Tugas Besar - Eksplorasi Jupyter Notebook

1. Tobi Santoso (14115029)
2. Agung Prawoso (14115038)

Best Regards,

Informatics Student, Sumatera Institute of Technology

A. Membaca dataset standar iris dan dataset play-tennis

1. Membaca dataset standar iris menggunakan sklearn.datasets

In [114]:

```
from sklearn import datasets
df_iris = datasets.load_iris()
print('Fill in the iris dataframe:')
for x in df_iris: print(x)
```

```
Fill in the iris dataframe:
data
target
target_names
DESCR
feature_names
filename
```

In [2]:

```
print(df_iris.DESCR)
print()
print('Data sample and target')
print(df_iris.feature_names)
print(df_iris.data[:10])
print(df_iris.target_names)
print(df_iris.target[:10])
```

```
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the
class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset
is taken
from Fisher's paper. Note that it's the same as in R, but not as in
the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the
field and
is referenced frequently to this day.  (See Duda & Hart, for exampl
e.)  The
data set contains 3 classes of 50 instances each, where each class r
efers to a
type of iris plant.  One class is linearly separable from the other
2; the
latter are NOT linearly separable from each other.

.. topic:: References

   - Fisher, R.A. "The use of multiple measurements in taxonomic pro
blems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contribut
ions to
     Mathematical Statistics" (John Wiley, NY, 1950).
```

```
   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scen
e Analysis.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 21
8.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New S
ystem
     Structure and Classification Rule for Recognition in Partially
Exposed
     Environments".  IEEE Transactions on Pattern Analysis and Machi
ne
     Intelligence, Vol. PAMI-2, No. 1, 67-71.
   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE T
ransactions
     on Information Theory, May 1972, 431-433.
   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOC
LASS II
     conceptual clustering system finds 3 classes in the data.
   - Many, many more ...

Data sample and target
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'peta
l width (cm)']
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]]
['setosa' 'versicolor' 'virginica']
[0 0 0 0 0 0 0 0 0 0]
```

2. Membaca dataset play-tennis (dataset eksternal dalam format csv)

In [3]:

```
import pandas as pd

import os

os.getcwd()
```

Out[3]:

```
'/Users/Mac'
```

In [4]:

```
os.listdir(os.getcwd())
```

Out[4]:

```
['Tucil2_13515096.ipynb',
 '.config',
 'Music',
 '.condarc',
 'go',
 '.DS_Store',
 'VirtualBox VMs',
 '.CFUserTextEncoding',
 '.bash_profile.save',
 '.ServiceHub',
 '.templateengine',
 '.serverauth.24776',
 '.local',
 'Creative Cloud Files',
 '.serverauth.15438',
 'Pictures',
 '.gnome2',
 'TUCIL 2 AI.ipynb',
 '.ipython',
 'Desktop',
 'Library',
 '.matplotlib',
 '.bitnami',
 '.emulator_console_auth_token',
 'iCloud Drive (Archive)',
 '.serverauth.3483',
 '.IdentityService',
 '.nuuid.ini',
 '.android',
 '.cups',
 '.bash_sessions',
 'iris.csv',
 'a.out',
 'Public',
 'geosvg.svg',
 'tennis.csv',
 '.anaconda',
 'Movies',
 'Applications',
 '.gradle',
 '.Trash',
 '.ipynb_checkpoints',
 'New document 1.2018_09_14_20_57_51.0.svg',
 '.jupyter',
 'Mainyuk.svg',
 'Documents',
 '.mono',
 'NetBeansProjects',
 '.bash_profile',
 'AndroidStudioProjects',
 '.Xauthority',
 'Downloads',
 '.cache',
 'Tubes_14115029_14115038.ipynb',
 '.bash_history',
 '.conda']
```

In [5]:

```
pd.read_csv('tennis.csv')
```

Out[5]:

	outlook	temp	humidity	windy	play
0	sunny	hot	high	False	no
1	sunny	hot	high	True	no
2	overcast	hot	high	False	yes
3	rainy	mild	high	False	yes
4	rainy	cool	normal	False	yes
5	rainy	cool	normal	True	no
6	overcast	cool	normal	True	yes
7	sunny	mild	high	False	no
8	sunny	cool	normal	False	yes
9	rainy	mild	normal	False	yes
10	sunny	mild	normal	True	yes
11	overcast	mild	high	True	yes
12	overcast	hot	normal	False	yes
13	rainy	mild	high	True	no

B. Melakukan Pembelajaran FULL TRAINING

1. NaiveBayes
2. DecisionTree ID3
3. kNN
4. Neural Network ML

Melakukan pembelajaran untuk dataset iris dengan skema full-training, dan menampilkan modelnya.

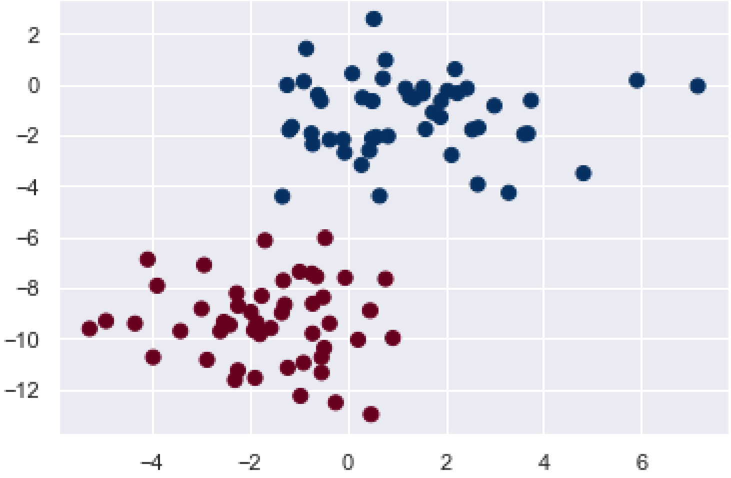
In [116]:

```
from sklearn.naive_bayes import GaussianNB
from sklearn import tree
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
```

1. NaiveBayes

In [10]:

```
from sklearn.datasets import make_blobs
X, y = make_blobs(100, 2, centers=2, random_state=2, cluster_std=1.5)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='RdBu');
```



In [11]:

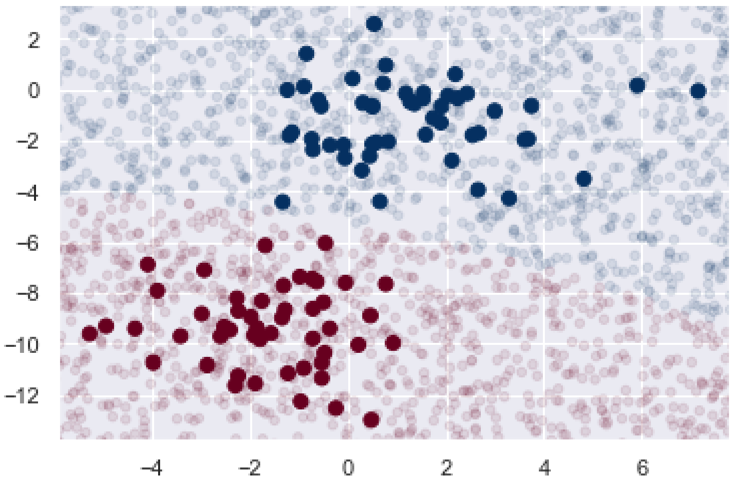
```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X, y);
```

In [12]:

```
rng = np.random.RandomState(0)
Xnew = [-6, -14] + [14, 18] * rng.rand(2000, 2)
ynew = model.predict(Xnew)
```

In [13]:

```
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='RdBu')
lim = plt.axis()
plt.scatter(Xnew[:, 0], Xnew[:, 1], c=ynew, s=20, cmap='RdBu', alpha=0.1)
plt.axis(lim);
```



In [14]:

```
yprob = model.predict_proba(Xnew)
yprob[-8:].round(2)
```

Out[14]:

```
array([[0.89, 0.11],
       [1.  , 0.  ],
       [1.  , 0.  ],
       [1.  , 0.  ],
       [1.  , 0.  ],
       [1.  , 0.  ],
       [0.  , 1.  ],
       [0.15, 0.85]])
```

1. NaiveBayes - 2

In [117]:

```
clf_GNB = GaussianNB()
clf_GNB.fit(X_train_full, y_train_full)
```

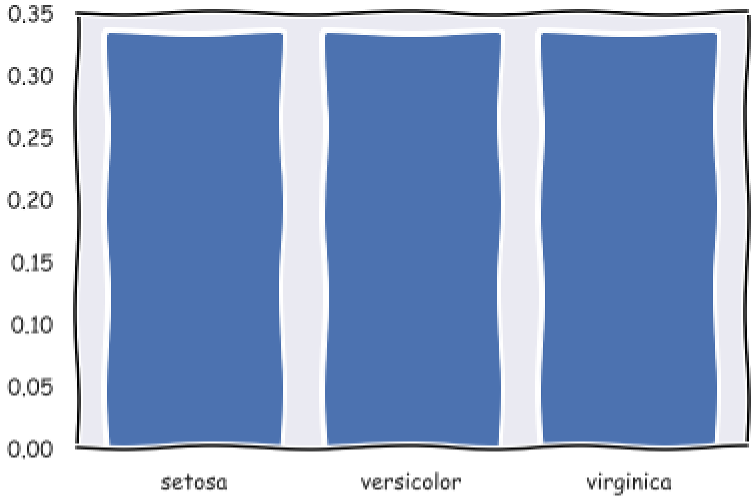
Out[117]:

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [17]:

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.xkcd()

print('Prior probability')
for i in range(3):
    print(df_iris.target_names[i] + ':', clf_GNB.class_prior_[i])
plt.bar([0,1,2], clf_GNB.class_prior_)
plt.xticks([0,1,2],df_iris.target_names)
plt.show()
```

```
Prior probability
setosa: 0.3333333333333333
versicolor: 0.3333333333333333
virginica: 0.3333333333333333
```



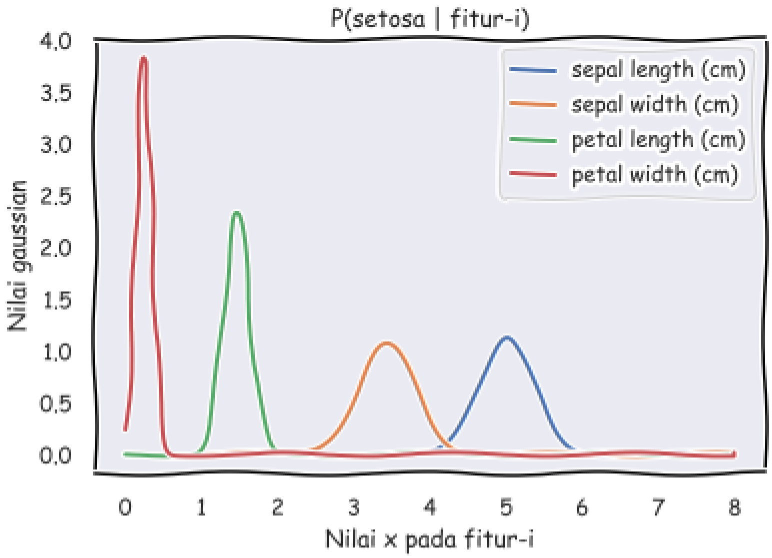
In [118]:

```
from math import exp, sqrt, pi
import numpy as np

# Untuk mengeluarkan nilai gaussian
def gaussian(x, theta, sigma):
    return 1/sqrt(2*pi*sigma) * exp((-(x-theta)**2)/(2*sigma))

# Untuk menggambar grafik Gaussian
def model_gnb(lot, los, tar):
    print('Untuk kelas', df_iris.target_names[tar])
    print('Nilai theta')
    for j in range(4): print(df_iris.target_names[tar] + ' | ' + df_iris.feature
_names[j] + ':', clf_GNB.theta_[tar][j])

    print('\nNilai sigma')
    for j in range(4): print(df_iris.target_names[tar] + ' | ' + df_iris.feature
_names[j] + ':', clf_GNB.sigma_[tar][j])

    eks = list(np.arange(0.0, 8.0, 0.01))
    plt.xlabel('Nilai x pada fitur-i')
    plt.ylabel('Nilai gaussian')
    for feat in range(4):
        ye = [gaussian(elem, lot[tar][feat], los[tar][feat]) for elem in eks]
        plt.title('P(' + df_iris.target_names[tar] + ' | fitur-i)')
        plt.plot(eks, ye, label = df_iris.feature_names[feat])
    plt.legend(loc='upper right')
    plt.show()
```

In [19]:

```
model_gnb(clf_GNB.theta_, clf_GNB.sigma_, 0)
```

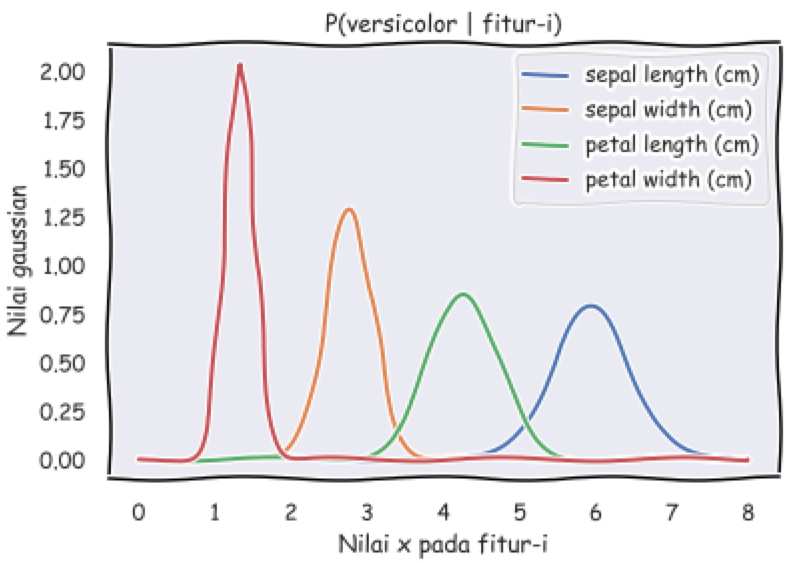
```
Untuk kelas setosa
Nilai theta
setosa | sepal length (cm): 5.005999999999999
setosa | sepal width (cm): 3.428000000000001
setosa | petal length (cm): 1.4620000000000002
setosa | petal width (cm): 0.2459999999999999

Nilai sigma
setosa | sepal length (cm): 0.12176400309550259
setosa | sepal width (cm): 0.14081600309550263
setosa | petal length (cm): 0.029556003095502676
setosa | petal width (cm): 0.010884003095502673
```



In [20]:

```
model_gnb(clf_GNB.theta_, clf_GNB.sigma_, 1)
```

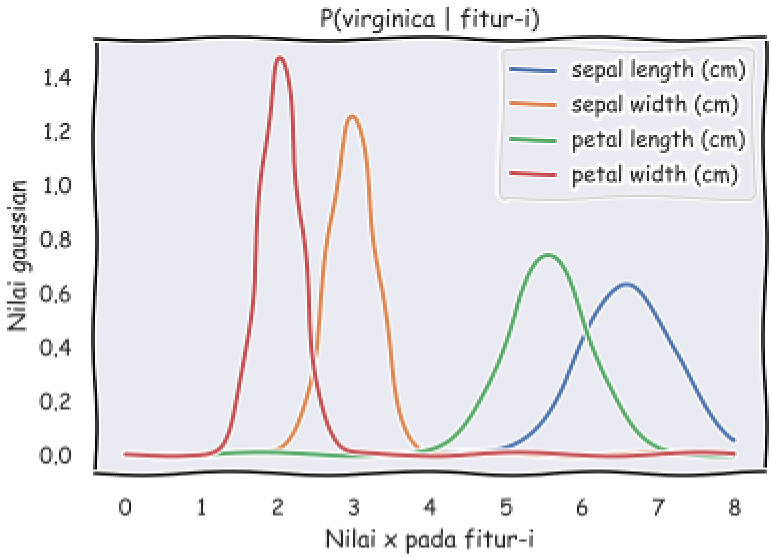
```
Untuk kelas versicolor
Nilai theta
versicolor | sepal length (cm): 5.936
versicolor | sepal width (cm): 2.7700000000000005
versicolor | petal length (cm): 4.26
versicolor | petal width (cm): 1.3259999999999998

Nilai sigma
versicolor | sepal length (cm): 0.2611040030955028
versicolor | sepal width (cm): 0.09650000309550268
versicolor | petal length (cm): 0.21640000309550278
versicolor | petal width (cm): 0.03832400309550265
```



In [21]:

```
model_gnb(clf_GNB.theta_, clf_GNB.sigma_, 2)
```

```
Untuk kelas virginica
Nilai theta
virginica | sepal length (cm): 6.587999999999998
virginica | sepal width (cm): 2.9739999999999998
virginica | petal length (cm): 5.552
virginica | petal width (cm): 2.026

Nilai sigma
virginica | sepal length (cm): 0.39625600309550263
virginica | sepal width (cm): 0.10192400309550273
virginica | petal length (cm): 0.2984960030955029
virginica | petal width (cm): 0.07392400309550265
```



2. DecisionTree ID3

In [10]:

```
from sklearn.datasets import *
from sklearn import tree

from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import label_binarize

n_classes = 3
wine = load_wine()
clf = tree.DecisionTreeClassifier()

train_x, test_x, train_y, test_y =train_test_split(wine.data, wine.target,
                                                   test_size=0.2, random_state=6
66)
# binarize class labels to plot ROC
train_y = label_binarize(train_y, classes=[0, 1, 2])
test_y = label_binarize(test_y, classes=[0, 1, 2])

y_score = clf.fit(train_x, train_y).predict(test_x)

fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(test_y[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(test_y.ravel(), y_score.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

#ROC curve for a specific class here for all classes
print(roc_auc)
```

```
{0: 0.9807692307692308, 1: 0.975, 2: 1.0, 'micro': 0.979166666666666
9}
```

In [11]:

```
from sklearn.datasets import load_iris
from sklearn import tree
from sklearn.metrics import classification_report, accuracy_score
from sklearn.model_selection import train_test_split
import graphviz

iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test
_size=0.4, random_state=17)

clf = tree.DecisionTreeClassifier(random_state=17)
clf = clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred, target_names=iris.target_names))
print('\nAccuracy: {0:.4f}'.format(accuracy_score(y_test, y_pred)))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        15
  versicolor       0.96      0.93      0.94        27
   virginica       0.89      0.94      0.92        18

   micro avg       0.95      0.95      0.95        60
   macro avg       0.95      0.96      0.95        60
weighted avg       0.95      0.95      0.95        60


