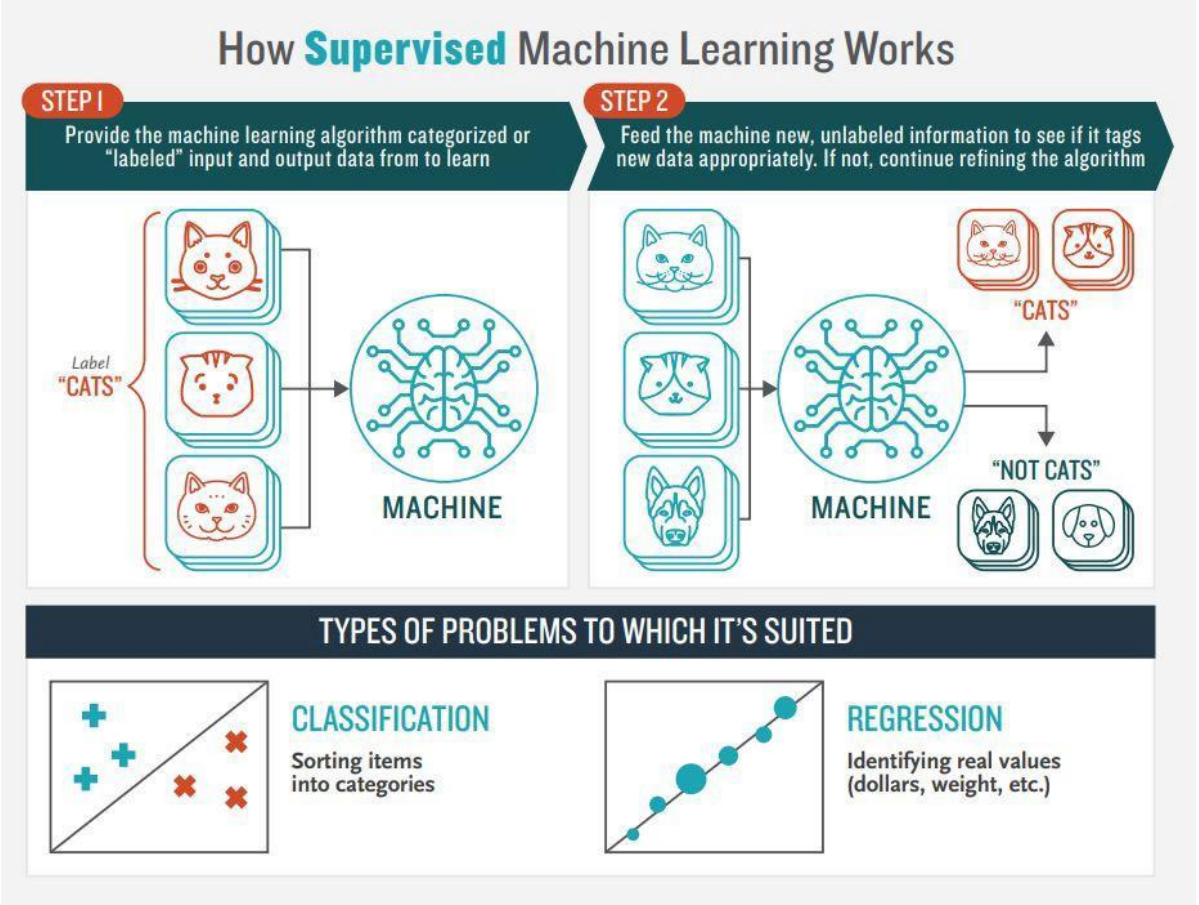
Modul Praktikum

Kecerdasan Buatan

5

Supervised Learning



1. Apa itu Supervised Learning

Supervised Learning merupakan salah satu pendekatan dalam Machine Learning (ML), yakni komputer membuat algoritma atau peraturan berdasarkan data yang diberikan. Supervised Learning dalam bahasa indonesia adalah pembelajaran mesin yang ada supervisornya, yaitu *label* pada tiap data.

Label adalah tag, keputusan, atau penamaan pada setiap nilai record. Sebagai contoh, gambar kucing diberi tag "kucing" pada tiap masing masing gambar kucing dan gambar anjing di tag "anjing" di tiap masing gambar anjing.

Model (algoritma/peraturan) *machine learning* akan dilatih hingga dapat mendeteksi ***patterns*** (pola) dan hubungan antara data yang di-*input* dan label *output*-nya. Hal ini akan menambah akurasi pada hasil *labeling* saat digunakan pada data yang belum pernah dipelajari atau dilihat sebelumnya.

Cara kerja algoritma *supervised* yaitu mampu menerapkan informasi pada suatu data yang dilakukan dengan cara memberi label tertentu, contohnya seperti data yang sebelumnya sudah terklasifikasi.

Jenis algoritma ini dilakukan dengan cara perbandingan dalam pengalaman belajar yang sumbernya dari peristiwa sebelumnya, selain itu algoritma ini juga memiliki kemampuan dalam memberi target pada suatu output.

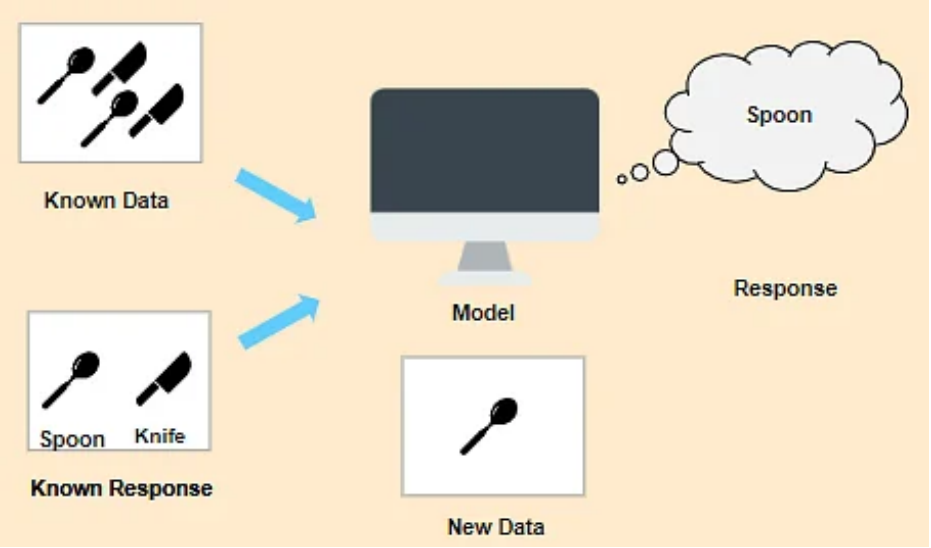
Supervised Learning terdiri dari variabel input dan variabel output sehingga kita dapat meramal apa output selanjutnya ketika ingin memasuki input baru. Dataset pada metode ini harus dilabeli dengan baik agar hasilnya akurat.

