Machine Learning dengan Python

Scikit-learn

- Install scikit-learn untuk machine learning
- Data untuk worksp ini tersedia di http://wcw.cs.ui.ac.id/publik/iris.csv

Load Dataset

```
import pandas as pd
data = pd.read_csv('iris.csv')
```

Summarize Dataset

- Dimension of dataset
  - `data.shape`

    (150, 5)
- Peek the Data
  - `data.head()`
  - `data.tail()`
  - `data.info()`

Statistical Summary

```
data.describe()
```

	sepal-length	sepal-width	petal-length	petal-width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Class Distribution

```
data.groupby('class').size()
```

```
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

Visualization

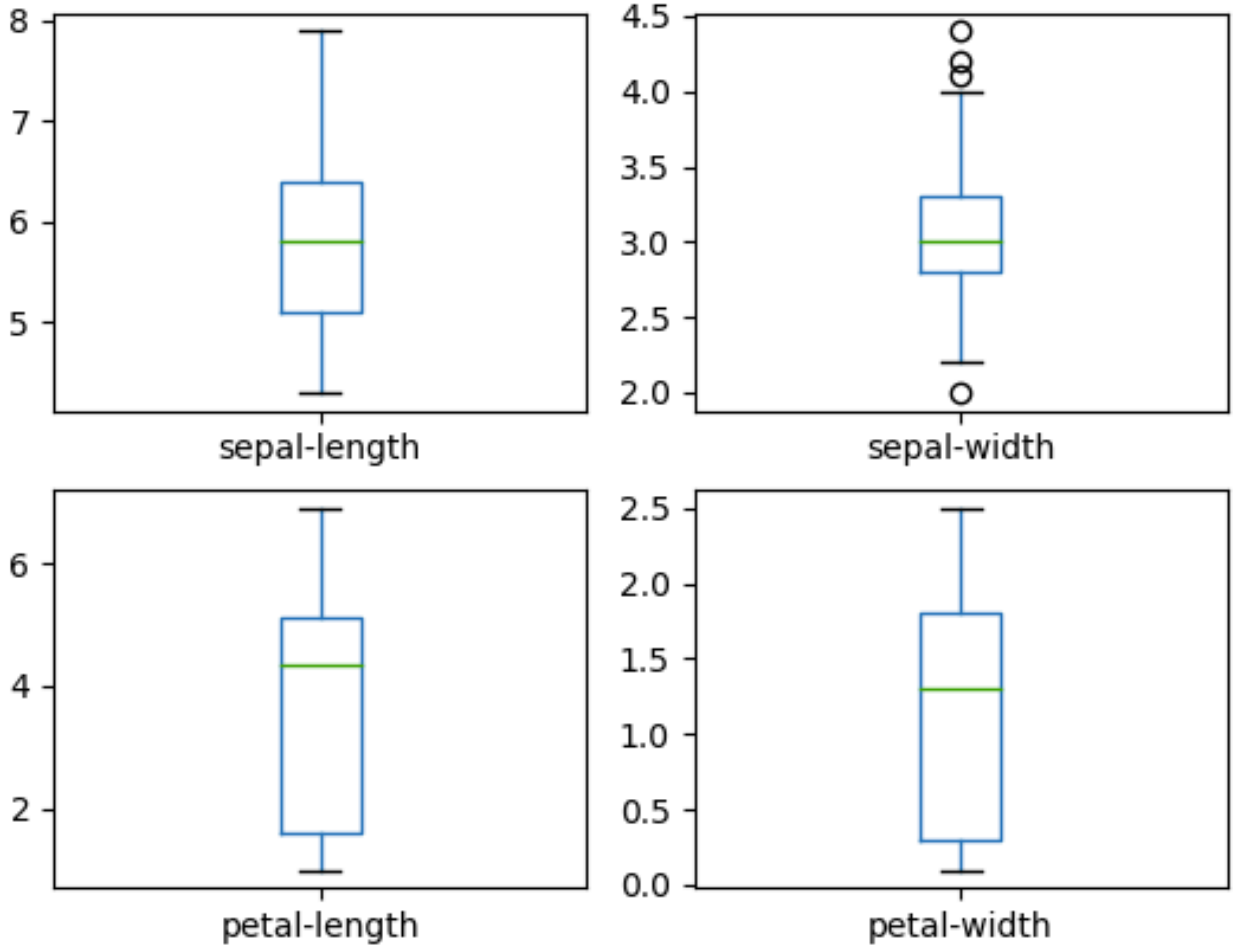
- Univariate Plots
  - To better understand each attribute

- Multivariate
  - To better understand relationship between attributes
- Lets remove the dependent variable **class** and create a new data frame: **dataset**

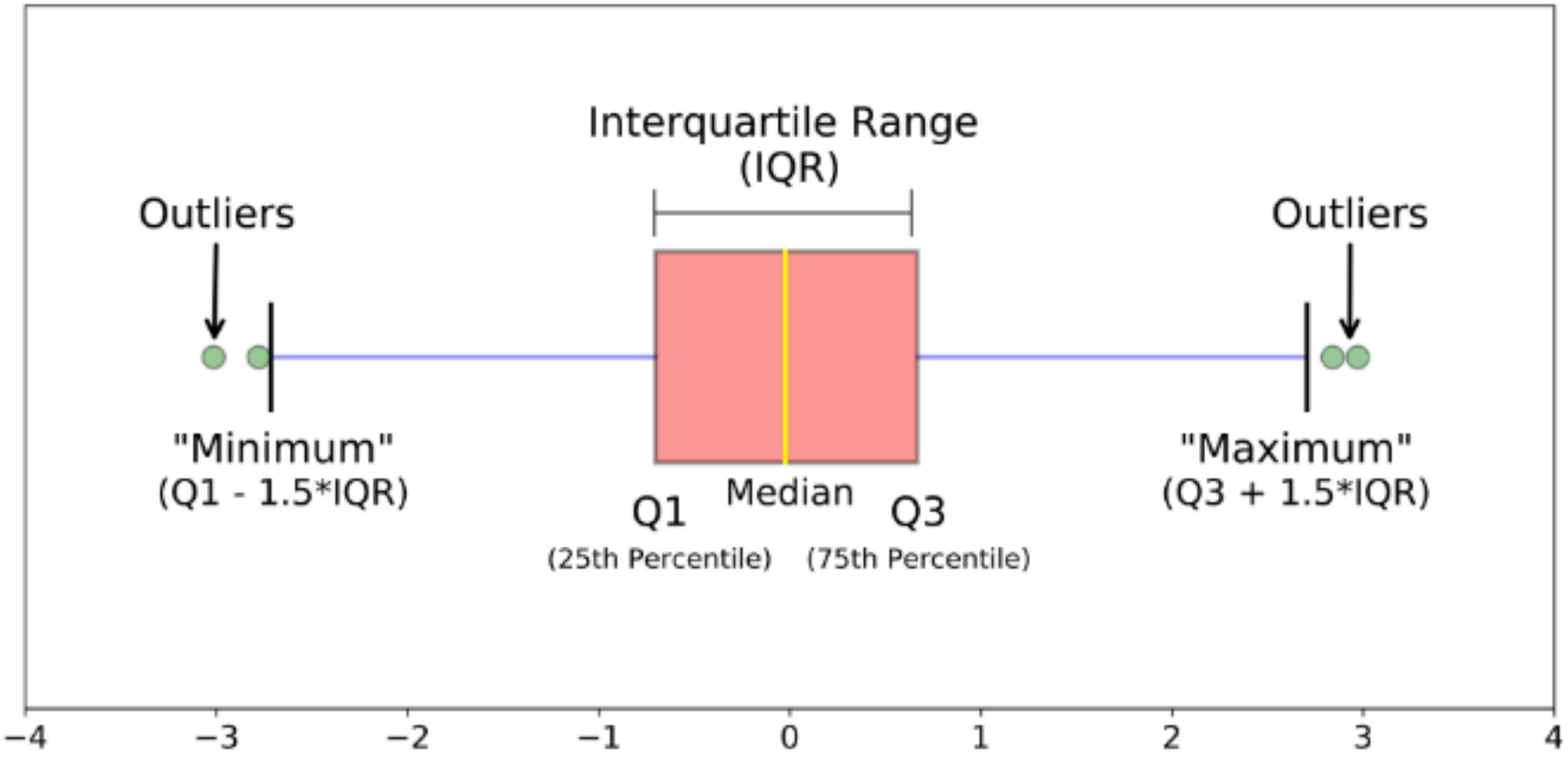
```
dataset = data.drop('class', axis=1)
Import matplotlib.pyplot as plt
```

Visualization: Univariate Plot

