Lab 4 Data Science & Analysis
Dimention Reduction & Machine Learning
Deadline: 26 April 2018 Pukul 23.55

Petunjuk:

1. Dalam tutorial Lab-4 ini Anda akan mencoba langkah demi langkah dalam Machine Learning
2. **Anda tidak perlu membuka Anaconda**, Silahkan Anda langsung membuka **Jupyter Notebook** atau **editor lainnya** yang biasa Anda gunakan.
3. File yang harus dikumpulkan:
   - Tutorial (20%) : File tutorial berisi screenshoot hasil dan code dari kotak dengan tulisan berwarna **hitam** pada tutorial. Bagian teks yang berwarna **biru** dapat anda coba tetapi tidak perlu di dokumentasikan/dikumpulkan. Format penamaan file **Tutorial_Lab3_Nama_NPM.pdf**
   - Jawaban (80%): File jawaban berisi jawaban dari hasil pekerjaan anda yang terdiri dari penjelasan, screenshot hasil, dan code. Code bisa langsung di screenshot. Format penamaan file **Jawaban_Lab3_Nama_NPM.pdf**
   - File.py (-10% jika tidak dikumpulkan) : Berisi code dari jawaban soal. Anda tidak perlu melampirkan code untuk tutorial. Format penamaan file **codejawaban.py**
   - Zip tutorial, jawaban, dan *file*.py Anda. Format penamaan file **Lab4_Nama_NPM.zip**
4. **Penalti keterlambatan pengumpulan 10%** dari total nilai Lab 4.
5. **Penalti salah submit pengumpulan 10%** dari total nilai Lab 4.

A. Dimentionality Reduction

Salah satu masalah yang sering dijumpai dalam mengolah data adalah tingginya dimensi data yang diolah. Mengapa tingginya dimensi ini membuat masalah jadi sulit? Bukankah semakin banyak item yang diobservasi, akan membuat klasifikasi suatu data menjadi mudah dan hasilnya lebih baik? Pendapat ini ada benarnya. Umumnya penambahan atribut pada suatu data akan membuat proses klasifikasi menjadi lebih mudah dan hasilnya lebih baik. Tetapi jika jumlah atribut terus bertambah, sedangkan jumlah sampel data terbatas, maka akurasi klasifikasi ini pada titik tertentu akan menurun. Fenomena inilah yang disebut "***Curse of dimensionality***"

Masalah ***Curse of Dimensionality*** (kutukan dimensi) dapat diselesaikan dengan menambahkan lebih banyak datapoint (not always possible) atau mengurangi dimensi. Pengurangan dimensi yang dimaksud adalah pengurangan pada jumlah kolom dari data. Beberapa cara yang dapat dilakukan adalah:

- ***Feature Selection***: mencari korelasi antar atribut dan memilih fitur terbaik
- ***Feature Extraction***: Menggunakan rumus matematik untuk memperoleh fitur yang baru (seringkali lebih baik dari fitur original). Salah satu metode yang dapat digunakan adalah *Principal Component Analysis* (PCA).

PCA akan mengekstrak sejumlah kolom/fitur baru dengan jumlah yang lebih sedikit sebagai representatif dari data asli.

1) Text

Pada tutorial ini akan dilakukan prediksi apakah seseorang memberikan ulasan bintang 5 atau 1 berdasarkan kata-kata/ *text* yang mereka gunakan dalam ulasan.

- **Load Data**

```
import pandas as pd
yelp = pd.read_csv("yelp.csv",encoding='unicode-escape')
yelp_best_worst = yelp[(yelp.stars==5) | (yelp.stars==1)]
print (yelp.shape)
print (yelp_best_worst.shape)

#output
(10000, 10)
(4086, 10)
```

- **Partisi Data**

Membagi dataset ke dalam data pelatihan dan pengujian adalah salah satu bagian penting dalam membangun dan mengevaluasi model. Model yang dihasilkan diharapkan tidak 'overfit' maupun 'underfit' terhadap data training, agar ketika diberi data baru yang tidak berlabel (data testing) model dapat mengenali dengan baik. Untuk itu kita harus menyesuaikan berapa proporsi yang pas untuk data training dan testing. Contoh berikut membagi data *training* 75% dan data *testing* 25%.

```
X = yelp_best_worst.text
y = yelp_best_worst.stars == 5
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=100)
```

- **Menghilangkan *stopword***

Stop word merupakan kata-kata yang tidak penting dalam hal pencarian suatu dokumen. Kata-kata ini biasanya berjumlah sangat banyak dalam suatu data. Dalam bahasa Inggris contohnya *the, at, is*. Kata-kata ini perlu dihilangkan sebelum melakukan text mining

```
from sklearn.feature_extraction.text import CountVectorizer
vect = CountVectorizer(stop_words='english')
```

- **Tokenisasi Data**

Tokenisasi merupakan proses *spliting* data menjadi token-token. Dalam kasus ini, 1 token merupakan sebuah kata. Pemisahan dokumen menjadi token kata dilakukan berdasarkan karakter spasi. *Library* **sklearn** menyediakan sebuah fungsi yang akan Mengkonversi kumpulan dokumen teks ke matriks jumlah token yang bernama Document Term Matriks (DTM).

```
X_train_dtm = vect.fit_transform(X_train)
X_test_dtm = vect.transform(X_test)
```

- **Building model**

Pada tutorial ini akan digunakan Logistic Regression sebagai classifier dengan menggunakan semua fitur dari DTM.

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train_dtm, y_train)
lr.score(X_test_dtm, y_test)
print (X_train_dtm.shape)
print (X_test_dtm.shape)

#output
(3064, 16507)
(1022, 16507)
0.9119373776908023
```

Selanjutnya dilakukan pelatihan model dengan hanya menggunakan 300 *top words* saja sebagai fitur

```
vect = CountVectorizer(stop_words='english', max_features=300)
X_train_dtm = vect.fit_transform(X_train)
X_test_dtm = vect.transform(X_test)
print (X_train_dtm.shape)
print (X_test_dtm.shape)
lr.fit(X_train_dtm, y_train)
lr.score(X_test_dtm, y_test)

#output:
(3064, 300)
(1022, 300)
0.8816046966731899
```

Dapat dilihat bahwa akurasi yang diperoleh turun menjadi 88% saja. Hal ini *makes sense* karena sebagaian besar fitur (±80%) dihilangkan.

Selanjutnya dilakukan pelatihan model dengan menggunakan pendekatan yang berbeda, yaitu dengan menggunakan PCA untuk membuat 300 fitur baru.

