Evaluasi Algoritma - Tugas Pertemuan 2 Praktikum ML

Marcella Komunita Pasaribu - 24060120120026
Pembelajaran Mesin C

Notebook: https://colab.research.google.com/drive/1qCcCkMt5ddBY3_eBnoy8tgc1dO9Y_YD3?usp=sharing (https://colab.research.google.com/drive/1qCcCkMt5ddBY3_eBnoy8tgc1dO9Y_YD3?usp=sharing)
Dataset: https://archive.ics.uci.edu/ml/datasets/User+Knowledge+Modeling (https://archive.ics.uci.edu/ml/datasets/User+Knowledge+Modeling)

Import Library

In [ ]:
```
# data preparation
import pandas as pd

# model preparation
from sklearn import model_selection

# build model
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB

from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier tree.DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression # Import Logistic Regression linear_model.LogisticRegress
ion
from sklearn.linear_model import Perceptron # Import Perceptron linear_model.Perceptron
from sklearn.ensemble import GradientBoostingClassifier # Import Gradient Boosting Classifier ensemble.Gradie
ntBoostingClassifier

# test model
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

Pertama, akan dilakukan impor terhadap library yang dibutuhkan dalam penugasan ini, yaitu pandas dan sklearn (model_selection, model, dan metrics). Pandas adalah library yang umum digunakan untuk memanipulasi data, dimana salah satu tipe datanya adalah dataframe. Dataframe adalah tipe data menyerupai tabel atau yang setiap kolom dan barisnya berisikan value. Terdapat tujuh model yang digunakan dalam penugasan ini, yaitu sebagai berikut.

> Model pada modul (tugas 1 dan 2)
>
> - SVC
> - KNeighborClassifier
> - GaussianNB
>
> Model eksplorasi (tugas 1)
>
> - DecisionTreeClassifier
> - LogisticRegression
> - Perceptron
> - GradientBoostingClassifier

Method sklearn.model_selection adalah method untuk mempersiapkan sebuah blueprint yang akan digunakan dalam analisis data dan menghasilkan prediksi terhadap data baru. Pemilihan model yang tepat berguna agar dapat menghasilkan hasil yang akurat dalam prediksi.

Library sklearn.metrics berisi method yang digunakan untuk menguji model yang dibuat berdasarkan label prediksi yang dihasilkan dengan label aktual. Terdapat tiga jenis metrik yang digunakan, yaitu akurasi, confussion matrix, dan classification report.

Tugas 1

Input Dataset

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

Kode di atas adalah kode untuk memuat dataset yang ada di internet. Kemudian, kolom-kolom dari dataset dispesifikasikan dalam variabel names. Dataset tersebut diinisalisasi ke dalam bentuk dataframe dengan memanggil method read_csv() dari library pandas.

In [ ]:
```
dataset
```

Out[ ]:

	sepal-length	sepal-width	petal-length	petal-width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

Untuk membuktikan bahwa dataset sudah berhasil dimuat ke dalam notebook, dataset dapat ditampilkan dengan memanggil nama variabel, yaitu dataset. Pada pojok kiri juga ditampilkan dimensi dari dataset, dapat dilihat bahwa dataset iris ini memiliki 150 baris/sampel data dan 5 kolom/fitur.

Membuat Validasi Dataset

Validasi dataset dilakukan untuk mengetahui tingkat kebaikan atau keakuratan dari suatu model. Validasi dataset dibuat dengan menggunakan metode statistik untuk memperkirakan keakuratan model dari hasil prediksi data testing terhadap data aktualnya.

In [ ]:
```
# split-out validation dataset
array = dataset.values
X = array[:, 0:4]
Y = array[:, 4]

# membagi ukuran dataset
validation_size = 0.2

# set random_state ke integer
seed = 7
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, Y, test_size=validation_si
ze, random_state=seed)
```

Pada kode di atas, dataset dibagi ke dalam dua variabel, yaitu X yang berisi variabel fitur (kolom pertama sampai keempat) dan Y yang berisi variabel target (kolom kelima). Selanjutnya, dipanggil method train_test_split() untuk membagi data ini menjadi data train dan test/validation. Pada pemanggilan method, diatur atribut test_size = 0.2, yang berarti ukuran dari data latih adalah 80%, sedangkan ukuran dari data uji adalah 20%. Selain itu, diatur random_state menjadi integer (7), agar method train_test_split ini mengembalikan hasil (pembagian data) yang sama untuk setiap eksekusi.

In [ ]:

