Tutorial 2

Machine Learning

Nama : Zaki Raihan

NPM : 1606878505

Basic Statistics

1. Import data insurance.csv dan hitung dimensi:

```
In [31]: import pandas as pd
         insurance = pd.read_csv("Dataset Tutorial 2/insurance.csv")
         insurance.head() #Look at the head of the data (just the first few rows)
```

Out[31]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

```
In [33]: insurance.shape
Out[33]: (1338, 7)
```

Dengan menggunakan command insurance.shape, saya mendapatkan bahwa terdapat 1338 Row dan 7 Column pada data insurance.csv

2. Mengubah semua kolom yang berisi kategorikal data menjadi numerik. Misal “male” menjadi 0, “female” menjadi 1. Simpan data hasil perubahan ini kedalam insurance_modif.csv

```
In [32]: insurance['sex'] = insurance['sex'].map({'female': 1, 'male': 0})
         insurance['smoker'] = insurance['smoker'].map({'yes': 1, 'no': 0})
         insurance['region'] = insurance['region'].map({'southwest': 0, 'southeast': 1, 'northwest': 2, 'northeast': 3})
```

```
In [33]: insurance.to_csv("insurance_modif.csv", sep=',', encoding='utf-8', index=False)
         insurance.head()
```

Out[33]:

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	1	0	16884.92400
1	18	0	33.770	1	0	1	1725.55230
2	28	0	33.000	3	0	1	4449.46200
3	33	0	22.705	0	0	2	21984.47061
4	32	0	28.880	0	0	2	3866.85520

3. Melakukan random sampling sederhana pada data Insurance dimana k = 15!

```
In [34]: insurance.sample(n=15)
```

Out[34]:

	age	sex	bmi	children	smoker	region	charges
1229	58	0	30.305	0	0	3	11938.25595
153	42	1	23.370	0	1	3	19964.74630
138	54	1	31.900	3	0	1	27322.73386
1054	27	1	21.470	0	0	2	3353.47030
591	47	0	19.570	1	0	2	8428.06930
718	51	1	36.670	2	0	2	10848.13430
724	50	1	27.075	1	0	3	10106.13425
1297	28	1	26.510	2	0	1	4340.44090
1331	23	1	33.400	0	0	0	10795.93733
1118	33	0	35.750	1	1	1	38282.74950
55	58	0	36.955	2	1	2	47496.49445
1179	31	0	29.810	0	1	1	19350.36890
900	49	0	22.515	0	0	3	8688.85885
183	44	1	26.410	0	0	2	7419.47790
798	58	1	33.100	0	0	0	11848.14100

4. Melakukan uji korelasi spearman dan pearson

Variable yang saya uji korelasinya adalah pengaruh '*age*' terhadap '*charges*', pengaruh '*bmi*' terhadap '*charges*' dan pengaruh '*smoker*' terhadap '*charges*'.

Saya memilih *age*, *bmi*, dan *smoker* karena saya merasa ketiga variable tersebut dapat mempengaruhi besaran biaya asuransi untuk seseorang (*charges*)

Uji korelasi dengan pearson:

```
In [35]: insurance_pearson__ageXcharges = insurance['charges'].corr(insurance['age'], method='pearson')
print ("Korelasi dari ages dan charges ('Pearson') = " + str(insurance_pearson__ageXcharges))
insurance_pearson__bmiXcharges = insurance['charges'].corr(insurance['bmi'], method='pearson')
print ("Korelasi dari bmi dan charges ('Pearson') = " + str(insurance_pearson__bmiXcharges))
insurance_pearson__smokerXcharges = insurance['charges'].corr(insurance['smoker'], method='pearson')
print ("Korelasi dari smoker dan charges ('Pearson') = " + str(insurance_pearson__smokerXcharges))
```

```
Korelasi dari ages dan charges ('Pearson') = 0.2990081933306476
Korelasi dari bmi dan charges ('Pearson') = 0.19834096883362884
Korelasi dari smoker dan charges ('Pearson') = 0.7872514304984767
```

