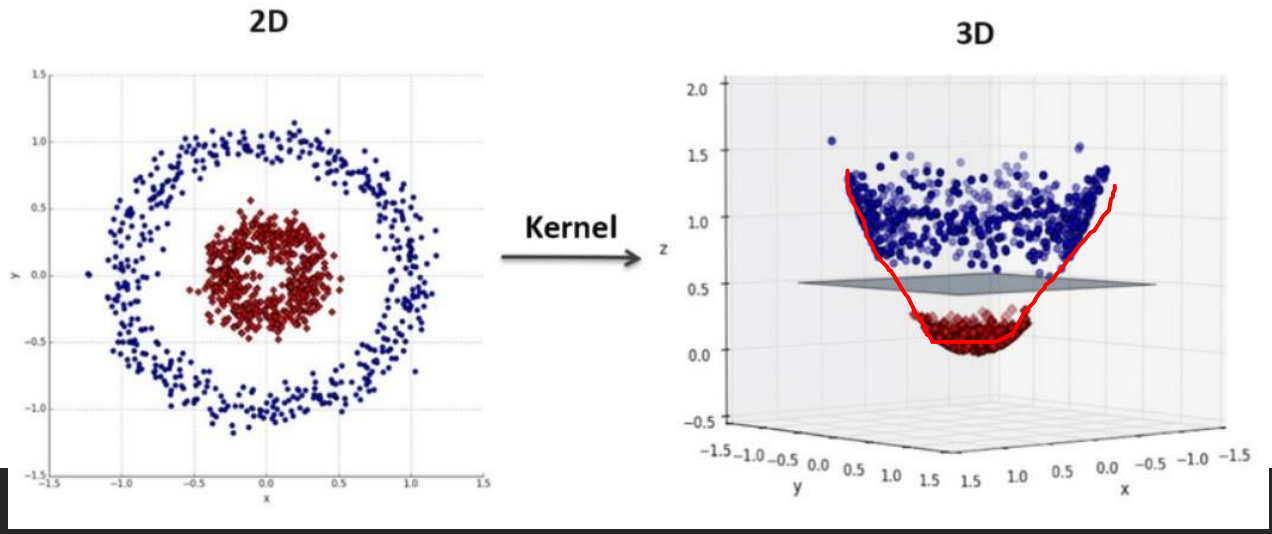
Kita bisa membuat sebuah model klasifikasi yang memisahkan antara kedua kelas tersebut menggunakan SVM. Menurut Aurelien Geron dalam buku Hands on Machine Learning, SVM bekerja dengan membuat decision boundary atau sebuah bidang yang mampu memisahkan dua buah kelas



SVM Classifier (Non Linear)

Data berikut merupakan data yang tidak bisa dipisahkan secara linier sehingga kita menyebutnya sebagai data non-linear. Pada data non-linear, decision boundary yang dihitung algoritma SVM bukan berbentuk garis lurus.

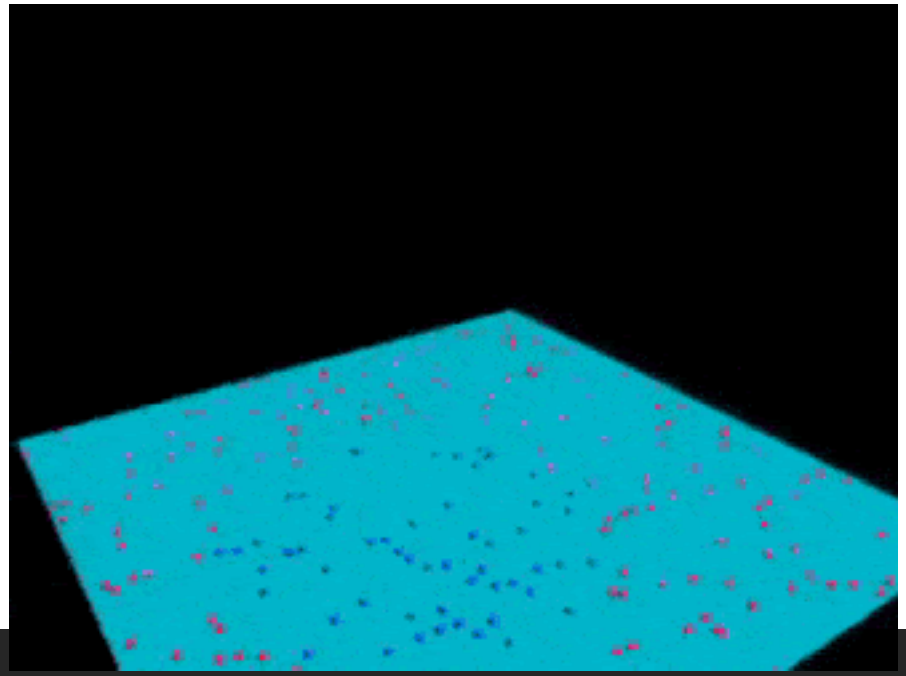
Untuk data seperti di atas, Support Vector Classifier menggunakan metode “kernel trick”, yaitu sebuah metode untuk mengubah data pada dimensi tertentu (misal 2D) ke dalam dimensi yang lebih tinggi (3D) sehingga dapat menghasilkan hyperplane yang optimal.