```
X_train
# Y_train
```

```
Out[ ]: array([[6.2, 2.8, 4.8, 1.8],
               [5.7, 2.6, 3.5, 1.0],
               [4.6, 3.6, 1.0, 0.2],
               [6.9, 3.1, 5.4, 2.1],
               [6.4, 2.9, 4.3, 1.3],
               [4.8, 3.0, 1.4, 0.3],
               [5.5, 3.5, 1.3, 0.2],
               [5.4, 3.9, 1.7, 0.4],
               [5.1, 3.5, 1.4, 0.3],
               [7.1, 3.0, 5.9, 2.1],
               [6.7, 3.3, 5.7, 2.1],
               [6.8, 2.8, 4.8, 1.4],
               [6.4, 2.8, 5.6, 2.2],
               [6.5, 3.0, 5.5, 1.8],
               [5.7, 3.0, 4.2, 1.2],
               [5.0, 3.3, 1.4, 0.2],
               [6.7, 3.1, 4.4, 1.4],
               [6.0, 2.2, 4.0, 1.0],
               [6.4, 2.7, 5.3, 1.9],
               [4.7, 3.2, 1.6, 0.2],
               [4.6, 3.1, 1.5, 0.2],
               [5.1, 3.4, 1.5, 0.2],
               [7.7, 3.8, 6.7, 2.2],
               [4.3, 3.0, 1.1, 0.1],
               [6.3, 3.3, 6.0, 2.5],
               [5.5, 2.4, 3.7, 1.0],
               [5.0, 2.0, 3.5, 1.0],
               [6.5, 2.8, 4.6, 1.5],
               [5.0, 3.4, 1.6, 0.4],
               [4.4, 2.9, 1.4, 0.2],
               [5.0, 3.5, 1.6, 0.6],
               [6.7, 3.1, 4.7, 1.5],
               [7.3, 2.9, 6.3, 1.8],
               [5.5, 2.6, 4.4, 1.2],
               [5.2, 2.7, 3.9, 1.4],
               [5.7, 4.4, 1.5, 0.4],
               [7.2, 3.2, 6.0, 1.8],
               [5.4, 3.4, 1.7, 0.2],
               [5.8, 4.0, 1.2, 0.2],
               [6.1, 2.6, 5.6, 1.4],
               [5.7, 2.5, 5.0, 2.0],
               [4.8, 3.0, 1.4, 0.1],
               [6.5, 3.0, 5.8, 2.2],
```

```
[4.6, 3.2, 1.4, 0.2],
[6.6, 2.9, 4.6, 1.3],
[6.7, 3.0, 5.2, 2.3],
[6.1, 3.0, 4.6, 1.4],
[5.7, 3.8, 1.7, 0.3],
[7.0, 3.2, 4.7, 1.4],
[4.7, 3.2, 1.3, 0.2],
[6.5, 3.0, 5.2, 2.0],
[7.7, 2.6, 6.9, 2.3],
[4.9, 2.4, 3.3, 1.0],
[4.8, 3.1, 1.6, 0.2],
[5.5, 4.2, 1.4, 0.2],
[5.6, 3.0, 4.1, 1.3],
[6.4, 3.2, 5.3, 2.3],
[5.2, 3.5, 1.5, 0.2],
[7.9, 3.8, 6.4, 2.0],
[5.8, 2.8, 5.1, 2.4],
[5.7, 2.9, 4.2, 1.3],
[5.1, 3.7, 1.5, 0.4],
[5.1, 2.5, 3.0, 1.1],
[5.0, 3.4, 1.5, 0.2],
[7.7, 2.8, 6.7, 2.0],
[7.6, 3.0, 6.6, 2.1],
[5.0, 3.2, 1.2, 0.2],
[5.4, 3.7, 1.5, 0.2],
[6.7, 3.3, 5.7, 2.5],
[6.1, 2.8, 4.0, 1.3],
[6.3, 2.5, 5.0, 1.9],
[7.4, 2.8, 6.1, 1.9],
[5.0, 2.3, 3.3, 1.0],
[5.4, 3.4, 1.5, 0.4],
[5.4, 3.9, 1.3, 0.4],
[5.6, 2.8, 4.9, 2.0],
[4.9, 3.0, 1.4, 0.2],
[5.3, 3.7, 1.5, 0.2],
[6.0, 2.7, 5.1, 1.6],
[6.8, 3.2, 5.9, 2.3],
[6.2, 3.4, 5.4, 2.3],
[5.7, 2.8, 4.1, 1.3],
[6.3, 2.3, 4.4, 1.3],
[4.9, 3.1, 1.5, 0.1],
[6.9, 3.1, 5.1, 2.3],
[5.0, 3.6, 1.4, 0.2],
```

