Hasil Split & KFold MyMLP

In [0]:

```
import sys
sys.path.append('/content/gdrive/My Drive/6 University/Machine Learning/MLP')

import pandas as pd
import numpy as np
from MultilayerPerceptron import *
from sklearn.model_selection import train_test_split

# Confusion Matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

# Kfold
from sklearn.model_selection import KFold

def experiment(mlp, filename):
    mlp.save_model_to_file(filename)
    mlp.load_model_from_file(filename)

def split():
    print("Split 90-10")

    data = pd.read_csv("/content/gdrive/My Drive/6 University/Machine Learning/MLP/i
    inputs = data.drop('species', axis = 1)
    target = data['species']

    #training
    df = pd.concat([inputs, target], axis=1)
    train, test = train_test_split(df, test_size=0.1)

    trainX = train[['sepal_length','sepal_width','petal_length','petal_width']] # ta
    trainY = train.species # output of our training data
    testX = test[['sepal_length','sepal_width','petal_length','petal_width']] # taki
    testY = test.species   # output value of test data

    trainX = trainX.reset_index(drop=True)
    trainY = trainY.reset_index(drop=True)
    testX = testX.reset_index(drop=True)
    testY = testY.reset_index(drop=True)
    mlp = myMLP()
    mlp.fit(trainX, trainY)
    experiment(mlp, "Output-MLP-Split-Test.txt")
    prediction = mlp.predict(testX)
    confusion_matrix_results = confusion_matrix(testY.values, prediction)

    print("Confusion Matrix: ")
    print(confusion_matrix_results)
    print("Accuracy Score: ")
    print(accuracy_score(testY.values, prediction))
    print("Classification Report: ")
    print(classification_report(testY.values, prediction))



def kfold():
    print("K-Fold")

    #Training k-fold
```

```
    data = pd.read_csv("/content/gdrive/My Drive/6 University/Machine Learning/MLP/i
    inputs = data.drop('species', axis = 1)
    target = data['species']
    df = pd.concat([inputs, target], axis=1)
    train, test = train_test_split(df, test_size=0.2)  # divide 150 data to x and y

    inputX = train[['sepal_length','sepal_width','petal_length','petal_width']] # ta
    inputY = train.species # output of our training data
    testX = test[['sepal_length','sepal_width','petal_length','petal_width']] # taki
    testY = test.species   # output value of test data

    inputX = inputX.reset_index(drop=True)
    inputY = inputY.reset_index(drop=True)
    testX = testX.reset_index(drop=True)
    testY = testY.reset_index(drop=True)

    #k-fold
    kf = KFold(n_splits=10)
    kf.get_n_splits(inputX)
    indices = kf.split(inputX)

    mainmlp = myMLP()
    maxacc = 0

    for train_index, test_index in indices:
        trainX, valX = inputX.iloc[train_index], inputX.iloc[test_index]
        trainY, valY = inputY.iloc[train_index], inputY.iloc[test_index]

        trainX = trainX.reset_index(drop=True)
        trainY = trainY.reset_index(drop=True)
        valX = valX.reset_index(drop=True)
        valY = valY.reset_index(drop=True)

        mlp = myMLP()
        mlp.fit(trainX, trainY)
        experiment(mlp, "Output-MLP-Kfold.txt")
        prediction = mlp.predict(valX)
        if (accuracy_score(valY.values, prediction) > maxacc):
            mainmlp = mlp
            maxacc = accuracy_score(valY.values, prediction)
    finalPrediction = mainmlp.predict(testX)
    print("Accuracy Score: ")
    print(accuracy_score(testY.values, finalPrediction))


split()
kfold()
```

```
Split 90-10
Confusion Matrix:
[[3 0 0]
 [0 4 1]
 [0 0 7]]
Accuracy Score:
0.9333333333333333
Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         3
Iris-versicolor       1.00      0.80      0.89         5
 Iris-virginica       0.88      1.00      0.93         7
```

```
       accuracy                           0.93        15
      macro avg       0.96      0.93      0.94        15
   weighted avg       0.94      0.93      0.93        15
```

```
K-Fold
Accuracy Score:
0.9666666666666667
```