```
from sklearn import decomposition
vect = CountVectorizer(stop_words='english')
pca = decomposition.PCA(n_components=300)
X_train_dtm = vect.fit_transform(X_train).todense()
X_train_dtm = pca.fit_transform(X_train_dtm)
X_test_dtm = vect.transform(X_test).todense()
X_test_dtm = pca.transform(X_test_dtm)
print (X_train_dtm.shape)
print (X_test_dtm.shape)
lr.fit(X_train_dtm, y_train)
lr.score(X_test_dtm, y_test)
```

```
#output
(3064, 300)
(1022, 300)
0.9090019569471625
```

Dapat dilihat bahwa dengan menggunakan 300 fitur saja, dapat diperoleh akurasi yang hampir sama dengan ketika menggunakan semua fitur. Selain itu, nilai akurasi yang diperoleh juga lebih tinggi dari model yang menggunakan 300 *top words*.

2) Image

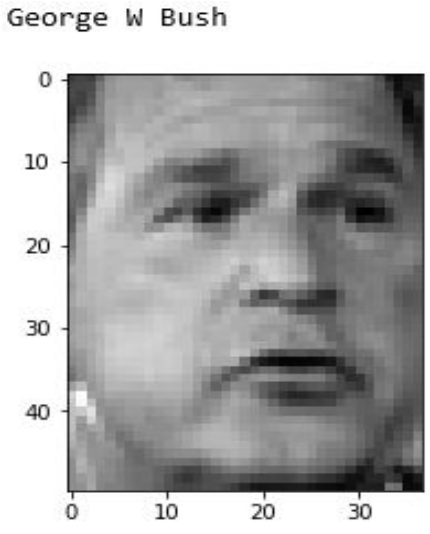
Pada contoh berikut akan dilakukan prediksi wajah dengan menggunakan dataset **faces** yang telah disediakan dalam *library sklearn*.

- **Load Data**

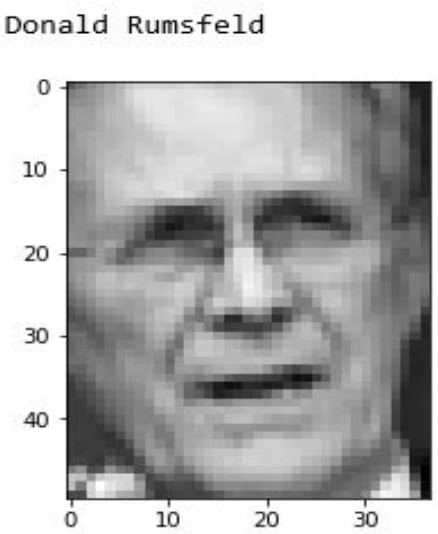
```
from sklearn.datasets import fetch_lfw_people
lfw_people = fetch_lfw_people(min_faces_per_person=70, resize=0.4)
n_samples, h, w = lfw_people.images.shape
X = lfw_people.data
y = lfw_people.target
n_features = X.shape[1]
X.shape

#output
(760, 1850) #760 datapoint & 1850 feature
```

- **Plotting Image**

```
from matplotlib import pyplot as
plt
plt.imshow(X[0].reshape((h, w)),
cmap=plt.cm.gray)
print
(lfw_people.target_names[y[0]])
plt.show()
```




```
plt.imshow(X[759].reshape((h, w)),
cmap=plt.cm.gray)
print
(lfw_people.target_names[y[759]])
plt.show()
```



- **Dataset Information**

```
target_names = lfw_people.target_names
n_classes = target_names.shape[0]
print("Total dataset size:")
print("n_samples: %d" % n_samples)
print("n_features: %d" % n_features)
print("n_classes: %d" % n_classes)
print("target names: %s" % target_names)
```

```
Total dataset size:
n_samples: 760
n_features: 1850
n_classes: 3
target names: ['Donald Rumsfeld' 'George W Bush' 'Gerhard Schroeder']
```

- **Building Model**

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from time import time
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=1)
t0 = time()
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
print (accuracy_score(y_pred, y_test))
print ((time() - t0))

#output
0.8947368421052632
1.868746042251587
```

Dengan menggunakan semua fitur diperoleh akurasi sebesar 89% dan running time 1,86 detik (hasil bergantung dari environment yang digunakan). Selanjutkan dilakukan training dengan menggunakan atribut yang di-*generate* oleh PCA.

```
from sklearn.cross_validation import train_test_split
n_components = 75
pca = decomposition.PCA(n_components=n_components,
whiten=True).fit(X_train)
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)
t0 = time()
logreg.fit(X_train_pca, y_train)
y_pred = logreg.predict(X_test_pca)
print (accuracy_score(y_pred, y_test))
print ((time() - t0))

#output
0.9
0.031249523162841797
```