Supervised Learning ini ideal untuk Binary Classification, Multi-Class Classification, Regression Modeling, dan Ensembling. Supervised Learning dapat memberikan hasil yang akurat pada data yang kompleks, tetapi memakan waktu dan berat dalam biaya komputasi. Sedangkan, Unsupervised Learning lebih cepat bahkan real-time dan tidak berat dalam biaya komputasinya, namun kurang kompleks dan sangat kurang akurat.

2. Dataset Supervised Learning

Machine Learning tidak dapat bekerja tanpa data. Oleh sebab itu, hal yang pertama disiapkan adalah dataset. Dataset pada algoritma ini umumnya dibagi menjadi 2 bagian, yaitu *training set* dan *testing set* seperti yang telah dijelaskan di Modul 4, pada bagian Data Splitting.

Training set nantinya akan digunakan untuk melatih algoritma dalam mencari model yang sesuai, sedangkan ***testing set*** akan dipakai untuk menguji dan mengetahui performa model yang didapatkan pada tahapan testing. Ilustrasinya dapat dilihat pada gambar berikut:



Keterangan:

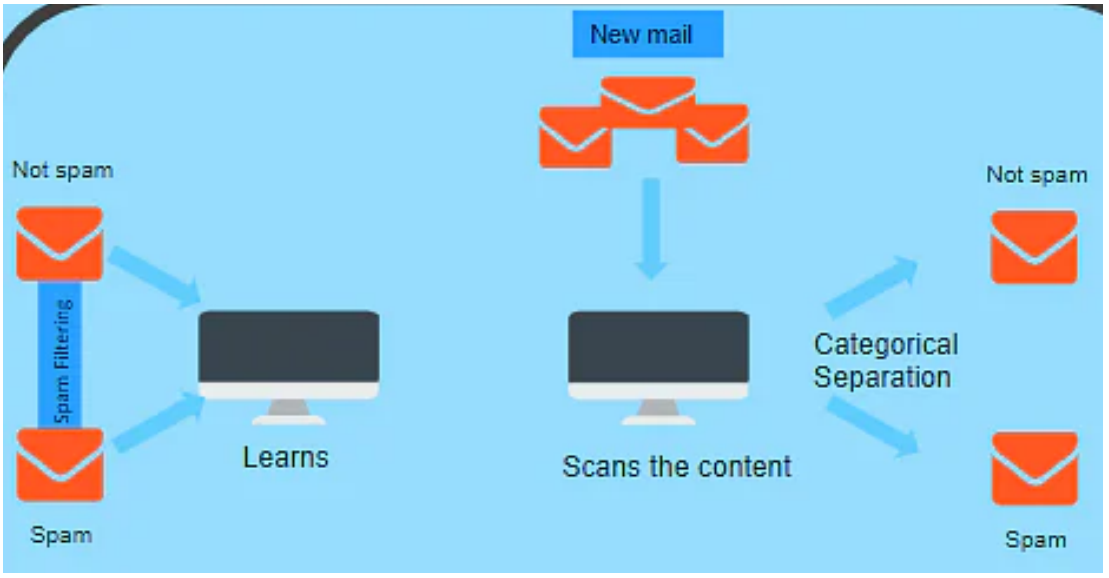
Known Data	: Data Training
Known Response	: Data Labelling
New Data	: Data Testing
Response	: Output

3. Jenis Supervised Learning

Supervised Learning ini sangat baik dalam klasifikasi dan regresi, contohnya seperti menentukan kategori artikel suatu berita atau memprediksi volume penjualan pada tanggal spesifik di masa depan.

a. Classification

Klasifikasi digunakan ketika variabel keluaran (output) adalah *categorical*. Misalnya suatu dataset memiliki 2 kelas atau lebih pada *value*-nya, seperti ya atau tidak, pria atau wanita, benar atau salah, besar atau sedang atau kecil, dan lain-lain. Sebagai contoh seperti pada gambar berikut:

