Tucil_13520101_13520119

February 15, 2023

1 Tugas Kecil 1 IF3270

1.1 Eksplorasi Library Algoritma Pembelajaran pada Jupyter Notebook

1.1.1 Anggota

1. 13520101 - Aira Thalca Avila Putra
2. 13520119 - Marchotridyo

2 Bagian 1: Mengenal dataset breast-cancer

[1]:
```
# Load data breast_cancer
from sklearn.datasets import load_breast_cancer
data = load_breast_cancer()
```

[2]:
```
# Mengenal isi dari data breast_cancer
import pandas as pd
import numpy as np

def sklearn_to_df(sklearn_dataset):
    df = pd.DataFrame(sklearn_dataset.data, columns=sklearn_dataset.
 ↪feature_names)
    df['target'] = pd.Series(sklearn_dataset.target)
    return df

dataset = sklearn_to_df(data)

dataset.head()
```

[2]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	\
0	17.99	10.38	122.80	1001.0	0.11840	
1	20.57	17.77	132.90	1326.0	0.08474	
2	19.69	21.25	130.00	1203.0	0.10960	
3	11.42	20.38	77.58	386.1	0.14250	
4	20.29	14.34	135.10	1297.0	0.10030	

	mean compactness	mean concavity	mean concave points	mean symmetry	\
0	0.27760	0.3001	0.14710	0.2419	

```
1           0.07864          0.0869            0.07017         0.1812
2           0.15990          0.1974            0.12790         0.2069
3           0.28390          0.2414            0.10520         0.2597
4           0.13280          0.1980            0.10430         0.1809

   mean fractal dimension  …  worst texture  worst perimeter  worst area  \
0                 0.07871  …          17.33           184.60      2019.0
1                 0.05667  …          23.41           158.80      1956.0
2                 0.05999  …          25.53           152.50      1709.0
3                 0.09744  …          26.50            98.87       567.7
4                 0.05883  …          16.67           152.20      1575.0

   worst smoothness  worst compactness  worst concavity  worst concave points  \
0            0.1622             0.6656           0.7119                0.2654
1            0.1238             0.1866           0.2416                0.1860
2            0.1444             0.4245           0.4504                0.2430
3            0.2098             0.8663           0.6869                0.2575
4            0.1374             0.2050           0.4000                0.1625

   worst symmetry  worst fractal dimension  target
0          0.4601                  0.11890       0
1          0.2750                  0.08902       0
2          0.3613                  0.08758       0
3          0.6638                  0.17300       0
4          0.2364                  0.07678       0

[5 rows x 31 columns]
```

[3]:
```
# Lihat data fitur apa saja yang dimiliki oleh dataset dan nama target variabelnya
print(data.feature_names)
print(data.target_names)
```

```
['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
['malignant' 'benign']
```

Jadi, apa yang ingin dilakukan setelah melakukan pembelajaran? Yang nantinya ingin dilakukan: Dari suatu data (yang berisikan nilai dari feature 'mean radius' s.d. 'worst fractal dimension') yang diberikan, kita ingin mengklasifikasikan data tersebut apakah bersifat 'malignant'

(0) atau 'benign' (1).

Sebelum dimulai learning, cek beberapa informasi tambahan dahulu...

[4]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  target                   569 non-null    int32
dtypes: float64(30), int32(1)
memory usage: 135.7 KB
```

Beberapa hal yang bisa diperhatikan dari pemanggilan df.info():

1. Tidak ada missing values, dan
2. Semua tipe datanya numerik

3 Bagian 2: Membagi dataset menjadi 80% data training dan 20% data testing

[5]:
```
from sklearn.model_selection import train_test_split