Accuracy: 0.9500
```

In [12]:

```
dot = tree.export_graphviz(clf, out_file=None, feature_names=iris.feature_names,
 class_names=iris.target_names, filled=True, rounded=True, special_characters=Tr
ue)

graph = graphviz.Source(dot)
graph.format = 'png'
graph.render('iris', view=True)
```

Out[12]:

```
'iris.png'
```

2. DecisionTree ID3 - 2

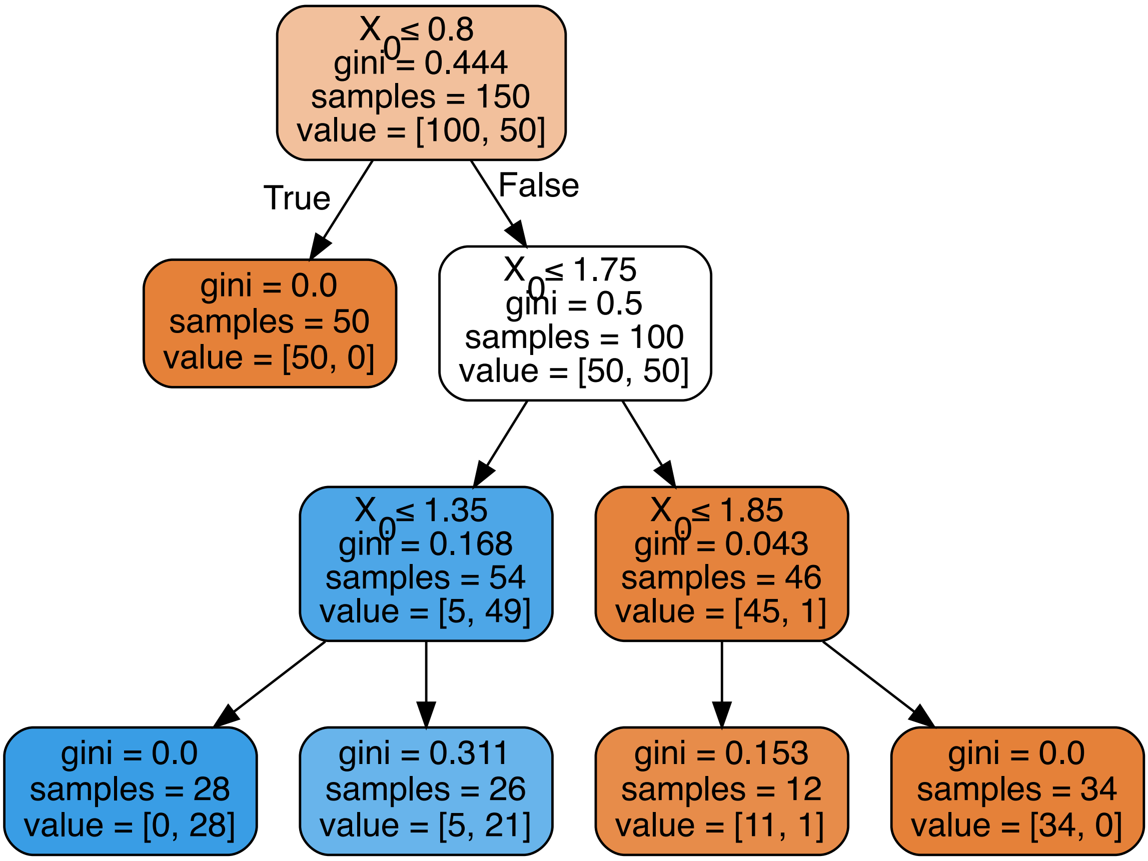
In [27]:

```
from sklearn.datasets import load_iris
from sklearn import tree
iris = load_iris()

target = iris.target

target[target == 2] = 0  # Convert to binary problem

clf_iris = tree.DecisionTreeClassifier(max_depth = 3)

clf_iris.fit(iris.data[:, 3].reshape(-1,1), target) # Classify using only one fe
ature

dot_data_iris = tree.export_graphviz(clf_iris, out_file=None,
                            filled=True, rounded=True,
                            special_characters=True)

graph_iris = graphviz.Source(dot_data_iris)
```

In [28]:

```
graph_iris
```

Out[28]:



3. kNN

In [112]:

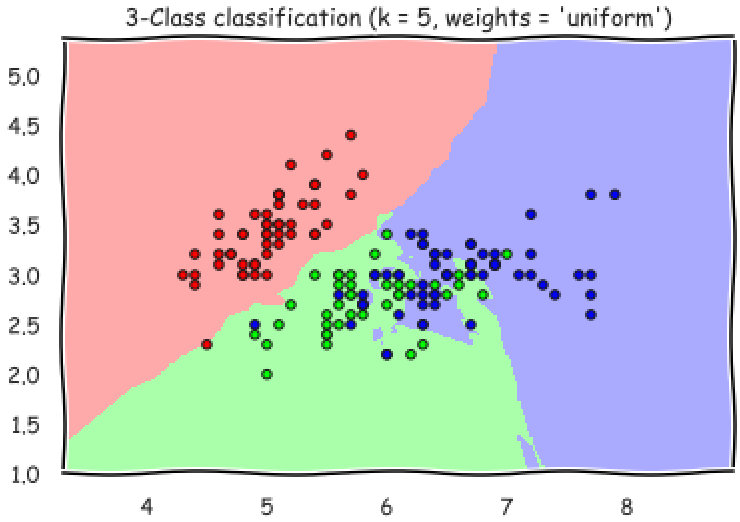
```
clf_KNN = KNeighborsClassifier()
clf_KNN.fit(X_train_full, y_train_full)
```

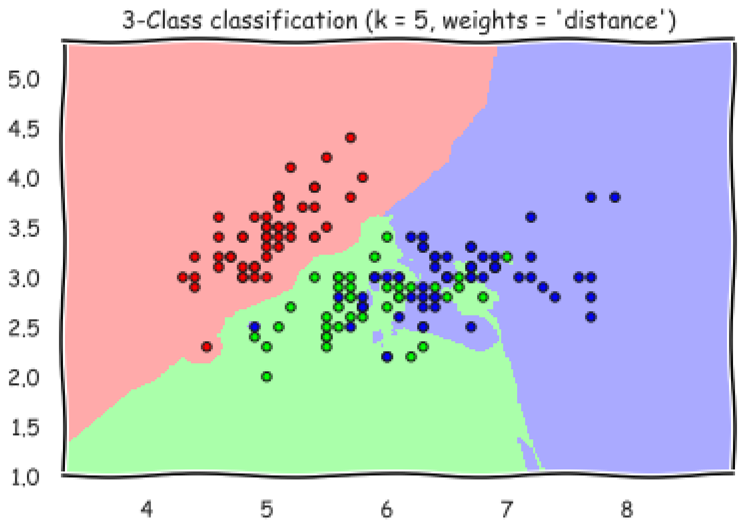
Out[112]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkows
ki',
           metric_params=None, n_jobs=None, n_neighbors=5, p=2,
           weights='uniform')
```