SVM Classifier (Non Linear)

Bagaimana trik kernel bekerja?

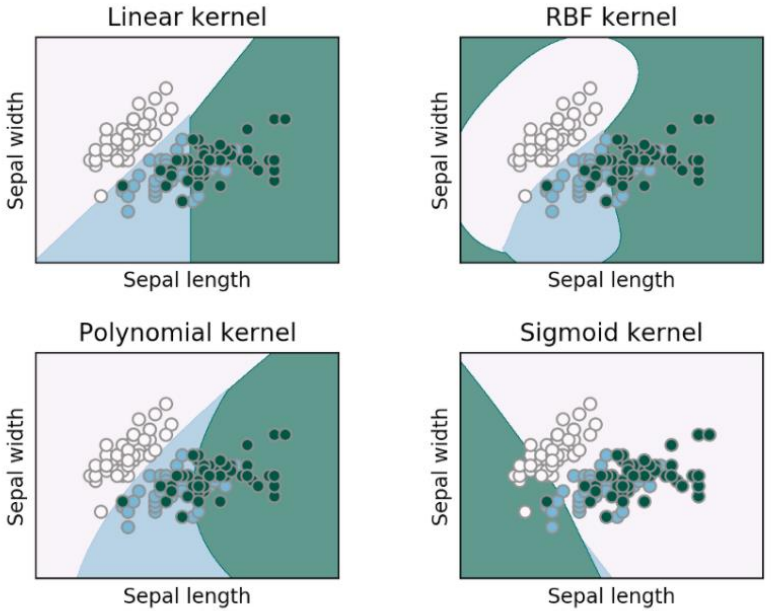
Pertama, kita perlu menghitung skor jarak dari dua titik data, misal x_i dan x_j. Skor akan bernilai lebih tinggi untuk titik data yang lebih dekat, dan sebaliknya. Lalu kita gunakan skor ini untuk memetakan data pada dimensi yang lebih tinggi (3D) menggunakan fungsi kernel.



Fungsi Kernel

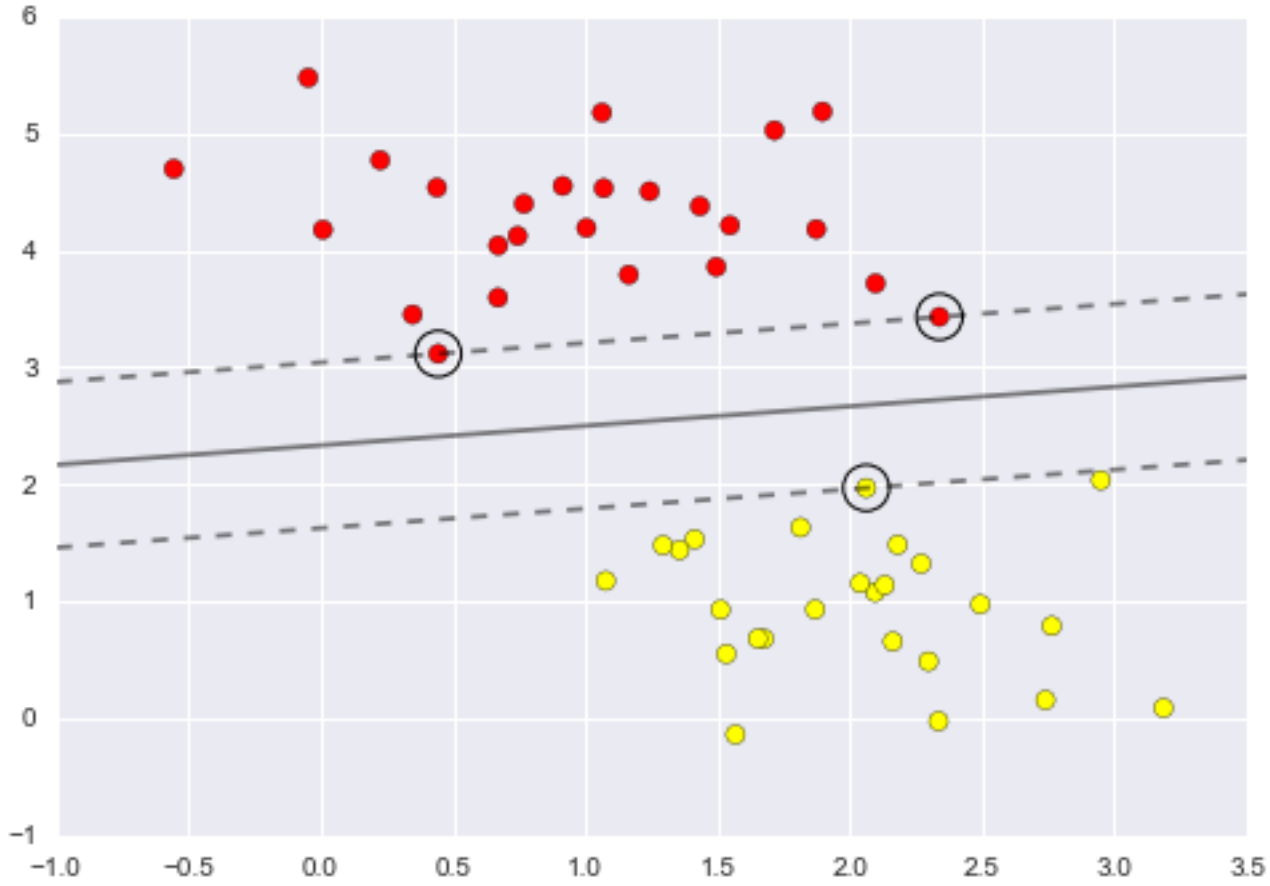
Berikut adalah beberapa fungsi kernel yang perlu Anda ketahui.