Hasil Split dan Kfold DTL

In [61]:

```
import pandas as pd
from myc45 import *

import sys
sys.path.append('/content/gdrive/My Drive/6 University/Machine Learning/MLP')

# Confusion Matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

# Kfold
from sklearn.model_selection import KFold

pd.options.mode.chained_assignment = None

def experiment(c45):
    c45.id3.tree_.save_to_file("Output-C45.txt")
    c45.id3.tree_.load_from_file("Output-C45.txt")
    dict = {
        "sepal_length": 5.5,
        "sepal_width": 3.3,
        "petal_length": 2.0,
        "petal_width": 0.65
    }
    print(c45.classify(c45.id3.tree_, dict))

def split():
    print("Split 90-10")

    df = pd.read_csv('/content/gdrive/My Drive/6 University/Machine Learning/MLP/iri
    train, test = train_test_split(df, test_size=0.1)

    # Main Program Goes Here:
    data = pd.read_csv("/content/gdrive/My Drive/6 University/Machine Learning/MLP/i

    inputs = data.drop('species', axis = 1)
    target = data['species']

    #training
    df = pd.concat([inputs, target], axis=1)
    train, test = train_test_split(df, test_size=0.1)
    train = train.reset_index(drop=True)

    testX = test[['sepal_length','sepal_width','petal_length','petal_width']] # taki
    testY = test.species   # output value of test data

    testX = testX.reset_index(drop=True)
    testY = testY.reset_index(drop=True)

    attributes = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
    target = 'species'

    c45 = myC45(train, target, attributes)
    prediction = []
    for i in range(len(testX)):
        prediction.append(c45.classify(c45.id3.tree_, testX.iloc[i]))
    confusion_matrix_results = confusion_matrix(testY.values, prediction)
```

```
    print("Confusion Matrix: ")
    print(confusion_matrix_results)
    print("Accuracy Score: ")
    print(accuracy_score(testY.values, prediction))
    print("Classification Report: ")
    print(classification_report(testY.values, prediction))
    # experiment(c45)


def kfold():
    print("K-Fold")
    #Training k-fold
    df = pd.read_csv('/content/gdrive/My Drive/6 University/Machine Learning/MLP/iri
    data, test = train_test_split(df, test_size=0.2)  # divide 150 data to x and y =

    dataX = data[['sepal_length','sepal_width','petal_length','petal_width']] # taki
    dataY = data.species # output of our training data
    testX = test[['sepal_length','sepal_width','petal_length','petal_width']] # taki
    testY = test.species   # output value of test data

    dataX = dataX.reset_index(drop=True)
    dataY = dataY.reset_index(drop=True)
    testX = testX.reset_index(drop=True)
    testY = testY.reset_index(drop=True)

    #k-fold
    kf = KFold(n_splits=10)
    kf.get_n_splits(data)
    indices = kf.split(data)

    mainc45 = None
    maxacc = 0;

    for train_index, test_index in indices:
        train = data.iloc[train_index]
        train = train.reset_index(drop=True)

        valX, valY = dataX.iloc[test_index], dataY.iloc[test_index]
        valX = valX.reset_index(drop=True)
        valY = valY.reset_index(drop=True)

        attributes = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
        target = 'species'
        c45 = myC45(train, target, attributes)
        prediction = []
        for i in range(len(valX)):
            prediction.append(c45.classify(c45.id3.tree_, valX.iloc[i]))
        if (accuracy_score(valY.values, prediction) > maxacc):
            mainc45 = c45
            maxacc = accuracy_score(valY.values, prediction)
    finalPrediction = []
    for i in range(len(testX)):
        finalPrediction.append(mainc45.classify(mainc45.id3.tree_, testX.iloc[i]))
    print("Accuracy Score: ")
    print(accuracy_score(testY.values, finalPrediction))
    # experiment(mainc45)


split()
kfold()
```

```
Split 90-10
```

```
-
Confusion Matrix:
[[3 0 0]
 [0 4 5]
 [0 0 3]]
Accuracy Score:
0.6666666666666666
Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         3
Iris-versicolor       1.00      0.44      0.62         9
 Iris-virginica       0.38      1.00      0.55         3

       accuracy                           0.67        15
      macro avg       0.79      0.81      0.72        15
   weighted avg       0.88      0.67      0.68        15

K-Fold
Accuracy Score:
0.6333333333333333
```