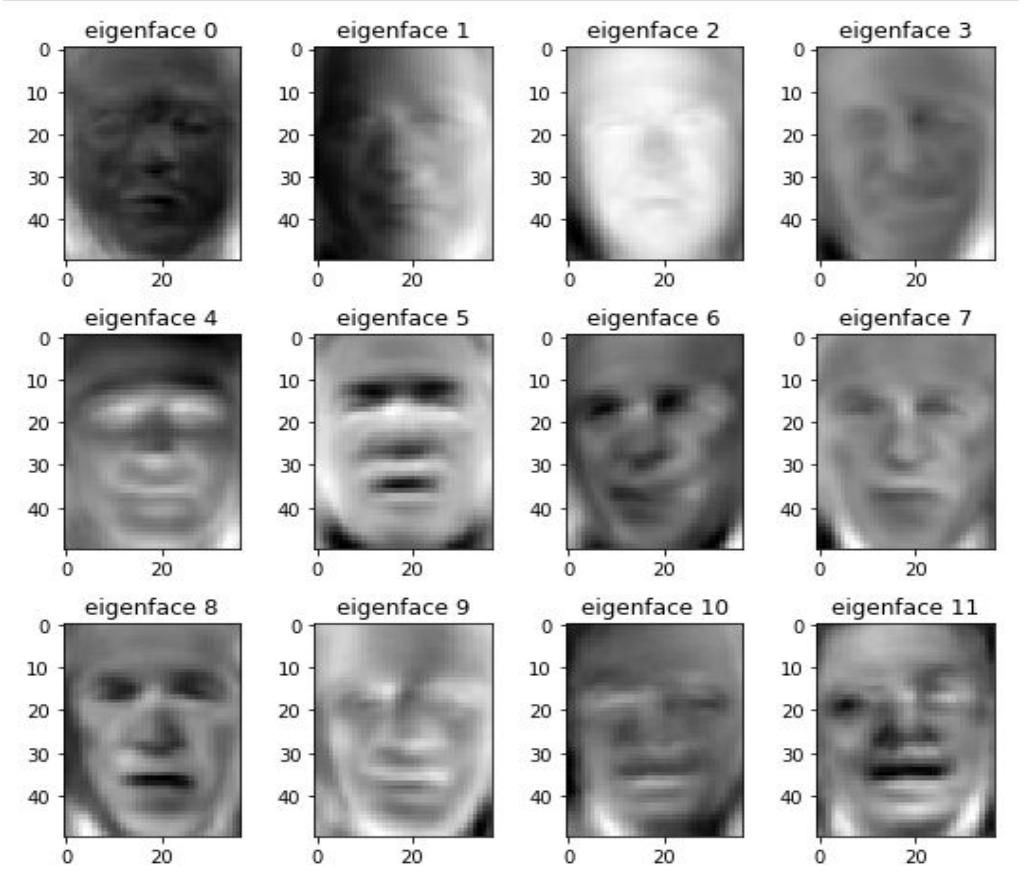
Berdasarkan hasil yang diperoleh, dapat dilihat bahwa dengan menggunakan hanya 75 fitur saja yang diperoleh dari PCA, model yang dibangun mampu menghasilkan nilai

akurasi yang lebih tinggi dibandingkan menggunakan semua fitur (1850). Selain itu, waktu komputasi dengan jumlah fitur yang lebih sedikit tentu akan lebih cepat juga, dalam hal ini waktu komputasi menjadi 60 kali lebih cepat.

Oleh karena itu, dapat disimpulkan bahwa PCA dan fitur ekstraksi secara umum dapat digunakan untuk memudahkan pengolahan data yang memilki dimensi tinggi (banyak kolom). Proses tersebut juga dapat mempercepat proses pembelajaran dan meningkatkan performa algoritma.

Berikut diberikan visualisasi hasil fitur yang di-*generate* oleh PCA

```
def plot_gallery(images, titles, n_row=3, n_col=4):
    """Helper function to plot a gallery of portraits"""
    plt.figure(figsize=(1.8 * n_col, 2.4 * n_row))
    plt.subplots_adjust(bottom=0, left=.01, right=.99, top=.90,
hspace=.35)
    for i in range(n_row * n_col):
        plt.subplot(n_row, n_col, i + 1)
        plt.imshow(images[i], cmap=plt.cm.gray)
        plt.title(titles[i], size=12)
eigenfaces = pca.components_.reshape((n_components, h, w))
eigenface_titles = ["eigenface %d" % i for i in
range(eigenfaces.shape[0])]
plot_gallery(eigenfaces, eigenface_titles)
plt.show()
```



B. Machine Learning (Supervised Learning - Regression & Classification)

Supervised learning atau pembelajaran terbimbing adalah sebuah pendekatan dimana sudah terdapat data yang dilatih, dan terdapat *variable* yang ditargetkan sehingga tujuan dari pendekatan ini adalah mengkelompokan suatu data ke target yang sudah ada. Salah satu contoh dari supervised learning adalah **Decision Tree**.

Decision Tree

Decision Tree merupakan metode *non-parametric supervised learning* yang digunakan untuk menyelesaikan permasalahan klasifikasi dan regresi. Tujuannya adalah untuk membuat suatu model yang dapat memprediksi nilai variabel target dengan cara mempelajari aturan keputusan sederhana yang disimpulkan dari fitur data.

1. Decision Tree Regression

Decision Tree (DT) Regression adalah metode DT yang diterapkan pada kasus regresi. Berikut gambaran mengenai DT Regression. Seperti yang sudah di jelaskan pada Lab 3, regresi merupakan pembelajaran yang melihat **relasi antara variabel numerik** yang dependen (variabel yang akan diprediksi) dengan satu atau lebih variabel yang independen (prediktornya).

Contoh:

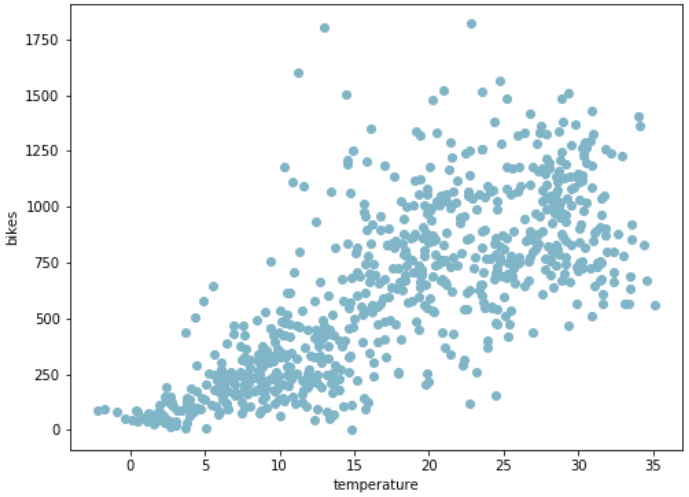
Pada tutorial kali ini Anda seolah-olah menjadi pegawai baru yang bekerja di perusahaan penyewaan sepeda di Depok. Perusahaan ini memiliki skema penyewaan sepeda yakni dimana pengguna dapat menyewa sepeda dari lokasi tertentu dan mengembalikannya di lokasi yang berbeda menggunakan mesin di tempat parkir sepeda. Anda bertanggung jawab untuk memprediksi berapa banyak sepeda yang akan digunakan di masa depan. Hal ini menjadi sangat penting untuk memprediksi pendapatan perusahaan dan merencanakan perbaikan infrastruktur. Tetapi Anda tidak tahu banyak tentang sistem penyewaan sepeda. Anda hanya diberikan data csv yang berisi jumlah sepeda yang di sewa setiap hari.
Silahkan download data **bikes.csv** di **SCeLE**!

```
# load data bikes.csv
import pandas as pd
bikes = pd.read_csv('bikes.csv')
bikes.head()
```

	date	temperature	humidity	windspeed	count
0	2011-01-03	2.716070	45.715346	21.414957	120.0
1	2011-01-04	2.896673	54.267219	15.136882	108.0
2	2011-01-05	4.235654	45.697702	17.034578	82.0
3	2011-01-06	3.112643	50.237349	10.091568	88.0
4	2011-01-07	2.723918	49.144928	15.738204	148.0

