MODUL 1: Prediksi harga rumah menggunakan model regresi linier dan model regresi polinomial

Tujuan Praktikum:

1. Melakukan data *prepocessing* pada dataset sebelum digunakan untuk melatih model regresi linier
2. Melatih model regresi linier untuk memprediksi harga rumah berdasarkan lokasi, kondisi linkungan, kondisi rumah, dan tarif pajak
3. Melatih model regresi polinomial untuk memprediksi harga rumah berdasarkan lokasi, kondisi linkungan, kondisi rumah, dan tarif pajak

Alat dan Bahan

1. Dataset Boston housing prices
2. AI workstation

Teori dan Definisi

1. Dataset Boston housing price

   Dataset Boston housing price berisi data yang berkaitan dengan harga rumah di kota Boston. Dataset ini dikumpulkan oleh U.S Census Service dan disimpan dalam arsip StatLib di Carnegie Mellon University. Pada dataset ini terdapat 506 data dengan 13 fitur dan satu target output (MDEV). Berikut ini adalah fitur-fitur dan target output pada dataset Boston house price:

   – CRIM: per capita crime rate by town

   – ZN: proportion of residential land zoned for lots over 25,000 sq.ft.

   – INDUS: proportion of non-retail business acres per town

   – CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)

   – NOX: nitric oxides concentration (parts per 10 million)

   – RM: average number of rooms per dwelling

   – AGE: proportion of owner-occupied units built prior to 1940

   – DIS: weighted distances to five Boston employment centres

   – RAD: index of accessibility to radial highways

   – TAX: full-value property-tax rate per $10,000

   – PTRATIO: pupil-teacher ratio by town

   – B: 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town

   – LSTAT: % lower status of the population

   – MEDV: median value of owner-occupied homes in $1000's

   Contoh lima data pertama pada dataset Boston housing price adalah sebagai berikut.

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

2. *Data preprocessing*

   Data preprocessing adalah langkah dalam machine learning yang mentransformasi data asli ke bentuk baru untuk memfasilitasi langkah selanjutnya. Salah satu metode dalam data *preprocessing* adalah penskalaan. Beberapa algoritma machine learning memerlukan asumsi bahwa semua fitur harus berada pada interval yang sama, misal [0,1]. Misalkan fitur X dengan nilai $x_1, x_2, \ldots, x_n$ akan di transformasi menjadi X' dengan nilai $x'_1, x'_2, \ldots, x'_n$ sedemikian rupa sehingga $x'_i \in [a, b]$. Untuk melakukan penskalaan agar nilai fitur berada pada $[a,b]$ dapat digunakan transformasi berikut.

$$x'_i = a + \frac{\big(x_i - max(X)\big)(b-a)}{max(X) - min(X)}$$

3. Regresi linier

   Model regresi adalah salah satu model machine learning yang digunakan untuk memprediksi nilai numerik dari target output (variabel tak bebas) berdasarkan nilai fitur input (variabel bebas). Berdasarkan bentuk hubungan antara input fitur dan target output model regresi dibagi menjadi dua bentuk, yaitu regresi linier dan regresi non linier. Regresi linier menggunakan asumsi bahwa hubungan antara input fitur dan target output adalah fungsi linier dari parameter model (koefisien/bobot dan bias/suku konstan) seperti pada persamaan berikut.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \varepsilon$$

   Dengan y adalah target output, $x_1, x_2, \ldots, x_n$ ada fitur input, $\beta_0, \beta_1, \beta_2, \ldots, \beta_n$ adalah parameter model, dan ε adalah suku error. Nilai $\beta_0, \beta_1, \beta_2, \ldots, \beta_n$ akan diestimasi pada saat proses *training* menggunakan data *training*. Salah satu metode yang dapat digunakan untuk mengestimasi nilai $\beta_0, \beta_1, \beta_2, \ldots, \beta_n$ adalah metode *ordinary least square* (OLS). Metode ini mencari nilai $\beta_0, \beta_1, \beta_2, \ldots, \beta_n$ yang meminimumkan jumlah kuadrat error (SSE) seperti pada persamaan berikut.

$$SSE = \sum_{i=1}^{m} \varepsilon_i^2 = \sum_{i=1}^{m} \big(y_i - (\beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \ldots + \beta_n x_{ni})\big)^2$$

4. Regresi polinomial

Regresi polinomial menggunakan asumsi bahwa hubungan antara input fitur dan target output adalah fungsi linier dari parameter model tetapi merupakan fungi nonlinier (fungsi polinomial) dari fitur input. Untuk melatihan model regresi polinomial, fitur input perlu ditransformasi dahulu menjadi fitur polinomial derajan n.

5. Evaluasi model regresi linier

   Beberapa nilai yang dapat digunakan untuk mengevaluasi model regresi linier adalah sebagai berikut:

   - Mean squared error (MSE)

$$MSE = \frac{1}{m} SSE$$

   - Coefficient of determination (R^2)

$$R^2 = \frac{SSR}{SST} = \frac{\sum_{i=1}^{m} (\hat{y}_i - \underline{y})^2}{\sum_{i=1}^{m} (y_i - \underline{y})^2}$$

   Dengan $\hat{y}_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \ldots + \beta_n x_{ni}$ dan $\underline{y}$ adalah rata-rata dari y.