```
dataset.plot(kind='box', subplots=True,
layout=(2,2), sharex=False, sharey=False)
plt.show()
```

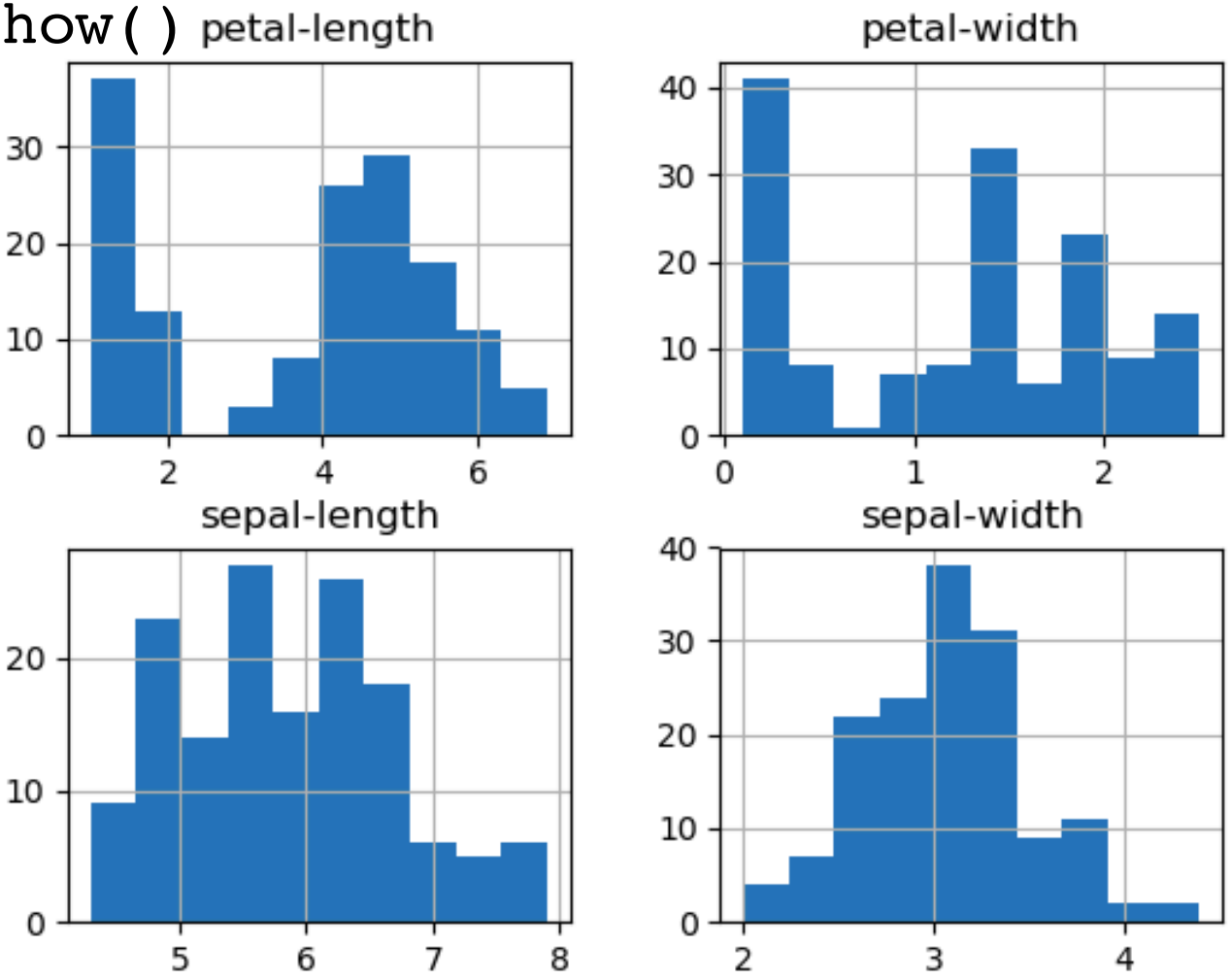


Understand Box Plot

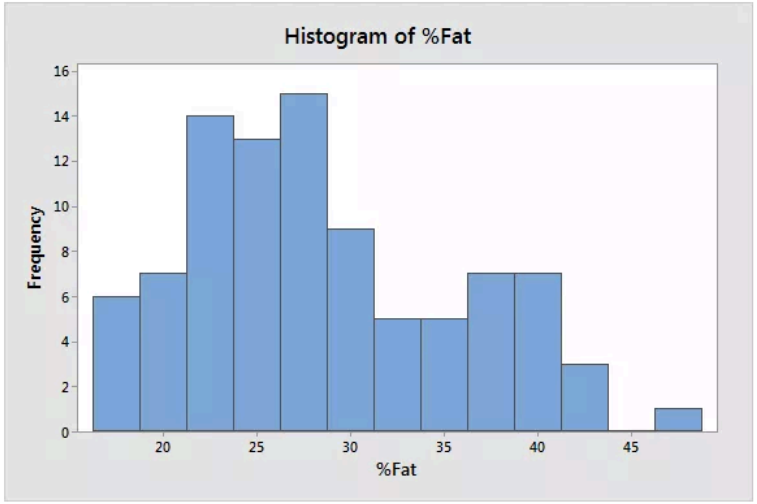


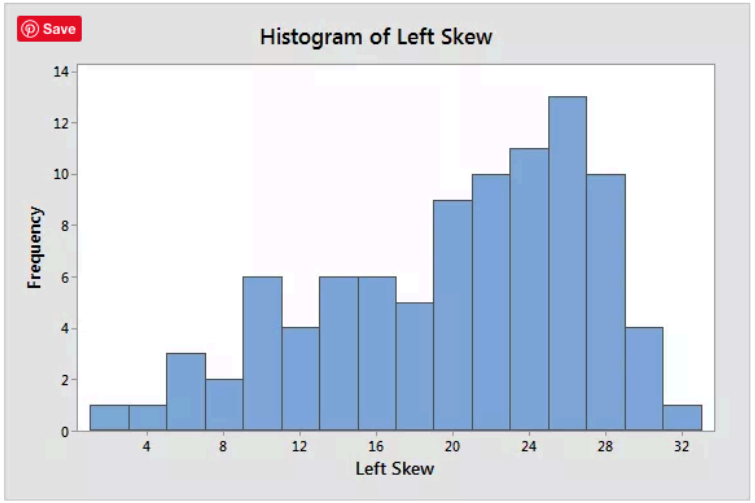
Visualization: Univariate Plot