Uji korelasi dengan spearman:

```
In [37]: insurance_spearman__ageXcharges = insurance['charges'].corr(insurance['age'], method='spearman')
print ("Korelasi dari ages dan charges ('spearman') = " + str(insurance_spearman__ageXcharges))
insurance_spearman__bmiXcharges = insurance['charges'].corr(insurance['bmi'], method='spearman')
print ("Korelasi dari bmi dan charges ('spearman') = " + str(insurance_spearman__bmiXcharges))
insurance_spearman__smokerXcharges = insurance['charges'].corr(insurance['smoker'], method='spearman')
print ("Korelasi dari smoker dan charges ('spearman') = " + str(insurance_spearman__smokerXcharges))
```

```
Korelasi dari ages dan charges ('spearman') = 0.534392133771846
Korelasi dari bmi dan charges ('spearman') = 0.11939590358331147
Korelasi dari smoker dan charges ('spearman') = 0.6634600597131322
```

5. Dari hasil pengujian korelasi terhadap variabel *age*, *bmi*, dan *smoker*, saya dapat melihat bahwa ketiga variabel tersebut memiliki korelasi positif terhadap besaran biaya asuransi yang dikenakan kepada seseorang (*charges*). Itu artinya jika yamng mengajukan merupakan perokok maka kemungkinan besaran biaya asuransi juga akan naik. Hal itu juga berlaku untuk *bmi* dan umur, walaupun kenaikannya tidak signifikan.

Regresi

1. Menggunakan data insurance_modiv.csv yang dihasilkan dari nomor 1

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

insurance_modif = pd.read_csv("insurance_modif.csv")
insurance_modif.head() #Look at the head of the data (just the first few rows)
```

Out[1]:

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	1	0	16884.92400
1	18	0	33.770	1	0	1	1725.55230
2	28	0	33.000	3	0	1	4449.46200
3	33	0	22.705	0	0	2	21984.47061
4	32	0	28.880	0	0	2	3866.85520

2. Memilih fitur yang digunakan sebagai model

```
feature_cols = ['age', 'smoker']
feature_data_in_insurance = insurance_modif[feature_cols]
label_data_in_insurance = insurance_modif['charges']
```

Saya memilih variabel *age* dan *smoker* karena nilai korelasinya yang cukup baik dengan variabel *charge*.

3. Menggunakan proporsi 80:20 untuk pembagian data training dan testing

```
from sklearn.model_selection import train_test_split

# Dividing data for training and test with 80:20 ratio
train, test, train_labels, test_labels = train_test_split(feature_data_in_insurance,
                                                          label_data_in_insurance,
                                                          test_size=0.2,
                                                          random_state=42)
```

Menggunakan Linear Regression untuk memprediksi nilai *charges* (data hasil prediksi tidak saya tampilkan semua):

Linear Regession

In [5]:
```
from sklearn.linear_model import LinearRegression

# instantiate a new model
linreg_insurance_modif = LinearRegression()

# fit the model to our data
model_linreg_insurance_modif = linreg_insurance_modif.fit(train, train_labels)

# Trying to predict with trained machine
preds_linreg_insurance_modif = linreg_insurance_modif.predict(test)

# Print result
print("Prediction: \n", preds_linreg_insurance_modif)
```

```
Prediction:
 [10033.0543404   7548.96991811 38965.83489094 10309.06372065
 26545.41277952  6996.9511576   2856.8104538  15277.23256522
  5340.89487608 11137.09186141 29581.51596231  8929.01681938
  5616.90425634 33997.66604637 37861.79736992 34273.67542662
 11137.09186141 34273.67542662 10309.06372065 33169.63790561
```

```
from sklearn import metrics
import numpy as np

# Count the MSE result from prediction with real label
print("Nilai MSE = ",metrics.mean_squared_error(test_labels, preds_linreg_insurance_modif))
print("Nilai MAE = ",metrics.mean_absolute_error(test_labels, preds_linreg_insurance_modif))
print("Nilai RMSE = ",np.sqrt(metrics.mean_squared_error(test_labels, preds_linreg_insurance_modif)))
```