   Nilai makasimum dari R^2 adalah 1 dan dapat bernilai negatif. Suatu model regresi linier yang baik dicirikan dengan nilai MSE yang kecil (dekat dengan 0) dan nilai R^2 yang besar (dekat dengan 1)

Langkah Kerja

1. *Import library* yang diperlukan untuk data melakukan data *preprocessing* dan melatih model regresi linier

```
from sklearn.datasets import load_boston
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
```

2. Load dataset Boston *house price* dan tampilkan ukuran datanya

```
boston = load_boston()
features = boston.data
target = boston.target
print("Features shape :" , features.shape)
print("Target shape :", target.shape)
```

```
Features shape : (506, 13)
Target shape : (506,)
```

3. Tampilkan beberapa contoh data dari dataset Boston *house price*

```
df_boston = pd.DataFrame(boston.data, columns=boston.feature_names)
df_boston['MEDV'] = boston.target
df_boston.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

4. Bagi dataset menjadi data *training* (60%) dan data *testing* (40%)

```
xtrain,xtest,ytrain,ytest=train_test_split(features,target,test_size=0.4,random_state=1)
print(xtrain.shape,xtest.shape)
print(ytrain.shape,ytest.shape)
```

```
(303, 13) (203, 13)
(303,) (203,)
```

5. Lakukan data *preprocesing* pada data *training* dan *testing* dengan mentransformasi semua fitur agar nilainya berada pada selang [0,1]

```
scaler=MinMaxScaler()
xstrain=scaler.fit_transform(xtrain)
xstest=scaler.transform(xtest)
```

6. Latih model regresi linier menggunakan semua fitur di data *training* yang sudah diskalakan. Kemudian tampilkan koefisien regresi dan suku kostannya

```
lr=LinearRegression()
lr.fit(xstrain,ytrain)
print('constant term:', lr.intercept_)
print('coefficients:',lr.coef_)
```

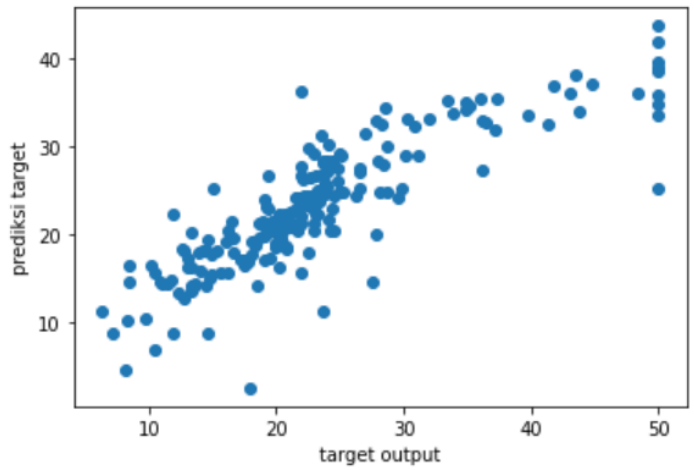
```
constant term: 25.572341408049482
coefficients: [ -6.58598655   6.73132853   1.37668315   2.18579583  -8.24138541
  16.48957721   0.19324514 -15.01197421   6.66026451  -6.42948378
  -7.84788938   3.72928282 -17.65541145]
```

7. Evaluasi model yang telah dilatih menggunakan data *testing* dengan menghitung MSE dan R^2 serta plot hubungan antara target output dan hasil prediksinya

```
ypred=lr.predict(xstest)
mse=mean_squared_error(ytest,ypred)
r2=lr.score(xstest,ytest)
print('MSE:',mse)
print('R^2:',r2)
```

```
MSE: 25.205774702366604
R^2: 0.7209056672661766
```

```
plt.figure()
plt.scatter(ytest,ypred)
plt.xlabel('target output')
plt.ylabel('prediksi target')
plt.show()
```



8. Transformasi semua fitur menjadi fitur polinomial derajat 2

```
polynomial = PolynomialFeatures(degree=2, include_bias=False)
xsptrain=polynomial.fit_transform(xstrain)
xsptest=polynomial.transform(xstest)
```

9. Latih model regresi linier menggunakan fitur polinomial