Analisis

File eksternal

Sebelum setiap metode (confusion matrix maupun k-fold) dilakukan, DTL maupun MLP men-save dan men-load file ke dalam file eksternal.

Model DTL

Model DTL (C45) disave dalam bentuk tree. Model DTL diload dengan cara membaca setiap spasi dari file eksternal, lalu jumlah spasi tersebut akan dijadikan referensi apakah dia nama atribut atau value dari nama atribut tersebut. Selain itu, jumlah spasi yang semakin dalam menandakan bahwa nama atribut / value dari nama atribut tersebut adalah milik pohon generasi yang lebih muda (contoh: spasi 1 = parent, spasi 2 = anak).

Model ANN

Model ANN (MLP) di save ke dalam file eksternal berdasarkan jumlah layer, weight, dan perceptron pada setiap layer. Model MLP di-load dari file eksternal berdasarkan lokasi dari layer, weight, dan perceptron. Setiap line baru pada file eksternal menandakan layer baru. Setiap weight akan dibatasi dengan "p" pada file eksternal. Sebelum terbaca huruf "p", weight adalah milik satu perceptron.

Contoh:

-5.31792429650554 5.08384530165187 -2.2912199946989236 p

Maka:

weight 1 = -5.31792429650554, weight 2 = 5.08384530165187, weight 3 = -2.2912199946989236 milik perceptron 1.

Notes About ANN

Pada kasus ANN (MLP), ANN(MLP) dilatih dengan hidden layer 3 buah dengan masing-masing: 10 perceptron, 5 perceptron, dan 2 perceptron. Learning rate yang digunakan adalah 0,05 dan Iterasi maksimumnya adalah 400. Diberikan batch_size = 32 dan treshold eror = 10^-4.

Train-Split Test

Pada DTL dan ANN (MLP), kami menggunakan data split yaitu 90-10, dan kami split menggunakan train_test_split yang tersedia di library `sklearn.model_selection` . Artinya, 90% digunakan sebagai data training dan 10% digunakan sebagai data testing. Pemilihan data training dan data testing dilakukan secara random!

Diperlukan Confusion Matrix pada Train-Split test. Untuk mengimplementasi Confusion Matrix, menghitung akurasi, dan membuat Classification Report, kami menggunakan library `sklearn.metrics` .

Pada Classification Report, terdapat:
Precision: ratio true positive dari seluruh hasil positive yang didapatkan (tp / (tp + fp)) di mana tp merupakan angka kasus true positive dan fp merupakan angka kasus false positive
Recall: ratio true positive dari hasil positive yang seharusnya ada (tp / (tp + fn)) di mana tp merupakan angka kasus true positive dan fn merupakan angka kasus false negative
F1-score: Harmonic mean dari Precision dan Recall
Support: jumlah kemunculan setiap kelas di hasil sebenarnya

Pada kasus MLP, Didapatkan akurasi 93.33%. Artinya hasil prediksi menggunakan MLP 93.33% benar terhadap hasil seharusnya.

Pada kasus DTL, Didapatkan akurasi 66.66%. Artinya hasil prediksi menggunakan DTL 66.66% benar terhadap hasil seharusnya.