In [119]:

```
from matplotlib.colors import ListedColormap
import numpy as np
# Create color maps
cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
cmap_bold = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])

h = .02
n_neighbors = 5
X = iris.data[:, :2]

for weights in ['uniform', 'distance']:
    # we create an instance of Neighbours Classifier and fit the data.
    clf = KNN(n_neighbors, weights=weights)
    clf.fit(X, Y)

    # Plot the decision boundary. For that, we will assign a color to each
    # point in the mesh [x_min, x_max]x[y_min, y_max].
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

    # Put the result into a color plot
    Z = Z.reshape(xx.shape)
    plt.figure()
    plt.pcolormesh(xx, yy, Z, cmap=cmap_light)

    # Plot also the training points
    plt.scatter(X[:, 0], X[:, 1], c=Y, cmap=cmap_bold,
                edgecolor='k', s=20)
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.title("3-Class classification (k = %i, weights = '%s')"
              % (n_neighbors, weights))

plt.show()
```





4. Neural Network MLP

In [111]:

```
clf_MLP = MLPClassifier(hidden_layer_sizes=(10,))
clf_MLP.fit(X_train_full, y_train_full)
```

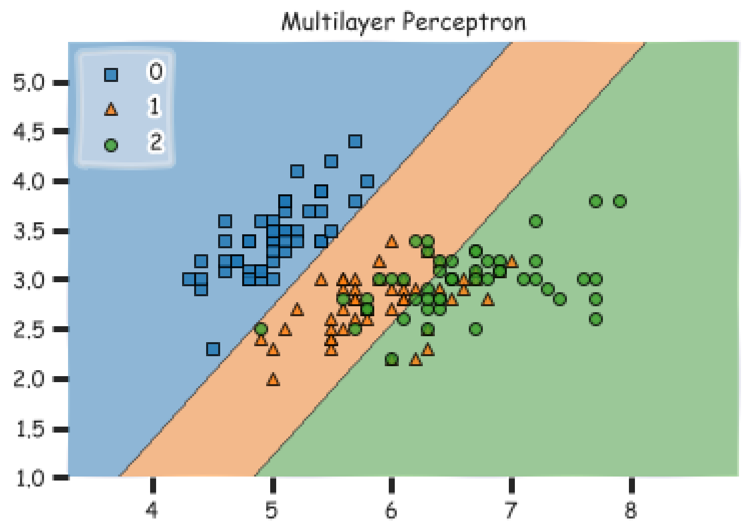
```
/Volumes/Data/Anaconda/anaconda3/lib/python3.7/site-packages/sklear
n/neural_network/multilayer_perceptron.py:562: ConvergenceWarning: S
tochastic Optimizer: Maximum iterations (200) reached and the optimi
zation hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

Out[111]:

```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', be
ta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(10,), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=None, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=False, warm_start=False)
```

In [62]:

```
mlp = MLP(solver='lbfgs',alpha=1e-5,hidden_layer_sizes=(5,2), random_state=1)
mlp.fit(X,Y)

from mlxtend.plotting import plot_decision_regions

fig = plot_decision_regions(X=X, y=Y, clf=mlp, legend=2)
plt.title('Multilayer Perceptron')
plt.show()
```



C. Melakukan pembelajaran 90:10

1. NaïveBayes
2. DecisionTree
3. kNN
4. MLP

untuk dataset iris dengan skema split train 90% dan test 10%, dan menampilkan kinerja serta confusion matrixnya.

In [68]:

```
import itertools
from sklearn import metrics
from sklearn.model_selection import train_test_split

def plot_confusion_matrix(cm, classes,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
#     print(cm)
#     plt.xkcd()
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="black" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

def kinerja(y_test, y_predict, title):
    print("Akurasi: {0:.4f}".format(metrics.accuracy_score(y_test, y_predict)))
    print()
    print("Classification Report")
    print(metrics.classification_report(y_test, y_predict, target_names=df_iris.
target_names))
    print()
    plt.figure()
    # print(metrics.confusion_matrix(y_test, y_predict, labels=[0,1,2]))
    plot_confusion_matrix(metrics.confusion_matrix(y_test, y_predict), classes=d
f_iris.target_names,
                          title= title + ' Confusion Matrix')
    plt.show()
```

In [65]:

```
X_train_90, X_test_90, y_train_90, y_test_90 = train_test_split(df_iris.data, df
_iris.target, test_size=0.1)
```

1. NaïveBayes

In [66]:

```
clf_GNB_90 = GaussianNB()
clf_GNB_90.fit(X_train_90, y_train_90)
```

Out[66]:

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

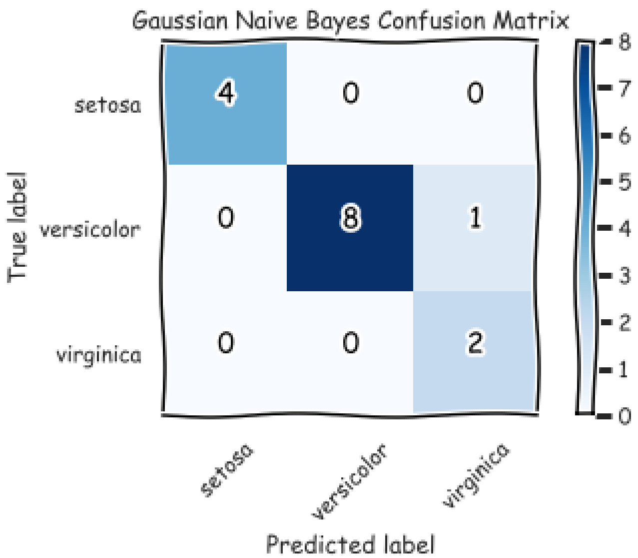
1. NaïveBayes

menampilkan kinerja serta confusion matrixnya

In [69]:

