Praktikum AI 2022

Pertemuan Ke-8

01

Machine Learning

Mesin belajar?

02

Boston House Price Forecast

Mempelajari machine learning menggunakan Boston house price data set

03

Linear Regression

Mempelajari linear regression dengan bantuan matplotlib dan numpy

Machine Learning

01

Definisi

Merupakan cabang dari Artificial Intelligence (AI) yang berfokus pada penggunaan data dan algoritma untuk meniru kemampuan berpikir manusia, memungkinkan sistem untuk belajar dan meningkatkan kemampuannya dari apa yang ia pelajari tanpa diprogram secara eksplisit.

Cara Kerja

UC Berkeley memecah sistem pembelajaran algoritma pembelajaran mesin menjadi tiga bagian utama :

1. Decision Process : Berdasarkan beberapa data input, yang dapat diberi label atau tidak, algoritma akan menghasilkan perkiraan tentang suatu pola dalam data.
2. An Error Function: Error function berfungsi untuk mengevaluasi prediksi model. Jika ada contoh yang diketahui, fungsi kesalahan dapat membuat perbandingan untuk menilai keakuratan model.
3. Model Optimization Process : Jika model dapat lebih cocok dengan titik data dalam set pelatihan, maka bobot disesuaikan untuk mengurangi perbedaan antara contoh yang diketahui dan estimasi model sampai ambang batas akurasi terpenuhi.

Implementasi Machine Learning

- Speech recognition
- Customer Service
- Computer Vision
- Recommendation Engines
- Automated Stock Trading
- dsb...

Boston House Price Forecast

02

Merupakan salah satu contoh dataset dari scikit untuk mempelajari cara pengimplementasian machine learning untuk memprediksi data.

Detail Dataset

Kasus ini didasarkan pada kumpulan data Boston, yang berisi 13 fitur dan 506 record. Setiap record berisi informasi sebagai berikut :

- CRIM: urban per capita crime rate
- ZN: proportion of residential land exceeds 25,000 square feet
- INDUS: proportion of non-retail commercial land in a town
- CHAS: Charles river empty variable (1 indicates that the boundary is a river; otherwise, the value is 0)
- NOX: Nitric oxide concentration
- RM: average number of rooms in a house
- AGE: proportion of private houses completed before 1940
- DIS: weighted distance to the five central regions of Boston
- RAD: proximity index of a radial highway
- TAX: full value property tax rate of $10,000
- PTRATIO: proportion of teachers and students in urban areas
- target: average price of private houses, unit: $1,000

MARI KITA NGODING!!!

Part 1 : Import Dependencies

```
#Prevent unnecessary warnings.
import warnings
warnings.filterwarnings("ignore")

#Introduce the basic package of data science.
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import pandas as pd
import scipy.stats as st
import seaborn as sns

#Introduce machine learning, preprocessing, model selection, and evaluation indicators.
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import r2_score

#Import the Boston dataset used this time.
from sklearn.datasets import load_boston

#Introduce algorithms.
from sklearn.linear_model import RidgeCV, LassoCV, LinearRegression, ElasticNet

#Compared with SVC, it is the regression form of SVM.
from sklearn.svm import SVR

#Integrate algorithms.
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from xgboost import XGBRegressor
```

Part 2 : Memasukkan dataset dan melihat atribut data

```
#Load the Boston house price data set.
boston = load_boston()

#x features, and y labels.
x = boston.data
y = boston.target

#Display related attributes.
print('Feature column name')
print(boston.feature_names)
print("Sample data volume: %d, number of features: %d"% x.shape)
print("Target sample data volume: %d"% y.shape[0])
```

```
Feature column name
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
Sample data volume: 506, number of features: 13
Target sample data volume: 506
```