Merujuk pada tabel di atas, Anda dapat melihat bahwa tidak hanya data jumlah sepeda yang disewa setiap hari, tetapi juga kondisi rata-rata cuaca pada setiap hari. Untuk itu, ada baiknya jika Anda memeriksa apakah terdapat hubungan antara cuaca dan jumlah sepeda yang di sewa. Anda dapat melakukan visualisasi data dengan cara mem-*ploting* suhu terhadap jumlah sepeda yang di sewa setiap hari.

```
from matplotlib import pyplot as plt

plt.figure(figsize=(8,6))
plt.plot(bikes['temperature'],bikes['count'], 'o')
plt.xlabel('temperature')
plt.ylabel('bikes')
plt.show()
```



Scatter Plot

Merujuk pada gambar di atas, Anda dapat melihat bahwa ada hubungan yang jelas antara jumlah sepeda yang di sewa dan suhu. Ketika suhu meningkat maka jumlah sepeda yang di sewa juga meningkat. Namun, bagaimana kita dapat menggunakan informasi ini untuk memprediksi jumlah sepeda yang di sewa?. Untuk menjawab pertanyaan tersebut, kita dapat menggunakan metode regresi. Dengan menggunakan metode tersebut kita dapat menangkap hubungan antara suhu dan jumlah sepeda yang di sewa menggunakan model yang dapat Anda peroleh menggunakan metode Decision Tree Regression.

```
from sklearn.tree import DecisionTreeRegressor
import numpy as np

regressor = DecisionTreeRegressor(max_depth=2)
regressor.fit(np.array([bikes['temperature']]).T, bikes['count'])
```

Pertanyaan 1:
Berapa banyak sepeda yang di sewa saat suhu 5 derajat?

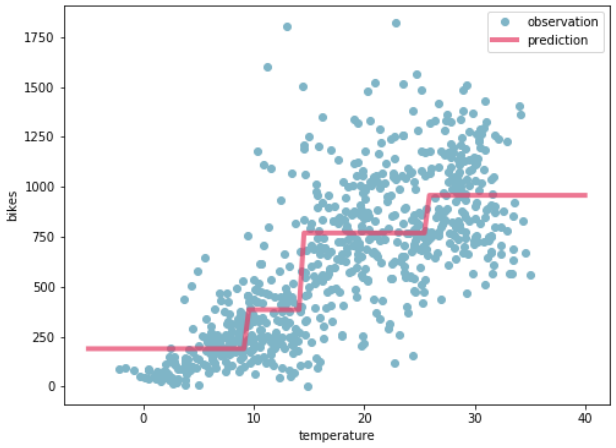
```
regressor.predict(5.)
# output
array([ 189.23183761])
```

Pertanyaan 2:
Berapa banyak sepeda yang di sewa saat suhu 20 derajat?

```
regressor.predict(20.)
# output
array([ 769.08756039])
```

Anda juga dapat melakukan visualisasi prediksi pada variasi suhu.

```
xx = np.array([np.linspace(-5, 40, 100)]).T

plt.figure(figsize=(8,6))
plt.plot(bikes['temperature'], bikes['count'], 'o', label='observation')
plt.plot(xx, regressor.predict(xx), linewidth=4, alpha=.7, label='prediction')
plt.xlabel('temperature')
plt.ylabel('bikes')
plt.legend()
plt.show()
```

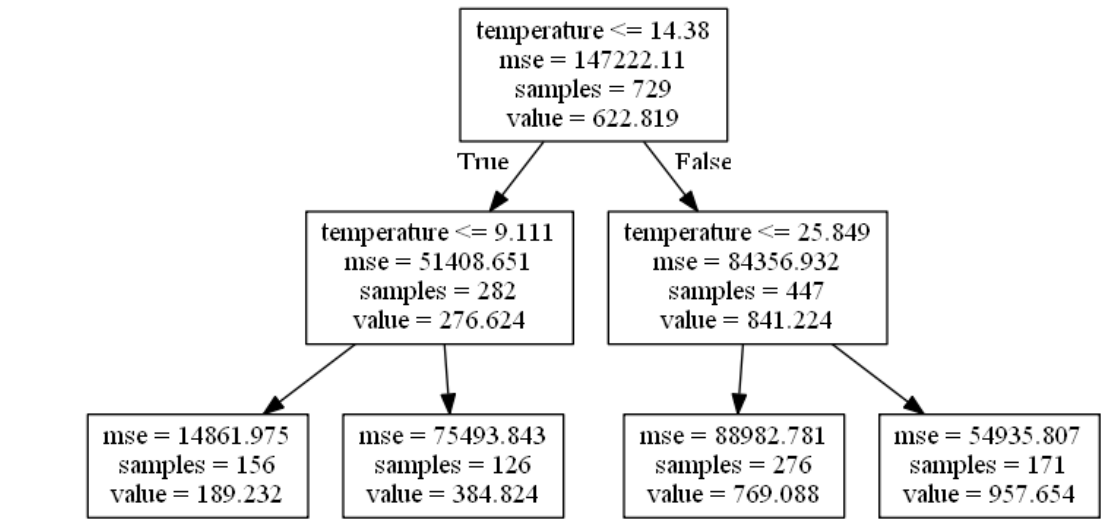



```
Decision Tree Regression
```

Merujuk pada gambar di atas, Anda dapat mencatat bahwa prediksi meningkat ketika suhu meningkat. Selain itu, Anda juga dapat mencatat bahwa prediksi merupakan fungsi yang bertahap. Hal ini sangat terkait dengan cara kerja Decision Tree. Decision tree membagi fitur input (hanya suhu dalam kasus ini) di beberapa wilayah dan memberikan nilai prediksi ke setiap wilayah. Pemilihan daerah dan nilai prediksi dalam suatu wilayah dipilih untuk menghasilkan prediksi yang paling sesuai dengan data. Data yang paling sesuai merupakan minimun jarak pengamatan dari prediksi.

