MODUL PRAKTIKUM KECERDASAN BUATAN

PERTEMUAN 10 – PEMBELAJARAN MESIN

Tools : Jupyter notebook
Bahasa Pemrograman : Python

Dalam praktikum kali ini, kita akan belajar tentang penerapan pembelajaran mesin sederhana menggunakan Python.

Pembelajaran Mesin (*Machine Learning*)

Pembelajaran mesin (*machine learning*) adalah cabang dari kecerdasan buatan yang memungkinkan mesin dan sistem komputer untuk belajar dari data tanpa harus secara eksplisit diprogram. Dengan algoritma yang ditingkatkan, mesin dapat belajar dengan sendirinya dari data yang diberikan dan dapat diadaptasi ke situasi atau masalah yang berkembang. Tujuan utama *machine learning* adalah untuk membuat mesin yang lebih cerdas atau yang mampu memberikan prediksi akurat dengan cepat dari suatu data tanpa intervensi manusia secara terus-menerus.

Tahapan dalam Pembelajaran Mesin (*Machine Learning*)

Tahapan dalam Machine Learning adalah sebagai berikut:

1. ***Gathering Data* (pengumpulan data)**: Tahap pertama dalam Machine Learning adalah pengumpulan data. Data tersebut dapat diperoleh dari berbagai sumber seperti pengamatan, survei, atau bahkan dari internet. Data tersebut secara umum dibagi menjadi dua kategori, yaitu data latih (training data) dan data uji (test data).
2. ***Data Cleaning* (pembersihan data)**: Tahap kedua adalah pembersihan data. Pada tahap ini, data yang sudah dikumpulkan harus dipastikan tidak mengandung data yang tidak relevan atau data yang tidak diperlukan. Data cleaning dapat melibatkan teknik seperti penghapusan data yang hilang, penghapusan data duplikat, dan sebagainya.
3. ***Data Preparation* (persiapan data)**: Tahap ketiga adalah persiapan data atau *data preparation*. Pada tahap ini, data yang telah dibersihkan perlu dipersiapkan agar dapat digunakan dalam algoritma Machine Learning. Persiapan data dapat melibatkan teknik seperti standarisasi data, normalisasi data, dan encoding data.
4. ***Model Training* (pelatihan model)**: Tahap keempat adalah pelatihan model. Pada tahap ini, algoritma Machine Learning akan digunakan untuk melatih model berdasarkan data latih yang telah dipersiapkan. Tujuannya adalah untuk membuat model yang dapat menghasilkan hasil yang akurat.

5. ***Model Evaluation*** **(evaluasi model)**: Tahap kelima adalah evaluasi model. Pada tahap ini, model yang telah dilatih akan dievaluasi menggunakan data uji yang telah dipersiapkan sebelumnya untuk memastikan bahwa model dapat menghasilkan hasil yang akurat.
6. ***Model Tuning*** **(penyetelan model)**: Tahap keenam adalah penyetelan model. Pada tahap ini, model akan disetel ulang atau diperbaiki jika hasil evaluasi tidak memenuhi standar yang diinginkan. Model yang telah disetel ulang kemudian akan dievaluasi dan dilatih kembali sampai mencapai tingkat akurasi yang diinginkan.
7. ***Deployment*** **(penyebaran)**: Tahap terakhir adalah penyebaran atau deployment. Pada tahap ini, model yang telah dilatih, dievaluasi, dan disetel ulang kemudian akan diterapkan pada data baru untuk memprediksi hasil yang diinginkan.

Metode dalam Pembelajaran Mesin (*Machine Learning*)

Terdapat beberapa metode dalam pembelajaran mesin, antara lain:

1. ***Supervised Learning***: *Mesin learning* yang menggunakan data input dan output yang sudah diketahui untuk melakukan prediksi. Artinya, mesin belajar dari data yang sudah diberi label dan membuat prediksi berdasarkan pola yang terdapat pada data tersebut. Contoh dari supervised learning adalah:
    a. Klasifikasi: misalnya membedakan spam email dari email yang asli.
    b. Regresi: misalnya memprediksi harga rumah berdasarkan luas tanah.
2. ***Unsupervised Learning***: *Mesin learning* tidak menggunakan data input/output yang sudah diketahui, namun mempelajari data tersebut melalui pengelompokan atau klasifikasi secara mandiri. Contoh dari unsupervised learning adalah:
    a. *Clustering*: memisahkan data menjadi beberapa kelompok berdasarkan pola yang terdapat pada data tersebut.
    b. *Dimensionality reduction*: mengurangi jumlah dimensi data untuk memudahkan analisa.
3. ***Reinforcement Learning***: Metode ini mengajarkan mesin untuk membuat keputusan berdasarkan keadaan lingkungan dan memberikan *feedback* yang tepat atau salah. Contoh dari *reinforcement learning* adalah:
    a. *Game* AI: sebuah agent belajar memainkan sebuah *game* dan bertindak melalui *trial* dan *error*.
    b. Navigasi: sebuah robot belajar bergerak di ruang lingkup tertentu dan menghindari hambatan.