1. Linear
2. Gaussian kernel
3. RBF (Radial Basis Function)
4. Polinomial
5. Sigmoid

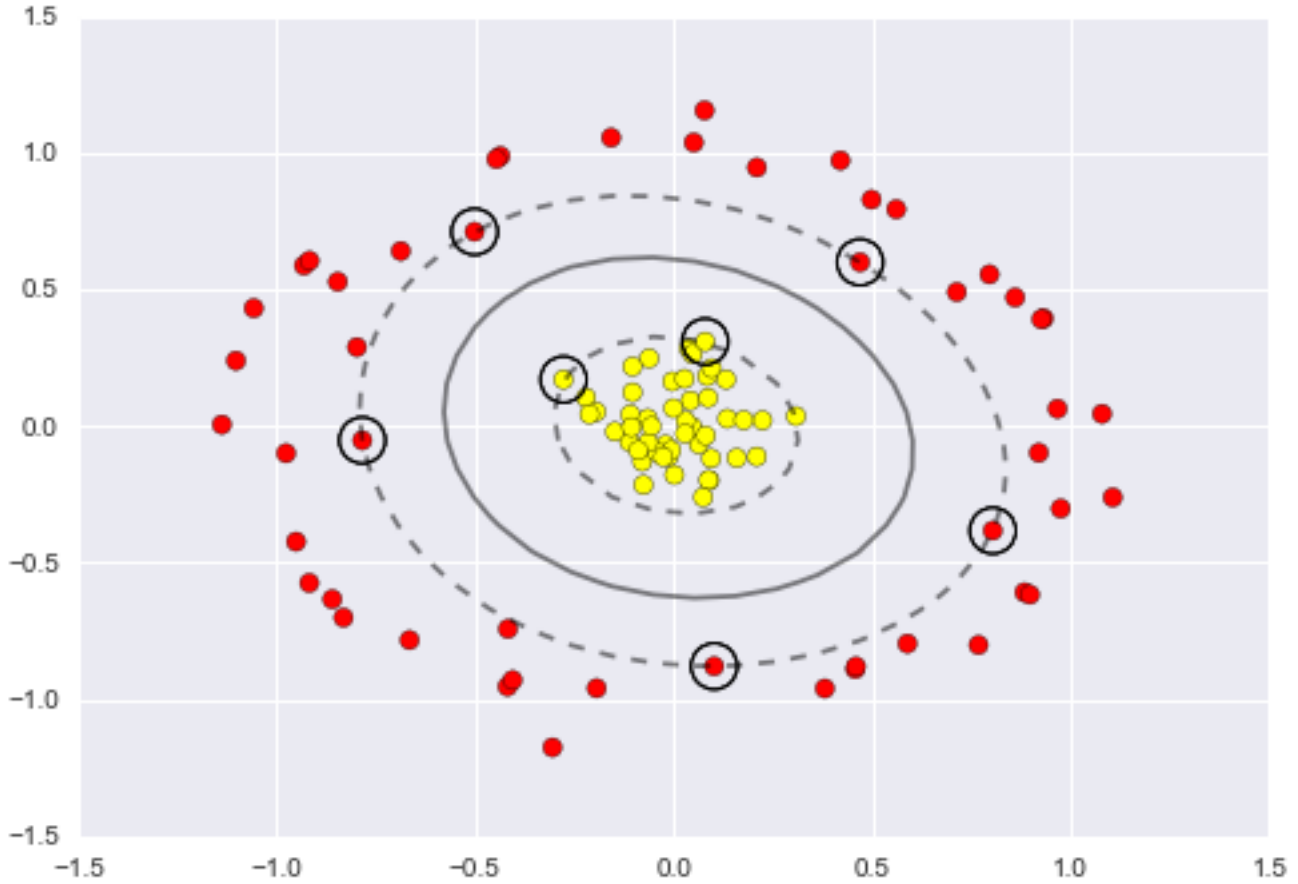


H. Marius. “Multiclass Classification with Support Vector Machine (SVM), Dual Problem, and Kernel Function”. Towards Data Science. link