```
       [4.4, 3.0, 1.3, 0.2],
       [6.0, 2.9, 4.5, 1.5],
       [6.1, 2.8, 4.7, 1.2],
       [6.4, 2.8, 5.6, 2.1],
       [4.9, 3.1, 1.5, 0.1],
       [5.6, 2.9, 3.6, 1.3],
       [5.9, 3.0, 4.2, 1.5],
       [6.3, 2.7, 4.9, 1.8],
       [6.8, 3.0, 5.5, 2.1],
       [5.5, 2.3, 4.0, 1.3],
       [6.3, 2.8, 5.1, 1.5],
       [4.8, 3.4, 1.9, 0.2],
       [6.3, 3.3, 4.7, 1.6],
       [5.6, 2.5, 3.9, 1.1],
       [5.1, 3.8, 1.9, 0.4],
       [5.1, 3.8, 1.5, 0.3],
       [4.6, 3.4, 1.4, 0.3],
       [5.7, 2.8, 4.5, 1.3],
       [6.6, 3.0, 4.4, 1.4],
       [5.1, 3.5, 1.4, 0.2],
       [7.7, 3.0, 6.1, 2.3],
       [6.1, 3.0, 4.9, 1.8],
       [6.2, 2.2, 4.5, 1.5],
       [6.3, 3.4, 5.6, 2.4],
       [4.4, 3.2, 1.3, 0.2],
       [6.5, 3.2, 5.1, 2.0],
       [5.5, 2.5, 4.0, 1.3],
       [6.3, 2.5, 4.9, 1.5],
       [5.1, 3.3, 1.7, 0.5],
       [5.8, 2.7, 5.1, 1.9],
       [5.8, 2.6, 4.0, 1.2],
       [6.3, 2.9, 5.6, 1.8],
       [5.8, 2.7, 4.1, 1.0],
       [5.0, 3.0, 1.6, 0.2]], dtype=object)
```

Untuk menguji pembagian atau splitting data sudah berhasil, data dapat ditampilkan dengan memanggil nama variabel (X_validation, X_train, Y_validation, atau Y_train).

Pembuatan Model

In [ ]:
```
# Spot Check Algorithms
models = []

# Algoritma yang digunakan di modul
models.append(('KNN', KNeighborsClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))

# Eksplorasi Algoritma
models.append(('DecisionTreeClassifier', DecisionTreeClassifier()))
models.append(('LogisticRegression', LogisticRegression()))
models.append(('Perceptron', Perceptron()))
models.append(('GradientBoostingClassifier', GradientBoostingClassifier()))
```

Model-model yang akan digunakan dihimpun dalam satu array yang dinamakan models. Untuk setiap model akan ditambahkan ke dalam array models dalam bentuk tuple, yaitu pasangan data nama model dengan method terkait yang sudah diimport sebelumnya.

K-Folds Cross Validation

Pada kode di bawah, diinisialisasi dua array kosong yang akan digunakan untuk menyimpan nama dari model (array names) dan hasil evaluasi model (results). Kemudian akan dilakukan iterasi pengevaluasian untuk setiap model dalam array models yang sudah di-append pada cell sebelumnya. Evaluasi dilakukan dengan memanggil method Kfold dan cross_val_score pada libraray sklearn.model_selection. Validasi dilakukan dengan menggunakan 10 kali lipat cross validation, sehingga dataset dibagi menjadi 10 bagian, 9 untuk latihan dan 1 untuk pengujian.

In [ ]:

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

```
KNN: 0.991667 (0.025000)
NB: 0.966667 (0.055277)
SVM: 0.983333 (0.033333)
DecisionTreeClassifier: 0.950000 (0.066667)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs faile
d to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs faile
d to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs faile
d to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs faile
d to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs faile
d to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
```

```
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

```
LogisticRegression: 0.983333 (0.033333)
Perceptron: 0.750000 (0.129099)
GradientBoostingClassifier: 0.950000 (0.066667)
```

Validation Datasets' Prediction

Cells di bawah adalah kode untuk melakukan prediksi dan menampilkan metrik evaluasi dari ketujuh model. Pertama, dilakukan instansiasi dari model. Kemudian, model diimplementasi pada data latih dengan menggunakan method fit(). Setelah dilakukan training menggunakan data latih, dilakukan pengujian pada data uji dengan menggunakan method predict() dari model. Terakhir, dilakukan pengujian metrik terhadap hasil prediksi. Adapun penugasan ini menggunakan tiga metrik, yaitu akurasi, confusion matrix, dan classification report.

Confusion matrix adalah matriks yang berisi evaluasi akurasi dari model klasifikasi. Baris pada matriks merujuk pada label aktual, sedangkan kolom merujuk pada label prediksi.

Precision adalah kemampuan dari model untuk tidak melabeli sebuah instan positif yang seharus negatif.
Recall adalah kemampuan dari model untuk menemukan semua instan postif.
F1-score adalah rata-rata dari precision dan recall, dimana skor terbaik adalah 1.0 dan skor terburuk adalah 0.0.

Macro average adalah rata-rata dari suatu metrik (precision, recall, atau f1-score), sedangkan weighted average adalah rata-rata dari metrik per kategori kelas, dengan mempertimbangkan support atau distribusi dari kelas dalam dataset.