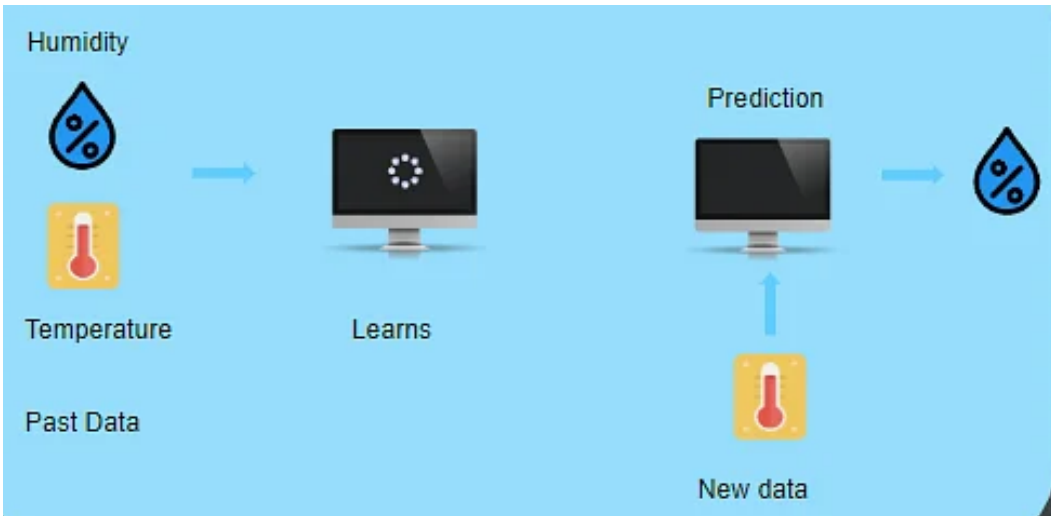


Untuk memprediksi suatu email adalah spam atau tidak, kita harus mengajari mesin seperti apa email yang memiliki unsur spam. Tentunya dengan cara mengkategorikan dari ulasan content dan header dari konten email, serta mencari informasi palsu yang terdapat pada email. Kata kunci dan filter blacklist juga digunakan dari pengirim spam yang telah di-blacklist. Fitur-fitur tersebut digunakan untuk memberi skor pada email.

Semakin tinggi skor pada suatu email, maka semakin dikategorikan sebagai email spam. Algoritma tersebut menentukan apakah harus disimpan pada folder Inbox atau folder Spam tergantung dari konten, label data, dan skor spam dari email yang datang (Simplilearn, 2022).

b. Regression

Regresi digunakan ketika variabel keluaran (output) bernilai real atau kontinyu. Pada kasus ini, terdapat hubungan antara dua variabel atau lebih misalnya suatu variabel berubah apabila ada perubahan dengan variabel lainnya. Sebagai contoh, suatu nilai rataan semester berubah apabila salah satu nilai seperti Matematika berubah.



Pada contoh di atas, terdapat dua variabel yaitu kelembapan sebagai variabel dependen dan temperatur sebagai variabel independen. Jika temperatur meningkat, maka kelembapan menurun. Dua variabel ini dimasukkan pada model dan komputer akan secara otomatis mempelajari hubungan antara mereka. Setelah *training* pada model selesai, model dapat memprediksi kelembapan tergantung dari temperatur yang diberikan.

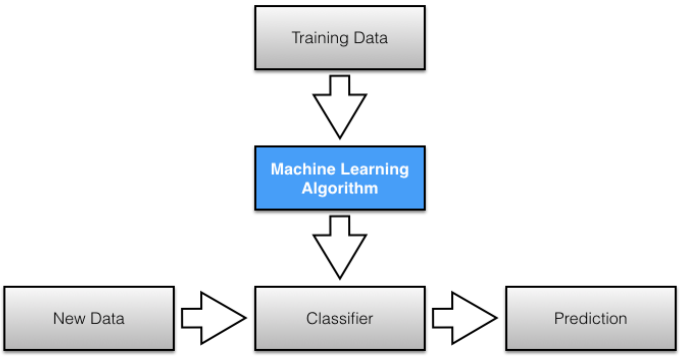
SQFT	BEDS	BATHS	PRICE
3,125	5	3	$530,000
2,100	4	2	$460,000
1,200	3	1.5	$250,000
3,950	6	4	???

Example of Data in ERP

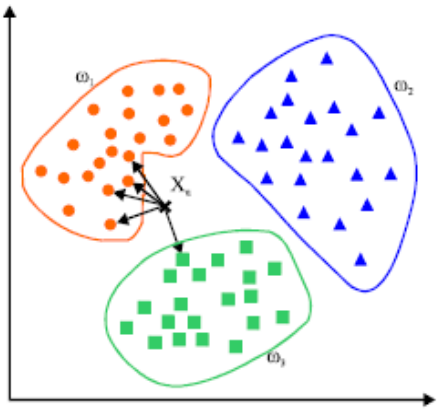
Source : https://medium.com/@gowthamy

Contoh lain yaitu pada gambar diatas, akan diprediksi harga rumah dengan luas tanah 3.950 square foot dengan 6 kamar tidur dan 4 kamar mandi, dengan harga sebagai target. Kemudian dilatih suatu *model machine learning* dengan 3 data sebelumnya dengan harga propertinya. Setelah itu, jika akurasinya kurang dari 80% maka diperlukan *refining model* hingga mendapat akurasi yang paling besar dan *loss* paling kecil.

4. Algoritma Supervised Learning



a. K-Nearest Neighbour (KNN)



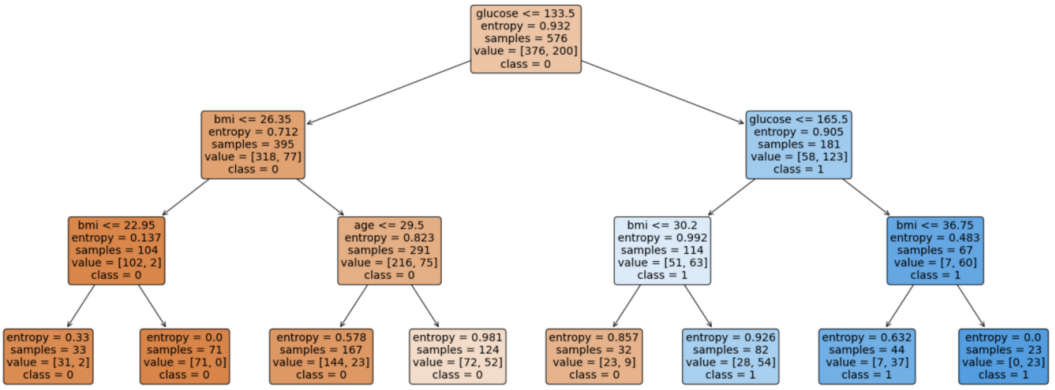
K-Nearest Neighbour atau algoritma KNN menggunakan algoritma non-parametrik yang mengelompokkan poin data berdasarkan kedekatan dan juga asosiasi mereka dengan data lain.