Penerapan Pembelajaran Mesin Sederhana

Pada sesi kali ini, kita akan menerapkan *machine learning* sederhana menggunakan Python. Adapun metode yang digunakan adalah *supervised learning*. *Machine learning* digunakan untuk melakukan prediksi adanya penyakit jantung berdasarkan data The Heart-Disease-Dataset (https://www.kaggle.com/datasets/ineubytes/heart-disease-dataset). Adapun langkah - langkah yang perlu dilakukan adalah sebagai berikut:

1. Mengimpor *library* yang relevan

Pertama, kita perlu impor semua modul, fungsi, dan objek yang akan kita gunakan dalam menerapkan *machine learning*.

Simple Machine Learning Project

Import libraries

In [42]:
```
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
```

2. Memuat *dataset*

Load dataset

In [43]:
```
dataset = read_csv("heart.csv")
```

3. Melihat ringkasan *dataset*

a. Melihat dimensi data

Summarize The Dataset

In [44]:
```
# shape
print(dataset.shape)
```

Output:

```
(1025, 14)
```

Hal tersebut menunjukkan bahwa dalam *dataset* The Heart-Disease-Dataset terdapat 1025 obyek data dan 14 fitur atau atribut.

b. Melihat cuplikan data

```
In [45]: # head
         print(dataset.head(10))
```

Output:

```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   52    1   0       125   212    0        1      168      0      1.0      2
1   53    1   0       140   203    1        0      155      1      3.1      0
2   70    1   0       145   174    0        1      125      1      2.6      0
3   61    1   0       148   203    0        1      161      0      0.0      2
4   62    0   0       138   294    1        1      106      0      1.9      1
5   58    0   0       100   248    0        0      122      0      1.0      1
6   58    1   0       114   318    0        2      140      0      4.4      0
7   55    1   0       160   289    0        0      145      1      0.8      1
8   46    1   0       120   249    0        0      144      0      0.8      2
9   54    1   0       122   286    0        0      116      1      3.2      1

   ca  thal  target
0   2     3       0
1   0     3       0
2   0     3       0
3   1     3       0
4   3     2       0
5   0     2       1
6   3     1       0
7   1     3       0
8   0     3       0
9   2     2       0
```

Fungsi head() digunakan untuk melihat data teratas. Dalam hal ini, kita akan melihat 10 data teratas dari *dataset*.

c. Melihat ringkasan statistik

Kali ini, kita akan menampilkan nilai count, mean, min dan max serta beberapa persentil dari beberapa atribut dalam *dataset*.

```
In [46]: # statistical summarize
         # descriptions
         print(dataset[["age", "trestbps", "chol", "fbs"]].describe())
```

Output:

```
                age     trestbps        chol         fbs
count  1025.000000  1025.000000  1025.00000  1025.000000
mean     54.434146   131.611707   246.00000     0.149268
std       9.072290    17.516718    51.59251     0.356527
min      29.000000    94.000000   126.00000     0.000000
25%      48.000000   120.000000   211.00000     0.000000
50%      56.000000   130.000000   240.00000     0.000000
75%      61.000000   140.000000   275.00000     0.000000
max      77.000000   200.000000   564.00000     1.000000
```

d. Melihat distribusi kelas

Sekarang kita akan melihat jumlah data yang dimiliki oleh tiap kelas sehingga dapat terlihat bahwa kelas 0 (normal) terdiri dari 499 data dan kelas 1 (sakit jantung) terdiri dari 526 data.

```
In [47]: # class distribution
         print(dataset.groupby('target').size())
```

Output:

```
target
0    499
1    526
dtype: int64
```

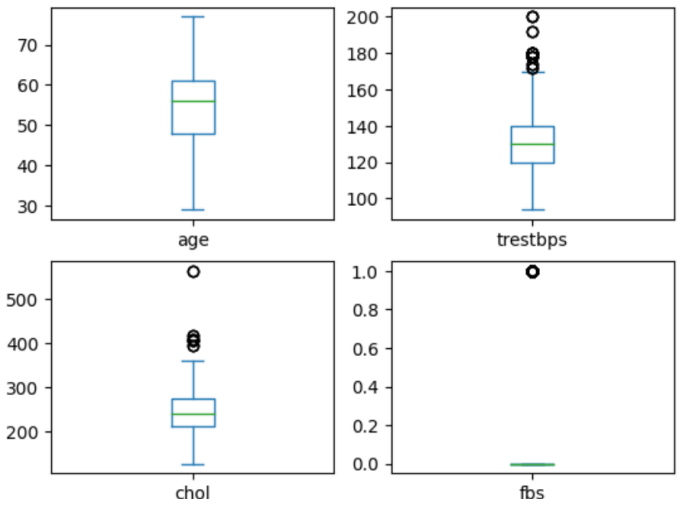
4. Visualisasi data

a. Plot univariat

Kita akan melihat distribusi nilai atribut menggunakan *boxplot*.

```
In [58]: # box and whisker plots
         dataset[["age", "trestbps", "chol", "fbs"]].plot(kind='box', subplots=True, \
         layout=(2,2), sharex=False, sharey=False)
         pyplot.show()
```