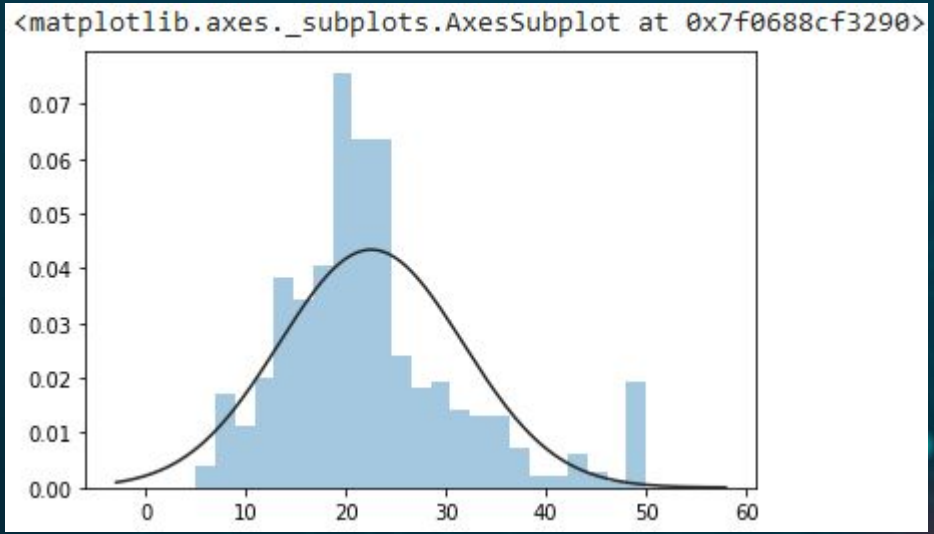
Part 3 : Konversi dataset ke format DataFrame

```
x = pd.DataFrame(boston.data, columns=boston.feature_names)
x.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
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
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
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
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
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

Part 4 : Visualisasi Distribusi Label

```
sns.distplot(tuple(y), kde=False, fit=st.norm)
```



Part 5 : Pisah dan pre-process dataset

```
#Segment the data.
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=28)

#Standardize the data set.
ss = StandardScaler()
x_train = ss.fit_transform(x_train)
x_test = ss.transform(x_test)
x_train[0:100]
```

```
array([[-0.35703125, -0.49503678, -0.15692398, ..., -0.01188637,
         0.42050162, -0.29153411],
       [-0.39135992, -0.49503678, -0.02431196, ...,  0.35398749,
         0.37314392, -0.97290358],
       [ 0.5001037 , -0.49503678,  1.03804143, ...,  0.81132983,
         0.4391143 ,  1.18523567],
       ...,
       [-0.34697089, -0.49503678, -0.15692398, ..., -0.01188637,
         0.4391143 , -1.11086682],
       [-0.39762221,  2.80452783, -0.87827504, ...,  0.35398749,
         0.4391143 , -1.28120919],
       [-0.38331362,  0.41234349, -0.74566303, ...,  0.30825326,
         0.19472652, -0.40978832]])
```

Part 6 : Modelkan data set

```python
# Set the model name.
names = ['LinerRegression',
'Ridge',
'Lasso',
'Random Forrest',
'GBDT',
'Support Vector Regression',
'ElasticNet',
'XgBoost']

# Define the model.
# cv is the cross-validation idea here.
models = [LinearRegression(),
RidgeCV(alphas=(0.001,0.1,1),cv=3),
LassoCV(alphas=(0.001,0.1,1),cv=5),
RandomForestRegressor(n_estimators=10),
GradientBoostingRegressor(n_estimators=30),
SVR(),
ElasticNet(alpha=0.001,max_iter=10000),
XGBRegressor()]

# Output the R2 scores of all regression models.
# Define the R2 scoring function.
def R2(model,x_train, x_test, y_train, y_test):
  model_fitted = model.fit(x_train,y_train)
  y_pred = model_fitted.predict(x_test)
  score = r2_score(y_test, y_pred)
  return score

#Traverse all models to score.
for name,model in zip(names,models):
  score = R2(model,x_train, x_test, y_train, y_test)
  print("{}: {:.6f}, {:.4f}".format(name,score.mean(),score.std()))
```