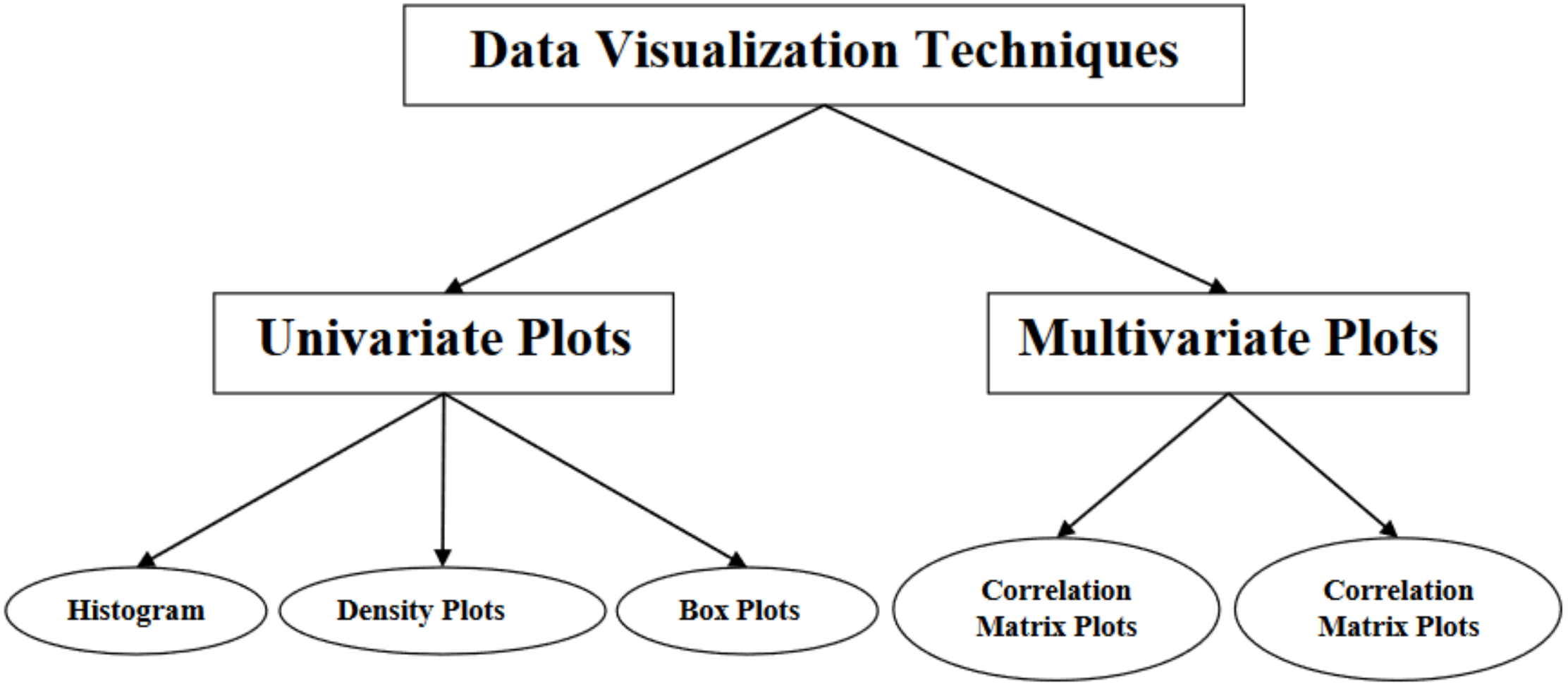
```
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')
plot_svc_decision_function(model);
```



```
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')
plot_svc_decision_function(clf)
plt.scatter(clf.support_vectors_[:, 0], clf.support_vectors_[:, 1],
            s=300, lw=1, facecolors='none');
```

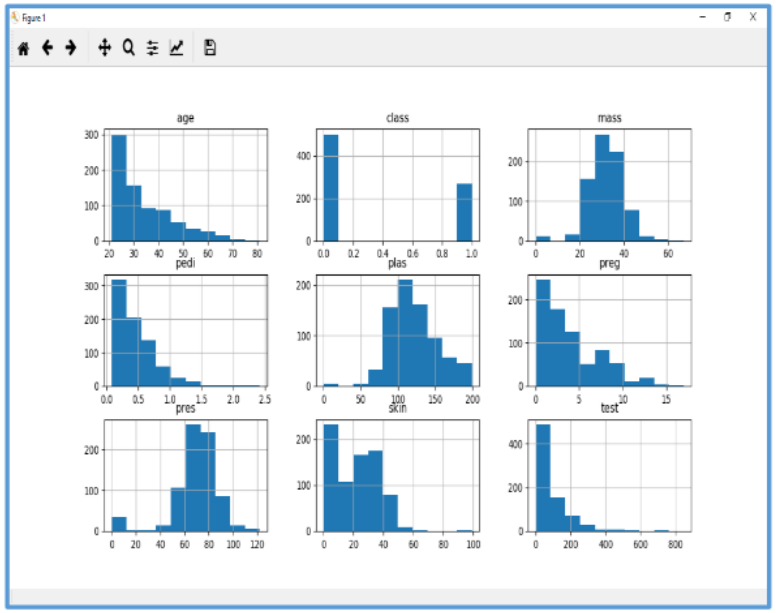


Data Visualisasi


Data Visualization Techniques
Univariate Plots
Multivariate Plots
Histogram
Density Plots
Box Plots
Correlation Matrix Plots
Correlation Matrix Plots