X = data.data
Y = data.target

# Bagi menjadi 80% data training dan 20% data testing
X_train, X_test, Y_train, Y_test =  train_test_split(X, Y, test_size = 0.20)
```

4 Bagian 3: Melakukan pembelajaran untuk membentuk classifier dengan berbagai algoritma

4.1 Decision Tree

[6]:
```
from sklearn import tree
decisionTreeClassifier = tree.DecisionTreeClassifier()
decisionTreeClassifier = decisionTreeClassifier.fit(X_train, Y_train)

# Model pohon decision tree
print(tree.export_text(decisionTreeClassifier, feature_names=list(data.
  ↪feature_names)))
```

```
|--- worst perimeter <= 105.95
|   |--- worst concave points <= 0.16
|   |   |--- worst smoothness <= 0.18
|   |   |   |--- perimeter error <= 6.60
|   |   |   |   |--- area error <= 48.98
|   |   |   |   |   |--- worst perimeter <= 102.40
|   |   |   |   |   |   |--- worst texture <= 33.35
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- worst texture >  33.35
|   |   |   |   |   |   |   |--- worst texture <= 33.56
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- worst texture >  33.56
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- worst perimeter >  102.40
|   |   |   |   |   |   |--- worst texture <= 29.68
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- worst texture >  29.68
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- area error >  48.98
|   |   |   |   |   |--- radius error <= 0.69
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- radius error >  0.69
```

```
|   |   |   |   |   |   |--- class: 1
|   |   |   |--- perimeter error >  6.60
|   |   |   |   |--- class: 0
|   |   |--- worst smoothness >  0.18
|   |   |   |--- mean concavity <= 0.07
|   |   |   |   |--- class: 1
|   |   |   |--- mean concavity >  0.07
|   |   |   |   |--- class: 0
|   |--- worst concave points >  0.16
|   |   |--- class: 0
|--- worst perimeter >  105.95
|   |--- mean concave points <= 0.05
|   |   |--- worst area <= 957.45
|   |   |   |--- mean area <= 600.55
|   |   |   |   |--- class: 0
|   |   |   |--- mean area >  600.55
|   |   |   |   |--- class: 1
|   |   |--- worst area >  957.45
|   |   |   |--- worst fractal dimension <= 0.06
|   |   |   |   |--- class: 1
|   |   |   |--- worst fractal dimension >  0.06
|   |   |   |   |--- class: 0
|   |--- mean concave points >  0.05
|   |   |--- worst texture <= 20.36
|   |   |   |--- worst perimeter <= 119.70
|   |   |   |   |--- class: 1
|   |   |   |--- worst perimeter >  119.70
|   |   |   |   |--- class: 0
|   |   |--- worst texture >  20.36
|   |   |   |--- fractal dimension error <= 0.01
|   |   |   |   |--- class: 0
|   |   |   |--- fractal dimension error >  0.01
|   |   |   |   |--- class: 1
```

4.2 ID3

[7]:
```
import six
import sys
sys.modules['sklearn.externals.six'] = six

from id3 import Id3Estimator

id3Classifier = Id3Estimator()
id3Classifier = id3Classifier.fit(X_train, Y_train)
```

4.3 K-Means

[8]:
```
from sklearn.cluster import KMeans
kMeansClassifier = KMeans(n_clusters = 2, n_init = 10)
kMeansClassifier = kMeansClassifier.fit(X_train)
```

4.4 Logistic Regression

[9]:
```
# @Aira
```

4.5 Neural Network

[10]:
```
# @Aira
```

4.6 SVM

[11]:
```
# @Aira
```

5 Bagian 4: Menyimpan model hasil pembelajaran

[12]:
```
import pickle