```
Nilai MSE =  38274699.675041825
Nilai MAE =  3990.979515251796
Nilai RMSE =  6186.654966542244
```

Menggunakan DecisionTree Regressor untuk memprediksi nilai *charges* (data hasil prediksi tidak saya tampilkan semua):

Decision Tree Regression

In [6]:
```
from sklearn.tree import DecisionTreeRegressor
import matplotlib.pyplot as plt

# instantiate a new model
decision_tree_insurance_modif = DecisionTreeRegressor(max_depth=3)

# fit the model to our data
model_decision_tree_insurance_modif = decision_tree_insurance_modif.fit(train, train_labels)

# Trying to predict with trained machine
preds_decision_tree_insurance_modif = decision_tree_insurance_modif.predict(test)

# Print result
print("Prediction: \n", preds_decision_tree_insurance_modif)
```

```
Prediction:
 [10406.77138663  6350.82542396 41548.3632756  10406.77138663
 25740.60476472  6350.82542396  3364.76173199 13752.28669534
  6350.82542396 10406.77138663 29302.57654471  6350.82542396
  6350.82542396 35925.50978255 41548.3632756  35925.50978255
 10406.77138663 35925.50978255 10406.77138663 29302.57654471
  6350.82542396 10406.77138663  3364.76173199  3364.76173199
 10406.77138663 13752.28669534 13752.28669534  6350.82542396
 10406.77138663  3364.76173199  6350.82542396 13752.28669534
```

In [10]:
```
# Count the MSE result from prediction with real label
print("Nilai MSE = ",metrics.mean_squared_error(test_labels, preds_decision_tree_insurance_modif))
print("Nilai MAE = ",metrics.mean_absolute_error(test_labels, preds_decision_tree_insurance_modif))
print("Nilai RMSE = ",np.sqrt(metrics.mean_squared_error(test_labels, preds_decision_tree_insurance_modif)))
```

```
Nilai MSE =  39481103.90452745
Nilai MAE =  4106.097544656881
Nilai RMSE =  6283.399072518588
```

4. Dari hasil perhitungan menggunakan linear regression dan decision tree regressor, maka terlihat bahwa hasil MAE dari linear regression lebih kecil ketimbang menggunakan decision tree regressor.

Klasifikasi

1. menggunakan data adults.csv dan melakukan perubahan kolom yang berisi kategorikal data ke numerik