```
lr=LinearRegression()
lr.fit(xsptrain,ytrain)
print('constant term:', lr.intercept_)
print('coefficients:',lr.coef_)
```

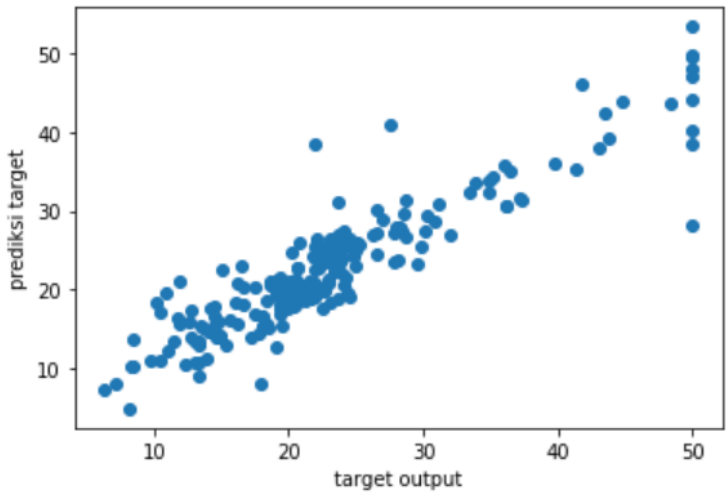
```
constant term: 5.944828104104683
coefficients: [ 2.75258545e-01  3.96140945e+01  2.99458996e+01 -2.56250885e+01
  3.03388896e+01 -2.42448039e+01 -6.93914683e+00  2.72599101e+01
  4.29783784e+00  3.66400658e+00 -4.86734388e+00 -1.10035342e+01
 -2.58692806e+01 -3.84752830e+01  1.73451207e-02 -1.60167559e+01
 -2.58867573e+00  2.39832031e+01  1.37412650e+00 -4.63656957e+01
  1.98962655e+00  3.15227757e+00  4.16901456e+01 -3.57840497e+01
 -1.31983648e+01 -7.23118485e+00  3.86504194e+01]
```

10. Evaluasi model yang telah dilatih menggunakan fitur polinomial

```
ypred=lr.predict(xstest)
mse=mean_squared_error(ytest,ypred)
r2=lr.score(xstest,ytest)
print('MSE:',mse)
print('R^2:',r2)
```

```
MSE: 25.205774702366604
R^2: 0.7209056672661766
```

```
plt.figure()
plt.scatter(ytest,ypred)
plt.xlabel('target output')
plt.ylabel('prediksi target')
plt.show()
```



MODUL 2: Klasifikasi persetujuan pengajuan kredit menggunakan artificial neural network

Tujuan Praktikum

Alat dan Bahan

1. Dataset simulasi histori pengajuan kredit bank
2. AI Workstation

Teori dan Definisi

1. Dataset simulasi histori pengajuan kredit bank

   Dataset pengajuan kredit bank didapat dari industri perbankan. Dataset ini berisi histori persetujuan pengajuan kredit yang diajukan oleh nasabah dan disetujui atau tidak oleh pihak bank. Beberapa industri perbankan masih menerapkan persetujuan kredit secara manual. Persetujuan secara manual membutuhkan waktu yang lama. Dengan memanfaatkan algoritma klasifikasi di Machine Learning, proses penentuan persetujuan kredit ini bisa diotomatisasi atau paling tidak bisa memberikan rekomendasi oleh pihak bank. Berikut ini adalah contoh histori pengajuan kredit bank.

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
1	15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101348.88	1
2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
3	15619304	Onio	502	France	Female	42	8	159660.8	3	1	0	113931.57	1
4	15701354	Boni	699	France	Female	39	1	0	2	0	0	93826.63	0
5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.1	0

2. *Encoding*

   Encoding adalah salah satu tahap data pre-processing dalam algoritma *machine learning*. Dalam mengerjakan projek *data science* ataupun *machine learning*, kita akan sangat mungkin menemukan satu atau beberapa fitur yang bertipe kategori, misalnya ‘Sangat Baik’, ‘Baik’, ‘Tidak Baik. Beberapa algoritma klasifikasi tidak dapat memproses data dalam bentuk kategori, sehingga diperlukan untuk mengubah data tersebut menjadi data numerik. Proses ini disebut dengan *encoding*. Untuk nilai yang memiliki 2 jenis kategori, seperti jenis kelamin, yaitu: *male* dan *female*, setelah dilakukan encoding maka setelah dilakukan encoding, maka nilai *male* menjadi 1 dan *female* menjadi 0.

   One-Hot encoding adalah salah satu metode *encoding*. Metode ini merepresentasikan data bertipe kategori sebagai vektor biner yang bernilai integer, 0 dan 1, dimana semua elemen akan bernilai 0 kecuali satu elemen yang bernilai 1, yaitu elemen yang memiliki nilai kategori tersebut. Contohnya pada kolom geography, terdapat 3 jenis negara, yaitu: France, Germany, dan Spain. Setelah dilakukan encoding, maka nilai *France* menjadi (1,0,0), *Germany* menjadi (0,1,0), dan *Spain* menjadi (0,0,1).

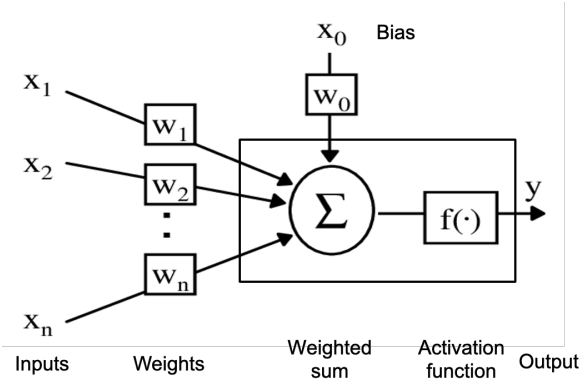
3. *Standardization*

   Standardization digunakan untuk mengubah fitur ke bentuk baru sehingga memiliki *mean* 0 dan standar deviasi 1. Misal, *x* = [15,57,36,88,37] adalah fitur, lakukan *standardization* sehingga hasil standardization *x'* memiliki nilai mean 0 dan standard deviasi 1. Gunakan persamaan berikut untuk menjadikan nilai *x* menjadi *x'*.

$$x'_i = \frac{(x_i - \mu_x)}{\sigma_x}$$

4. *Artificial Neural Network*

   Artificial Neural Network (ANN) model pembelajaran mesin yang terinspirasi oleh sistem saraf biologis. Menggabungkan kompleksitas beberapa teknik statistik dan tujuan machine learning untuk meniru kecerdasan manusia. ANN memiliki kemampuan belajar mandiri dan membentuk model nonlinier, yang sangat sederhana dalam komputasi dan memiliki kemampuan untuk memecahkan masalah nyata yang kompleks. ANN telah diterapkan untuk memecahkan masalah regresi dan klasifikasi di beberapa bidang seperti prediksi kebangkrutan, pengenalan tulisan tangan, inspeksi produk, dan diagnosis medis. Model ANN dapat dilihat di Gambar berikut ini.



5. Evaluasi model klasifikasi

   Akurasi adalah rasio prediksi Benar (positif dan negatif) dengan keseluruhan data. Perhitungan akurasi dapat diliuhat pada persamaan berikut ini.

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

   Dengan arti TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative.

Langkah Kerja

1. *Import Library* yang diperlukan untuk klasifikasi