Anda juga dapat memeriksa serangkaian aturan yang dibuat selama proses pelatihan.

```
penting!
silahkan anda install lib --> conda install python-graphviz

from sklearn.tree import export_graphviz
export_graphviz(regressor, out_file='tree.dot', feature_names=['temperature'])
from IPython.display import Image
Image(filename='tree.png')
```



Decision Tree

Aturan-aturan tersebut disusun dalam sebuah pohon biner: setiap kali Anda memperkirakan jumlah sepeda yang di sewa, suhu diperiksa dengan aturan mulai dari root sampai bagian bawah mengikuti jalan yang "di dikte" dari hasil aturan. Misalnya, masukan suhu adalah 16.5, pertama yang akan diperiksa adalah (suhu $\leq$ 14,3) lalu akan memberikan hasil negatif yang mengarahkan kita ke simpul anak kanannya di tingkat pohon berikutnya (suhu $\leq$ 25.8), kali ini Anda memiliki respon positif dan Anda akan berakhir pada anak kirinya. Node ini disebut sebagai leaf, yang berarti tidak mengandung aturan melainkan nilai prediksi. Pada kasus ini prediksi jumlah sepeda tersebut adalah 796 sepeda.
https://cambridgespark.com/content/tutorials/getting-started-with-regression-and-decision-trees/index.html

2. Decision Tree Classification

Decision Tree (DT) Classification adalah metode DT yang diterapkan pada kasus klasifikasi. Berikut gambaran mengenai DT Classification.

http://www.saedsayad.com/decision_tree.htm

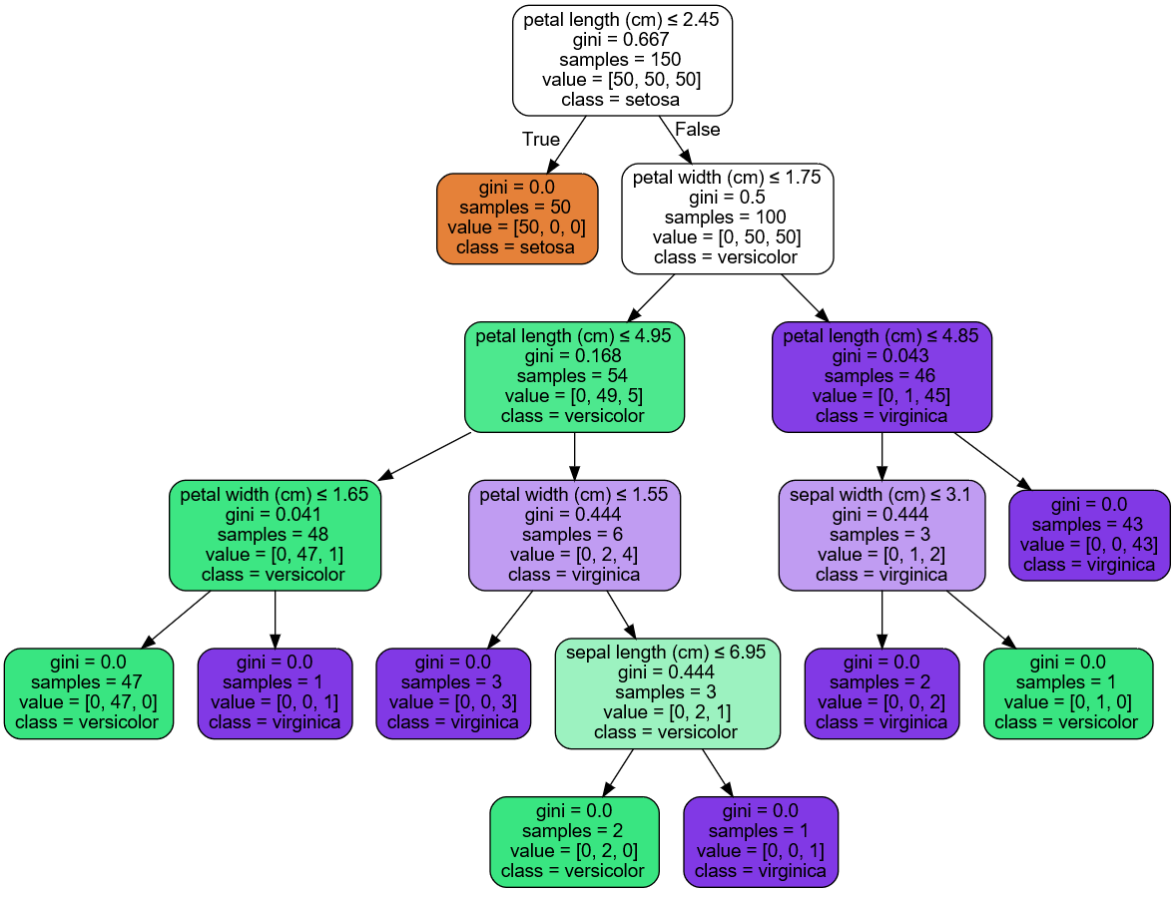
Contoh:
Pada tutorial kali ini Anda akan melakukan klasifikasi (multi kelas) pada data iris menggunakan metode Decision Tree Classification.

```
from sklearn.datasets import load_iris
from sklearn import tree

# load dataset iris
iris = load_iris()
```

```
# create model
clf = tree.DecisionTreeClassifier()
clf = clf.fit(iris.data, iris.target)
```

