

LAPORAN PRAKTIKUM
MACHINE LEARNING PRAKTIKUM KE-2
EVALUATION ALGORITHM



Disusun Oleh
Khoeru Roziqin
24060119120031

Dataset: Lung Cancer

FAKULTAS SAINS DAN MATEMATIKA
PRODI INFORMATIKA/ILMU KOMPUTER KELAS B
UNIVERSITAS DIPONEGORO
SEMARANG
2021

A. Dasar Teori

Pada praktikum minggu lalu telah membahas mengenai *python* dan melakukan *summary dataset* yang terdiri dari menentukan dimensi dari dataset, melihat isi dataset, distribusi kelas data, melihat ringkasan statistik dan cara visualisasi data menggunakan *plot univariat* dan *multivariat*. Pada praktikum minggu ini akan membahas mengenai evaluasi algoritma. Evaluasi algoritma digunakan untuk memperkirakan akurasi dari data yang belum diketahui sebelumnya. Terdapat beberapa model yang dapat digunakan pada evaluasi algoritma antara lain:

1. K-Nearest Neighbors (KNN)
2. Gaussian Naive Bayes (NB)
3. Support Vector Machines (SVM).

Untuk memperkirakan akurasi pada data yang tidak diketahui sebelumnya terdapat beberapa data yang dapat dilakukan:

1. Membuat Validasi Dataset

Validasi dilakukan untuk mengetahui bahwa model yang dibuat itu bagus. Untuk memperkirakan keakuratan model yang dibuat pada data yang tidak terlihat dapat menggunakan metode statistik juga dengan mengevaluasi data aktual yang tidak terlihat. Artinya, kita akan menahan beberapa data yang tidak dapat dilihat oleh algoritma dan akan menggunakan data ini untuk mendapatkan informasi tentang seberapa akurat model terbaik sebenarnya. Datateset akan dibagi menjadi dua, 80% diantaranya akan digunakan untuk melatih model dan 20% digunakan untuk data validasi.

2. K-Folds cross Validation

Menggunakan validasi silang 10 kali lipat untuk memperkirakan akurasi. Untuk itu dataset dibagi menjadi 10 bagian, 9 untuk latihan dan 1 untuk pengujian dan ulangi untuk semua kombinasi.

3. Membangun Model

Untuk mengetahui algoritma yang cocok dengan studi kasus ini maka harus mengevaluasi dengan beberapa algoritma. Diantaranya yaitu K-Nearest Neighbors (KNN), Gaussian Naive Bayes (NB) dan Support Vector Machines (SVM).

4. Memilih model terbaik

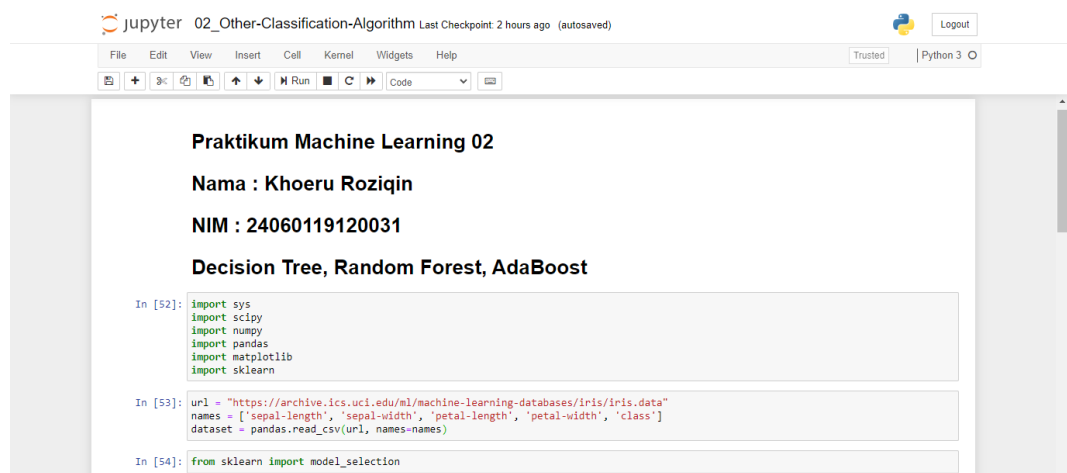
Jika sudah memiliki hasil evaluasi dari ketiga model diatas untuk memilih model terbaik dilakukan dari perbandingan satu sama lainnya dan dipilih yang paling akurat dengan melihat hasil nilai masing masing. Selanjutnya kita dapat mencoba melakukan pengujian tentang keakuratan model terhadap data yang ada.

B. Rumusan Masalah

1. Lakukan Eksplorasi terhadap algoritma klasifikasi lain yang ada!
2. Buatlah evaluasi algoritma dengan dataset yang telah dicoba pada tugas praktikum sebelumnya (dengan menggunakan 3 model yaitu KNN, NB dan SVM)!