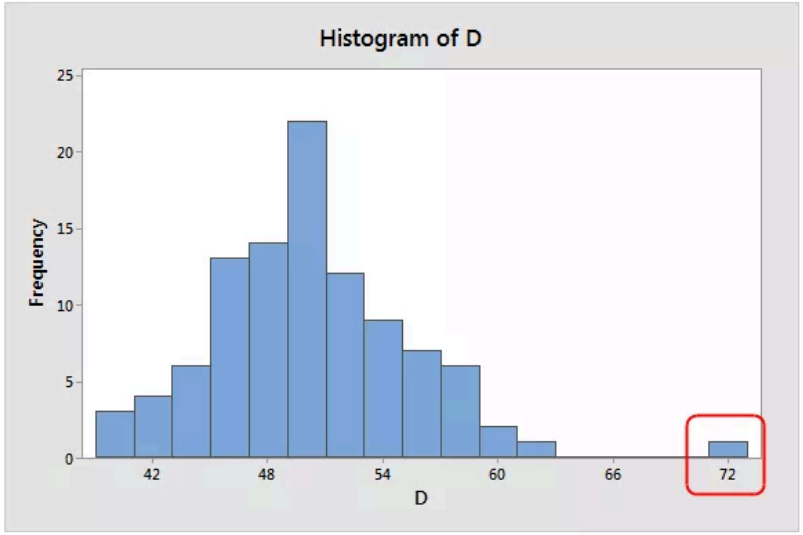
```
dataset.hist()
plt.show()
```



Understand Histogram

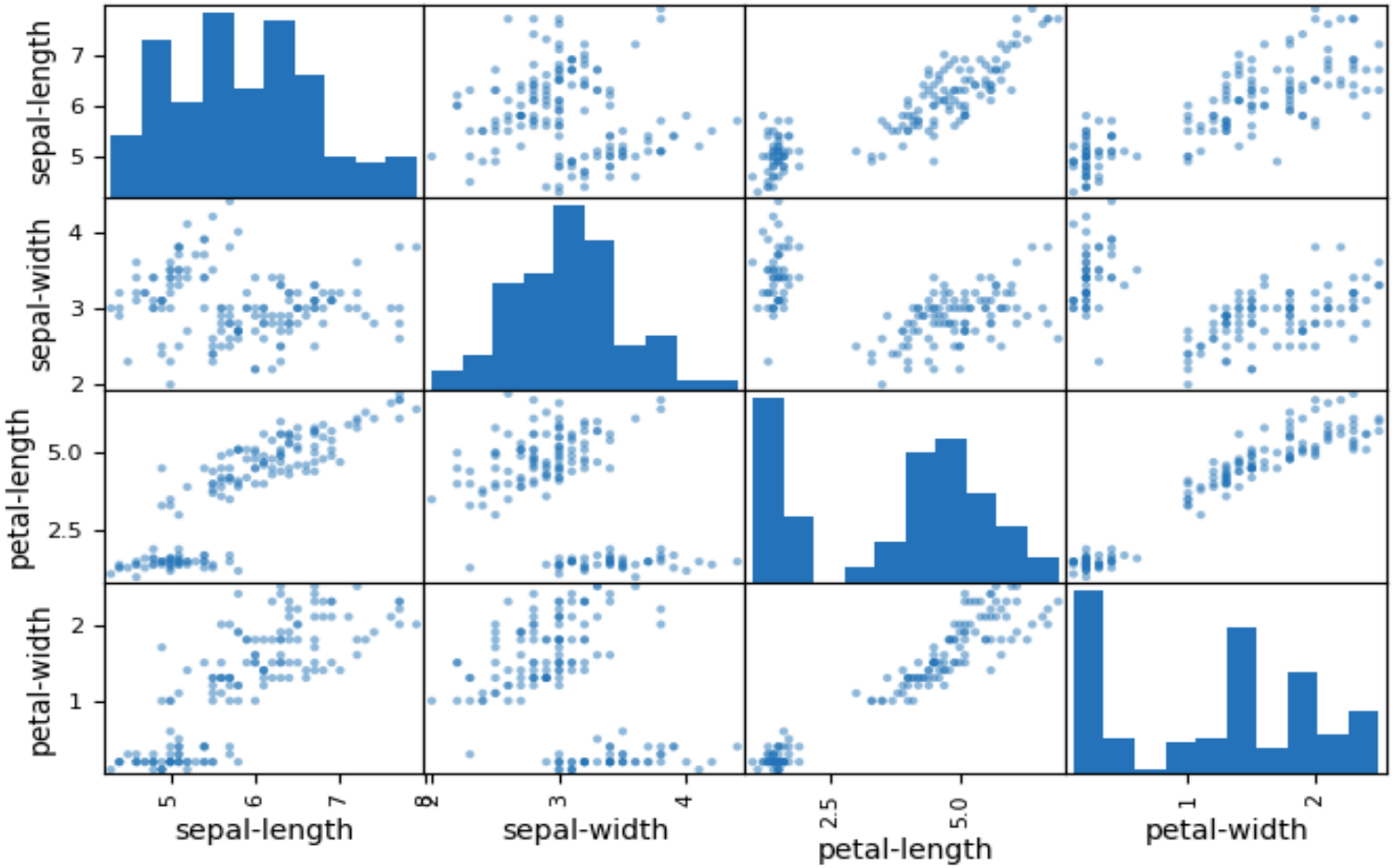






Visualization: Multivariate Plot

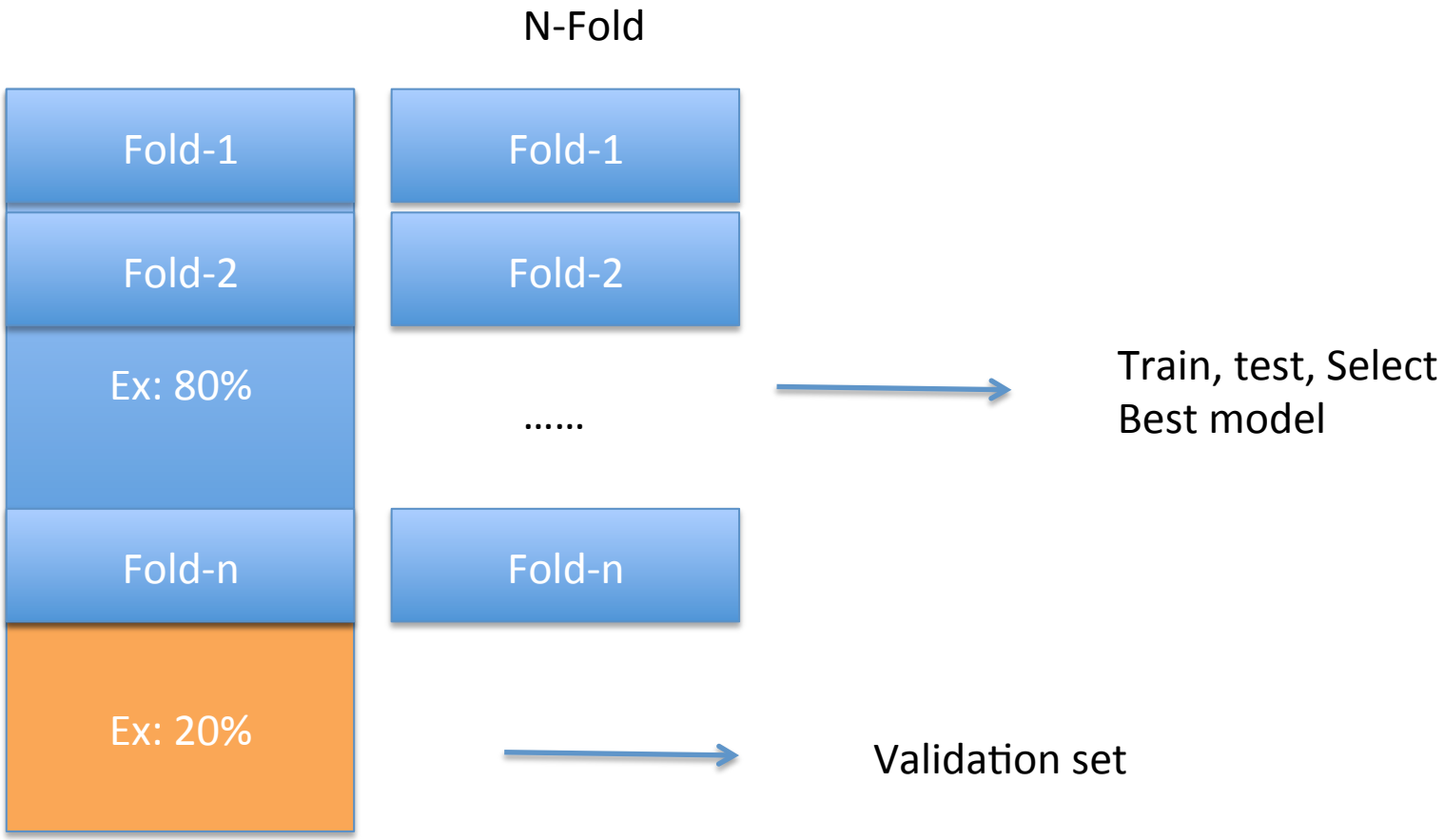
```
pd.plotting.scatter_matrix(dataset,
diagonal='hist')
plt.show()
```



Understand Scatter Plot

- No Association
- Has Association
  - Direction
    - -- positive
    - -- negative
  - Form
    - -- linear
    - -- non-linear
  - Strength
    - -- weak
    - – moderate
    - – strong
  - Outlier

Create Train & Validation Set



Create Train & Validation Set

- from sklearn.model_selection import train_test_split
- array = data.values
- X = array[:,0:4] ----- independent attributes
- Y = array[:,4] ----- dependent attribute
- X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=0.20, random_state=1)

Create Train & Validation Set