penting!
silahkan anda install lib --> conda install python-graphviz

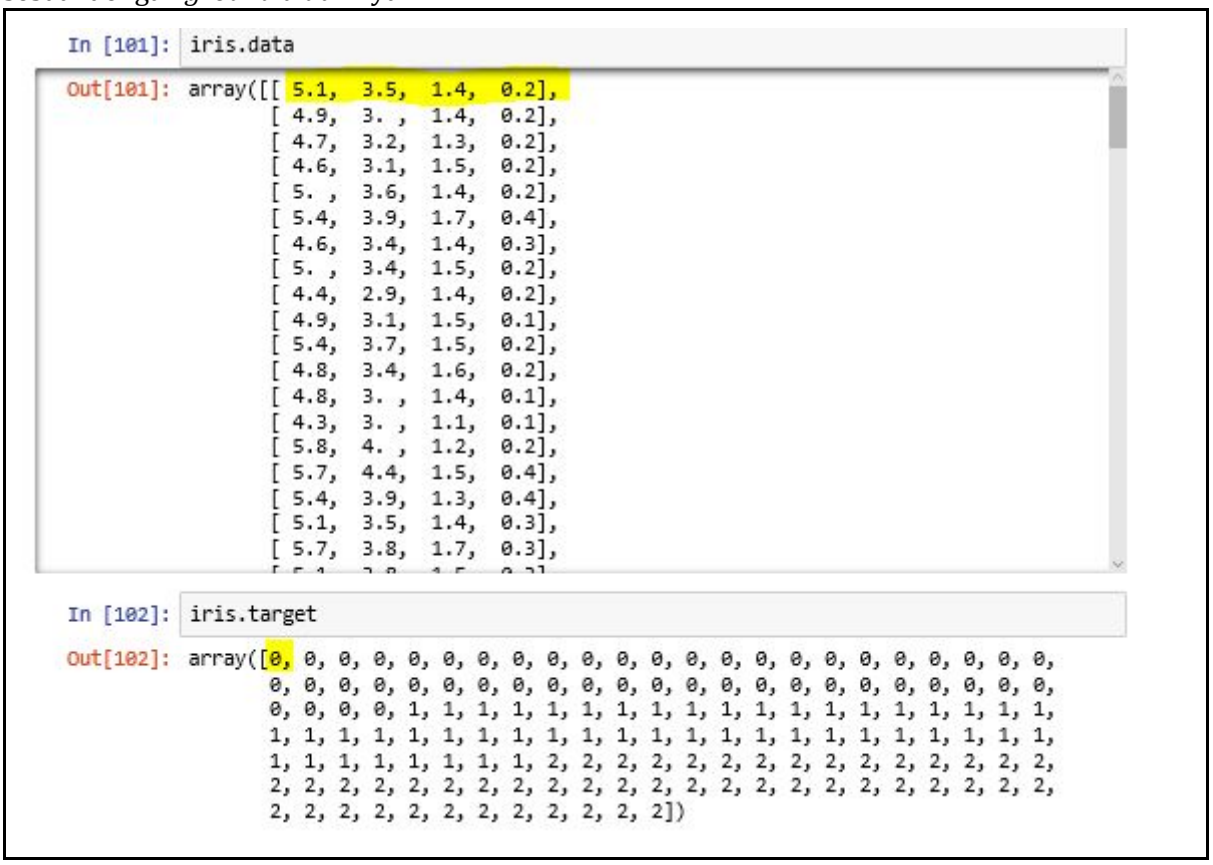
```
# visualisasi aturan pada decison tree menggunakan Lib graphviz
import graphviz
dot_data = tree.export_graphviz(clf, out_file=None)
graph = graphviz.Source(dot_data)
graph.render("iris")
dot_data = tree.export_graphviz(clf, out_file=None,
                         feature_names=iris.feature_names,
                         class_names=iris.target_names,
                         filled=True, rounded=True,
                         special_characters=True)
graph = graphviz.Source(dot_data)
graph
```



Anda akan melakukan prediksi pada data baris pertama.

```
clf.predict(iris.data[:1, :])
#output
array([0])
```

Bandingkan dengan data *ground truth*. Perhatikan tulisan yang di stabilo kuning! Hasil prediksi sesuai dengan *ground truth*-nya.

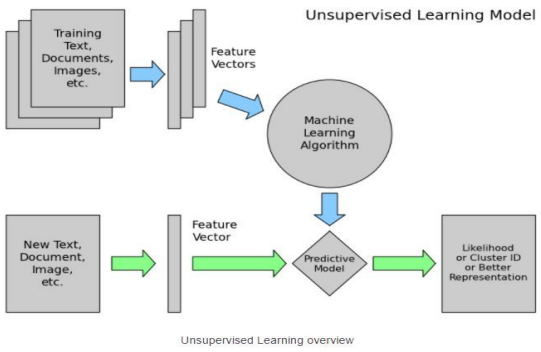


```
In [101]: iris.data

Out[101]: array([[ 5.1,  3.5,  1.4,  0.2],
                 [ 4.9,  3. ,  1.4,  0.2],
                 [ 4.7,  3.2,  1.3,  0.2],
                 [ 4.6,  3.1,  1.5,  0.2],
                 [ 5. ,  3.6,  1.4,  0.2],
                 [ 5.4,  3.9,  1.7,  0.4],
                 [ 4.6,  3.4,  1.4,  0.3],
                 [ 5. ,  3.4,  1.5,  0.2],
                 [ 4.4,  2.9,  1.4,  0.2],
                 [ 4.9,  3.1,  1.5,  0.1],
                 [ 5.4,  3.7,  1.5,  0.2],
                 [ 4.8,  3.4,  1.6,  0.2],
                 [ 4.8,  3. ,  1.4,  0.1],
                 [ 4.3,  3. ,  1.1,  0.1],
                 [ 5.8,  4. ,  1.2,  0.2],
                 [ 5.7,  4.4,  1.5,  0.4],
                 [ 5.4,  3.9,  1.3,  0.4],
                 [ 5.1,  3.5,  1.4,  0.3],
                 [ 5.7,  3.8,  1.7,  0.3],

In [102]: iris.target

Out[102]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
                 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
                 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

http://scikit-learn.org/stable/modules/tree.html

C. Machine Learning (Unsupervised Learning - Clustering)

Clustering atau sering disebut pembelajaran tidak terbimbing (*unsupervised learning*) merupakan pekerjaan yang memisahkan data/vektor ke dalam sejumlah kelompok (*cluster*) menurut karakteristiknya masing-masing. Data-data yang mempunyai kemiripan karakteristik akan berkumpul dalam *cluster* yang sama, dan data-data dengan karakteristik berbeda akan terpisah dalam *cluster* yang berbeda. Dalam clustering tidak diperlukan label kelas untuk setiap data karena nantinya label baru bisa diberikan ketika *cluster* sudah terbentuk. Berikut gambaran sederhana mengenai cara kerja dari *unsupervised learning*.



Unsupervised Learning overview

K-means

K-means merupakan salah satu metode *clustering* berbasis partisi yang melakukan partisi set data ke dalam sejumlah K *cluster* yang sudah ditetapkan di awal. **Perlu diingat bahwa, tipe data untuk k-means adalah data numerik sehingga harus dilakukan konversi data untuk atribut-atribut yang *non*-numerik**. Parameter yang harus dimasukkan ketika menggunakan k-means adalah nilai K (jumlah *cluster*). Nilai K yang digunakan biasanya didasarkan pada informasi yang diketahui sebelumnya tentang berapa banyak *cluster* yang muncul dari data, berapa banyak *cluster* yang dibutuhkan untuk penerapannya, atau jenis *cluster* dicari dengan mengeksplorasi/melakukan percobaan dengan beberapa nilai K.

Contoh 1:

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
import sklearn.metrics as sm

import pandas as pd
import numpy as np

%matplotlib inline
```

```
# import iris dataset
iris = datasets.load_iris()
```