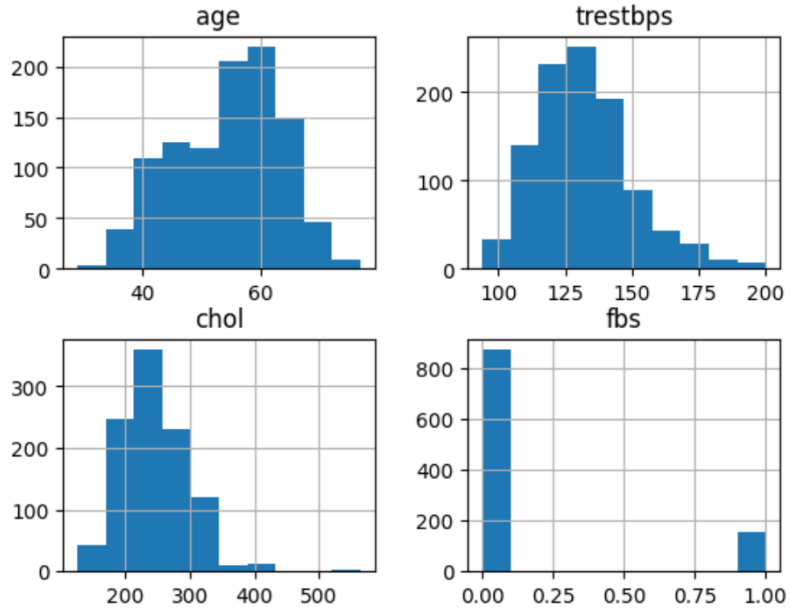
Output:



Kita juga dapat melihat distribusi nilai atribut menggunakan histogram.

```
In [49]: # histograms
         dataset[["age", "trestbps", "chol", "fbs"]].hist()
         pyplot.show()
```

Output:

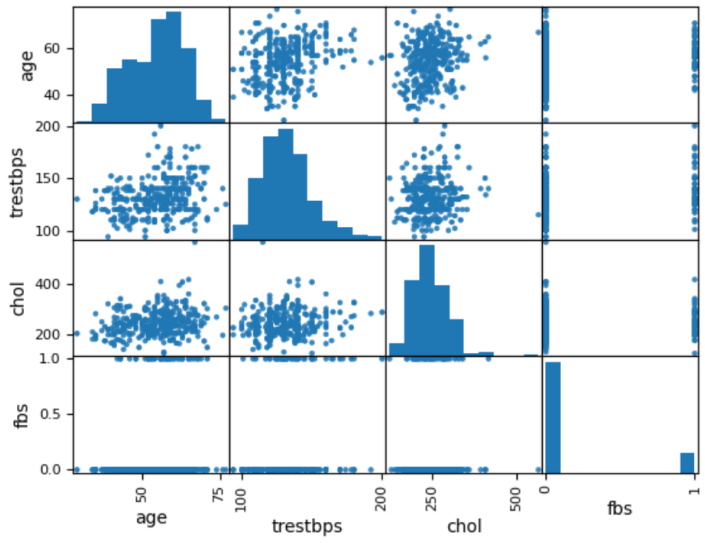


b. Plot multivariat

Kita akan melihat interaksi antar atribut menggunakan plot multivariat. Kali ini, kita akan mengambil contoh interaksi beberapa atribut saja.

```
In [50]: # scatter plot matrix
         scatter_matrix(dataset[["age", "trestbps", "chol", "fbs"]])
         pyplot.show()
```

Output:



5. Evaluasi algoritma

a. Membuat *dataset* validasi

Kita akan membagi *dataset* menjadi dua bagian, 70% di antaranya akan digunakan untuk melatih, mengevaluasi, dan memilih model, dan 30% akan kita tahan sebagai *dataset* validasi.

Evaluate Some Algorithms

Create a Validation Dataset

In [73]:
```
# Split-out validation dataset
array = dataset.values
X = array[:,0:13]
y = array[:,13]
X_train, X_validation, Y_train, Y_validation = train_test_split(X, y, test_size=0.30, random_state=1)
```

b. Membangun model dana membagi *dataset* menggunakan *cross-validation*

Kali ini kita akan menguji 6 algoritma, antara lain:

1. *Logistic Regression* (LR)
2. *Linear Discriminant Analysis* (LDA)
3. *K-Nearest Neighbors* (KNN).
4. *Classification and Regression Trees* (CART).
5. *Gaussian Naive Bayes* (NB).
6. *Support Vector Machines* (SVM).