```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

2. Import Dataset histori pengajuan kredit dan tampilkan dataset. Link dataset dapat diambil dari https://drive.google.com/uc?id=1_FXPTX8ap2LfgKKVF2zjShieopWDqHwL

```
import pandas as pd
import numpy as np

dataset = pd.read_csv('https://drive.google.com/uc?id=1_FXPTX8ap2LfgKKVF2zjShieopWDqHwL')
dataset.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumO
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	
3	4	15701354	Boni	699	France	Female	39	1	0.00	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	

3. *Encoding* atribut *Gender* menggunakan *LabelEncoder* agar menjadi bentuk numerik. Sehingga nilai *Male* menjadi 1 dan *Female* menjadi 0. Kemudian tampilkan data setelah dilakukan *Encoding*.

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:,2] = le.fit_transform(X[:,2])
```

```
print(X)
```

```
[[619 'France' 0 ... 1 1 101348.88]
 [608 'Spain' 0 ... 0 1 112542.58]
 [502 'France' 0 ... 1 0 113931.57]
 ...
 [709 'France' 0 ... 0 1 42085.58]
 [772 'Germany' 1 ... 1 0 92888.52]
 [792 'France' 0 ... 1 0 38190.78]]
```

4. *Encoding* atribut *Geography* menggunakan *LabelEncoder*. Setelah dilakukan *encoding*, maka atribut *Geography* menjadi 3 kolom dan terletak di kolom 0 hingga 2. Nilai *France* menjadi (1,0,0), *Germany* menjadi (0,1,0), dan *Spain* menjadi (0,0,1).

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1])],
                       remainder='passthrough')
X = ct.fit_transform(X)
```

```
print(X)
```

```
[[1.0 0.0 0.0 ... 1 1 101348.88]
 [0.0 0.0 1.0 ... 0 1 112542.58]
 [1.0 0.0 0.0 ... 1 0 113931.57]
 ...
 [1.0 0.0 0.0 ... 0 1 42085.58]
 [0.0 1.0 0.0 ... 1 0 92888.52]
 [1.0 0.0 0.0 ... 1 0 38190.78]]
```

5. Split dataset menjadi *training set* dan *testing set*

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

6. *Scaling* atribut menggunakan *Standard Scaler*. Setelah itu tampilkan data setelah dilakukan *scaling*.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
print(X_train)
```

```
[[-1.01460667 -0.5698444   1.74309049 ...  0.64259497 -1.03227043
   1.10643166]
 [-1.01460667  1.75486502 -0.57369368 ...  0.64259497  0.9687384
  -0.74866447]
 [ 0.98560362 -0.5698444  -0.57369368 ...  0.64259497 -1.03227043
   1.48533467]
 ...
 [ 0.98560362 -0.5698444  -0.57369368 ...  0.64259497 -1.03227043
   1.41231994]
 [-1.01460667 -0.5698444   1.74309049 ...  0.64259497  0.9687384
   0.84432121]
 [-1.01460667  1.75486502 -0.57369368 ...  0.64259497 -1.03227043
   0.32472465]]
```

```
print(X_test)
```

```
[[-1.01460667  1.75486502 -0.57369368 ...  0.64259497  0.9687384
   1.61085707]
 [ 0.98560362 -0.5698444  -0.57369368 ...  0.64259497 -1.03227043
   0.49587037]
 [-1.01460667 -0.5698444   1.74309049 ...  0.64259497  0.9687384
  -0.42478674]
 ...
 [-1.01460667 -0.5698444   1.74309049 ...  0.64259497 -1.03227043
   0.71888467]
 [-1.01460667  1.75486502 -0.57369368 ...  0.64259497  0.9687384
  -1.54507805]
 [-1.01460667  1.75486502 -0.57369368 ...  0.64259497 -1.03227043
   1.61255917]]
```