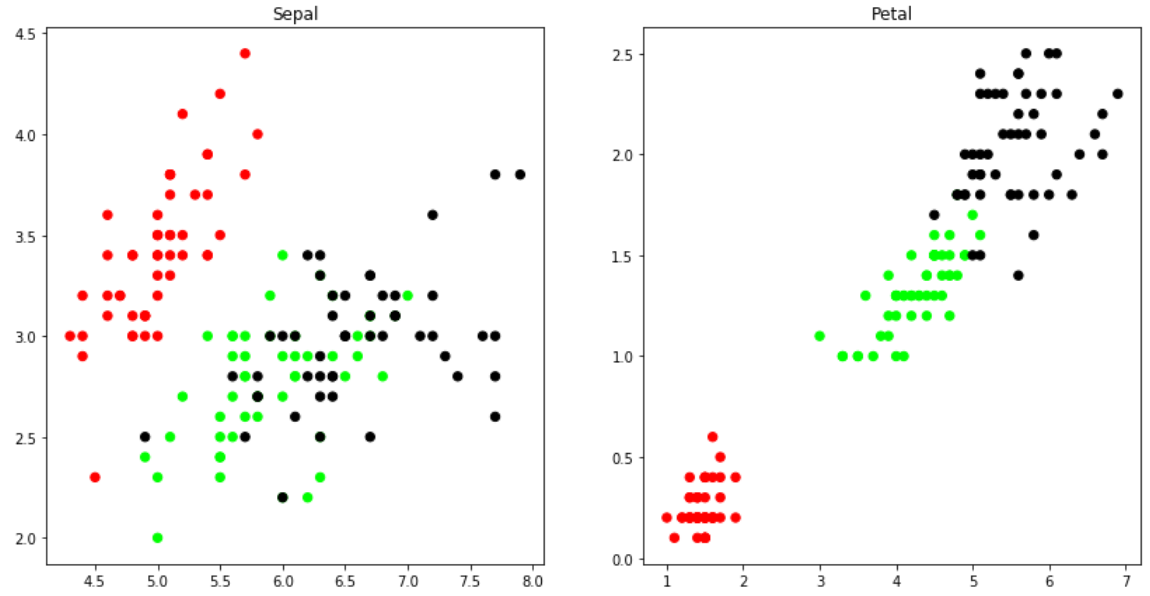
```
# deskripsi data
iris.data
iris.feature_names
iris.target
iris.target_names
```

```
# simpan input sebagai Dataframe dan set nama kolom
x = pd.DataFrame(iris.data)
x.columns = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']

y = pd.DataFrame(iris.target)
y.columns = ['Targets']
```

```
data = np.array(x)
```

```
# Set the size of the plot
plt.figure(figsize=(14,7))
```

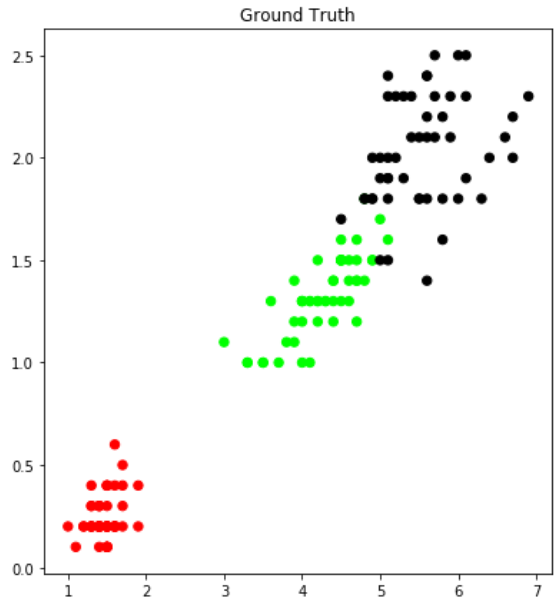
```
# Create a colormap
colormap = np.array(['red', 'lime', 'black'])

# Plot Sepal
plt.subplot(1, 2, 1)
plt.scatter(x.Sepal_Length, x.Sepal_Width, c=colormap[y.Targets], s=40)
plt.title('Sepal')

plt.subplot(1, 2, 2)
plt.scatter(x.Petal_Length, x.Petal_Width, c=colormap[y.Targets], s=40)
plt.title('Petal')
```

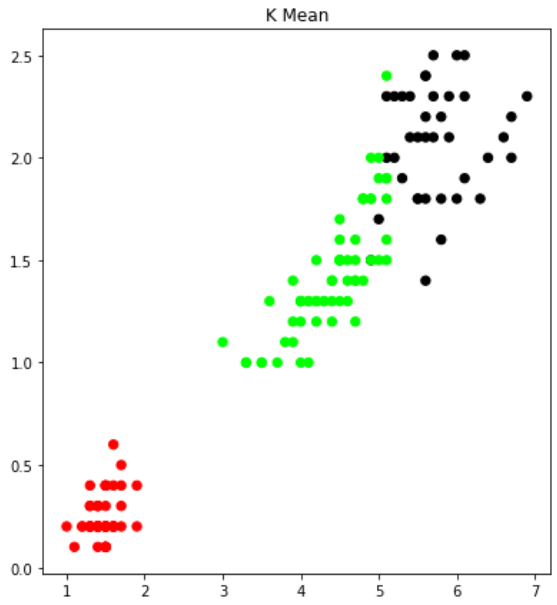



```
# K Means Cluster
model = KMeans(n_clusters=3)
model.fit(x)
```