```
LinerRegression: 0.564115, 0.0000
Ridge: 0.563673, 0.0000
Lasso: 0.564049, 0.0000
Random Forrest: 0.691201, 0.0000
GBDT: 0.727440, 0.0000
Support Vector Regression: 0.517260, 0.0000
ElasticNet: 0.563992, 0.0000
[15:52:29] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
XgBoost: 0.761123, 0.0000
```

Part 7 : Buat model machine learning

```
'''
'kernel': kernel function
'C': SVR regularization factor
'gamma': 'rbf', 'poly' and 'sigmoid' kernel function coefficient, which affects the model performance
'''

parameters = {
'kernel': ['linear', 'rbf'],
'C': [0.1, 0.5,0.9,1,5],
'gamma': [0.001,0.01,0.1,1]
}

#Use grid search and perform cross validation.
model = GridSearchCV(SVR(), param_grid=parameters, cv=3)
model.fit(x_train, y_train)
```

```
GridSearchCV(cv=3, estimator=SVR(),
             param_grid={'C': [0.1, 0.5, 0.9, 1, 5],
                         'gamma': [0.001, 0.01, 0.1, 1],
                         'kernel': ['linear', 'rbf']})
```

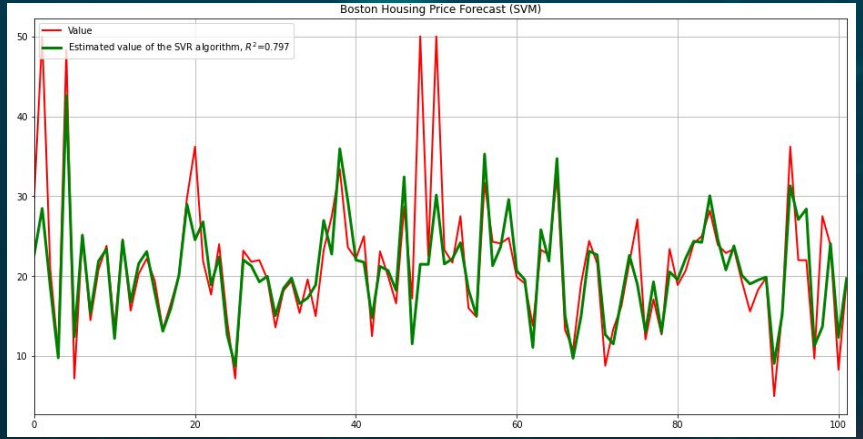
Part 8 : Ambil parameter optimal

```
print("Optimal parameter list:", model.best_params_)
print("Optimal model:", model.best_estimator_)
print("Optimal R2 value:", model.best_score_)
```

```
Optimal parameter list: {'C': 5, 'gamma': 0.1, 'kernel': 'rbf'}
Optimal model: SVR(C=5, gamma=0.1)
Optimal R2 value: 0.7965173649188232
```

Part 9 : Buat Visualisasi

```
##Perform visualization.
ln_x_test = range(len(x_test))
y_predict = model.predict(x_test)

#Set the canvas.
plt.figure(figsize=(16,8), facecolor='w')

#Draw with a red solid line.
plt.plot (ln_x_test, y_test, 'r-', lw=2, label=u'Value')

#Draw with a green solid line.
plt.plot (ln_x_test, y_predict, 'g-', lw = 3, label=u'Estimated value of the SVR algorithm, $R^2$=%.3f' %
(model.best_score_))