In [ ]:

```python
# Make predictions on validation dataset
# model k-neighbor classifier

knn = KNeighborsClassifier()
knn.fit(X_train, Y_train)
predictions = knn.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 11  1]
 [ 0  2  9]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       0.85      0.92      0.88        12
 Iris-virginica       0.90      0.82      0.86        11

       accuracy                           0.90        30
      macro avg       0.92      0.91      0.91        30
   weighted avg       0.90      0.90      0.90        30
```

Hasil evaluasi dari model KNeighborClassifier adalah sebagai berikut.

1. Akurasi dari model adalah 0.9
2. Terdapat tiga kesalahan prediksi, yaitu dua hasil yang memprediksikan instan sebagai Iris-versicolor (true: Iris-virginica) dan satu hasil yang memprediksikan instan sebagai Iris-virginica (true: Iris-versicolor)
3. Classification report

- **Precision**: Semua prediksi yang menghasilkan label Iris-setosa benar; semua prediksi yang menghasilkan label Iris-versicolor hanya 85% yang benar; semua prediksi yang menghasilkan label Iris-virginica hanya 90% yang benar
- **Recall**: Dari semua bunga yang diprediksi sebagai Iris-setosa, 100% hasil prediksi benar; dari semua bunga yang diprediksi sebagai Iris-versicolor, hanya 92% hasil prediksi yang benar; dari semua bunga yang diprediksi sebagai Iris-virginica, hanya 82% hasil prediksi yang benar
- **F1-score**: Karena nilai F1-Score mendekati 1, maka dapat dikatakan bahwa model bekerja dengan cukup baik untuk melakukan klasifikasi dari ketiga kelas
- **Support**: Model memprediksikan terdapat 7 Iris-setosa, 12 Iris-versicolor, dan 11 Iris-virginica

In [ ]:

```python
# Make predictions on validation dataset
# model gaussian

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))
```

```
0.8333333333333334
[[7 0 0]
 [0 9 3]
 [0 2 9]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       0.82      0.75      0.78        12
 Iris-virginica       0.75      0.82      0.78        11

       accuracy                           0.83        30
      macro avg       0.86      0.86      0.86        30
   weighted avg       0.84      0.83      0.83        30
```

Hasil evaluasi dari model GaussianNB adalah sebagai berikut.

1. Akurasi dari model adalah 0.83
2. Terdapat lima kesalahan prediksi, yaitu tiga hasil yang memprediksikan instan sebagai Iris-versicolor (true: Iris-virginica) dan dua hasil yang memprediksikan instan sebagai Iris-virginica (true: Iris-versicolor)
3. Classification report

- **Precision**: Semua prediksi yang menghasilkan label Iris-setosa benar; semua prediksi yang menghasilkan label Iris-versicolor hanya 82% yang benar; semua prediksi yang menghasilkan label Iris-virginica hanya 75% yang benar
- **Recall**: Dari semua bunga yang diprediksi sebagai Iris-setosa, 100% hasil prediksi benar; dari semua bunga yang diprediksi sebagai Iris-versicolor, hanya 75% hasil prediksi yang benar; dari semua bunga yang diprediksi sebagai Iris-virginica, hanya 82% hasil prediksi yang benar
- **F1-score**: Karena nilai F1-Score mendekati 1, maka dapat dikatakan bahwa model bekerja dengan cukup baik untuk melakukan klasifikasi dari ketiga kelas
- **Support**: Model memprediksikan terdapat 7 Iris-setosa, 12 Iris-versicolor, dan 11 Iris-virginica

In [ ]:

```python
# Make predictions on validation dataset
# model svc

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))
```

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

Hasil evaluasi dari model SVC adalah sebagai berikut.

1. Akurasi dari model adalah 0.867
2. Terdapat empat kesalahan prediksi, yaitu dua hasil yang memprediksikan instan sebagai Iris-versicolor (true: Iris-virginica) dan dua hasil yang memprediksikan instan sebagai Iris-virginica (true: Iris-versicolor)
3. Classification report

- **Precision**: Semua prediksi yang menghasilkan label Iris-setosa benar; semua prediksi yang menghasilkan label Iris-versicolor hanya 83% yang benar; semua prediksi yang menghasilkan label Iris-virginica hanya 82% yang benar
- **Recall**: Dari semua bunga yang diprediksi sebagai Iris-setosa, 100% hasil prediksi benar; dari semua bunga yang diprediksi sebagai Iris-versicolor, hanya 83% hasil prediksi yang benar; dari semua bunga yang diprediksi sebagai Iris-virginica, hanya 82% hasil prediksi yang benar
- **F1-score**: Karena nilai F1-Score mendekati 1, maka dapat dikatakan bahwa model bekerja dengan cukup baik untuk melakukan klasifikasi dari ketiga kelas
- **Support**: Model memprediksikan terdapat 7 Iris-setosa, 12 Iris-versicolor, dan 11 Iris-virginica

In [ ]:
```
# Make predictions on validation dataset
# model decision tree
dtc = DecisionTreeClassifier()
dtc.fit(X_train, Y_train)
predictions = dtc.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                           0.90        30
      macro avg       0.91      0.91      0.91        30
   weighted avg       0.90      0.90      0.90        30
```

Hasil evaluasi dari model DecisionTreeClassifier adalah sebagai berikut.