Algoritma KNN mengasumsikan bahwa titik data yang serupa akan selalu bisa ditemukan di sekitarnya. Konsekuensinya, algoritma ini pun selalu berupaya untuk menghitung jarak antar titik data (biasanya dengan Euclidean distance) dan kemudian menentukan kategori berdasarkan jenis yang paling sering muncul.

KNN ini disukai banyak ilmuwan data atau *data scientists*. Sebab, penggunaannya relatif mudah dan waktu perhitungannya pun cukup rendah. Namun, saat terus dilakukan uji dataset, waktu pemrosesan pun menjadi semakin lama. Oleh karena itu, KNN lebih sering dimanfaatkan untuk *recommendation system* dan *image recognition*.

```
import pandas as pd
import numpy as np
from sklearn.cross_validation import train_test_split
from sklearn.neighbors import KNeighborsClassifier

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

knn = KNeighborsClassifier(n_neighbors = 5)

knn.fit(X_train, y_train)
#KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
#   metric_params=None, n_jobs=1, n_neighbors=5, p=2, weights='uniform')

knn.score(X_test, y_test)
0.53333333333333333
```

b. Decision Tree



Decision Tree adalah sebuah tipe model yang digunakan untuk Supervised Learning. Decision Tree dapat digunakan untuk menyelesaikan masalah klasifikasi dan regresi, namun lebih sering digunakan untuk masalah klasifikasi.

Decision Tree memiliki bentuk seperti pohon, dimana tree memiliki node akar (**root node**), **decision node** dan **node daun** (leaf node). Leaf node adalah node akhir yang tidak dapat dipecah dan yang akan menentukan hasil prediksi decision tree.

Decision Tree bekerja dengan melakukan klasifikasi berdasarkan atribut yang paling membedakan. Algoritma decision tree akan dimulai pada node akar, kemudian akan melakukan perbandingan antara nilai yang ada pada dataset dan data yang ingin diprediksi. Berdasarkan perbandingan tersebut, algoritma akan turun ke bawah dan melanjutkan ke sub-node yang lebih dalam. Algoritma ini akan berulang hingga mencapai leaf node, dimana keputusan prediksi bisa dibuat.

Kelebihan	Kekurangan
Model simpel dan mudah dimengerti (mirip seperti cara manusia membuat keputusan)	Untuk klasifikasi dengan banyak label, akan membuat komputasi model lebih kompleks
Dapat digunakan untuk mencari semua kemungkinan keputusan	Dapat memiliki masalah overfitting
Tidak perlu melakukan banyak pembersihan atau augmentasi data	Tree dapat memiliki banyak layer, yang membuat model kompleks

```
###### IMPORT MODULES #######
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier

#### TRAIN TEST SPLIT ####
train_predictor, test_predictor, train_target, test_target = \
train_test_split(predictor, target, test_size=0.25)

print test_predictor.shape
print train_predictor.shape
(38, 4)
(112, 4)

#### CREATE MODEL ####
clf = DecisionTreeClassifier()

#### FIT MODEL ####
model = clf.fit(train_predictor, train_target)
print model
DecisionTreeClassifier(class_weight=None, criterion='gini',
            max_depth=None, max_features=None,
            max_leaf_nodes=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best')

#### TEST MODEL ####
predictions = model.predict(test_predictor)

print sklearn.metrics.confusion_matrix(test_target,predictions)
print sklearn.metrics.accuracy_score(test_target,
predictions)*100, '%'
[[14  0  0]
 [ 0 13  0]
 [ 0  1 10]]
97.3684210526 %

######## FEATURE IMPORTANCE ########
f_impt= pd.DataFrame(model.feature_importances_,
index=df.columns[:-2])
f_impt = f_impt.sort_values(by=0,ascending=False)
f_impt.columns = ['feature importance']

f_impt
petal width (cm)      0.952542
petal length (cm)     0.029591
sepal length (cm)     0.017867
sepal width (cm)      0.000000
```

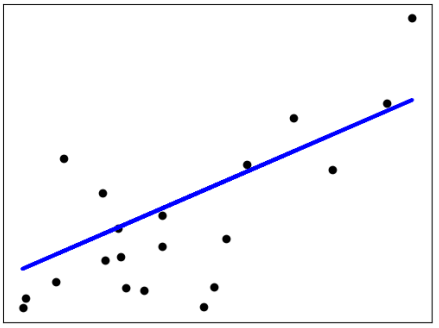
c. Linear Regression

Algoritma supervised learning yang satu ini biasanya digunakan dalam identifikasi hubungan antara variabel dependen dengan satu (atau lebih) variabel independen. Identifikasi tersebut kemudian digunakan untuk memprediksi hasil di masa depan.

Jika hanya ada satu variabel dependen dan satu variabel independen, maka disebut dengan simple Regresi Linear. Namun, jika terdapat banyak variabel sekaligus, disebut dengan istilah multiple Regresi Linear.

Contoh penggunaan Linear Regression [source]:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

# Load the diabetes dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)

# Use only one feature
diabetes_X = diabetes_X[:, np.newaxis, 2]

# Split the data into training/testing sets
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]

# Split the targets into training/testing sets
diabetes_y_train = diabetes_y[:-20]
diabetes_y_test = diabetes_y[-20:]

# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)

# Make predictions using the testing set
diabetes_y_pred = regr.predict(diabetes_X_test)

# Plot outputs
plt.scatter(diabetes_X_test, diabetes_y_test, color="black")
plt.plot(diabetes_X_test, diabetes_y_pred, color="blue", linewidth=3)
plt.xticks(())
plt.yticks(())
plt.show()
```



d. Naïve Bayes

Berikutnya ada Naïve Bayes, yang mengadopsi prinsip kemandirian kelas bersyarat dari Teorema Bayes. Dalam prinsip tersebut, ada tidaknya satu elemen tidak akan memengaruhi komponen lain dalam probabilitas hasil yang akan diberikan, dengan *predictor* yang mendapatkan efek yang sama.

Naïve Bayes kemudian dibagi menjadi tiga menurut penggolonganya: *Multinomial Naïve Bayes*, *Bernoulli Naïve Bayes*, dan *Gaussian Naïve Bayes*. [source]. Teknik ini umumnya digunakan dalam *text classification*, *recommendation system*, serta *spam detection*.