#Display in a diagram.
plt.legend(loc ='upper left')
plt.grid(True)
plt.title(u"Boston Housing Price Forecast (SVM)")
plt.xlim(0, 101)
plt.show()
```



Source Code

Untuk lebih lanjut, notebook untuk kode ini ada di link berikut :

https://colab.research.google.com/drive/1tORyTMowJqMyX2_qeSxERcIckOiROV_1?usp=sharing

Detail of Linear Regression

03

About This Experiment

Eksperimen ini bertujuan untuk membuat dan melihat bagaimana iterasi yang terjadi di dalam algoritma linear regression yang sesuai dengan distribusi data yang ada

Tujuan:

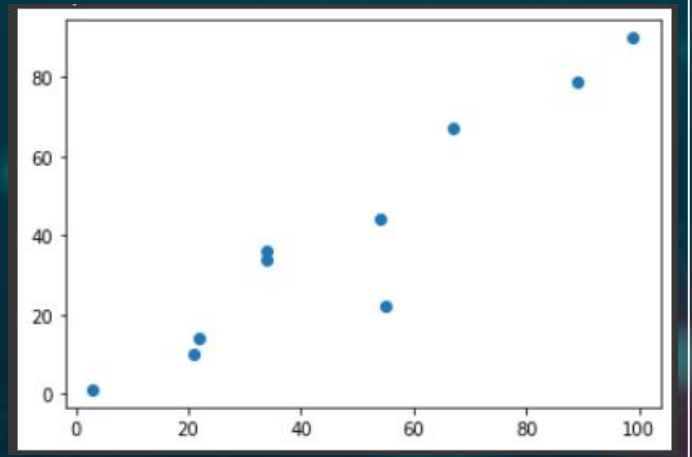
- Familiar dengan sintaks *python basic*
- Menguasai langkah-langkah ketika mengimplementasi linear regression

MARI KITA NGODING!!!

Persiapan Data

Terdapat 10 buah data random yang memiliki linear relationship.

```
#Import the required modules, numpy for calculation, and Matplotlib for
drawing
import numpy as np
import matplotlib.pyplot as plt

#define data, and change list to array
x = [3,21,22,34,54,34,55,67,89,99]
x = np.array(x)
y = [1,10,14,34,44,36,22,67,79,90]
y = np.array(y)

#Show the effect of a scatter plot
plt.scatter(x,y)
```



Mendefinisikan Fungsi Terkait

- Model function: mendefinisikan linear regression model **wx+b**
- Loss function: merupakan fungsi untuk mengukur error atau MSE (mean square error)
- Optimization funct. : metode penurunan gradien untuk mencari turunan parsial dari w dan b

```
#The basic linear regression model is wx+b

def model(a, b, x):
    return a*x + b

#loss function

def loss_function(a, b, x, y):
    num = len(x)
    prediction=model(a,b,x)
    return (0.5/num) * (np.square(prediction-y)).sum()
```

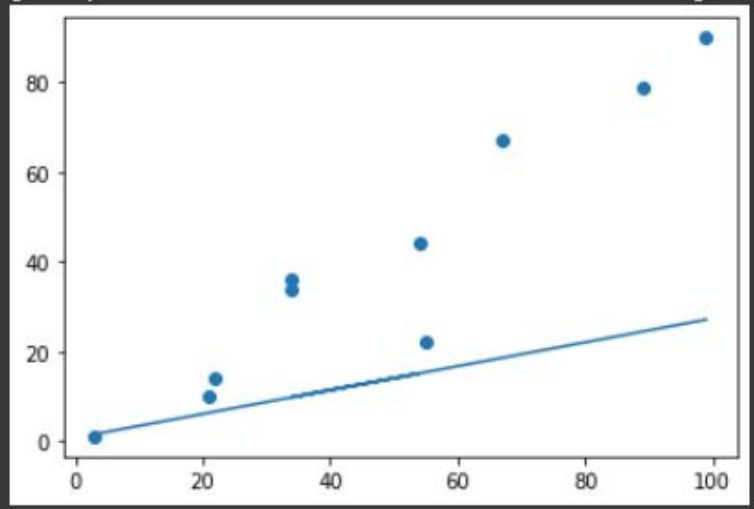
Mendefinisikan Fungsi Terkait (lanjutan)

```
#The optimization function
def optimize(a,b,x,y):
     num = len(x)
     prediction = model(a,b,x)
     #Update the values of A and B by finding the partial derivatives of the loss
     function on a and b
     da = (1.0/num) * ((prediction - y)*x).sum()
     db = (1.0/num) * ((prediction - y).sum())
     a = a - Lr * da
     b = b - Lr * db
     return a, b