7. Proses Training dan Evaluasi menggunakan perhitungan *accuracy*.

```
for nh in range (2,11):
  ann=MLPClassifier(hidden_layer_sizes=(nh,),max_iter=10000,random_state=1)
  ann.fit(X_train,y_train)
  acc=100*ann.score(X_test,y_test)
  print('Neuron in hidden layer: %g, accuracy %.2f %%'%(nh,acc))
```

```
Neuron in hidden layer: 2, accuracy 83.75 %
Neuron in hidden layer: 3, accuracy 85.15 %
Neuron in hidden layer: 4, accuracy 84.20 %
Neuron in hidden layer: 5, accuracy 86.35 %
Neuron in hidden layer: 6, accuracy 86.15 %
Neuron in hidden layer: 7, accuracy 86.20 %
Neuron in hidden layer: 8, accuracy 85.70 %
Neuron in hidden layer: 9, accuracy 86.25 %
Neuron in hidden layer: 10, accuracy 86.15 %
```

8. Gunakan model ANN dengan jumlah neuron dalam hidden layer berjumlah 3 dikarenakan memiliki akurasi yang terbaik untuk menentukan apakah nasabah dengan informasi berikut ini disetujui atau tidak pengajuan kreditnya.

Geography: France

Credit Score: 600

Gender: Male

Age: 40 years old

Tenure: 3 years

Balance: $ 60000

Number of Products: 2

Does this customer have a credit card ? Yes

Is this customer an Active Member: Yes

Estimated Salary: $ 50000

```
C = np.array([[600,'France','Male', 40, 3, 60000, 2, 1, 1, 50000]])

#transform the gender column
C[:,2] = le.transform(C[:,2])

#transform geographic column
C = ct.transform(C)

#scale value
C = sc.transform(C)

pred = ann.predict(C)

if(pred == 0):
  print('Tolak')
else:
  print('Terima')
```

```
Tolak
```

MODUL 3: Klastering data penduduk menggunakan k-Means

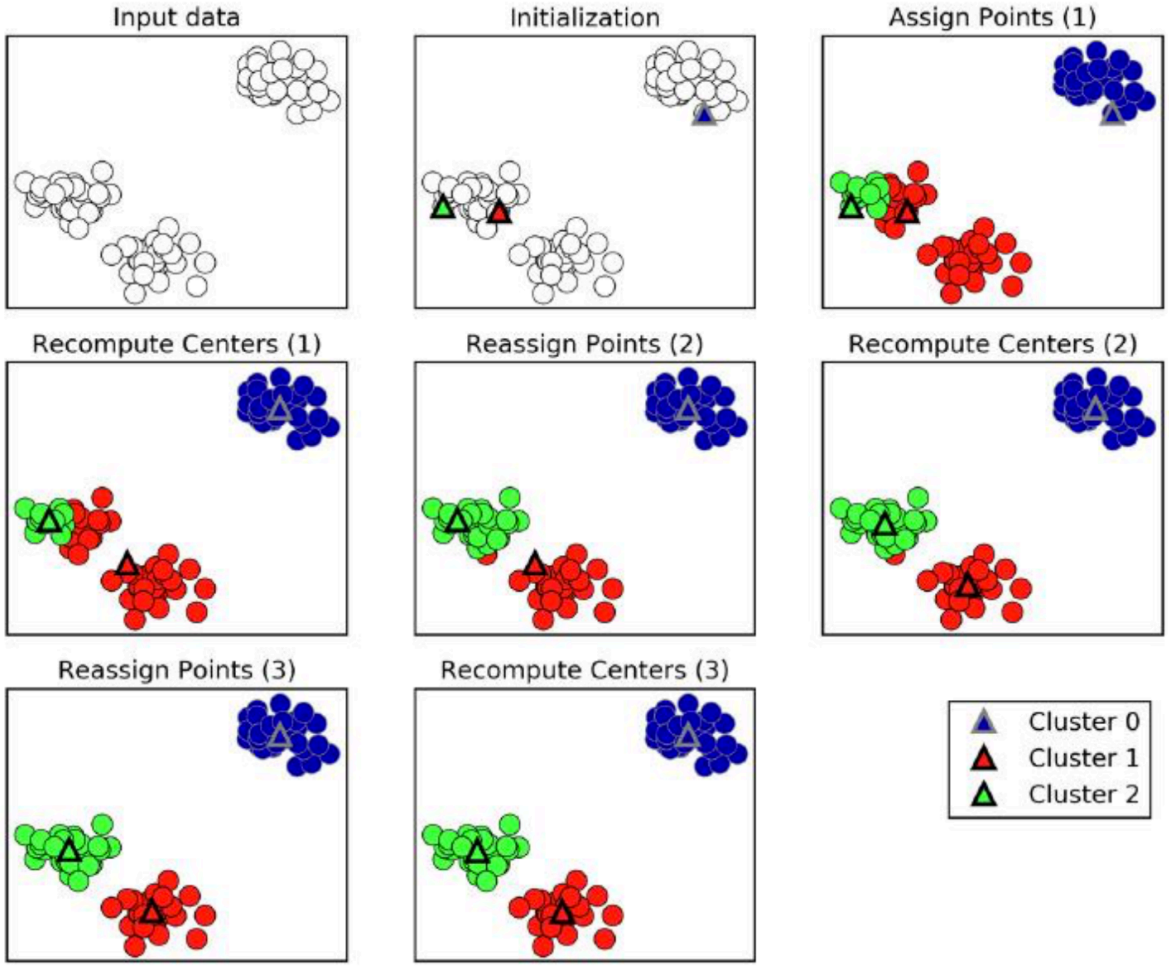
Tujuan Praktikum:
1. Mahasiswa dapat mengklasterkan data penduduk