```
>>> len(X_train)
    120
>>> len(Y_train)
    120
>>> len(X_validation)
    30
>>> len(Y_validation)
    30
```

Test Harness

- Lets use 10 fold
- Stratified folding: each fold or split of the dataset will aim to have the same distribution of example by class as exist in the whole training dataset

Test Harness

```
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
```

Select Models

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
```

Evaluate Each Model

```
models = []
models.append(('KNN',
KNeighborsClassifier()))

models.append(('CART',
DecisionTreeClassifier()))

models.append(('NB', GaussianNB()))

models.append(('SVM', SVC(gamma='auto')))
```

Evaluate Each Model

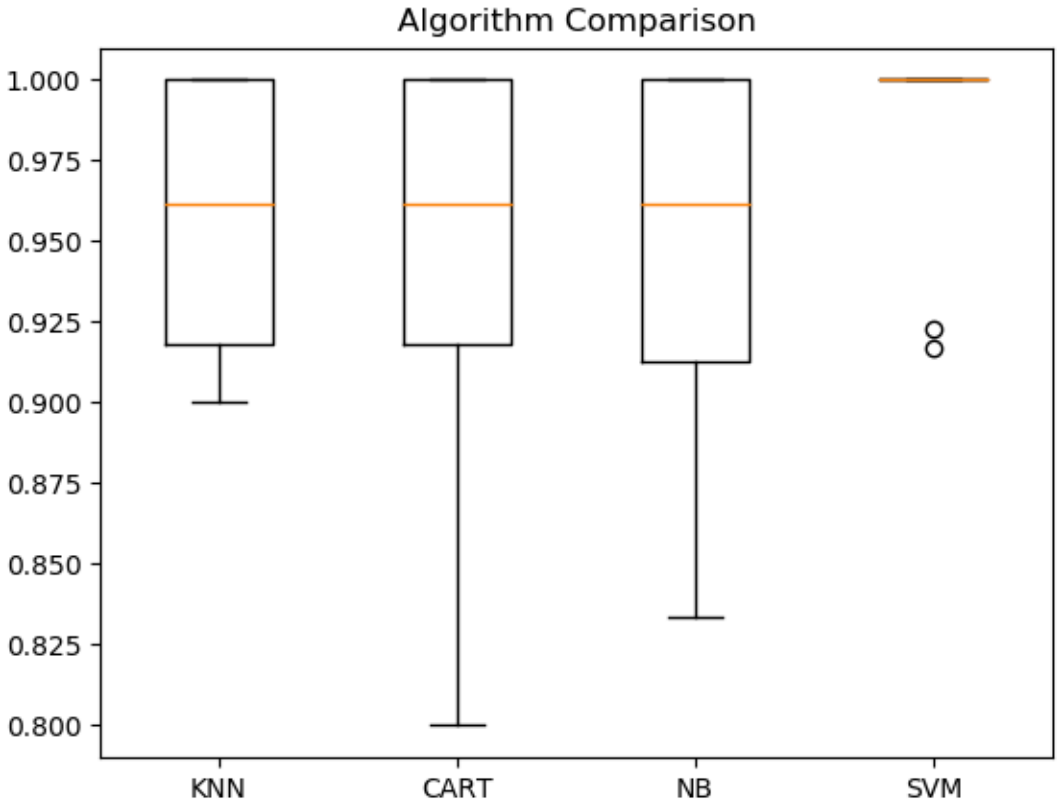
```
results = []
names = []
for name, model in models:
    kfold = StratifiedKFold(n_splits=10, random_state=1)
    cv_results = cross_val_score(model, X_train, Y_train,
    cv=kfold, scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    print('%s: %f (%f)' % (name, cv_results.mean(),
    cv_results.std()))
```

Results

- KNN: 0.957191 (0.043263)
- CART: 0.947191 (0.062574)
- NB: 0.948858 (0.056322)
- SVM: **0.983974** (0.032083)

Compare Results

- plt.boxplot(results, labels=names)
- plt.title('Algorithm Comparison')
- plt.show()



Make Prediction

- model = SVC(gamma='auto')
- model.fit(X_train, Y_train)
- predictions = model.predict(X_validation)