Analisis

File eksternal

Sebelum setiap metode (confusion matrix maupun k-fold) dilakukan, DTL maupun MLP men-save dan men-load file ke dalam file eksternal.

Model DTL

Model DTL (C45) disave dalam bentuk tree. Model DTL diload dengan cara membaca setiap spasi dari file eksternal, lalu jumlah spasi tersebut akan dijadikan referensi apakah dia nama atribut atau value dari nama atribut tersebut. Selain itu, jumlah spasi yang semakin dalam menandakan bahwa nama atribut / value dari nama atribut tersebut adalah milik pohon generasi yang lebih muda (contoh: spasi 1 = parent, spasi 2 = anak).

Model ANN

Model ANN (MLP) di save ke dalam file eksternal berdasarkan jumlah layer, weight, dan perceptron pada setiap layer. Model MLP di-load dari file eksternal berdasarkan lokasi dari layer, weight, dan perceptron. Setiap line baru pada file eksternal menandakan layer baru. Setiap weight akan dibatasi dengan "p" pada file eksternal. Sebelum terbaca huruf "p", weight adalah milik satu perceptron.

Contoh:

-5.31792429650554 5.08384530165187 -2.2912199946989236 p

Maka:

weight 1 = -5.31792429650554, weight 2 = 5.08384530165187, weight 3 = -2.2912199946989236 milik perceptron 1.

Notes About ANN

Pada kasus ANN (MLP), ANN(MLP) dilatih dengan hidden layer 3 buah dengan masing-masing: 10 perceptron, 5 perceptron, dan 2 perceptron. Learning rate yang digunakan adalah 0,05 dan Iterasi maksimumnya adalah 400. Diberikan batch_size = 32 dan treshold eror = 10^-4.

Terminologi

Pada Classification Report, terdapat:
Precision: ratio true positive dari seluruh hasil positive yang didapatkan (tp / (tp + fp)) di mana tp merupakan angka kasus true positive dan fp merupakan angka kasus false positive
Recall: ratio true positive dari hasil positive yang seharusnya ada (tp / (tp + fn)) di mana tp merupakan angka kasus true positive dan fn merupakan angka kasus false negative
F1-score: Harmonic mean dari Precision dan Recall
Support: jumlah kemunculan setiap kelas di hasil sebenarnya

Train-Split Test

Pada DTL dan ANN (MLP), kami menggunakan data split yaitu 90-10, dan kami split menggunakan train_test_split yang tersedia di library `sklearn.model_selection`. Artinya, 90% digunakan sebagai data training dan 10% digunakan sebagai data testing. Pemilihan data training dan data testing dilakukan secara random!

Diperlukan Confusion Matrix pada Train-Split test. Untuk mengimplementasi Confusion Matrix, menghitung akurasi, dan membuat Classification Report, kami menggunakan library `sklearn.metrics`.

Kasus ANN

Pada kasus ANN, Didapatkan akurasi `93,3333%`. Perhatikan bahwa 93.3333% adalah hasil yang diuji kepada data testing. Confusion Matrix yang didapat adalah:

[[3 0 0] [0 4 1] [0 0 7]]

*Kelas 1 = Iris-setosa *Kelas 2 = Iris-versicolor *Kelas 3 = Iris-virginica

Artinya, 3 instance berhasil diklasifikasi di kelas 1 (true kelas 1), 4 instance berhasil diklasifikasi di kelas 2 (true kelas 2), namun terdapat 1 instance yang seharusnya berada di kelas 3 namun diklasifikasikan di kelas 2. Kemudian, terdapat 7 instance yang berhasil diklasifikasi di kelas 3 (true kelas 3).

Sesuai dengan terminologi, precision dari kelas Iris-setosa adalah 1, Iris-versicolor adalah 1, dan Iris-virginica adalah 0.88. Perhatikan bahwa 0.88 didapat dari `7 instance / (7 instance benar + 1 instance seharusnya kelas 3 namun terdeteksi di kelas 2)`.