Alat dan Bahan:
1. Komputer Lab
2. Editor: Sublime, Visual Code
3. Python

Teori dan Definisi
Klastering adalah suatu metode untuk mengelompokkan data menjadi beberapa kelompok yang disebut dengan nama klaster. Data yang ada akan terbagi menjadi beberapa kelompok di mana data yang ada dalam suatu klaster mempunyai kemiripan dan data yang ada di klaster berbeda mempunyai perbedaan.

Klastering menggunakan k-Means merupakan salah satu metode yang paling sederhana dan paling banyak dipakai untuk mempelajari algoritma klastering. Metode k-means akan mencari pusat-pusat klaster yang dapat mewakili suatu kelompok data.

Algoritma k-Means secara garis besar mempunyai dua tahapan, yaitu menentukan data mana saja yang dekat dengan suatu pusat klaster, dan memastikan tiap pusat klaster adalah rata-rata dari data dalam klaster tersebut. Algoritma ini akan berhenti ketika penentuan data ke suatu klaster tertentu tidak mengalami perubahan. Ilustrasi dari algoritma k-Means dapat dilihat pada Gambar 5.1 di mana terjadi tiga kali iterasi sampai dengan algoritma ini berhenti. Pusat data digambarkan dalam simbol segitiga dan data digambarkan dalam bentuk lingkaran. Tiga warna yang berbeda dipakai untuk menunjukkan kelompok data tersebut.



Gambar 5.1. Cara kerja algoritma k-means

Langkah Kerja:

Eksperimen 1

1. Buka editor untuk menulis bahasa pemrograman Python.
2. Pastikan scikit-learn sudah ter-install.
3. Tulis kode berikut untuk mencoba k-means, di mana k=3

```
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans

# generate synthetic two-dimensional data
X, y = make_blobs(random_state=1)

# build the clustering model
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
```

4. Tampilkan data yang ada masuk ke dalam klaster yang mana dengan menggunakan kode berikut:

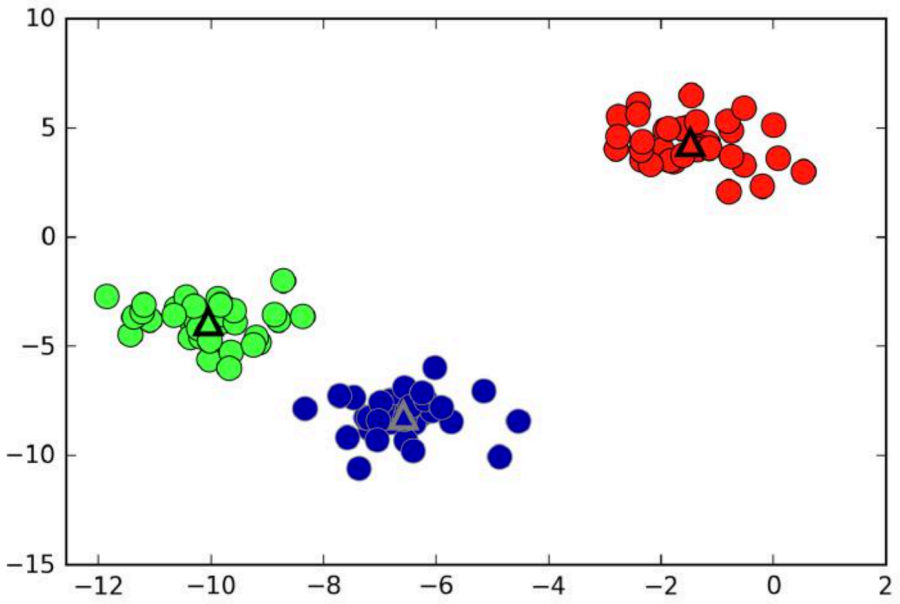
```
print("Cluster memberships:\n{}".format(kmeans.labels_))
```

```
Cluster memberships:
[1 2 2 2 0 0 0 2 1 1 2 2 0 1 0 0 0 1 2 2 0 2 0 1 2 0 0 1 1 0 1 1 0 1 2 0 2
 2 2 0 0 2 1 2 2 0 1 1 1 1 2 0 0 0 1 0 2 2 1 1 2 0 0 2 2 0 1 0 1 2 2 2 0 1
 1 2 0 0 1 2 1 2 2 0 1 1 1 1 2 1 0 1 1 2 2 0 0 1 0 1]
```

5. Tampilkan visualisasi hasil klastering menggunakan kode berikut

```
mglearn.discrete_scatter(X[:, 0], X[:, 1], kmeans.labels_, markers='o')
mglearn.discrete_scatter(
    kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], [0, 1, 2],
    markers='^', markeredgewidth=2)
```

:



6. Klasterkan data tersebut menjadi 2 klaster (k=2) menggunakan kode berikut:

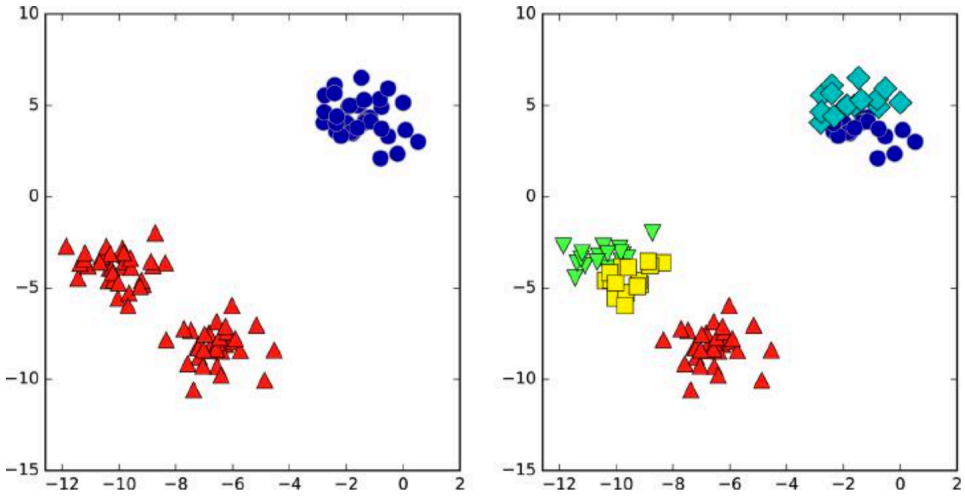
```
fig, axes = plt.subplots(1, 2, figsize=(10, 5))

# using two cluster centers:
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)
assignments = kmeans.labels_

mglearn.discrete_scatter(X[:, 0], X[:, 1], assignments, ax=axes[0])

# using five cluster centers:
kmeans = KMeans(n_clusters=5)
kmeans.fit(X)
assignments = kmeans.labels_

mglearn.discrete_scatter(X[:, 0], X[:, 1], assignments, ax=axes[1])
```



Eksperimen 2

1. Unduh data untuk mengklasterkan data penduduk di tautan berikut:

   California housing (https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html).

2. Baca deskripsi isi data yang diunduh tersebut sebelum melakukan klastering.
3. Klasterkan data tersebut dengan beberapa nilai k. Mulai dengan nilai k=6.

MODUL 4: Klastering data pengunjung mall menggunakan DBSCAN

Tujuan Praktikum:

1. Mahasiswa dapat mengklasterkan data pengunjung mall mengguakan DBSCAN

Alat dan Bahan:
1. Komputer Lab
2. Editor: Sublime, Visual Code
3. Python

Teori dan Definisi:

DBSCAN (Density Based Spatial Clustering of Applications with Noise) merupakan salah satu metode klastering di mana metode ini tidak membutuhkan informasi berapa banyak jumlah klaster di awal. DBSCAN bisa melakukan klastering untuk pola-pola yang berbentuk kompleks. Metode ini juga bisa mengetahui data yang bukan bagian dari klaster.

Cara kerja DBSCAN berdasar pada kepadatan data, di mana banyak data berdekatan satu dengan yang lain. Area data yang berdekatan ini disebut dengan area padat (dense). Konsep dasar dari DBSCAN adalah mengklasterkan data di area yang pada dan memisahkan dengan area yang relatif kosong.

Data yang ada dalam area yang padat disebut dengan data core. Data core ini didefinisikan berdasar dari 2 parameter, yaitu jumlah data dan eps. Suatu data dikatakan sebagai data core jika terpenuhi data sejumlah minimum jumlah data yang ada dalam jarak eps. Data core mempunyai jarak yang dekat dalam jangkauan jarak eps merupakan data yang berada dalam klaster yang sama. Ada tiga jenis data dalam DBSCAN, yaitu core, boundary, dan noise. Data boundary adalah data yang ada dalam jangkauan eps dari data core. Selain itu data merupakan noise.

Langkah Kerja:

Eksperimen 1

1. Buka editor untuk menulis bahasa pemrograman Python.
2. Pastikan scikit-learn sudah ter-install.
3. Tulis kode berikut untuk mengklasterkan menggunakan DBSCAN.

```
from sklearn.cluster import DBSCAN
X, y = make_blobs(random_state=0, n_samples=12)

dbscan = DBSCAN()
clusters = dbscan.fit_predict(X)
print("Cluster memberships:\n{}".format(clusters))
```

   Hasil dari kode tersebut adalah semua data merupakan noise.