Build Model & Test Harness

In [54]:
```
# Spot Check Algorithms
models = []
models.append(('LR', LogisticRegression(solver='liblinear', multi_class='ovr')))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))
# evaluate each model in turn
results = []
names = []
for name, model in models:
 kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
 cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring='accuracy')
 results.append(cv_results)
 names.append(name)
 print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))
```

Output:

```
LR: 0.835544 (0.047796)
LDA: 0.832707 (0.047211)
KNN: 0.729480 (0.041518)
CART: 0.983275 (0.016221)
NB: 0.836894 (0.049667)
SVM: 0.972183 (0.023241)
```

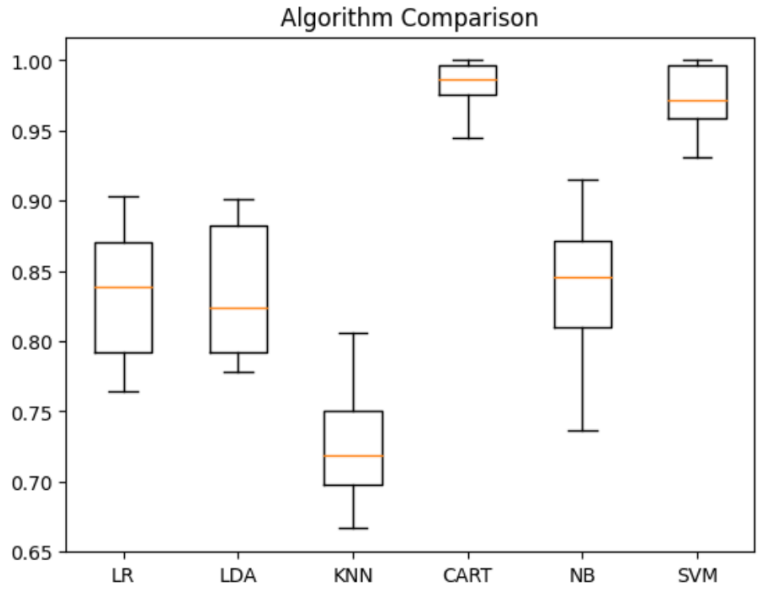
c. Memilih model terbaik

Kita juga dapat membuat plot dari hasil evaluasi model dan membandingkan akurasi rata-rata dari masing-masing model.

Select Best Model

In [55]:
```
# Compare Algorithms
pyplot.boxplot(results, labels=names)
pyplot.title('Algorithm Comparison')
pyplot.show()
```

Output:



Berdasarkan plot di atas dapat diketahui bahwa *Classification and Regression Trees* (CART) memiliki estimasi nilai akurasi tertinggi yaitu sebesar 98,3275%.

6. Melakukan prediksi

a. Melakukan prediksi

Berdasarkan hasil pengujian algoritma pada langkah sebelumnya, kita akan menggunakan *Classification and Regression Trees* (CART) untuk melakukan prediksi.

Make Predictions

In [56]:
```
# Make predictions on validation dataset
model = DecisionTreeClassifier()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
```

b. Evaluasi prediksi

Evaluate Predictions

```
In [57]: # Evaluate predictions
         print(accuracy_score(Y_validation, predictions))
         print(confusion_matrix(Y_validation, predictions))
         print(classification_report(Y_validation, predictions))
```

Output:

```
0.9902597402597403
[[161   0]
 [  3 144]]
              precision    recall  f1-score   support

         0.0       0.98      1.00      0.99       161
         1.0       1.00      0.98      0.99       147

    accuracy                           0.99       308
   macro avg       0.99      0.99      0.99       308
weighted avg       0.99      0.99      0.99       308
```

Kita dapat melihat bahwa akurasi hasil prediksinya adalah 0,99 atau sekitar 99 % pada *dataset* yang bertahan. Adapun laporan klasifikasi memberikan perincian setiap kelas berdasarkan *precision*, *recall*, *f1-score* and *support* yang menunjukkan hasil yang sangat baik (mengingat set data validasinya kecil).

Latihan: Melakukan prediksi kebahagian

1. Terapkan *machine learning* untuk melakukan prediksi kebahagiaan menggunakan *dataset* happydata.csv(https://www.kaggle.com/datasets/priyanshusethi/happiness-classification-dataset)!
2. Tunjukkan hasil evaluasi dari penerapan di atas!

Referensi:

https://machinelearningmastery.com/machine-learning-in-python-step-by-step/

https://www.w3schools.com/python/python_ml_getting_started.asp

https://www.ibm.com/topics/machine-learning