Recall dari kelas Iris-setosa adalah 1, Iris-versicolor adalah 0.8, dan Iris-virginica adalah 1. Perhatikan bahwa 0.8 didapat dari 4 instance / (4 instance benar + 1 instance seharusnya bukan kelas 2)

f1-score dari kelas Iris-virginica adalah 1, Iris-versicolor adalah 0.89, dan Iris-virginica adalah 0.93. f1-score didapat dari rata-rata harmonik antara presisi dan recall.

Support adalah jumlah kemunculan instance pada hasil sebenarnya: Iris-setosa = 3, Iris-versicolor = 5, Iris-virginica = 7.

Kasus DTL

Pada kasus DTL, Didapatkan akurasi `66.666%` . Perhatikan bahwa 66.666% adalah hasil yang diuji kepada data testing. Confusion Matrix yang didapat adalah:

[[3 0 0] [0 4 5] [0 0 3]]

*Kelas 1 = Iris-setosa *Kelas 2 = Iris-versicolor *Kelas 3 = Iris-virginica

Artinya, 3 instance berhasil diklasifikasi di kelas 1 (true kelas 1), 4 instance berhasil diklasifikasi di kelas 2 (true kelas 2), namun terdapat 5 instance yang seharusnya berada di kelas 3 namun diklasifikasikan di kelas 2. Kemudian, terdapat 3 instance yang berhasil diklasifikasi di kelas 3 (true kelas 3).

Sesuai dengan terminologi, precision dari kelas Iris-setosa adalah 1, Iris-versicolor adalah 1, dan Iris-virginica adalah 0.38. 0.38 didapat dari `3 instance benar / 3 instance benar + 5 instance yang seharusnya diklasifikasikan di kelas 1` .

Recall dari kelas Iris-setosa adalah 1, Iris-versicolor adalah 0.44, dan Iris-virginica adalah 1. Perhatikan bahwa 0.44 didapat dari `4 instance benar / (4 instance benar + 5 instance seharusnya bukan kelas 2)` .

f1-score dari kelas Iris-virginica adalah 1, Iris-versicolor adalah 0.62, dan Iris-virginica adalah 0.55. f1-score didapat dari rata-rata harmonik antara presisi dan recall.

Support adalah jumlah kemunculan instance pada hasil sebenarnya: Iris-setosa = 3, Iris-versicolor = 9, Iris-virginica = 3.

Kfold

Untuk mengimplementasi 10-fold, kami menggunakan library sklearn.model_selection. Dari seluruh data yang ada, kami mengambil split 80-20. Artinya, 80% digunakan sebagai data training dan 20% digunakan sebagai data testing, dengan pemilihan data training dan data testing dilakukan secara random. Lalu, dari 80% data training tersebut pada kfold, akan dilakukan pembagian, dengan setiap pembagian akan mengandung 1 data validasi dan 9 data training. Lalu akan dicek setiap pembagian tersebut dan akan dibandingkan dengan data training baru, lalu mengambil pembagian yang memiliki akurasi prediksi yang paling tinggi.

Pada kasus MLP, Didapatkan akurasi 96.67%. Artinya hasil prediksi menggunakan MLP 96.67% benar terhadap hasil seharusnya.

Pada kasus DTL, Didapatkan akurasi 63.33%. Artinya hasil prediksi menggunakan DTL 63.33% benar terhadap hasil seharusnya.

Kesimpulan

Menurut hasil yang sudah didapatkan, ANN menggunakan myMLP lebih bagus dibandingkan DTL karena akurasinya lebih tinggi. Alasannya adalah, jumlah hidden layer yang tepat (3 buah), iterasi sebanyak 400, dan Xavier Initialization untuk menentukan weight awal.

In [0]:

```
"/"
```