TUGAS LKP 7 PRAKTIKUM PENGENALAN POLA
Dosen Pengampu : Dr. Toto Hardiyanto, S.Kom, M.Si

MACHINE LEARNING BERBASIS ALGORITMA TREE

Di susun oleh :
SUFIATUL MARYANA
G6601211012

PROGRAM PASCASARJANA
PROGRAM STUDI ILMU KOMPUTER
DEPARTEMEN ILMU KOMPUTER
INSTITUT PERTANIAN BOGOR
2021

PERTEMUAN 7

Machine Learning berbasis Algoritma Tree

Link github : https://github.com/sufiatulmaryana/RKP7Pola1

TUJUAN PRAKTIKUM

Pada praktikum Pereteman 8 akan menerapkan beberapa konsep antara lain :

1. Decision Tree
2. CART
3. Bagging dan Random Forest

Keempat konsep tersebut diterapkan pada sebuah permasalahan dengan menggunakan tools / bahasa pemrograman Python.

1. Decision Tree

```
#Import Library
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics


#Melakukan pembacaaan dataset

col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 
'age', 'label']

# load dataset

pima = pd.read_csv("pima-indians-diabetes.csv", header=None, names=col_names)
#print(pima)


#split dataset in features and target variable

feature_cols = ['pregnant', 'insulin', 'bmi','age','glucose','bp','pedigree']

x = pima[feature_cols] # Features
y = pima.label # Target variable


# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.25, 
random_state=3)


# Membuat objek DT
 # Dapat dioptimalkan dengan menghitung Entropy
   clf = DecisionTreeClassifier()
   clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)
```

```
# Melakukan Pelatihan DT
  clf = clf.fit(X_train,y_train)


# Memprediksi

y_pred = clf.predict(X_test)


# Menghitung akurasi model
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

2. CART (Classification And Regression Tree)

\# Load Dataset

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from sklearn.datasets import load_boston
boston_dataset = load_boston()

boston = pd.DataFrame(boston_dataset.data, columns=boston_dataset.feature_names)

boston['MEDV'] = boston_dataset.target
names = boston_dataset.feature_names
```

#Library CART pada python
from sklearn.tree import DecisionTreeRegressor

```
array = boston.values

X = array[:,0:13]
Y = array[:,13]
#print(X)
#print(Y)

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3,
random_state=1234)

#model = DecisionTreeRegressor(max_leaf_nodes = 20)

model = DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           max_leaf_nodes=50,  min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           random_state=None, splitter='best')
```

#Evaluasi

```
rt = model.fit(X_train, Y_train)
rt

import random as rnd

rnd.seed(123458)
X_new = X[rnd.randrange(X.shape[0])]
X_new = X_new.reshape(1,13)

#Prediksi Model
YHat = model.predict(X_new)


df = pd.DataFrame(X_new, columns = names)
df["Predicted Price"] = YHat
df.head(1)

from sklearn.metrics import r2_score
YHat = model.predict(X_test)
print(YHat)
```

#Menghitung Rata-rata Kuadrat

```
r2 = r2_score(Y_test, YHat)
print("R-Squared = ", r2)
```

3. Bagging

#Impor Library

```
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
from sklearn.ensemble import BaggingClassifier
from sklearn.model_selection import GridSearchCV
```

#Load cancer dataset

```
bc = datasets.load_breast_cancer()
X = bc.data
y = bc.target


#membagi dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=1, stratify=y)
```

#Melakukan pipelining

```
pipeline = make_pipeline(StandardScaler(),
                         LogisticRegression(random_state=1))
```

#Skema bagging

```
bgclassifier = BaggingClassifier(base_estimator=pipeline, n_estimators=100,
                                 max_features=8,
                                 max_samples=80,
                                 random_state=1, n_jobs=5)


bgclassifier.fit(X_train, y_train)


print('Model test Score: %.3f, ' %bgclassifier.score(X_test, y_test),
      'Model training Score: %.3f' %bgclassifier.score(X_train, y_train))
```

TUGAS PRAKTIKUM

1. Melakukan visualisasi Tree
   a. Decision Tree

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

```
[Running] python -u "d:\ANNA\S3\Tugas7\TugasPrak7.py"
     pregnant  glucose  bp  skin  insulin   bmi  pedigree  age  label
0           6      148  72    35        0  33.6     0.627   50      1
1           1       85  66    29        0  26.6     0.351   31      0
2           8      183  64     0        0  23.3     0.672   32      1
3           1       89  66    23       94  28.1     0.167   21      0
4           0      137  40    35      168  43.1     2.288   33      1
..        ...      ...  ..   ...      ...   ...       ...  ...    ...
763        10      101  76    48      180  32.9     0.171   63      0
764         2      122  70    27        0  36.8     0.340   27      0
765         5      121  72    23      112  26.2     0.245   30      0
766         1      126  60     0        0  30.1     0.349   47      1
767         1       93  70    31        0  30.4     0.315   23      0

[768 rows x 9 columns]
Accuracy: 0.671875

[Done] exited with code=0 in 0.854 seconds
```