Data plot

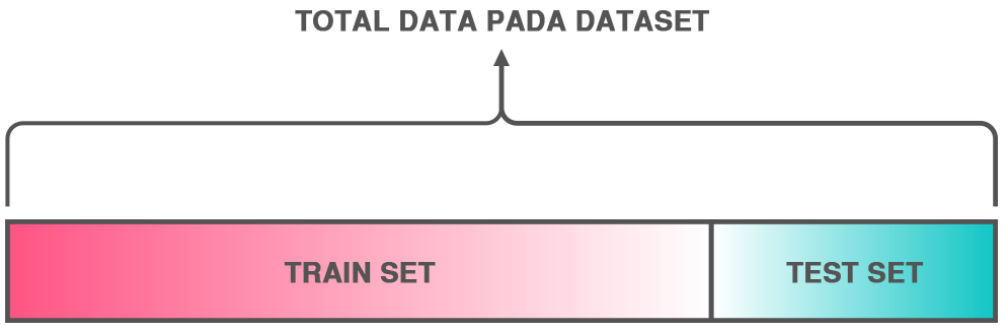
```
from matplotlib import pyplot
from pandas import read_csv
path = r"C:\pima-indians-diabetes.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age',
'class']
data = read_csv(path, names=names)
data.hist()
pyplot.show()
```



Modelling

Training, Testing, Fitting

b. Training & Testing ----> Production

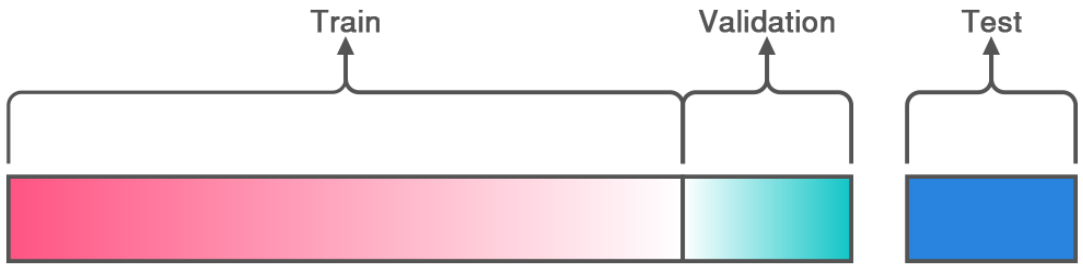


Perhatikan contoh kode berikut.

```
1.  from sklearn.model_selection import train_test_split
2.  x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1 )
```

Dengan fungsi train_test_split dari library sklearn, kita membagi array X dan y ke dalam 20% data testing (test_size=0.2). Misal total dataset A yang kita miliki adalah 1000 record, dengan test_size=0.2, maka data testing kita berjumlah 200 record dan jumlah data training sebesar 800 (80%).