- print(accuracy_score(Y_validation, predictions))
- print(confusion_matrix(Y_validation, predictions))
- print(classification_report(Y_validation, predictions))

Make Prediction

```
>>> print(accuracy_score(Y_validation, predictions))
0.9666666666666667
>>> print(confusion_matrix(Y_validation, predictions))
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
>>> print(classification_report(Y_validation, predictions))
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	11
Iris-versicolor	1.00	0.92	0.96	13
Iris-virginica	0.86	1.00	0.92	6
micro avg	0.97	0.97	0.97	30
macro avg	0.95	0.97	0.96	30
weighted avg	0.97	0.97	0.97	30

Tugas 2

Lakukanlah Klasifikasi dengan menggunakan Machine Learning dengan membandingkan kinerja model DecisionTreeClassifier,

KNeighborsClassifier, GaussianNB, serta SVM

Data yang digunakan adalah data possibility of diabetes yang dapat diunduh di
http://wcw.cs.ui.ac.id/publik/diabetes.csv
Keterangan tentang data terdapat di
http://wcw.cs.ui.ac.id/publik/diabetes.txt

Jawablah Pertanyaan Berikut

- Jelaskan persepsi sdr tentang setiap atribut pada data!
- Jelaskan persepsi sdr tentang hubungan antar variabel!
- Jelaskan hasil setiap model, mengapa? diperoleh hasil seperti itu?
- Model mana kah yang terbaik?
- Jelaskan hasil dari prediksi validasi
- Lampirkanlah script dari program
- Tugas dikumpulkan dalam bentuk hardcopy paling lambat pada tanggal 28 November 2019