1. Akurasi dari model adalah 0.9
2. Terdapat tiga kesalahan prediksi, yaitu dua hasil yang memprediksikan instan sebagai Iris-versicolor (true: Iris-virginica) dan satu hasil yang memprediksikan instan sebagai Iris-virginica (true: Iris-versicolor)
3. Classification report

- **Precision**: Semua prediksi yang menghasilkan label Iris-setosa benar; semua prediksi yang menghasilkan label Iris-versicolor hanya 91% yang benar; semua prediksi yang menghasilkan label Iris-virginica hanya 83% yang benar
- **Recall**: Dari semua bunga yang diprediksi sebagai Iris-setosa, 100% hasil prediksi benar; dari semua bunga yang diprediksi sebagai Iris-versicolor, hanya 83% hasil prediksi yang benar; dari semua bunga yang diprediksi sebagai Iris-virginica, hanya 91% hasil prediksi yang benar
- **F1-score**: Karena nilai F1-Score mendekati 1, maka dapat dikatakan bahwa model bekerja dengan cukup baik untuk melakukan klasifikasi dari ketiga kelas
- **Support**: Model memprediksikan terdapat 7 Iris-setosa, 12 Iris-versicolor, dan 11 Iris-virginica

In [ ]:

```python
# Make predictions on validation dataset
# model perceptron

perceptron = Perceptron()
perceptron.fit(X_train, Y_train)
predictions = perceptron.predict(X_validation)
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
```

```
0.7666666666666667
[[ 6  1  0]
 [ 0  7  5]
 [ 0  1 10]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      0.86      0.92         7
Iris-versicolor       0.78      0.58      0.67        12
 Iris-virginica       0.67      0.91      0.77        11

       accuracy                           0.77        30
      macro avg       0.81      0.78      0.79        30
   weighted avg       0.79      0.77      0.76        30
```

Hasil evaluasi dari model Perceptron adalah sebagai berikut.

1. Akurasi dari model adalah 0.767
2. Terdapat tujuh kesalahan prediksi, yaitu satu hasil yang memprediksikan instan sebagai Iris-setosa (true: Iris-versicolor), lima hasil yang memprediksikan instan sebagai Iris-versicolor (true: Iris-virginica), dan satu hasil yang memprediksikan instan sebagai Iris-virginica (true: Iris-versicolor)
3. Classification report

- **Precision**: Semua prediksi yang menghasilkan label Iris-setosa benar; semua prediksi yang menghasilkan label Iris-versicolor hanya 78% yang benar; semua prediksi yang menghasilkan label Iris-virginica hanya 67% yang benar
- **Recall**: Dari semua bunga yang diprediksi sebagai Iris-setosa, 86% hasil prediksi benar; dari semua bunga yang diprediksi sebagai Iris-versicolor, hanya 58% hasil prediksi yang benar; dari semua bunga yang diprediksi sebagai Iris-virginica, hanya 91% hasil prediksi yang benar
- **F1-score**: Karena nilai F1-Score dari kelas kedua, kurang mendekati 1, model ini dapat dikatakan kurang baik dalam memprediksi kelas klasifikasi
- **Support**: Model memprediksikan terdapat 7 Iris-setosa, 12 Iris-versicolor, dan 11 Iris-virginica

In [ ]:

```python
# Make predictions on validation dataset
# model gradient boosting classifier

gbc = GradientBoostingClassifier()
gbc.fit(X_train, Y_train)
predictions = gbc.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  9  3]
 [ 0  1 10]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       0.90      0.75      0.82        12
 Iris-virginica       0.77      0.91      0.83        11

       accuracy                           0.87        30
      macro avg       0.89      0.89      0.88        30
   weighted avg       0.88      0.87      0.87        30
```

Hasil evaluasi dari model GradientBoostingClassifier adalah sebagai berikut.