c. Training, Validating & Testing ----> Production

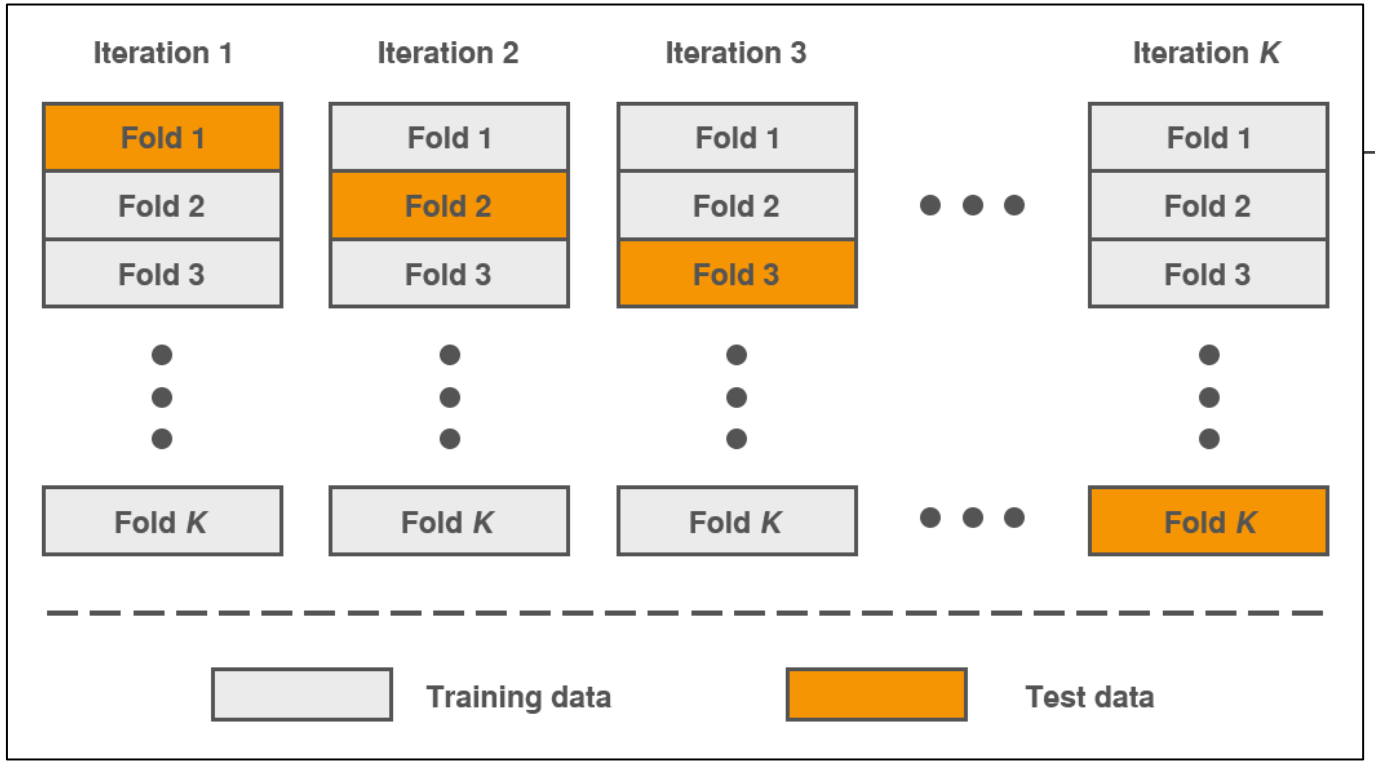


Perhatikan contoh kode berikut.

```
1.  from sklearn.model_selection import train_test_split
2.  x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1 )
```

Validation set atau holdout validation adalah bagian dari train set yang dipakai untuk pengujian model pada tahap awal. Secara sederhana, kita menguji beberapa model dengan hyperparameter yang berbeda pada data training yang telah dikurangi data untuk validation. Lalu kita pilih model serta hyperparameter yang bekerja paling baik pada validation set. Setelah proses pengujian pada holdout validation, kita bisa melatih model menggunakan data training yang utuh (data training termasuk data validation) untuk mendapatkan model final. Terakhir kita mengevaluasi model final pada test set untuk melihat tingkat erornya.

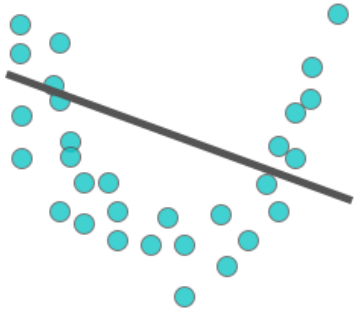
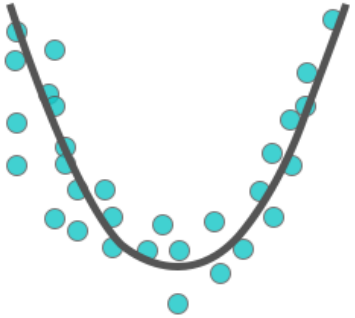
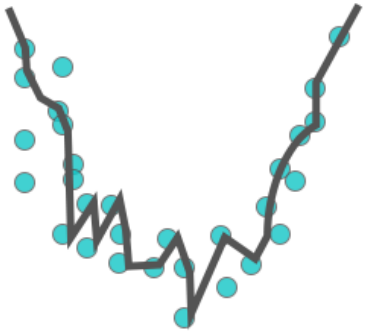
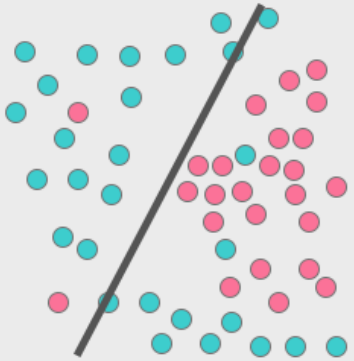
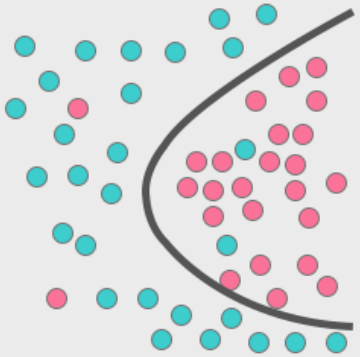
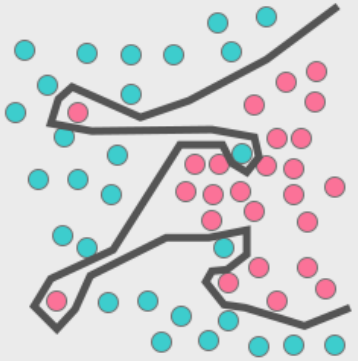
d. Cross Validation




```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(clf, x, y, cv=5)
scores
```