```
# load the iris dataset
from sklearn.datasets import load_iris
iris = load_iris()

# store the feature matrix (X) and response vector (y)
X = iris.data
y = iris.target

# splitting X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.4, random_state=1)

# training the model on training set
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# making predictions on the testing set
y_pred = gnb.predict(X_test)

# comparing actual response values (y_test) with predicted
response values (y_pred)
from sklearn import metrics
print("Gaussian Naive Bayes model accuracy(in %):",
metrics.accuracy_score(y_test, y_pred)*100)
```

```
Output: Gaussian Naive Bayes model accuracy(in %): 95.0
```

e. Support Vector Machine (SVM)

Support Vector Machine atau sering disingkat SVM merupakan metode algoritma Supervised Learning yang dikembangkan oleh Vladimir Vapnik. Metode ini biasanya digunakan dalam *data classification* dan juga *regression*.

Metode Support Vector Machine sering dimanfaatkan dalam masalah klasifikasi serta pembangunan *hyperplane* atau batas keputusan yang memisahkan kelas-kelas titik data. [source]

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

bankdata = pd.read_csv("D:/Datasets/bill_authentication.csv")
bankdata.shape
bankdata.head()
```

	Variance	Skewness	Curtosis	Entropy	Class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

```
# Data Preprocessing
X = bankdata.drop('Class', axis=1)
y = bankdata['Class']

# Data Splitting
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.20)

# Melatih Algoritma
from sklearn.svm import SVC
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)

# Buat Prediksi
y_pred = svclassifier.predict(X_test)

# Mengevaluasi Algoritma
from sklearn.metrics import classification_report,
confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

OUTPUT:

```
[[152    0]
 [  1  122]]
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       152
           1       1.00      0.99      1.00       123

avg / total       1.00      1.00      1.00       275
```

5. Evaluasi (Classification Report)

Ada 3 API yang berbeda untuk mengevaluasi kualitas dari prediksi suatu model:

- **Estimator score method**: Para Estimators memiliki sebuah metode `score` yang memberikan kriteria evaluasi *default* untuk masalah yang memang didesain untuk diselesaikan oleh para estimator.
- **Scoring parameter**: Alat *model-evaluation* yang menggunakan cross-validation seperti (`model_selection.cross_val_score` dan `model_selection.GridSearchCV`) yang bergantung dengan strategi *scoring* internal.
- **Metric functions**: Module `sklearn.metrics` mengimplementasi fungsi- fungsi yang dapat memberikan *prediction error* untuk tujuan yang spesifik. Metrics ini dijelaskan lebih lanjut di Classification metrics, Multilabel ranking metrics, Regression metrics dan Clustering metrics.

Untuk mengevaluasi, gunakan fungsi dibawah ini: [source]

```
sklearn.metrics.classification_report(y_true, y_pred)
```

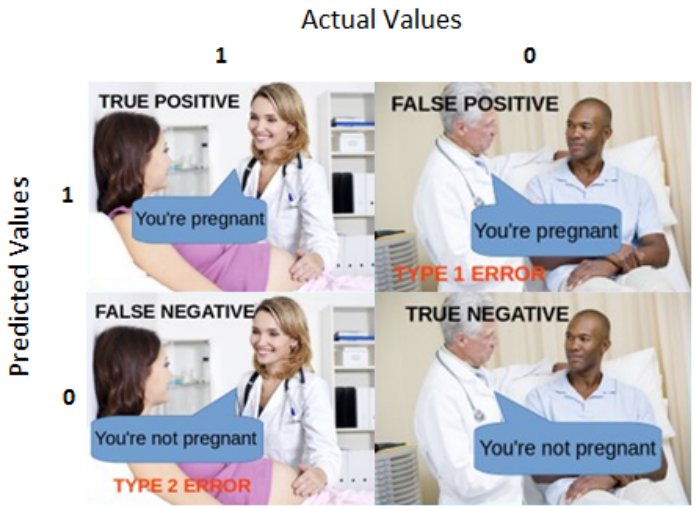
Contoh penggunaan Classification Report:

```
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
data =
pd.read_csv("https://raw.githubusercontent.com/amankharwal/SMS-Spam-D
etection/master/spam.csv", encoding= 'latin-1')
data = data[["class", "message"]]
x = np.array(data["message"])
y = np.array(data["class"])

# Fit Data menggunakan CountVectorizer
cv = CountVectorizer()
X = cv.fit_transform(x)

# Data Splitting
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=42)

# Classification menggunakan Multinomial Naive Bayes
clf = MultinomialNB()
clf.fit(X_train,y_train)
predictions = clf.predict(X_test)