```
kinerja(y_test_90, clf_GNB_90.predict(X_test_90), 'Gaussian Naive Bayes')
```

```
Akurasi: 0.9333

Classification Report
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00         4
  versicolor       1.00      0.89      0.94         9
   virginica       0.67      1.00      0.80         2

   micro avg       0.93      0.93      0.93        15
   macro avg       0.89      0.96      0.91        15
weighted avg       0.96      0.93      0.94        15
```



2. DecisionTree

In [70]:

```
clf_DT_90 = tree.DecisionTreeClassifier(criterion='entropy')
clf_DT_90.fit(X_train_90, y_train_90)
```

Out[70]:

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_d
epth=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=False, random_stat
e=None,
            splitter='best')
```

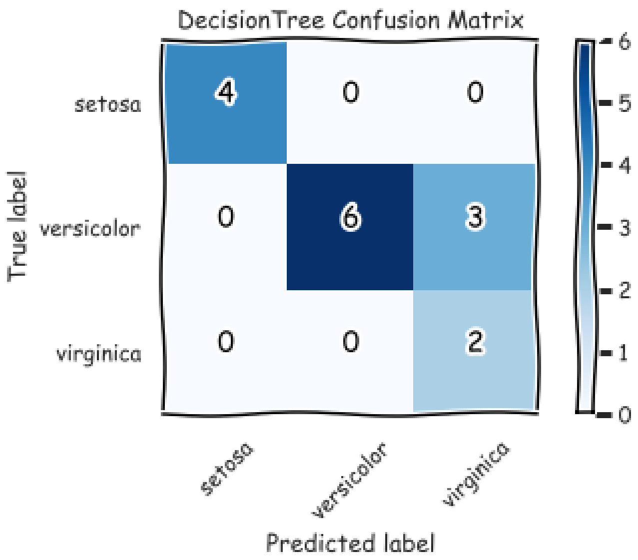
1. DecisionTree

menampilkan kinerja serta confusion matrixnya

In [71]:

```
kinerja(y_test_90, clf_DT_90.predict(X_test_90), 'DecisionTree')
```

```
Akurasi: 0.8000

Classification Report
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00         4
  versicolor       1.00      0.67      0.80         9
   virginica       0.40      1.00      0.57         2

   micro avg       0.80      0.80      0.80        15
   macro avg       0.80      0.89      0.79        15
weighted avg       0.92      0.80      0.82        15
```



3. kNN

In [73]:

```
clf_KNN_90 = KNeighborsClassifier(n_neighbors=1)
clf_KNN_90.fit(X_train_90, y_train_90)
```

Out[73]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkows
ki',
           metric_params=None, n_jobs=None, n_neighbors=1, p=2,
           weights='uniform')
```

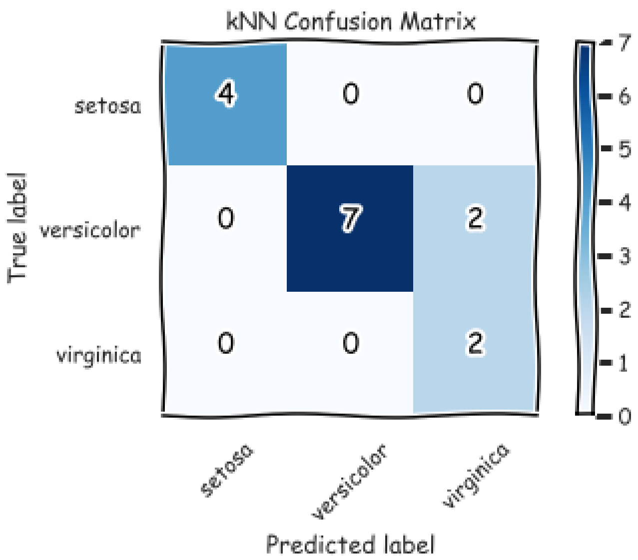
1. kNN

menampilkan kinerja serta confusion matrixnya

In [74]:

```
kinerja(y_test_90, clf_KNN_90.predict(X_test_90), 'kNN')
```

```
Akurasi: 0.8667

Classification Report
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00         4
  versicolor       1.00      0.78      0.88         9
   virginica       0.50      1.00      0.67         2

   micro avg       0.87      0.87      0.87        15
   macro avg       0.83      0.93      0.85        15
weighted avg       0.93      0.87      0.88        15
```



4. MLP

In [93]:

```
clf_MLP_90 = MLPClassifier()
clf_MLP_90.fit(X_train_90, y_train_90)
```

```
/Volumes/Data/Anaconda/anaconda3/lib/python3.7/site-packages/sklear
n/neural_network/multilayer_perceptron.py:562: ConvergenceWarning: S
tochastic Optimizer: Maximum iterations (200) reached and the optimi
zation hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

Out[93]:

```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', be
ta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(100,), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=None, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=False, warm_start=False)
```