```
import pandas as pd
adults = pd.read_csv("Dataset Tutorial 2/adults.csv")
adults.head() #Look at the head of the data (just the first few rows)
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	class
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

```
adults[' class'].describe()
adults[' class'] = adults[' class'].map({' <=50K': 0, ' >50K': 1})
adults[' sex'] = adults[' sex'].map({' Female': 0, ' Male': 1})
adults['workclass'] = adults['workclass'].map({' Private':0, ' Self-emp-not-inc':1, ' Self-emp-inc':2, ' Federal-gov':3, ' Local-
adults[' race'] = adults[' race'].map({' White': 0, ' Black': 4, ' Asian-Pac-Islander':1, ' Amer-Indian-Eskimo':2, ' Other':3})
adults[' occupation'] = adults[' occupation'].map({" Tech-support":0, " Craft-repair":1, " Other-service":2, " Sales":3, " Exec-m
adults[' marital-status'] = adults[' marital-status'].map({" Married-civ-spouse":0, " Divorced":1, " Never-married":2, " Separate
adults[' relationship'] = adults[' relationship'].map({" Wife":0, " Own-child":1, " Husband":2, " Not-in-family":3, " Other-relat
adults.head() #Look at the head of the data (just the first few rows)
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	class
0	39	5.0	77516	Bachelors	13	2	8.0	3	0	1	2174	0	40	United-States	0
1	50	1.0	83311	Bachelors	13	0	4.0	2	0	1	0	0	13	United-States	0
2	38	0.0	215646	HS-grad	9	1	6.0	3	0	1	0	0	40	United-States	0
3	53	0.0	234721	11th	7	0	6.0	2	4	1	0	0	40	United-States	0
4	28	0.0	338409	Bachelors	13	0	5.0	0	4	0	0	0	40	Cuba	0

*note: jika gambar kurang jelas dapat melihat Tutorial 2 - Klasifikasi.ipynb yang telah disertakan

2. mamiilih fitur yang akan digunakan sebagai model
   Saya memilih *age, education-num* (representasi dari *education*), *capital-gain,* dan *hours-per-week* sebagai vitur yang saya gunakan untuk mengklasifikasikan tingkatan pendapatan seseorang atau class(<=50K ataupun >50K)
   Saya memilih keempat fitur tersebut karena memiliki korelasi yang cukup kuat dengan variabel class, hal itu terlihat dari perhitungan korelasi dengan menggunakan pearson dan spearman di bawah:
   Pearson:

	age	workclass	fnlwgt	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	class
age	1.0	0.12	-0.077	0.037	-0.22	0.021	0.13	-0.03	0.089	0.078	0.058	0.069	0.23
workclass	0.12	1.0	-0.022	0.081	0.011	0.13	-0.019	0.038	-0.047	0.0012	0.0068	-0.09	0.015
fnlwgt	-0.077	-0.022	1.0	-0.043	0.024	0.0039	0.015	0.098	0.027	0.00043	-0.01	-0.019	-0.0095
education-num	0.037	0.081	-0.043	1.0	-0.11	-0.037	-0.032	-0.079	0.012	0.12	0.08	0.15	0.34
marital-status	-0.22	0.011	0.024	-0.11	1.0	0.017	0.36	0.13	-0.38	-0.074	-0.067	-0.22	-0.38
occupation	0.021	0.13	0.0039	-0.037	0.017	1.0	0.011	0.042	-0.044	-0.012	-0.015	0.04	-0.045
relationship	0.13	-0.019	0.015	-0.032	0.36	0.011	1.0	0.12	-0.17	-0.027	-0.031	0.057	-0.17
race	-0.03	0.038	0.098	-0.079	0.13	0.042	0.12	1.0	-0.12	-0.02	-0.024	-0.054	-0.097
sex	0.089	-0.047	0.027	0.012	-0.38	-0.044	-0.17	-0.12	1.0	0.048	0.046	0.23	0.22
capital-gain	0.078	0.0012	0.00043	0.12	-0.074	-0.012	-0.027	-0.02	0.048	1.0	-0.032	0.078	0.22
capital-loss	0.058	0.0068	-0.01	0.08	-0.067	-0.015	-0.031	-0.024	0.046	-0.032	1.0	0.054	0.15
hours-per-week	0.069	-0.09	-0.019	0.15	-0.22	0.04	0.057	-0.054	0.23	0.078	0.054	1.0	0.23
class	0.23	0.015	-0.0095	0.34	-0.38	-0.045	-0.17	-0.097	0.22	0.22	0.15	0.23	1.0

Spearman:

	age	workclass	fnlwgt	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	class
age	1.0	0.17	-0.078	0.066	-0.36	0.019	0.17	-0.028	0.1	0.12	0.058	0.14	0.27
workclass	0.17	1.0	-0.04	0.11	-0.06	0.095	-0.024	0.015	0.0065	0.031	0.019	-0.024	0.059
fnlwgt	-0.078	-0.04	1.0	-0.036	0.032	0.0046	0.012	0.054	0.025	-0.006	-0.0069	-0.022	-0.011
education-num	0.066	0.11	-0.036	1.0	-0.096	0.0098	-0.0037	-0.058	0.0063	0.12	0.075	0.17	0.33
marital-status	-0.36	-0.06	0.032	-0.096	1.0	0.022	0.37	0.13	-0.4	-0.13	-0.074	-0.26	-0.42
occupation	0.019	0.095	0.0046	0.0098	0.022	1.0	0.011	0.031	-0.064	0.00098	-0.01	0.027	-0.027
relationship	0.17	-0.024	0.012	-0.0037	0.37	0.011	1.0	0.089	-0.16	-0.029	-0.022	0.066	-0.16
race	-0.028	0.015	0.054	-0.058	0.13	0.031	0.089	1.0	-0.11	-0.03	-0.021	-0.08	-0.088
sex	0.1	0.0065	0.025	0.0063	-0.4	-0.064	-0.16	-0.11	1.0	0.067	0.042	0.26	0.22
capital-gain	0.12	0.031	-0.006	0.12	-0.13	0.00098	-0.029	-0.03	0.067	1.0	-0.067	0.093	0.28
capital-loss	0.058	0.019	-0.0069	0.075	-0.074	-0.01	-0.022	-0.021	0.042	-0.067	1.0	0.06	0.14
hours-per-week	0.14	-0.024	-0.022	0.17	-0.26	0.027	0.066	-0.08	0.26	0.093	0.06	1.0	0.27
class	0.27	0.059	-0.011	0.33	-0.42	-0.027	-0.16	-0.088	0.22	0.28	0.14	0.27	1.0

3. Menggunakan proporsi 80:20 untuk melakukan pembagian data training dan testing.