# Classification Report
from sklearn.metrics import classification_report
print(classification_report(y_test, predictions))
```

	precision	recall	f1-score	support
ham	0.99	0.99	0.99	1587
spam	0.93	0.92	0.92	252
accuracy			0.98	1839
macro avg	0.96	0.95	0.96	1839
weighted avg	0.98	0.98	0.98	1839

a. Confusion Matrix (Classification)

Confusion Matrix adalah pengukuran performa untuk masalah klasifikasi machine learning dimana keluaran dapat berupa dua kelas atau lebih. Confusion Matrix merupakan sebuah tabel dengan 4 kombinasi berbeda dari nilai prediksi dan nilai aktual. Ada empat istilah yang merupakan representasi hasil proses klasifikasi pada confusion matrix yaitu:

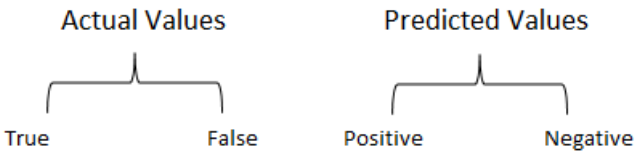


- **True Positive (TP) :**
  Interpretasi: Anda memprediksi positif dan itu benar.
  Anda memprediksikan bahwa seorang wanita hamil dan memang benar hamil.
- **True Negative (TN):**
  Interpretasi: Anda memprediksi negatif dan itu benar.
  Anda memprediksikan bahwa seorang pria tidak hamil dan benar, pria tidak mungkin hamil.
- **False Positive (FP):** (Kesalahan Tipe 1)
  Interpretasi: Anda memprediksi positif dan itu salah.
  Anda memprediksikan bahwa seorang pria hamil tetapi tidak mungkin pria bisa hamil.

- **False Negative (FN):** (Kesalahan Tipe 2/Sangat berbahaya)
  Interpretasi: Anda memprediksi negatif dan itu salah.
  Anda memperkirakan bahwa seorang wanita tidak hamil tetapi sebenarnya wanita tersebut hamil.

Dari contoh di atas dapat digambarkan bahwa:

- **Nilai Prediksi** : keluaran dari program yang nilainya Positif dan Negatif
- **Nilai Aktual** : nilai sebenarnya yang nilainya True dan False



Untuk mengenali Confusion Matrix lebih lanjut, lihat tabel ini:

y	y pred	output for threshold 0.6	Recall	Precision	Accuracy
0	0.5	0			
1	0.9	1			
0	0.7	1			
1	0.7	1	**1/2**	**2/3**	**4/7**
1	0.3	0			
0	0.4	0			
1	0.5	0			

$$Recall = \frac{TP}{TP+FN} \qquad Precision = \frac{TP}{TP+FP}$$

$$Accuracy = \frac{TP+TN}{Total} \qquad F\ Measure = \frac{2 \times Recall \times Precision}{Recall + Precision}$$

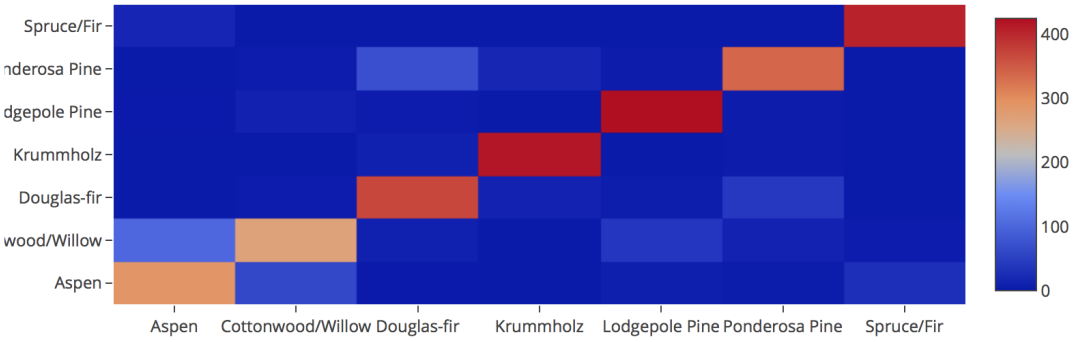
Recall: Dari semua kelas positif, berapa yang telah diprediksi secara benar.

Precision: Dari semua kelas yang telah diprediksi sebagai positif, berapa banyak yang sebenarnya positif.

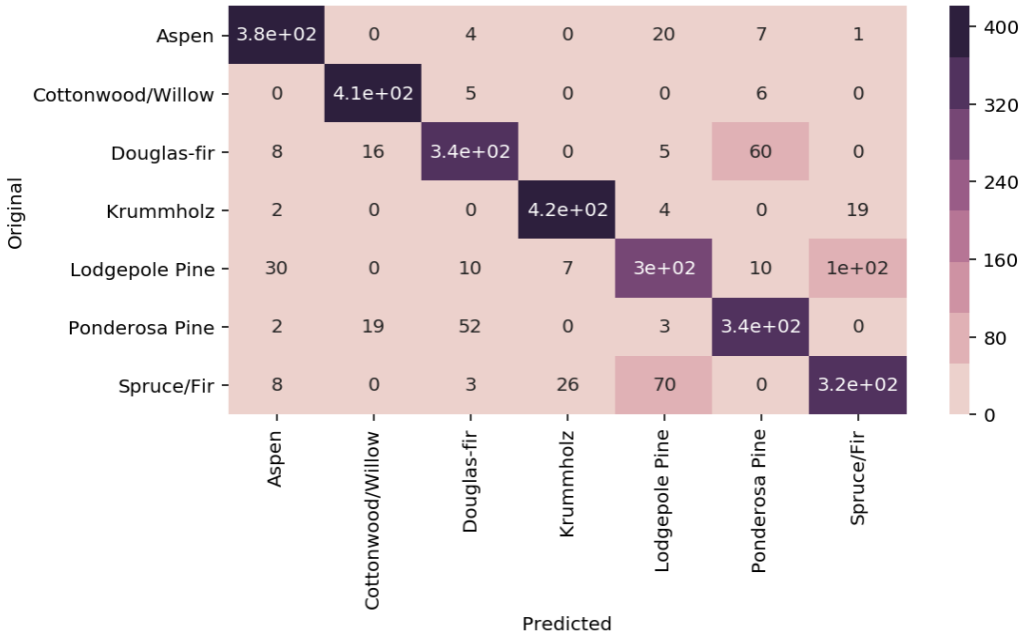
Accuracy: Dari semua kelas positif dan negatif, berapa banyak yang telah diprediksi secara benar.

F-Score: Jika sulit untuk membandingkan dua model dengan nilai Precision yang rendah dan Recall dengan nilai yang tinggi (atau sebaliknya), maka gunakan F-Score yang dapat membantu mengukur Recall dan Precision dalam satu waktu.