1. Akurasi dari model adalah 0.9
2. Terdapat tiga kesalahan prediksi, yaitu dua hasil yang memprediksikan instan sebagai Iris-versicolor (true: Iris-virginica) dan satu hasil yang memprediksikan instan sebagai Iris-virginica (true: Iris-versicolor)
3. Classification report

- **Precision**: Semua prediksi yang menghasilkan label Iris-setosa benar; semua prediksi yang menghasilkan label Iris-versicolor hanya 91% yang benar; semua prediksi yang menghasilkan label Iris-virginica hanya 83% yang benar
- **Recall**: Dari semua bunga yang diprediksi sebagai Iris-setosa, 100% hasil prediksi benar; dari semua bunga yang diprediksi sebagai Iris-versicolor, hanya 83% hasil prediksi yang benar; dari semua bunga yang diprediksi sebagai Iris-virginica, hanya 91% hasil prediksi yang benar
- **F1-score**: Karena nilai F1-Score mendekati 1, maka dapat dikatakan bahwa model bekerja dengan cukup baik untuk melakukan klasifikasi dari ketiga kelas
- **Support**: Model memprediksikan terdapat 7 Iris-setosa, 12 Iris-versicolor, dan 11 Iris-virginica

Kesimpulan

Dari ketujuh model, terdapat tiga model yang paling baik dalam melakukan klasifikasi, yaitu KNeighborClassifier, DecisionTreeClassifier, dan GradientBoostingClassifier dengan akurasi 0.9 dan macro avg F1-score sebesar 0.91.

Tugas 2

Input Dataset

In [ ]:
```
!pip install --upgrade xlrd
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/00257/Data_User_Modeling_Dataset_Hamdi%20Tol
ga%20KAHRAMAN.xls"
names = ['STG', 'SCG', 'STR', 'LPR', 'PEG', 'UNS']
dataset = pd.read_excel(url, sheet_name="Training_Data", names=names, usecols="A:F")
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: xlrd in /usr/local/lib/python3.7/dist-packages (2.0.1)
```

Kode di atas adalah kode untuk memuat dataset yang ada di internet. Kemudian, kolom-kolom dari dataset dispesifikasikan dalam variabel names. Dataset tersebut diinisalisasi ke dalam bentuk dataframe dengan memanggil method read_excel() dari library pandas.

Terdapat enam kolom dalam data. STG adalah derajat waktu belajar untuk objek tujuan, SCG adalah derajat dari jumlah repetisi dari pengguna untuk objek tujuan, STR adalah derajat waktu belajar dari pengguna untuk objek yang berkaitan dengan objek tujuan, LTR adalah performa test dari pengguna dalam objek yang berkaitan dengan objek tujuan, dan PEG adalah performa test dari test objek tujuan. UNS adalah level knowledge dari pengguna, yang terdiri atas empat kelas, yaitu very_low, low, middle, dan high.

In [ ]:
```
dataset
```

Out[ ]:

	STG	SCG	STR	LPR	PEG	UNS
0	0.00	0.00	0.00	0.00	0.00	very_low
1	0.08	0.08	0.10	0.24	0.90	High
2	0.06	0.06	0.05	0.25	0.33	Low
3	0.10	0.10	0.15	0.65	0.30	Middle
4	0.08	0.08	0.08	0.98	0.24	Low
...	...	...	...	...	...	...
253	0.61	0.78	0.69	0.92	0.58	High
254	0.78	0.61	0.71	0.19	0.60	Middle
255	0.54	0.82	0.71	0.29	0.77	High
256	0.50	0.75	0.81	0.61	0.26	Middle
257	0.66	0.90	0.76	0.87	0.74	High

258 rows × 6 columns

Untuk membuktikan bahwa dataset sudah berhasil dimuat ke dalam notebook, dataset dapat ditampilkan dengan memanggil nama variabel, yaitu dataset. Pada pojok kiri juga ditampilkan dimensi dari dataset, dapat dilihat bahwa dataset iris ini memiliki 403 baris/sampel data dan 6 kolom/fitur.

Membuat Validasi Dataset

Validasi dataset dilakukan untuk mengetahui tingkat kebaikan atau keakuratan dari suatu model. Validasi dataset dibuat dengan menggunakan metode statistik untuk memperkirakan keakuratan model dari hasil prediksi data testing terhadap data aktualnya.

In [ ]:
```
# split-out validation dataset
array = dataset.values
X = array[:, 0:5]
Y = array[:, 5]
validation_size = 0.2
seed = 7
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, Y, test_size=validation_si
ze, random_state=seed)
```

Pada kode di atas, dataset dibagi ke dalam dua variabel, yaitu X yang berisi variabel fitur (kolom pertama sampai keempat) dan Y yang berisi variabel target (kolom kelima). Selanjutnya, dipanggil method train_test_split() untuk membagi data ini menjadi data train dan test/validation. Pada pemanggilan method, diatur atribut test_size = 0.2, yang berarti ukuran dari data latih adalah 80%, sedangkan ukuran dari data uji adalah 20%. Selain itu, diatur random_state menjadi integer (7), agar method train_test_split ini mengembalikan hasil (pembagian data) yang sama untuk setiap eksekusi.