```
feature_cols = ['age',' education-num', ' capital-gain', ' hours-per-week']
feature_data_in_adults = adults[feature_cols]
label_data_in_adults = adults[' class']

from sklearn.model_selection import train_test_split

# Dividing data for training and test with 80:20 ratio
train, test, train_labels, test_labels = train_test_split(feature_data_in_adults,
                                                          label_data_in_adults,
                                                          test_size=0.2,
                                                          random_state=42)
```

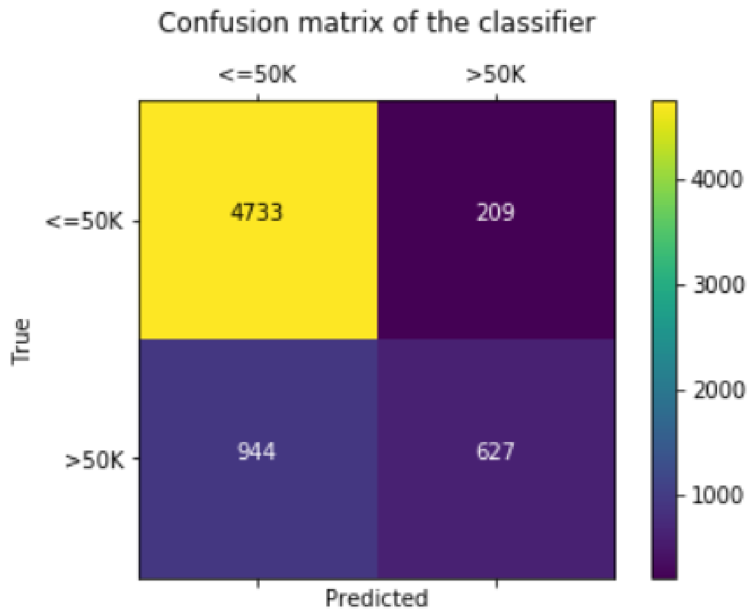
Menggunakan algoritma Decision Tree untuk melakukan pengelompokkan class di data tersebut.

```
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt

# instantiate a new model
decision_tree_adults = DecisionTreeClassifier(max_depth=4, random_state=1)

# fit the model to our data
model_decision_tree_adults = decision_tree_adults.fit(train, train_labels)

# Trying to predict with trained machine
preds_decision_tree_adults = decision_tree_adults.predict(test)

# Print result
pd.DataFrame({'feature':feature_cols, 'importance':decision_tree_adults.feature_importances_})
```

	feature	importance
0	age	0.211062
1	education-num	0.244790
2	capital-gain	0.480501
3	hours-per-week	0.063646

4. Visualisasi dari confussion matrix hasil klasifikasi:

```
[[4733  209]
 [ 944  627]]
```



5. Akurasi, precision dan recall pada hasil klasifikasi model:

```
accuracy score:
 0.8229694457239367


precision score:
 0.75


recall score:
 0.39910884786760026
```

Code untuk menampilkan confussion matrix, akurasi, precision dan recall:

Confussion Matrix:

```python
# Plot confusion matrix
from sklearn.metrics import confusion_matrix
import itertools

labels = ['<=50K', '>50K']
conf_mat = confusion_matrix(test_labels,preds_decision_tree_adults)
print(conf_mat)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(conf_mat)
plt.title('Confusion matrix of the classifier\n')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
fmt = 'd'
thresh = conf_mat.max() / 2.
for i, j in itertools.product(range(conf_mat.shape[0]), range(conf_mat.shape[1])):
    plt.text(j, i, format(conf_mat[i, j], fmt),
             horizontalalignment="center",
             color="black" if conf_mat[i, j] > thresh else "white")
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

Akurasi, precision dan recall:

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score

print('accuracy score: \n', accuracy_score(test_labels, preds_decision_tree_adults))
print("\n")
print('precision score: \n', precision_score(test_labels,preds_decision_tree_adults))
print("\n")
print('recall score: \n', recall_score(test_labels,preds_decision_tree_adults))
```

6. Penjelasan mengenai interpretasi hasil evaluasi dan error analysis.
   Berdasarkan confussion matrix terlohat bahwa:
   - Banyaknya data diprediksi termasuk dalam kelas <=50K dan kenyataannya benar termasuk kelas <=50K adalah sebanyak: 4733 data
   - Banyaknya data diprediksi termasuk dalam kelas <=50K dan kenyataannya termasuk kelas >50K adalah sebanyak: 944 data
   - Banyaknya data diprediksi termasuk dalam kelas >50K dan kenyataannya benar termasuk kelas >50K adalah sebanyak: 627 data
   - Banyaknya data diprediksi termasuk dalam kelas >50K dan kenyataannya termasuk kelas <=50K adalah sebanyak: 209 data
   - Dari sini dapat dilihat bahwa peluang perkiraan benar sekitar: (4733+627)/6513 = 0.82 atau 82%, sesuai dengan accuracy score

Clustering

1. Menggunakan data water-treatment.csv sebagai input data untuk clustering.

```python
import pandas as pd
water_treatment = pd.read_csv("Dataset Tutorial 2/water-treatment.csv", index_col=False)
water_treatment.head() #Look at the head of the data (just the first few rows)
```

	Date	Q-E	ZN-E	PH-E	DBO-E	DQO-E	SS-E	SSV-E	SED-E	COND-E	...	COND-S	RD-DBO-P	RD-SS-P	RD-SED-P	RD-DBO-S	RD-DQO-S	RD-DBO-G	RD-DQO-G	RD-SS-G	RD-SED-G
0	D-1/3/90	44101.0	1.5	7.8	NaN	407.0	166.0	66.3	4.5	2110	...	2000.0	NaN	58.8	95.5	NaN	70.0	NaN	79.4	87.3	99.6
1	D-2/3/90	39024.0	3.0	7.7	NaN	443.0	214.0	69.2	6.5	2660	...	2590.0	NaN	60.7	94.8	NaN	80.8	NaN	79.5	92.1	100.0
2	D-4/3/90	32229.0	5.0	7.6	NaN	528.0	186.0	69.9	3.4	1666	...	1888.0	NaN	58.2	95.6	NaN	52.9	NaN	75.8	88.7	98.5
3	D-5/3/90	35023.0	3.5	7.9	205.0	588.0	192.0	65.6	4.5	2430	...	1840.0	33.1	64.2	95.3	87.3	72.3	90.2	82.3	89.6	100.0
4	D-6/3/90	36924.0	1.5	8.0	242.0	496.0	176.0	64.8	4.0	2110	...	2120.0	NaN	62.7	95.6	NaN	71.0	92.1	78.2	87.5	99.5

5 rows × 39 columns