```
print (sklearn.metrics.confusion_matrix(test_target,predictions))
array([[288,  64,   1,   0,   7,   3,  31],
       [104, 268,  11,   0,  43,  15,   5],
       [  0,   5, 367,  15,   6,  46,   0],
       [  0,   0,  11, 416,   0,   4,   0],
       [  1,  13,   5,   0, 424,   4,   0],
       [  0,   5,  75,  22,   4, 337,   0],
       [ 20,   0,   0,   0,   0,   0, 404]])

# Buat heatmap menggunakan plotly
from plotly.offline import iplot
from plotly.offline import init_notebook_mode
import plotly.graph_objs as go
init_notebook_mode(connected=True)
layout = go.Layout(width=800, height=400)
data = go.Heatmap(z=x,x=title,y=title)
fig = go.Figure(data=[data], layout=layout)
iplot(fig)
```




```
# Tambahkan Confusion Matrix dari pd.crosstab sebelumnya
plt.figure(figsize=(10, 5))
sns.heatmap(confusion,annot=True,cmap=sns.cubehelix_palette(8));
```



b. Regression

R^2 (R-Squared) — proporsi varian dalam Y yang dapat dijelaskan oleh X.

Mean Squared Error (MSE) — Mengkalkulasi rata-rata dari setiap error menggunakan kuadrat (mengubah negatif ke positif) untuk melihat perbedaan antara nilai prediksi dan nilai aktualnya.

Root Mean Squared Error (RMSE) — Akar kuadrat dari Mean Squared Error untuk mengembalikan nilai error dengan skala yang sama.

Mean Absolute Error (MAE) — Rata-rata dari perbedaan antara nilai prediksi dan nilai aktualnya. Mengukur seberapa jauh prediksi dari keluaran yang telah diuji.

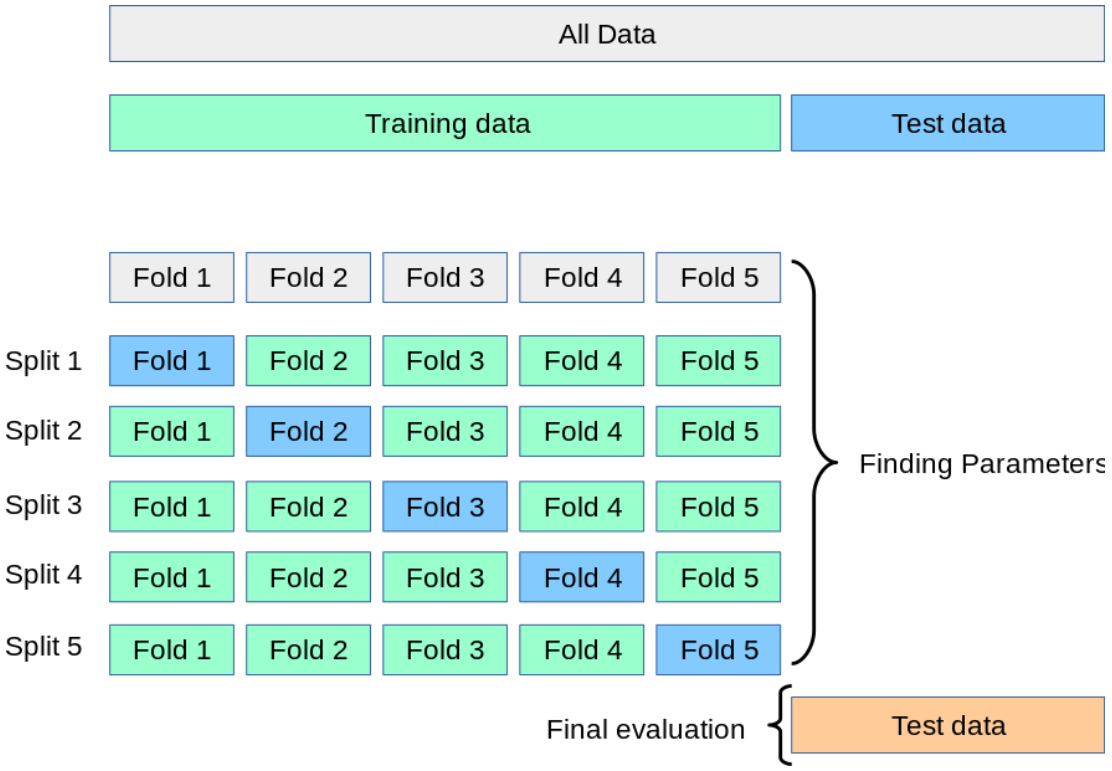
Hasil RMSE akan selalu lebih besar atau sama dengan MAE. Jika semua error memiliki magnitudo yang sama, maka RMSE = MAE. Karena tiap error dikuadratkan sebelum dirata-ratakan, RMSE memberikan bobot yang lebih terhadap nilai error yang tinggi, yang mana menyatakan bahwa RMSE akan lebih berguna jika error yang besar tersebut cenderung tidak diinginkan.

Contoh evaluasi regresi dengan Linear Regression:

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import
from sklearn.linear_model import LinearRegression
import numpy as np

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20, random_state=42)

mean_absolute_error, r2_score, mean_squared_error

model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("R^2 : ", r2_score(y_test, y_pred))
print("MAE :", mean_absolute_error(y_test,y_pred))
print("RMSE:",np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
R^2 :  0.6508427991759342
MAE : 0.07476031320105749
RMSE: 0.09761343652989583
```

c. Cross Validation (KFold)



K-Fold Validation memberikan hasil yang paling akurat dan efektif untuk diterapkan di data training yang berbeda di dataset yang sama.

Bagian gelap dari tabel 10-Fold Validation mewakilkan training set dan bagian terang dari tabel tersebut mewakilkan testing set.

Metode ini dapat digunakan untuk mengukur parameter algoritma yang berbeda, dan pemilihan beberapa model yang telah ditentukan sebelumnya, namun K-Fold Validation memiliki kekurangan yaitu estimasi performa memiliki sampel yang kecil. Biasanya CV K-fold digunakan karena dapat mengurangi waktu komputasi dengan tetap menjaga keakuratan estimasi.