pickle.dump(decisionTreeClassifier, open("decisionTreeClassifier.sav", "wb"))
pickle.dump(id3Classifier, open("id3Classifier.sav", "wb"))
pickle.dump(kMeansClassifier, open("kMeansClassifier.sav", "wb"))
# @Aira for logistic regression, neural network, dan SVM
```

6 Bagian 5: Melakukan prediksi dengan loaded model

6.1 Set up

[13]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score, confusion_matrix

def predictFromModel(modelFileName):
  loadedModel = pickle.load(open(modelFileName, "rb"))
  modelName = modelFileName.rstrip(".sav")
  # Lakukan prediksi
  Y_pred = loadedModel.predict(X_test)
  accuracy = accuracy_score(Y_test, Y_pred)
  precision = precision_score(Y_test, Y_pred)
  recall = recall_score(Y_test, Y_pred)
  f1 = f1_score(Y_test, Y_pred)
  cmat = confusion_matrix(Y_test, Y_pred)
  # Output!
```

```
  print(f'''Hasil prediksi dari model {modelName}:

  Accuracy: {accuracy}
  Precision: {precision}
  Recall: {recall}
  F1: {f1}

  Confusion matrix:
  {cmat}
  ''')
```

6.2 Decision Tree

[14]:
```
predictFromModel("decisionTreeClassifier.sav")
```

```
Hasil prediksi dari model decisionTreeClassifier:

  Accuracy: 0.9122807017543859
  Precision: 0.9558823529411765
  Recall: 0.9027777777777778
  F1: 0.9285714285714286

  Confusion matrix:
  [[39  3]
 [ 7 65]]
```

6.3 ID3

[15]:
```
predictFromModel("id3Classifier.sav")
```

```
Hasil prediksi dari model id3Classifier:

  Accuracy: 0.9298245614035088
  Precision: 0.9571428571428572
  Recall: 0.9305555555555556
  F1: 0.943661971830986

  Confusion matrix:
  [[39  3]
 [ 5 67]]
```

6.4 K-Means

[16]:
```
predictFromModel("kMeansClassifier.sav")
```

```
Hasil prediksi dari model kMeansClassifier:

  Accuracy: 0.13157894736842105
  Precision: 0.0
  Recall: 0.0
  F1: 0.0

  Confusion matrix:
  [[15 27]
 [72  0]]
```

6.5 Logistic Regression

[17]:
```
# @Aira
```

6.6 Neural Network

[18]:
```
# @Aira
```

6.7 SVM

[19]:
```
# @Aira
```

7 Bagian 6: Analisis hasil metrik evaluasi

Dari beberapa model yang sudah dibuat dan diuji, didapatkan hasil bahwa: 1. Skor accuracy tertinggi didapatkan oleh model ... 2. Skor precision tertinggi didapatkan oleh model ... 3. Skor recall tertinggi didapatkan oleh model ... 4. Skor F1 tertinggi didapatkan oleh model ...

Hal menarik lainnya yang ditemukan adalah karena K-Means adalah tipe learning unsupervised, skor yang didapatkan bisa rendah sekali jika ia mengklasifikan kelasnya salah (misal, yang seharusnya targetnya 0 dia klasifikasikan menjadi 1).

8 Bagian 7: K-Fold Cross Validation pada DecisionTreeClassifier

[20]:
```
from sklearn.model_selection import cross_validate

scores = cross_validate(decisionTreeClassifier, X, Y, cv=10,␣
 ↪scoring=["accuracy", "f1"])
print(f"""Data 10-fold cross validation:
  Accuracy: {scores["test_accuracy"]}, with mean {np.
 ↪mean(scores["test_accuracy"])}
  F1: {scores["test_f1"]}, with mean {np.mean(scores["test_f1"])}
""")
```

```
Data 10-fold cross validation:
  Accuracy: [0.92982456 0.85964912 0.92982456 0.89473684 0.92982456 0.9122807
 0.84210526 0.94736842 0.92982456 0.94642857], with mean 0.9121867167919799
  F1: [0.94285714 0.88888889 0.94594595 0.91666667 0.94444444 0.93150685
 0.87671233 0.95774648 0.94117647 0.95652174], with mean 0.9302466955477191
```

8.1 Perbandingan dengan skor prediksi DecisionTreeClassifier sebelumnya

Dapat dilihat bahwa ...