K-Fold Cross Validation atau lebih sering disebut cross validation adalah salah satu teknik yang populer dipakai dalam evaluasi model ML. Pada cross validation dataset dibagi sebanyak K lipatan. Pada setiap iterasi setiap lipatan akan dipakai satu kali sebagai data uji dan lipatan sisanya dipakai sebagai data latih. Dengan menggunakan cross validation kita akan memperoleh hasil evaluasi yang lebih akurat karena model dievaluasi dengan seluruh data. Berikut adalah ilustrasi dari K-cross validation.

Model Fitting

	Underfitting	Just right	Overfitting
Symptoms	• High training error • Training error close to test error • High bias	• Training error slightly lower than test error	• Very low training error • Training error much lower than test error • High variance
Regression illustration			
Classification illustration			

Model Fitting (Cara 1)

Mengubah parameter untuk meningkatkan performa

Parameters:

criterion : *{"gini", "entropy"}, default="gini"*

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

splitter : *{"best", "random"}, default="best"*

The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

max_depth : *int, default=None*

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : *int or float, default=2*

The minimum number of samples required to split an internal node:

- If int, then consider `min_samples_split` as the minimum number.
- If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

Model Fitting (Cara 2)

Ada salah satu teknik untuk menguji beberapa parameter sekaligus. Teknik ini disebut dengan *Grid Search*.

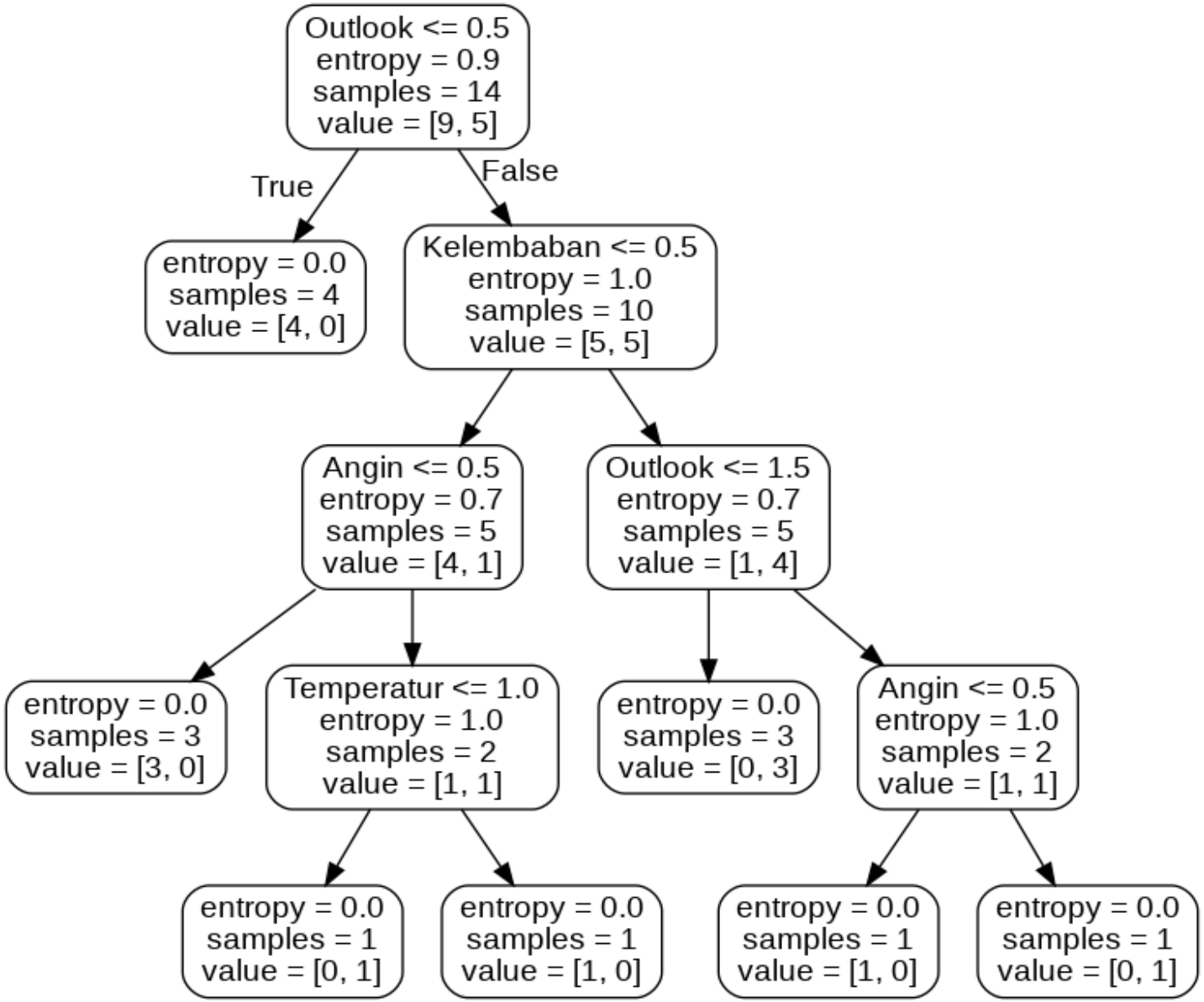
```
from sklearn.model_selection import train_test_split
Xtrain, Xtest, ytrain, ytest = train_test_split(faces.data, faces.target,
                                                random_state=42)
from sklearn.model_selection import GridSearchCV
param_grid = {'svc__C': [1, 5, 10, 50],
              'svc__gamma': [0.0001, 0.0005, 0.001, 0.005]}
grid = GridSearchCV(model, param_grid)

%time grid.fit(Xtrain, ytrain)
print(grid.best_params_)
```

```
CPU times: user 1min 5s, sys: 36.6 s, total: 1min 42s
Wall time: 1min 1s
{'svc__C': 10, 'svc__gamma': 0.001}
```