Ada 6 jenis dari Cross Validation:

1. K-Fold cross-validation (yang dipelajari)
2. Stratified k-fold cross-validation
3. Leave P Out cross-validation
4. Leave One Out cross-validation
5. Time Series cross-validation
6. Shuffle Split cross-validation

```
#### CARA 1 : Menggunakan Metrics
from sklearn.datasets import load_breast_cancer
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score

# Input Dataset
data = load_breast_cancer(as_frame = True)
df = data.frame

# Pisahkan fitur (variabel) dependent and independent
X = df.iloc[:,:-1]
Y = df.iloc[:,-1]

# Implementasi k-fold cross-validation
k = 5
k_fold = KFold(n_splits = k, random_state = None)
Lr = LogisticRegression(solver = 'liblinear')
acc_scores = []

# Loop setiap split untuk mendapatkan akurasinya
for training_index, testing_index in k_fold.split(X):
    X_train, X_test = X.iloc[training_index,:],
                      X.iloc[testing_index,:]
    Y_train, Y_test = Y.iloc[training_index],
                      Y.iloc[testing_index]

# Masukkan data training ke dalam model
Lr.fit(X_train,Y_train)

# Prediksi nilai-nilai data training untuk data testing
Y_pred = Lr.predict(X_test)

# Kalkulasi nilai akurasi menggunakan sklearn.accuracy_score()
acc = accuracy_score(Y_pred, Y_test)
acc_scores.append(acc)

# Kalkulasi rata-rata nilai akurasi
mean_acc_score = sum(acc_scores) / k

print("Nilai akurasi setiap fold: ", acc_scores)
print("Rata-rata nilai akurasi: ", mean_acc_score)
```

OUTPUT:

```
Nilai akurasi setiap fold:
[0.9122807017543859, 0.9473684210526315,
0.9736842105263158, 0.9736842105263158, 0.9557522123893806]
Rata-rata nilai akurasi:  0.952553951249806
```

```
#### CARA 2 : Menggunakan Metrics
from sklearn.datasets import load_breast_cancer
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score

# Input Dataset
data = load_breast_cancer(as_frame = True)
df = data.frame

# Pisahkan fitur (variabel) dependent and independent
X = df.iloc[:,:-1]
Y = df.iloc[:,-1]

# Implementasi k-fold cross-validation
k = 5
k_fold = KFold(n_splits = k, random_state = None)
Lr = LogisticRegression(solver = 'liblinear')
Lr.fit(X,Y)

# Temukan nilai akurasi dengan menggunakan metode cross_val_score
acc_score = cross_val_score(Lr, X, Y, cv=k_fold)

# Temukan nilai akurasi dengan menggunakan metode cross_val_score
mean_acc_score = sum(score)/len(score)

print("Nilai akurasi setiap fold: ", acc_scores)
print("Rata-rata nilai akurasi: ", mean_acc_score)
```

OUTPUT:

```
Nilai akurasi setiap fold:
[0.9122807017543859, 0.9473684210526315, 0.9736842105263158,
0.9736842105263158, 0.9557522123893806]
Rata-rata nilai akurasi:  0.952553951249806
```

d. HyperParameter Tuning

Hyperparameter tuning adalah proses untuk menemukan nilai hyperparameter. Tujuannya adalah untuk menemukan nilai hyperparameter yang dapat menghasilkan model dengan performa yang paling baik.

Contoh yang paling mudah misalnya adalah menentukan nilai dari learning rate pada neural network. Ketika nilai learning rate terlalu kecil, model akan belajar sangat lambat, sedangkan learning rate yang besar bisa berakibat model kesulitan menemukan global optima. Mencari angka yang tepat ini perlu proses eksperimen, proses inilah yang disebut dengan *hyperparameter tuning*.

Ada tiga metode dari hyperparameters tuning, yaitu Grid-Search, Random Search dan Auto-Tuning (Bayesian Tuning. Namun untuk mempersingkat modul hanya akan dibahas Grid-Search.

```
from scipy.stats import loguniform
from pandas import read_csv
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import RandomizedSearchCV

# load dataset
url =
'https://raw.githubusercontent.com/jbrownlee/Datasets/master/sonar
.csv'
dataframe = read_csv(url, header=None)

# split into input and output elements
data = dataframe.values
X, y = data[:, :-1], data[:, -1]

# definisikan model
model = LogisticRegression()

# definisikan evaluation
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3,
random_state=1)

# definisikan ruang search (search space)
space = dict()
space['solver'] = ['newton-cg', 'lbfgs', 'liblinear']
space['penalty'] = ['none', 'l1', 'l2', 'elasticnet']
space['C'] = [1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1, 10, 100]

# definisikan search menggunakan Grid-Search
search = GridSearchCV(model, space, scoring='accuracy', n_jobs=-1,
cv=cv)

# eksekusi search
result = search.fit(X, y)

# simpulkan result
print('Best Score: %s' % result.best_score_)
print('Best Hyperparameters: %s' % result.best_params_)
```

```
Best Score:
0.7828571428571429

Best Hyperparameters:
{'C': 1, 'penalty': 'l2', 'solver': 'newton-cg'}
```

Referensi:

6 Metode Algoritma Supervised Learning dalam Data Science - Algoritma

Supervised and Unsupervised Learning in (Machine Learning) (simplilearn.com)

Module 3 - Supervised Learning - Part One Course | Cloud Academy

Perbedaan Supervised Learning and Unsupervised Learning | Information Communication Technology (uc.ac.id)

What is Supervised Learning? (techtarget.com)

Perbedaan Supervised Learning dan Unsupervised Learning - Algoritma

Algoritma Supervised Vs Unsupervised Learning (dqlab.id)

How Do Machines Learn? (boozallen.com)

9. Supervised Learning — Data Science 0.1 documentation (python-data-science.readthedocs.io)

Istilah penting di Neural Network dan Deep Learning (bagian 2) - Structilmy

Hyperparameter Optimization With Random Search and Grid Search (machinelearningmastery.com)

6 Metode Algoritma Supervised Learning dalam Data Science

Algoritma Supervised Vs Unsupervised Learning, Cari Tahu Bedanya!

What is Supervised Learning? | IBM

What is Supervised Learning?

Supervised and Unsupervised Learning in (Machine Learning)

Perbedaan Supervised Learning and Unsupervised Learning | Information Communication Technology