#iterated function, return a and b
def iterate(a,b,x,y,times):
     for i in range(times):
          a,b = optimize(a,b,x,y)
     return a,b
```

Memulai iterasi

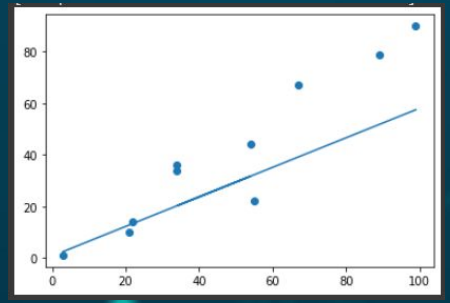
```
#Initialize parameters and display
a = np.random.rand(1)
print(a)
b = np.random.rand(1)
print(b)
Lr = 1e-4

#For the first iteration, the parameter
values, losses, and visualization after the
iteration are displayed
a,b = iterate(a,b,x,y,1)
prediction=model(a,b,x)
loss = loss_function(a, b, x, y)
print(a,b,loss)
plt.scatter(x,y)
plt.plot(x,prediction)
```

```
[0.00081188]
[0.67691892]
[0.26649129] [0.68081735] 576.0016842047816
[<matplotlib.lines.Line2D at 0x7ff5b664c650>]
```

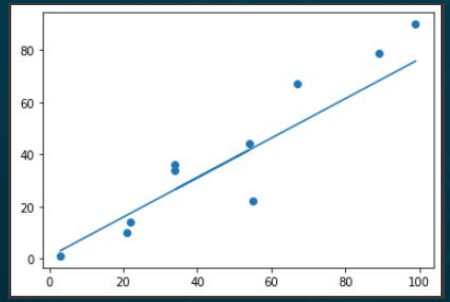


Jalankan iterasi selanjutnya

```
a,b = iterate(a,b,x,y,2)
prediction=model(a,b,x)
loss = loss_function(a, b, x, y)
print(a,b,loss)
plt.scatter(x,y)
plt.plot(x,prediction)
```

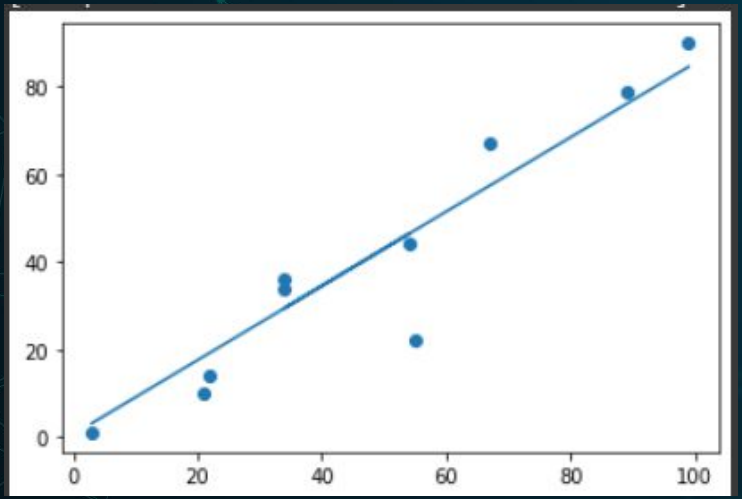



```
a,b = iterate(a,b,x,y,3)
prediction=model(a,b,x)
loss = loss_function(a, b, x, y)
print(a,b,loss)
plt.scatter(x,y)
plt.plot(x,prediction)
```



Jalankan iterasi selanjutnya

```
a,b = iterate(a,b,x,y,1000)
prediction=model(a,b,x)
loss = loss_function(a, b, x, y)
print(a,b,loss)
plt.scatter(x,y)
plt.plot(x,prediction)
```



Source Code

Untuk lebih lanjut, notebook untuk kode ini ada di link berikut :

https://colab.research.google.com/drive/1JT3F9bdXb85RK0Z5NnCs1KTqCK8-1Nq0?usp=sharing

Tugas

Tugas No 1

Buatlah implementasi decision tree algorithm menggunakan package pandas, math, dan numpy dari dataset pada link berikut :

https://data-certification.obs.cn-east-2.myhuaweicloud.com/ENG/HCIA-AI/V3.0/ML-Dataset.rar

buat dalam satu file bernama **NPM_decisionTree.py**

THANKS!

Do you have any questions?