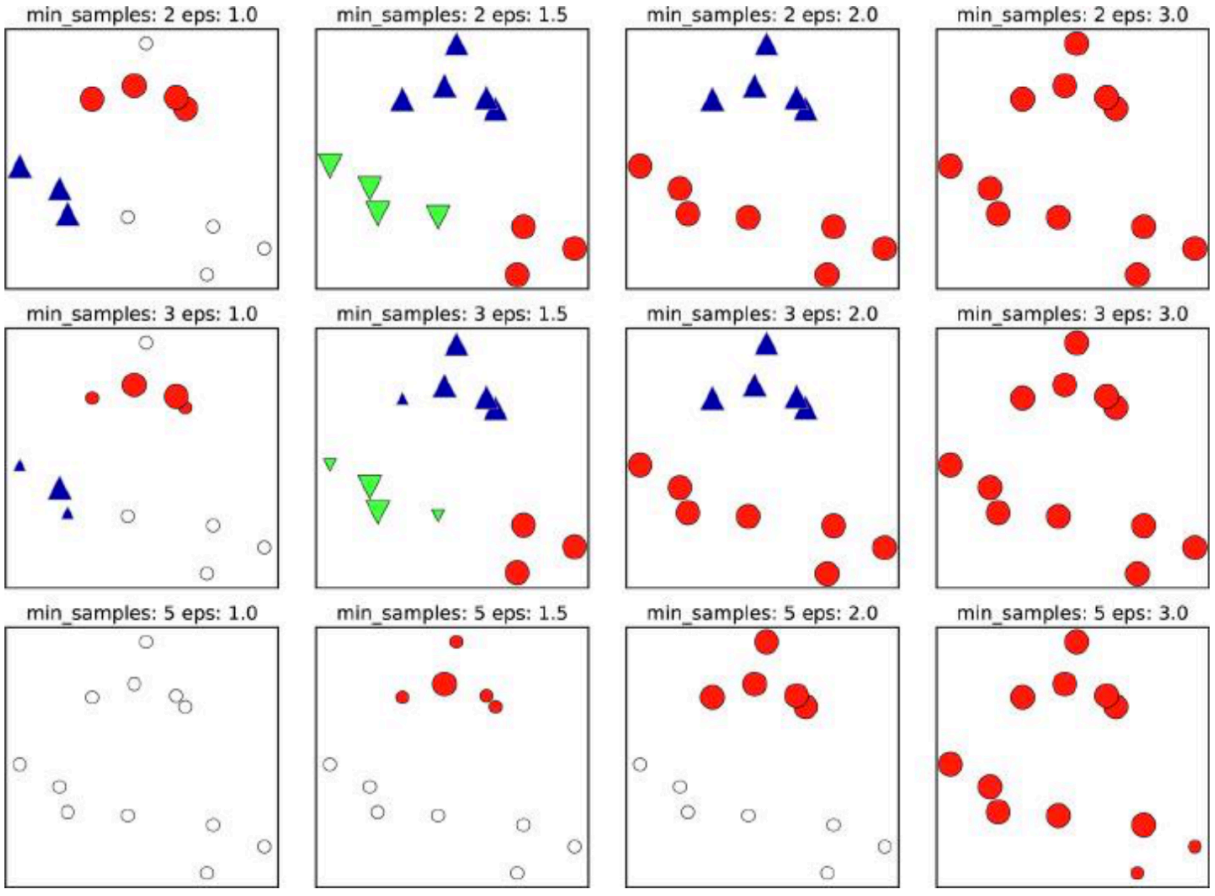
```
Cluster memberships:
[-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
```

4. Coba gunakan nilai n_samples dan eps yang berbeda-beda:

```
mglearn.plots.plot_dbscan()
```

5. Hasil dari kode tersebut adalah:

```
min_samples: 2 eps: 1.000000  cluster: [-1  0  0 -1  0 -1  1  1  0  1 -1 -1]
min_samples: 2 eps: 1.500000  cluster: [0 1 1 1 1 0 2 2 1 2 2 0]
min_samples: 2 eps: 2.000000  cluster: [0 1 1 1 1 0 0 0 1 0 0 0]
min_samples: 2 eps: 3.000000  cluster: [0 0 0 0 0 0 0 0 0 0 0 0]
min_samples: 3 eps: 1.000000  cluster: [-1  0  0 -1  0 -1  1  1  0  1 -1 -1]
min_samples: 3 eps: 1.500000  cluster: [0 1 1 1 1 0 2 2 1 2 2 0]
min_samples: 3 eps: 2.000000  cluster: [0 1 1 1 1 0 0 0 1 0 0 0]
min_samples: 3 eps: 3.000000  cluster: [0 0 0 0 0 0 0 0 0 0 0 0]
min_samples: 5 eps: 1.000000  cluster: [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
min_samples: 5 eps: 1.500000  cluster: [-1  0  0  0  0 -1 -1 -1  0 -1 -1 -1]
min_samples: 5 eps: 2.000000  cluster: [-1  0  0  0  0 -1 -1 -1  0 -1 -1 -1]
min_samples: 5 eps: 3.000000  cluster: [0 0 0 0 0 0 0 0 0 0 0 0]
```



Eksperimen 2

1. Unduh data untuk mengklasterkan data pengunjung mall di tautan berikut:

https://github.com/SteffiPeTaffy/machineLearningAZ/blob/master/Machine%20Learning%20A-Z%20Template%20Folder/Part%204%20-%20Clustering/Section%2025%20-%20Hierarchical%20Clustering/Mall_Customers.csv

2. Baca deskripsi isi data yang diunduh tersebut sebelum melakukan klastering.
3. Klasterkan data tersebut dengan beberapa nilai eps=3 dan n_samples=4.