Tubes-Experimental-Scheme

March 20, 2020

In [22]:
```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn import tree
from sklearn import neural_network
from sklearn.metrics import confusion_matrix
from sklearn.tree import export_text
from id3 import Id3Estimator
from id3 import export_text as id3export_text
import numpy as np
import joblib

from sklearn.model_selection import train_test_split
```

In [23]:
```
# Iris
# iris Features: [Sepal Length, Sepal Width, Petal Length, Petal Width]
# iris target: {Setosa, Versicolour, Virginica}
iris = datasets.load_iris()
iris_X = iris.data
iris_y = iris.target
print(iris.target_names)
```

```
['setosa' 'versicolor' 'virginica']
```

In [24]:
```
# DTL Skema Split Train
# TODO: Confusion matrix
dtl_model = tree.DecisionTreeClassifier()
dtl_train_X, dtl_test_X, dtl_train_y, dtl_test_y = train_test_split(iris_X, iris_y, te
dtl_model = dtl_model.fit(dtl_train_X, dtl_train_y)

y_pred = dtl_model.predict(dtl_test_X)

matrix = confusion_matrix(dtl_test_y, y_pred)

pred_column = []
truth_row = []

for i in range(0, len(matrix)):
```

```
            pred_column.append(1)
            truth_row.append(1)
            tempSumPred = 0
            tempSumTruth = 0
            for j in range(0, len(matrix[i])):
                tempSumPred += matrix[j][i]
                tempSumTruth += matrix[i][j]
            pred_column[i] = max(pred_column[i], tempSumPred)
            truth_row[i] = max(truth_row[i], tempSumTruth)

        tempSumPred = 0
        tempSumTruth = 0
        for i in range(0, len(matrix)):
            tempSumPred += matrix[i][i]/pred_column[i] # precision
            tempSumTruth += matrix[i][i]/truth_row[i] # recall
        tempSumPred /= len(matrix)
        tempSumTruth /= len(matrix)

        print("Precision: ", tempSumPred*100)
        print("Recall   : ", tempSumTruth*100)

        print(matrix)

        # Kinerja
        print("Kinerja DTL dengan skema Split Train =", dtl_model.score(dtl_test_X, dtl_test_y
```

```
Precision:  96.74603174603175
Recall   :  96.74603174603175
[[19  0  0]
 [ 0 20  1]
 [ 0  1 19]]
Kinerja DTL dengan skema Split Train = 96.66666666666667 %
```

```
In [25]: # DTL Skema 10-fold cross validation
         from sklearn.model_selection import cross_val_score

         ten_fold_score = cross_val_score(dtl_model, iris_X, iris_y, cv=10)
         print("Kinerja DTL dengan skema 10-fold cross validation =\n", list(ten_fold_score))
         print("Rata-rata akurasi = %0.2f (+/- %0.2f)" % (ten_fold_score.mean(), ten_fold_score
```

```
Kinerja DTL dengan skema 10-fold cross validation =
 [1.0, 0.9333333333333333, 1.0, 0.9333333333333333, 0.9333333333333333, 0.8666666666666667, 0.9
Rata-rata akurasi = 0.95 (+/- 0.09)
```

```
In [26]: # ANN Skema Split Train
         # TODO: Confusion Matrix
         ANN_model = neural_network.MLPClassifier(hidden_layer_sizes=(3, 2), max_iter=10000, ra
```

```
        ANN_train_X, ANN_test_X, ANN_train_y, ANN_test_y = train_test_split(iris_X, iris_y, te
        ANN_model = ANN_model.fit(ANN_train_X, ANN_train_y)

        y_pred = ANN_model.predict(ANN_test_X)

        matrix = confusion_matrix(ANN_test_y, y_pred)

        pred_column = []
        truth_row = []

        for i in range(0, len(matrix)):
            pred_column.append(1)
            truth_row.append(1)
            tempSumPred = 0
            tempSumTruth = 0
            for j in range(0, len(matrix[i])):
                tempSumPred += matrix[j][i]
                tempSumTruth += matrix[i][j]
            pred_column[i] = max(pred_column[i], tempSumPred)
            truth_row[i] = max(truth_row[i], tempSumTruth)

        tempSumPred = 0
        tempSumTruth = 0
        for i in range(0, len(matrix)):
            tempSumPred += matrix[i][i]/pred_column[i] # precision
            tempSumTruth += matrix[i][i]/truth_row[i] # recall
        tempSumPred /= len(matrix)
        tempSumTruth /= len(matrix)

        print("Precision: ", tempSumPred*100)
        print("Recall   : ", tempSumTruth*100)

        print(matrix)
        print("Kinerja ANN dengan skema Split Train =", ANN_model.score(ANN_test_X, ANN_test_y
```

```
Precision:  96.96969696969697
Recall   :  96.82539682539682
[[19  0  0]
 [ 0 19  2]
 [ 0  0 20]]
Kinerja ANN dengan skema Split Train = 96.66666666666667 %
```

```
In [27]: # ANN Skema 10-fold cross validation
         from sklearn.model_selection import cross_val_score

         ten_fold_score = cross_val_score(ANN_model, iris_X, iris_y, cv=10)
         print("Kinerja ANN dengan skema 10-fold cross validation =\n", list(ten_fold_score))
         print("Rata-rata akurasi = %0.2f (+/- %0.2f)" % (ten_fold_score.mean(), ten_fold_score
```

```
Kinerja ANN dengan skema 10-fold cross validation =
 [1.0, 1.0, 1.0, 0.9333333333333333, 0.8666666666666667, 0.8666666666666667, 0.9333333333333333
Rata-rata akurasi = 0.96 (+/- 0.11)
```

```
In [28]: # Export DTL Model
         print(dtl_model)
         filename = 'finalized_model.sav'
         joblib.dump(dtl_model, filename)

         loaded_model = joblib.load(filename)
         result = loaded_model.score(dtl_test_X, dtl_test_y)
         print(result)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
0.9666666666666667
```

```
In [29]: # Export ANN Model
         print(ANN_model)
         filename = 'finalized_mlp_model.sav'
         joblib.dump(ANN_model, filename)

         loaded_model = joblib.load(filename)
         result = loaded_model.score(ANN_test_X, ANN_test_y)
         print(result)
```

```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(3, 2), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=10000,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=3, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
0.9666666666666667
```