C. Pembahasan

1. Evaluasi Algoritma dengan dataset “Iris” dengan model Decision Tree, Random Forest, dan AdaBoost



```
Praktikum Machine Learning 02
Nama : Khoeru Roziqin
NIM : 24060119120031
Decision Tree, Random Forest, AdaBoost

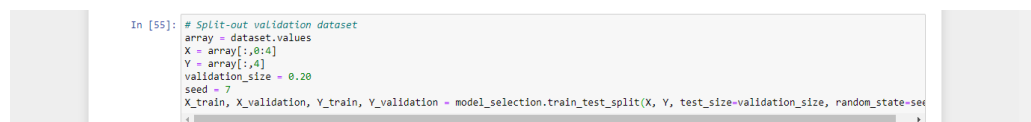
In [52]: import sys
import scipy
import numpy
import pandas
import matplotlib
import sklearn

In [53]: url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
dataset = pandas.read_csv(url, names=names)

In [54]: from sklearn import model_selection
```

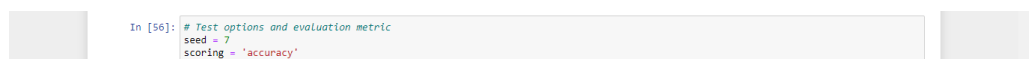
Import beberapa modul yang digunakan dan load dataset.data dan dataset.nama.

a. Membuat Validasi Dataset



```
In [55]: # Split-out validation dataset
array = dataset.values
X = array[:,0:4]
Y = array[:,4]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, Y, test_size=validation_size, random_state=seed)
```

b. K-Folds cross Validation



```
In [56]: # Test options and evaluation metric
seed = 7
scoring = 'accuracy'
```

c. Membangun Model

```
In [57]: from sklearn.tree import DecisionTreeClassifier
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.ensemble import AdaBoostClassifier

In [58]: # Spot Check Algorithms
        models = []
        models.append(('Decision Tree', DecisionTreeClassifier()))
        models.append(('Random Forest', RandomForestClassifier()))
        models.append(('AdaBoost', AdaBoostClassifier()))

In [59]: # evaluate each model in turn
        results = []
        names = []
        for name, model in models:
            kfold = model_selection.KFold(n_splits=10, random_state=seed, shuffle=True)
            cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold, scoring=scoring)
            results.append(cv_results)
            names.append(name)
            msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
            print(msg)

Decision Tree: 0.950000 (0.076376)
Random Forest: 0.966667 (0.048825)
AdaBoost: 0.966667 (0.048825)

In [60]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

d. Evaluasi

i. Decision Tree

```
In [61]: # Make predictions on validation dataset with Decision Tree
        decision_tree = DecisionTreeClassifier()
        decision_tree.fit(X_train, Y_train)
        predictions = decision_tree.predict(X_validation)
        print(accuracy_score(Y_validation, predictions))
        print(confusion_matrix(Y_validation, predictions))
        print(classification_report(Y_validation, predictions))

0.9
[[ 7  0  0]
 [ 0 10  2]
 [ 0  1 10]]
      precision    recall  f1-score   support

 Iris-setosa      1.00      1.00      1.00         7
 Iris-versicolor  0.91      0.83      0.87        12
 Iris-virginica   0.83      0.91      0.87        11

   accuracy
macro avg   0.91      0.91      0.91         30
weighted avg 0.90      0.90      0.90         30
```

ii. Random Forest

```
In [62]: # Make predictions on validation dataset with Random Forest
        random_forest = RandomForestClassifier()
        random_forest.fit(X_train, Y_train)
        predictions = random_forest.predict(X_validation)
        print(accuracy_score(Y_validation, predictions))
        print(confusion_matrix(Y_validation, predictions))
        print(classification_report(Y_validation, predictions))

0.8666666666666667
[[ 7  0  0]
 [ 0 10  2]
 [ 0  2  9]]
      precision    recall  f1-score   support

 Iris-setosa      1.00      1.00      1.00         7
 Iris-versicolor  0.83      0.83      0.83        12
 Iris-virginica   0.82      0.82      0.82        11

   accuracy
macro avg   0.88      0.88      0.88         30
weighted avg 0.87      0.87      0.87         30
```

iii. AdaBoost

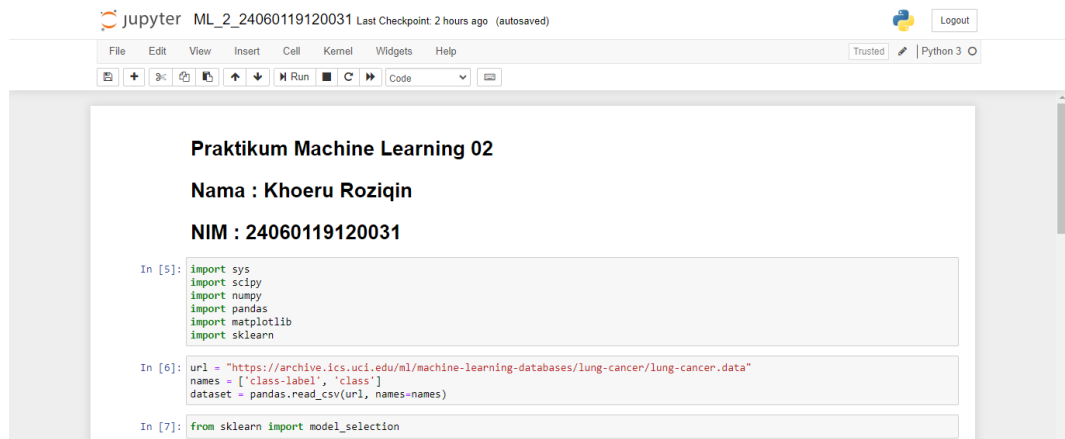
```
In [63]: # Make predictions on validation dataset with AdaBoost
        adaBoost = AdaBoostClassifier()
        adaBoost.fit(X_train, Y_train)
        predictions = adaBoost.predict(X_validation)
        print(accuracy_score(Y_validation, predictions))
        print(confusion_matrix(Y_validation, predictions))
        print(classification_report(Y_validation, predictions))