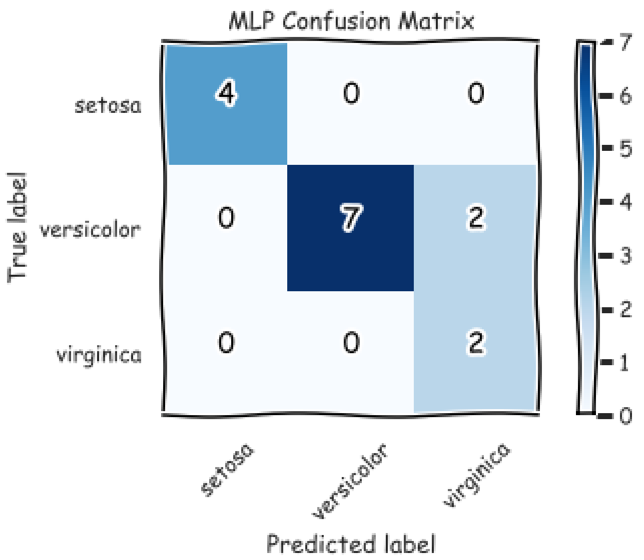
1. MLP

menampilkan kinerja serta confusion matrixnya

In [96]:

```
kinerja(y_test_90, clf_MLP_90.predict(X_test_90), 'MLP')
```

```
Akurasi: 0.8667

Classification Report
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00         4
  versicolor       1.00      0.78      0.88         9
   virginica       0.50      1.00      0.67         2

   micro avg       0.87      0.87      0.87        15
   macro avg       0.83      0.93      0.85        15
weighted avg       0.93      0.87      0.88        15
```



D. Melakukan pembelajaran 10-Fold

1. NaïveBayes
2. DecisionTree
3. kNN
4. MLP

untuk dataset iris dengan skema 10-fold cross validation, dan menampilkan kinerjanya.

In [97]:

```
# Classifier
clf_GNB_kf = GaussianNB()
clf_DT_kf = tree.DecisionTreeClassifier(criterion='entropy')
clf_KNN_kf = KNeighborsClassifier()
clf_MLP_kf = MLPClassifier()
```

In [98]:

```
from sklearn.model_selection import KFold

kf = KFold(10, shuffle=True)
kf
```

Out[98]:

```
KFold(n_splits=10, random_state=None, shuffle=True)
```

In [99]:

```
# Kontainer Metrik Kinerja
clf_GNB_kf_acc = []
clf_GNB_kf_prec = []
clf_GNB_kf_rec = []
clf_GNB_kf_f1 = []

clf_DT_kf_acc = []
clf_DT_kf_prec = []
clf_DT_kf_rec = []
clf_DT_kf_f1 = []

clf_KNN_kf_acc = []
clf_KNN_kf_prec = []
clf_KNN_kf_rec = []
clf_KNN_kf_f1 = []

clf_MLP_kf_acc = []
clf_MLP_kf_prec = []
clf_MLP_kf_rec = []
clf_MLP_kf_f1 = []

for train_index, test_index in kf.split(X_train_full):
#     print("TRAIN:", train_index, '\n', "TEST:", test_index, '\n\n')
    X_train, X_test = X_train_full[train_index], X_train_full[test_index]
    y_train, y_test = y_train_full[train_index], y_train_full[test_index]

    clf_GNB_kf.fit(X_train, y_train)
    y_test_predict_GNB = clf_GNB_kf.predict(X_test)
    clf_GNB_kf_acc.append(metrics.accuracy_score(y_test, y_test_predict_GNB))
    clf_GNB_kf_prec.append(metrics.precision_score(y_test, y_test_predict_GNB, a
verage='macro'))
    clf_GNB_kf_rec.append(metrics.recall_score(y_test, y_test_predict_GNB, avera
ge='macro'))
    clf_GNB_kf_f1.append(metrics.f1_score(y_test, y_test_predict_GNB, average='m
acro'))

    clf_DT_kf.fit(X_train, y_train)
    y_test_predict_DT = clf_DT_kf.predict(X_test)
    clf_DT_kf_acc.append(metrics.accuracy_score(y_test, y_test_predict_DT))
    clf_DT_kf_prec.append(metrics.precision_score(y_test, y_test_predict_DT, ave
rage='macro'))
    clf_DT_kf_rec.append(metrics.recall_score(y_test, y_test_predict_DT, average
='macro'))
    clf_DT_kf_f1.append(metrics.f1_score(y_test, y_test_predict_DT, average='mac
ro'))

    clf_KNN_kf.fit(X_train, y_train)
    y_test_predict_KNN = clf_KNN_kf.predict(X_test)
    clf_KNN_kf_acc.append(metrics.accuracy_score(y_test, y_test_predict_KNN))
    clf_KNN_kf_prec.append(metrics.precision_score(y_test, y_test_predict_KNN, a
verage='macro'))
    clf_KNN_kf_rec.append(metrics.recall_score(y_test, y_test_predict_KNN, avera
ge='macro'))
    clf_KNN_kf_f1.append(metrics.f1_score(y_test, y_test_predict_KNN, average='m
acro'))

    clf_MLP_kf.fit(X_train, y_train)
    y_test_predict_MLP = clf_MLP_kf.predict(X_test)
    clf_MLP_kf_acc.append(metrics.accuracy_score(y_test, y_test_predict_MLP))
```

```
    clf_MLP_kf_prec.append(metrics.precision_score(y_test, y_test_predict_MLP, a
verage='macro'))
    clf_MLP_kf_rec.append(metrics.recall_score(y_test, y_test_predict_MLP, avera
ge='macro'))
    clf_MLP_kf_f1.append(metrics.f1_score(y_test, y_test_predict_MLP, average='m
acro'))
```

```
/Volumes/Data/Anaconda/anaconda3/lib/python3.7/site-packages/sklear
n/neural_network/multilayer_perceptron.py:562: ConvergenceWarning: S
tochastic Optimizer: Maximum iterations (200) reached and the optimi
zation hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Volumes/Data/Anaconda/anaconda3/lib/python3.7/site-packages/sklear
n/neural_network/multilayer_perceptron.py:562: ConvergenceWarning: S
tochastic Optimizer: Maximum iterations (200) reached and the optimi
zation hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Volumes/Data/Anaconda/anaconda3/lib/python3.7/site-packages/sklear
n/neural_network/multilayer_perceptron.py:562: ConvergenceWarning: S
tochastic Optimizer: Maximum iterations (200) reached and the optimi
zation hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Volumes/Data/Anaconda/anaconda3/lib/python3.7/site-packages/sklear
n/neural_network/multilayer_perceptron.py:562: ConvergenceWarning: S
tochastic Optimizer: Maximum iterations (200) reached and the optimi
zation hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Volumes/Data/Anaconda/anaconda3/lib/python3.7/site-packages/sklear
n/neural_network/multilayer_perceptron.py:562: ConvergenceWarning: S
tochastic Optimizer: Maximum iterations (200) reached and the optimi
zation hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Volumes/Data/Anaconda/anaconda3/lib/python3.7/site-packages/sklear
n/neural_network/multilayer_perceptron.py:562: ConvergenceWarning: S
tochastic Optimizer: Maximum iterations (200) reached and the optimi
zation hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Volumes/Data/Anaconda/anaconda3/lib/python3.7/site-packages/sklear
n/neural_network/multilayer_perceptron.py:562: ConvergenceWarning: S
tochastic Optimizer: Maximum iterations (200) reached and the optimi
zation hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Volumes/Data/Anaconda/anaconda3/lib/python3.7/site-packages/sklear
n/neural_network/multilayer_perceptron.py:562: ConvergenceWarning: S
tochastic Optimizer: Maximum iterations (200) reached and the optimi
zation hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Volumes/Data/Anaconda/anaconda3/lib/python3.7/site-packages/sklear
n/neural_network/multilayer_perceptron.py:562: ConvergenceWarning: S
tochastic Optimizer: Maximum iterations (200) reached and the optimi
zation hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Volumes/Data/Anaconda/anaconda3/lib/python3.7/site-packages/sklear
n/neural_network/multilayer_perceptron.py:562: ConvergenceWarning: S
tochastic Optimizer: Maximum iterations (200) reached and the optimi
zation hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

Menampilkan Kinerjanya

In [100]:

```
print('GNB')
print('Rerata akurasi:', np.mean(clf_GNB_kf_acc))
print('Rerata presisi:', np.mean(clf_GNB_kf_prec))
print('Rerata recall:', np.mean(clf_GNB_kf_rec))
print('Rerata f1-score:', np.mean(clf_GNB_kf_f1))

print()
print('DT')
print('Rerata akurasi:', np.mean(clf_DT_kf_acc))
print('Rerata presisi:', np.mean(clf_DT_kf_prec))
print('Rerata recall:', np.mean(clf_DT_kf_rec))
print('Rerata f1-score:', np.mean(clf_DT_kf_f1))

print()
print('KNN')
print('Rerata akurasi:', np.mean(clf_KNN_kf_acc))
print('Rerata presisi:', np.mean(clf_KNN_kf_prec))
print('Rerata recall:', np.mean(clf_KNN_kf_rec))
print('Rerata f1-score:', np.mean(clf_KNN_kf_f1))

print()
print('MLP')
print('Rerata akurasi:', np.mean(clf_MLP_kf_acc))
print('Rerata presisi:', np.mean(clf_MLP_kf_prec))
print('Rerata recall:', np.mean(clf_MLP_kf_rec))
print('Rerata f1-score:', np.mean(clf_MLP_kf_f1))
```

```
GNB
Rerata akurasi: 0.9533333333333334
Rerata presisi: 0.9613888888888888
Rerata recall: 0.9597222222222224
Rerata f1-score: 0.9567291967291969

DT
Rerata akurasi: 0.9266666666666667
Rerata presisi: 0.9307142857142857
Rerata recall: 0.9211111111111111
Rerata f1-score: 0.9224928774928776

KNN
Rerata akurasi: 0.9666666666666668
Rerata presisi: 0.9691269841269842
Rerata recall: 0.9627777777777778
Rerata f1-score: 0.9634676434676436

MLP
Rerata akurasi: 0.9733333333333334
Rerata presisi: 0.9754761904761905
Rerata recall: 0.9713888888888889
Rerata f1-score: 0.9701098901098902
```

E. Menyimpan (save) model/hipotesis hasil pembelajaran ke sebuah file eksternal

Contoh: model yang akan disimpan adalah DecisionTree dengan skema full training

In [101]:

```
from sklearn.externals import joblib
```

In [102]:

```
joblib.dump(clf_DT, 'DT.model')
```

Out[102]:

```
['DT.model']
```

F. Membaca (read)model/hipotesis dari file eksternal

Contoh: model yang akan dibaca adalah DT.model

In [103]:

```
loaded_model = joblib.load('DT.model')
```

G. Membuat instance baru dengan memberi nilai untuk setiap atribut

In [106]:

```
instance_baru = [[2.6, 2.3, 4.1, 4.7],
                 [2.2, 4.4, 2.1, 4.1],
                 [5.2, 3.5, 4.6, 6.3],
                 [5.5, 1.6, 4.7, 4.3],
                 [3.1, 4.1, 1.3, 4.7]]
```

H. Melakukan klasifikasi dengan memanfaatkan model/hipotesisNaïveBayes, DecisionTree, dan kNN dan instance pada g

Contoh: Klasifikasi model yang dibentuk dengan skema Full Training

In [113]:

```
def klas(i): return df_iris.target_names[i]

hasil_GNB = clf_GNB.predict(instance_baru)
hasil_DT = clf_DT.predict(instance_baru)
hasil_KNN = clf_KNN.predict(instance_baru)
hasil_MLP = clf_MLP.predict(instance_baru)

for i in range(len(instance_baru)):
    print(instance_baru[i])
    print('GNB           :', klas(hasil_GNB[i]))
    print('Decision Tree :', klas(hasil_DT[i]))
    print('KNN           :', klas(hasil_KNN[i]))
    print('MLP           :', klas(hasil_MLP[i]))
    print()
```

```
[2.6, 2.3, 4.1, 4.7]
GNB           : virginica
Decision Tree : versicolor
KNN           : virginica
MLP           : virginica

[2.2, 4.4, 2.1, 4.1]
GNB           : virginica
Decision Tree : versicolor
KNN           : setosa
MLP           : versicolor

[5.2, 3.5, 4.6, 6.3]
GNB           : virginica
Decision Tree : versicolor
KNN           : virginica
MLP           : virginica

[5.5, 1.6, 4.7, 4.3]
GNB           : virginica
Decision Tree : versicolor
KNN           : virginica
MLP           : virginica

[3.1, 4.1, 1.3, 4.7]
GNB           : virginica
Decision Tree : versicolor
KNN           : setosa
MLP           : virginica
```

Catatan :

1. /Volumes/Data/Anaconda/anaconda3/lib/python3.7/site-packages/sklearn/neural_network/multilayer_perceptron.py:562: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.% self.max_iter, ConvergenceWarning) -> Bukan Error
2. In (X) -> Restart agar berurutan kembali X nya