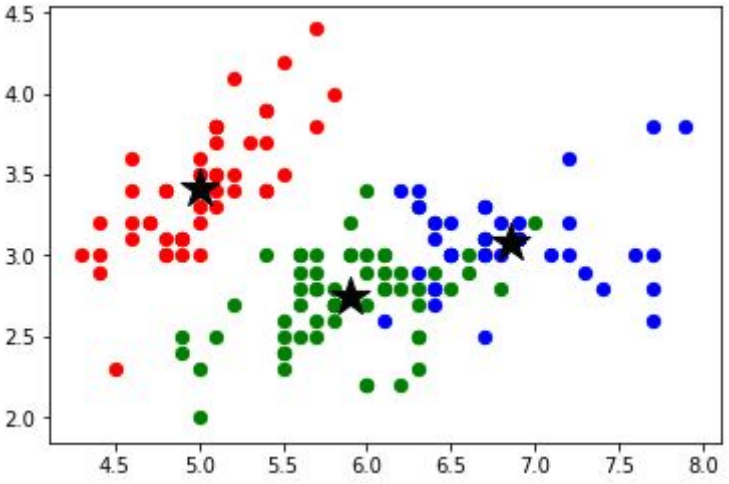
```
# The fix, we convert all the 1s to 0s and 0s to 1s.
predY = np.choose(model.labels_, [1, 0, 2]).astype(np.int64)
print (model.labels_)
print (predY)
```

```
# View the results
# Set the size of the plot
plt.figure(figsize=(14,7))

# Create a colormap
colormap = np.array(['red', 'lime', 'black'])

# Plot Orginal
plt.subplot(1, 2, 1)
plt.scatter(x.Petal_Length, x.Petal_Width, c=colormap[y.Targets], s=40)
plt.title('Ground Truth')

# Plot Predicted with corrected values
plt.subplot(1, 2, 2)
plt.scatter(x.Petal_Length, x.Petal_Width, c=colormap[predY], s=40)
plt.title('K Mean')
```






```
# Plot Centroid
colors = ['r', 'g', 'b']
k=3
labels = model.labels_
centroids = model.cluster_centers_
for i in range(k):
        points = np.array([x[j] for j in range(len(x)) if labels[j] == i])
        plt.scatter(points[:, 0], points[:, 1], s=40, c=colors[i])
plt.scatter(centroids[:, 0], centroids[:, 1], marker='*', s=400, c='#050505')
plt.show()
```



http://stamfordresearch.com/k-means-clustering-in-python/

Evaluasi hasil cluster

Evaluasi hasil *cluster* sebenarnya sangat bersifat subjektif, artinya sangat bergantung pada tujuan dan perspektif pengguna. Secara kuantitatif, ada 2 cara untuk memvalidasi hasil clustering yaitu menggunakan **validitas internal** dan **validitas eksternal**.

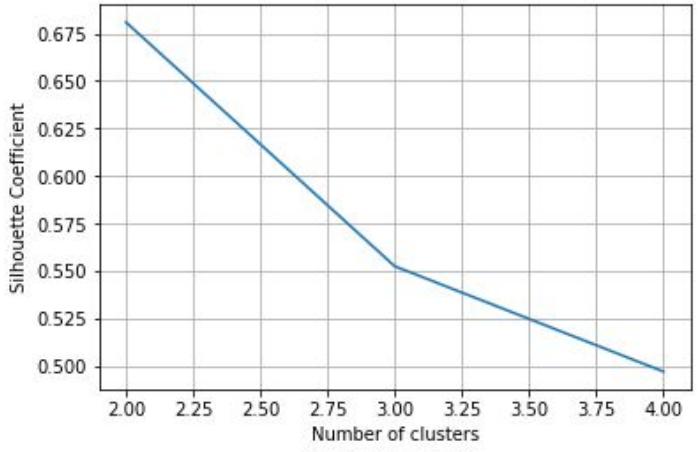
1. **Validitas internal**

   Validitas internal didasarkan pada nilai kohesi dan separasi. Kohesi dalam *clustering* didefinisikan sebagai jumlah dari kedekatan data terhadap *centroid* dari *cluster* yang diikutinya. Sedangkan separasi di antara dua *cluster* dapat diukur dengan kedekatan dua prototipe (*centroid*) *cluster*. Beberapa metode validasi internal di antaranya adalah Davies-Bouldin Index, Silhouette index, Dunn index, dan lain-lain.

   Pada bagian ini, akan dilakukan evaluasi hasil clustering dengan menggunakan **Silhouette Index (SI)**, metode ini merupakan metode yang paling banyak digunakan untuk validasi hasil *clustering* yang menggabungkan antara nilai kohesi dan separasi. Namun metode SI membutuhkan waktu yang cukup lama dan memori yang banyak untuk data yang besar.

```
from sklearn import metrics
# calculate SC for K=2 through K=5
k_range = range(2, 5)
scores = []
for k in k_range:
    km = KMeans(n_clusters=k, random_state=1)
    km.fit(x)
    scores.append(metrics.silhouette_score(x, km.labels_))
# plot the results
plt.plot(k_range, scores)
plt.xlabel('Number of clusters')
plt.ylabel('Silhouette Coefficient')
plt.grid(True)
print (scores)
```

```
[0.6808136202713507, 0.5525919445213676, 0.49722797262968016]
```



Nilai SI yang semakin tinggi menunjukkan hasil *cluster* yang semakin baik. *Unsupervised learning* sedikit berbeda dengan *supervised learning*, pada *unsupervised learning* biasanya

tidak dilakukan prediksi data baru tetapi hanya digunakan untuk menemukan pola dari data.

2. Validitas eksternal
   Validitas eksternal dapat dilakukan dengan mengetahui informasi eksternal (misal label kelas). Validitas dapat dilakukan dengan mengukur tingkat hubungan antara label *cluster* dengan label kelas seperti pada tahap klasifikasi.
   Contoh:

```
# Performance Metrics
sm.accuracy_score(y, predY)
# Confusion Matrix
sm.confusion_matrix(y, predY)
```