0.8666666666666667
[[ 7  0  0]
 [ 0 10  2]
 [ 0  2  9]]
      precision    recall  f1-score   support

 Iris-setosa      1.00      1.00      1.00         7
 Iris-versicolor  0.83      0.83      0.83        12
 Iris-virginica   0.82      0.82      0.82        11

   accuracy
macro avg   0.88      0.88      0.88         30
weighted avg 0.87      0.87      0.87         30
```

2. Evaluasi Algoritma dengan dataset “Lung-Cancer Dataset” dengan model KKN, Naïve Bayes, dan Support Vector Machine

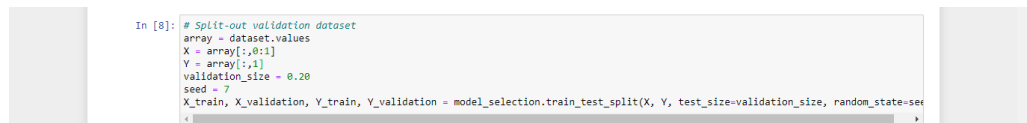


```
jupyter ML_2_24060119120031 Last Checkpoint: 2 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [5]: import sys
import scipy
import numpy
import pandas
import matplotlib
import sklearn

In [6]: url = "https://archive.ics.uci.edu/ml/machine-learning-databases/lung-cancer/lung-cancer.data"
names = ['class-label', 'class']
dataset = pandas.read_csv(url, names=names)

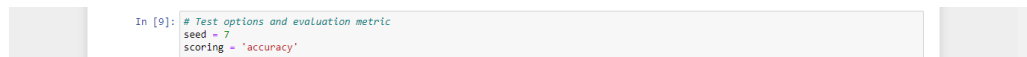
In [7]: from sklearn import model_selection
```

a. Membuat Validasi Dataset



```
In [8]: # Split-out validation dataset
array = dataset.values
X = array[:,0:1]
Y = array[:,1]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, Y, test_size=validation_size, random_state=seed)
```

b. K-Folds cross Validation



```
In [9]: # Test options and evaluation metric
seed = 7
scoring = 'accuracy'
```

c. Membangun Model



```
In [10]: from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

In [11]: # Spot Check Algorithms
models = []
models.append(('KNN', KNeighborsClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))

In [12]: # evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed, shuffle=True)
    cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)

KNN: 0.666667 (0.235702)
NB: 0.533333 (0.363624)
SVM: 0.666667 (0.235702)

In [13]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

d. Evaluasi dengan semua model

i. KKN

```
In [15]: # Make predictions on validation dataset with KKN
kkn = KNeighborsClassifier()
kkn.fit(X_train, Y_train)
predictions = kkn.predict(X_validation)
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions, zero_division=0))

0.8571428571428571
[[0 1]
 [0 6]]
      precision    recall  f1-score   support

     1       0.00       0.00       0.00      1
     2       0.86       1.00       0.92      6

   accuracy          0.43          0.50          0.46      7
  macro avg          0.43          0.50          0.46      7
 weighted avg          0.73          0.86          0.79      7
```

ii. Navie Bayes

```
In [16]: # Make predictions on validation dataset with NB
nb = GaussianNB()
nb.fit(X_train, Y_train)
predictions = nb.predict(X_validation)
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions, zero_division=0))

0.42857142857142855
[[1 0]
 [4 2]]
      precision    recall  f1-score   support

     1       0.20       1.00       0.33      1
     2       1.00       0.33       0.50      6

   accuracy          0.60          0.67          0.42      7
  macro avg          0.60          0.67          0.42      7
 weighted avg          0.89          0.43          0.48      7
```

iii. Support Vector Machine

```
In [14]: # Make predictions on validation dataset with SVM
svm = SVC()
svm.fit(X_train, Y_train)
predictions = svm.predict(X_validation)
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions, zero_division=0))

0.8571428571428571
[[0 1]
 [0 6]]
      precision    recall  f1-score   support

     1       0.00       0.00       0.00      1
     2       0.86       1.00       0.92      6

   accuracy          0.43          0.50          0.46      7
  macro avg          0.43          0.50          0.46      7
 weighted avg          0.73          0.86          0.79      7
```