```python
water_treatment = water_treatment.fillna(water_treatment.mean())
water_treatment.head()
```

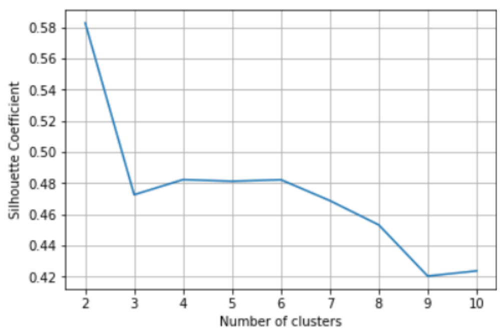
	Date	Q-E	ZN-E	PH-E	DBO-E	DQO-E	SS-E	SSV-E	SED-E	COND-E	...	COND-S	RD-DBO-P	RD-SS-P	RD-SED-P	RD-DBO-S	RD-DQO-S	RD-DBO-G	RD-DQO-G	RD-SS-G	RD SED
0	D-1/3/90	44101.0	1.5	7.8	188.714286	407.0	166.0	66.3	4.5	2110	...	2000.0	39.085806	58.8	95.5	83.448049	70.0	89.013646	79.4	87.3	99.
1	D-2/3/90	39024.0	3.0	7.7	188.714286	443.0	214.0	69.2	6.5	2660	...	2590.0	39.085806	60.7	94.8	83.448049	80.8	89.013646	79.5	92.1	100.
2	D-4/3/90	32229.0	5.0	7.6	188.714286	528.0	186.0	69.9	3.4	1666	...	1888.0	39.085806	58.2	95.6	83.448049	52.9	89.013646	75.8	88.7	98.
3	D-5/3/90	35023.0	3.5	7.9	205.000000	588.0	192.0	65.6	4.5	2430	...	1840.0	33.100000	64.2	95.3	87.300000	72.3	90.200000	82.3	89.6	100.
4	D-	36924.0	1.5	8.0	242.000000	496.0	176.0	64.8	4.0	2110		2120.0	39.085806	62.7	95.6	83.448049	71.0	92.100000	78.2	87.5	99

*note: saya melakukan persiapan data (data preparation) dengan asumsi bahwa data dengan '?' merupakan data yang tidak diketahui nilainya dan saya ganti dengan menggunakan rata2 dari kolom yang bersangkutan (jia '?' berada kolom Q-E maka nilainya saya ganti dengan rata-rata nilai Q-E)

2. Apakah anda memerlukan reduksi dimensi pada data tersebut?, Jika Iya berapa jumlah fitur yang anda gunakan dan berapa jumlah fitur yang anda hilangkan?
   Saya melakukan reduksi dengan menggunakan feature extraction dengan menggunakan modul PCA, dengan begitu maka tidak ada fitur yang dihilangkan melainkan saya menggabungkannya sehingga total hanya ada 10 fitur dari 38 fitur awal

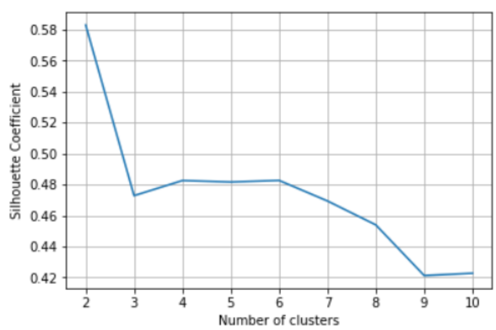
3. Hasil K-Means semua fitur:

```
K-Means Semua Fitur
    K      time     score
0   2  0.142211  0.582743
1   3  0.156272  0.472483
2   4  0.109395  0.482187
3   5  0.156273  0.481128
4   6  0.125021  0.482109
5   7  0.156273  0.468700
6   8  0.160929  0.453114
7   9  0.140638  0.420268
8  10  0.162448  0.423610
```



Hasil K-Means 10 fitur

```
K-Means 10 Fitur (Setelah Reduksi)
    K      time     score
0   2  0.101152  0.582906
1   3  0.125019  0.472774
2   4  0.093763  0.482529
3   5  0.125016  0.481602
4   6  0.109393  0.482598
5   7  0.131885  0.469342
6   8  0.127524  0.453902
7   9  0.134207  0.421217
8  10  0.100032  0.422698
```



4. Dari perbandingan kedua tabel hasil perhitungan K-Means untuk model diatas terlihat bahwa jika kita melakukan feature extraction maka waktu yang kita butuhkan akan lebih cepat ketimbang menggunakan semua fitur yang ada namun tidak begitu memiliki pengaruh yang berarti pada nilai Sihoulette Coeficientnya, sehingga dapat disimpulkan bahwa menggunakan feature extraction lebih efisien ketimbang menggunakan seluruh fitur yang ada.