b. CART

```
[Running] python -u "d:\ANNA\S3\Tugas7\cart.py"
[[6.3200e-03 1.8000e+01 2.3100e+00 ... 1.5300e+01
  3.9690e+02 4.9800e+00]
 [2.7310e-02 0.0000e+00 7.0700e+00 ... 1.7800e+01
  3.9690e+02 9.1400e+00]
 [2.7290e-02 0.0000e+00 7.0700e+00 ... 1.7800e+01
  3.9283e+02 4.0300e+00]
 ...
 [6.0760e-02 0.0000e+00 1.1930e+01 ... 2.1000e+01
  3.9690e+02 5.6400e+00]
 [1.0959e-01 0.0000e+00 1.1930e+01 ... 2.1000e+01
  3.9345e+02 6.4800e+00]
 [4.7410e-02 0.0000e+00 1.1930e+01 ... 2.1000e+01
  3.9690e+02 7.8800e+00]]
[24.  21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 15.
  18.9 21.7 20.4
 18.2 19.9 23.1 17.5 20.2 18.2 13.6 19.6 15.2 14.5 15.6
  13.9 16.6 14.8
 18.4 21.  12.7 14.5 13.2 13.1 13.5 18.9 20.  21.  24.7
  30.8 34.9 26.6
 25.3 24.7 21.2 19.3 20.  16.6 14.4 19.4 19.7 20.5 25.
  23.4 18.9 35.4
 24.7 31.6 23.3 19.6 18.7 16.  22.2 25.  33.  23.5 19.4
  22.  17.4 20.9
 24.2 21.7 22.8 23.4 24.1 21.4 20.  20.8 21.2 20.3 28.
  23.9 24.8 22.9
 23.9 26.6 22.5 22.2 23.6 28.7 22.6 22.  22.9 25.  20.6
  28.4 21.4 38.7
 43.8 33.2 27.5 26.5 18.6 19.3 20.1 19.5 19.5 20.4 19.8
  19.4 21.7 22.8
 18.8 18.7 18.5 18.3 21.2 19.2 20.4 19.3 22.  20.3 20.5
  17.3 18.8 21.4
 15.7 16.2 18.  14.3 19.2 19.6 23.  18.4 15.6 18.1 17.4
```

```
  17.1 13.3 17.8
14.  14.4 13.4 15.6 11.8 13.8 15.6 14.6 17.8 15.4 21.5
  19.6 15.3 19.4
17.  15.6 13.1 41.3 24.3 23.3 27.  50.  50.  50.  22.7
  25.  50.  23.8
23.8 22.3 17.4 19.1 23.1 23.6 22.6 29.4 23.2 24.6 29.9
  37.2 39.8 36.2
37.9 32.5 26.4 29.6 50.  32.  29.8 34.9 37.  30.5 36.4
  31.1 29.1 50.
33.3 30.3 34.6 34.9 32.9 24.1 42.3 48.5 50.  22.6 24.4
  22.5 24.4 20.
21.7 19.3 22.4 28.1 23.7 25.  23.3 28.7 21.5 23.  26.7
  21.7 27.5 30.1
44.8 50.  37.6 31.6 46.7 31.5 24.3 31.7 41.7 48.3 29.
  24.  25.1 31.5
23.7 23.3 22.  20.1 22.2 23.7 17.6 18.5 24.3 20.5 24.5
  26.2 24.4 24.8
29.6 42.8 21.9 20.9 44.  50.  36.  30.1 33.8 43.1 48.8
  31.  36.5 22.8
30.7 50.  43.5 20.7 21.1 25.2 24.4 35.2 32.4 32.  33.2
  33.1 29.1 35.1
45.4 35.4 46.  50.  32.2 22.  20.1 23.2 22.3 24.8 28.5
  37.3 27.9 23.9
21.7 28.6 27.1 20.3 22.5 29.  24.8 22.  26.4 33.1 36.1
  28.4 33.4 28.2
22.8 20.3 16.1 22.1 19.4 21.6 23.8 16.2 17.8 19.8 23.1
  21.  23.8 23.1
20.4 18.5 25.  24.6 23.  22.2 19.3 22.6 19.8 17.1 19.4
  22.2 20.7 21.1
19.5 18.5 20.6 19.  18.7 32.7 16.5 23.9 31.2 17.5 17.2
  23.1 24.5 26.6
22.9 24.1 18.6 30.1 18.2 20.6 17.8 21.7 22.7 22.6 25.
  19.9 20.8 16.8
21.9 27.5 21.9 23.1 50.  50.  50.  50.  50.  13.8 13.8
  15.  13.9 13.3
13.1 10.2 10.4 10.9 11.3 12.3  8.8  7.2 10.5  7.4 10.2
  11.5 15.1 23.2
 9.7 13.8 12.7 13.1 12.5  8.5  5.   6.3  5.6  7.2 12.1
  8.3  8.5  5.
11.9 27.9 17.2 27.5 15.  17.2 17.9 16.3  7.   7.2  7.5
  10.4  8.8  8.4
16.7 14.2 20.8 13.4 11.7  8.3 10.2 10.9 11.   9.5 14.5
  14.1 16.1 14.3
11.7 13.4  9.6  8.7  8.4 12.8 10.5 17.1 18.4 15.4 10.8
  11.8 14.9 12.6
14.1 13.  13.4 15.2 16.1 17.8 14.9 14.1 12.7 13.5 14.9
  20.  16.4 17.7
19.5 20.2 21.4 19.9 19.  19.1 19.1 20.1 19.9 19.6 23.2
```