In [ ]:
```
X_train
# Y_train
```

Out[ ]:
```
array([[0.51, 0.4, 0.4, 0.59, 0.23],
       [0.66, 0.36, 0.56, 0.4, 0.83],
       [0.44, 0.55, 0.11, 0.26, 0.83],
       ...,
       [0.09, 0.3, 0.68, 0.18, 0.85],
       [0.68, 0.19, 0.19, 0.48, 0.1],
       [0.38, 0.39, 0.79, 0.33, 0.3]], dtype=object)
```

Untuk menguji pembagian atau splitting data sudah berhasil, data dapat ditampilkan dengan memanggil nama variabel (X_validation, X_train, Y_validation, atau Y_train).

Pembuatan Model

In [ ]:
```
# Spot Check Algorithms
models = []

# Algoritma yang digunakan di modul
models.append(('KNN', KNeighborsClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))

for name, model in models:
  msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
  print(msg)
```

```
KNN: 0.950000 (0.066667)
NB: 0.950000 (0.066667)
SVM: 0.950000 (0.066667)
```

Model-model yang akan digunakan dihimpun dalam satu array yang dinamakan models. Untuk setiap model akan ditambahkan ke dalam array models dalam bentuk tuple, yaitu pasangan data nama model dengan method terkait yang sudah diimport sebelumnya.

K-Folds Cross Validation

Pada kode di bawah, diinisialisasi dua array kosong yang akan digunakan untuk menyimpan nama dari model (array names) dan hasil evaluasi model (results). Kemudian akan dilakukan iterasi pengevaluasian untuk setiap model dalam array models yang sudah di-append pada cell sebelumnya. Evaluasi dilakukan dengan memanggil method Kfold dan cross_val_score pada libraray sklearn.model_selection. Validasi dilakukan dengan menggunakan 10 kali lipat cross validation, sehingga dataset dibagi menjadi 10 bagian, 9 untuk latihan dan 1 untuk pengujian.

In [ ]:
```
# evaluate each model in turn
results = []
names = []
for name, model in models:
  kfold = model_selection.KFold(n_splits=10, random_state=7, shuffle=True)
  cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold, scoring='accuracy')
  results.append(cv_results)
  names.append(name)
  msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
  print(msg)
```

```
KNN: 0.819524 (0.070581)
NB: 0.883095 (0.044942)
SVM: 0.892619 (0.052871)
```

Validation Datasets' Prediction

Cells di bawah adalah kode untuk melakukan prediksi dan menampilkan metrik evaluasi dari ketujuh model. Pertama, dilakukan instansiasi dari model. Kemudian, model diimplementasi pada data latih dengan menggunakan method fit(). Setelah dilakukan training menggunakan data latih, dilakukan pengujian pada data uji dengan menggunakan method predict() dari model. Terakhir, dilakukan pengujian metrik terhadap hasil prediksi. Adapun penugasan ini menggunakan tiga metrik, yaitu akurasi, confusion matrix, dan classification report.

Confusion matrix adalah matriks yang berisi evaluasi akurasi dari model klasifikasi. Baris pada matriks merujuk pada label aktual, sedangkan kolom merujuk pada label prediksi.

Precision adalah kemampuan dari model untuk tidak melabeli sebuah instan positif yang seharus negatif.
Recall adalah kemampuan dari model untuk menemukan semua instan postif.
F1-score adalah rata-rata dari precision dan recall, dimana skor terbaik adalah 1.0 dan skor terburuk adalah 0.0.

Macro average adalah rata-rata dari suatu metrik (precision, recall, atau f1-score), sedangkan weighted average adalah rata-rata dari metrik per kategori kelas, dengan mempertimbangkan support atau distribusi dari kelas dalam dataset.

In [ ]:
```python
# Make predictions on validation dataset
kn = KNeighborsClassifier()
kn.fit(X_train, Y_train)
predictions = kn.predict(X_validation)
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
```

```
0.8269230769230769
[[ 9  0  0  0]
 [ 0 18  0  0]
 [ 0  5 14  0]
 [ 0  4  0  2]]
              precision    recall  f1-score   support

        High       1.00      1.00      1.00         9
         Low       0.67      1.00      0.80        18
      Middle       1.00      0.74      0.85        19
    very_low       1.00      0.33      0.50         6

    accuracy                           0.83        52
   macro avg       0.92      0.77      0.79        52
weighted avg       0.88      0.83      0.82        52
```

Hasil evaluasi dari model KNeighborsClassifier adalah sebagai berikut.