Decision Tree

	Outlook	Temperatur	Kelembaban	Angin	Play
0	cerah	panas	tinggi	tidak	Tidak Main
1	cerah	panas	tinggi	ya	Tidak Main
2	berawan	panas	tinggi	tidak	Main
3	hujan	sejuk	tinggi	tidak	Main
4	hujan	dingin	normal	tidak	Main


Outlook <= 0.5
entropy = 0.9
samples = 14
value = [9, 5]
True
False
entropy = 0.0
samples = 4
value = [4, 0]
Kelembaban <= 0.5
entropy = 1.0
samples = 10
value = [5, 5]
Angin <= 0.5
entropy = 0.7
samples = 5
value = [4, 1]
Outlook <= 1.5
entropy = 0.7
samples = 5
value = [1, 4]
entropy = 0.0
samples = 3
value = [3, 0]
Temperatur <= 1.0
entropy = 1.0
samples = 2
value = [1, 1]
entropy = 0.0
samples = 3
value = [0, 3]
Angin <= 0.5
entropy = 1.0
samples = 2
value = [1, 1]
entropy = 0.0
samples = 1
value = [0, 1]
entropy = 0.0
samples = 1
value = [1, 0]
entropy = 0.0
samples = 1
value = [1, 0]
entropy = 0.0
samples = 1
value = [0, 1]

Metode Evaluasi: Supervised Learning

Classification Metrics

Accuracy Score

```
>>> from sklearn.metrics import accuracy_score
>>> accuracy_score(y_test, y_pred)
```

Confusion Matrix

```
>>> from sklearn.metrics import confusion_matrix
>>> print(confusion_matrix(y_test, y_pred))
```

Classification Report

```
>>> from sklearn.metrics import classification_report
>>> print(classification_report(y_test, y_pred))
```

Cross-Validation

Adjusted Rand Index

```
>>> from sklearn.cross_validation import cross_val_score
>>> print(cross_val_score(knn, X_train, y_train, cv=4))
>>> print(cross_val_score(lr, X, y, cv=2))
```

Regression Metrics

Mean Absolute Error

```
>>> from sklearn.metrics import mean_absolute_error
>>> y_true = [3, -0.5, 2]
>>> mean absolute error(y true, y pred)
```

Mean Squared Error

```
>>> from sklearn.metrics import mean_squared_error
>>> mean_squared_error(y_test, y_pred)
```

R^2 Score

```
>>> from sklearn.metrics import r2_score
>>> r2_score(y_true, y_pred)
```

Metode Evaluasi: Unsupervised Learning

Clustering Metrics

Adjusted Rand Index

```
>>> from sklearn.metrics import adjusted_rand_score
>>> adjusted_rand_score(y_true, y_pred)
```

Homogeneity

```
>>> from sklearn.metrics import homogeneity_score
>>> homogeneity_score(y_true, y_pred)
```

V-measure

```
>>> from sklearn.metrics import v_measure_score
>>> metrics.v_measure_score(y_true, y_pred)
```

Referensi

- Field Cady, The Data Science Handbook, Wiley, (2017) pg. 21
- https://scikit-learn.org/stable/modules/feature_extraction.html
- https://pandas.pydata.org/pandas-docs/stable/user_guide/text.htm
- https://developers.google.com/machine-learning/problem-framing/big-questions?hl=id
- https://www.ibm.com/blogs/research/2017/08/ai-based-automated-feature-engineering/
- https://www.alamy.com/cross-industry-stardard-process-for-data-mining-data-science-process-presentation-main-steps-image263611143.html
- https://towardsdatascience.com/feature-engineering-for-machine-learning-3a5e293a5114
- https://datascience.stackexchange.com/questions/29006/feature-selection-vs-feature-extraction-which-to-use-when