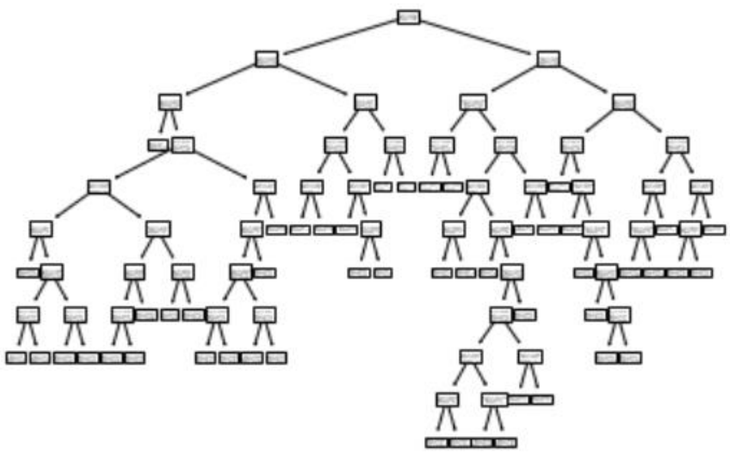
```
  29.8 13.8 13.3
 16.7 12.  14.6 21.4 23.  23.7 25.  21.8 20.6 21.2 19.1
  20.6 15.2  7.
  8.1 13.6 20.1 21.8 24.5 23.1 19.7 18.3 21.2 17.5 16.8
  22.4 20.6 23.9
   22.  11.9]
```

evaluasi

```
[35.51       24.26666667 10.38571429 22.35238095
14.02142857 23.71666667
 19.83421053 15.43571429 20.575      27.45
15.43571429 22.35238095
 22.35238095 18.31666667 17.71111111 23.71666667
8.49285714 16.81428571
 19.83421053 10.38571429 37.73333333 23.71666667
19.83421053 16.81428571
 27.45       19.77142857 19.83421053 22.46428571
31.52857143 22.83333333
 20.575      19.33333333 18.31666667 14.02142857
23.71666667 17.13333333
 20.50714286 22.35238095 10.38571429  8.49285714
8.49285714 18.31666667
 49.78571429 31.76       19.83421053 25.4375
25.4375     42.3
 14.02142857 25.4375     19.83421053 35.51
23.71666667 23.71666667
 50.         31.76       10.38571429 31.52857143
25.4375     22.35238095
 35.51       15.43571429 29.33333333 37.73333333
10.38571429 22.46428571
 35.51       20.50714286 49.78571429 49.15       10.95
23.71666667
 14.02142857 19.18333333 23.71666667 10.38571429 49.15
23.71666667
 15.08181818 15.08181818 42.75       49.78571429
23.71666667 15.08181818
 22.46428571 13.66666667 25.4375     31.52857143
19.77142857 22.46428571
 19.33333333 14.26666667 22.46428571 19.18333333
15.08181818 42.3
 15.43571429 24.26666667 25.4375     25.4375
19.83421053 19.83421053
 19.18333333 29.33333333 15.43571429 49.78571429
10.38571429 10.38571429
 31.06666667 19.83421053  8.49285714 25.4375
19.83421053 25.4375
 15.08181818 22.46428571 37.73333333 25.4375
```

```
15.08181818 49.15
 22.35238095 14.02142857 18.31666667 35.51
19.83421053 35.2
 23.71666667 22.46428571 19.83421053 22.35238095
22.83333333 19.83421053
 16.325      10.38571429 19.83421053 13.66666667
16.81428571 35.51
 17.13333333 29.33333333 19.83421053 10.38571429
24.26666667 19.83421053
 31.52857143 27.45       22.46428571 24.26666667
23.71666667  8.49285714
 19.83421053 19.77142857]

R-Squared =  0.8529712223699968
```



2. Melakukan tanpa skema bagging
   a. Pada Bagging

```
Model Linear test Score: 0.965,
Model Linear training Score: 0.991

Model Bagging test Score: 0.958,
Model Bagging training Score: 0.960
```

DAFTAR PUSTAKA

1. Richert W & Coelho LP. *Builidng Machine Learning System with Python*. 2013. Packt Publising. Birmingham, UK.