1. Akurasi dari model adalah 0.827
2. Terdapat sembilan kesalahan prediksi, yaitu lima hasil yang memprediksikan instan sebagai Middle (true: Low), empat hasil yang memprediksikan instan sebagai very_low (true: Low)
3. Classification report

- **Precision**: Semua prediksi yang menghasilkan label high, middle, dan very_low 100% benar; semua prediksi yang menghasilkan label low hanya 67% yang benar
- **Recall**: Dari semua user yang diprediksi sebagai High atau Low, 100% hasil prediksi benar; dari semua user yang diprediksi sebagai Middle, hanya 74% hasil prediksi yang benar; dari semua user yang diprediksi sebagai very_low, hanya 33% hasil prediksi yang benar
- **F1-score**: Karena nilai F1-Score dari kelas very_low tidak mendekati 1, dapat dikatakan model ini kurang baik dalam melakukan klasifikasi
- **Support**: Model memprediksikan terdapat 9 kelas High, 18 kelas Low, 19 kelas Middle, dan 6 kelas very_low

In [ ]:

```python
# Make predictions on validation dataset
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))
```

```
0.7884615384615384
[[ 8  0  1  0]
 [ 0 16  0  2]
 [ 0  5 14  0]
 [ 0  3  0  3]]
              precision    recall  f1-score   support

        High       1.00      0.89      0.94         9
         Low       0.67      0.89      0.76        18
      Middle       0.93      0.74      0.82        19
    very_low       0.60      0.50      0.55         6

    accuracy                           0.79        52
   macro avg       0.80      0.75      0.77        52
weighted avg       0.81      0.79      0.79        52
```

Hasil evaluasi dari model GaussianNB adalah sebagai berikut.

1. Akurasi dari model adalah 0.788
2. Terdapat sembilan kesalahan prediksi, yaitu lima hasil yang memprediksikan instan sebagai Middle (true: Low), satu hasil yang memprediksikan instan sebagai very_low (true: Low), dan satu hasil yang memprediksikan instan sebagai High (true: Middle)
3. Classification report

- **Precision**: Semua prediksi yang menghasilkan label high 100% benar; semua prediksi yang menghasilkan label low hanya 67% yang benar; semua prediksi yang menghasilkan label middle hanya 93% yang benar; semua prediksi yang menghasilkan label very_low hanya 60% yang benar
- **Recall**: Dari semua user yang diprediksi sebagai High atau Low, 89% hasil prediksi benar; dari semua user yang diprediksi sebagai Middle, hanya 74% hasil prediksi yang benar; dari semua user yang diprediksi sebagai very_low, hanya 50% hasil prediksi yang benar
- **F1-score**: Karena nilai F1-Score dari kelas very_low tidak mendekati 1, dapat dikatakan model ini kurang baik dalam melakukan klasifikasi
- **Support**: Model memprediksikan terdapat 9 kelas High, 18 kelas Low, 19 kelas Middle, dan 6 kelas very_low

In [ ]:
```python
# Make predictions on validation dataset
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))
```

```
0.8846153846153846
[[ 9  0  0  0]
 [ 0 18  0  0]
 [ 0  2 17  0]
 [ 0  4  0  2]]
              precision    recall  f1-score   support

        High       1.00      1.00      1.00         9
         Low       0.75      1.00      0.86        18
      Middle       1.00      0.89      0.94        19
    very_low       1.00      0.33      0.50         6

    accuracy                           0.88        52
   macro avg       0.94      0.81      0.83        52
weighted avg       0.91      0.88      0.87        52
```

Hasil evaluasi dari model SVC adalah sebagai berikut.

1. Akurasi dari model adalah 0.88
2. Terdapat enam kesalahan prediksi, yaitu dua hasil yang memprediksikan instan sebagai very_low (true: Low) dan empat hasil yang memprediksikan instan sebagai High (true: Middle)
3. Classification report

- **Precision**: Semua prediksi yang menghasilkan label high, middle, atau very low 100% benar; semua prediksi yang menghasilkan label low hanya 75% yang benar;
- **Recall**: Dari semua user yang diprediksi sebagai High atau Low, 100% hasil prediksi benar; dari semua user yang diprediksi sebagai Middle, hanya 89% hasil prediksi yang benar; dari semua user yang diprediksi sebagai very_low, hanya 33% hasil prediksi yang benar
- **F1-score**: Karena nilai F1-Score dari kelas very_low tidak mendekati 1, dapat dikatakan model ini kurang baik dalam melakukan klasifikasi
- **Support**: Model memprediksikan terdapat 9 kelas High, 18 kelas Low, 19 kelas Middle, dan 6 kelas very_low

Kesimpulan

Dari ketiga model, model yang paling baik dalam melakukan klasifikasi adalah SVC dengan akurasi 0.88 dan macro avg F1-score sebesar 0.83. Walaupun F1-score untuk kelas very_low cukup rendah (50%), dapat dilihat bahwa kelas ini memiliki distribusi yang paling sedikit (6).