**Evaluasi Model Machine Learning: Train/Test Split**

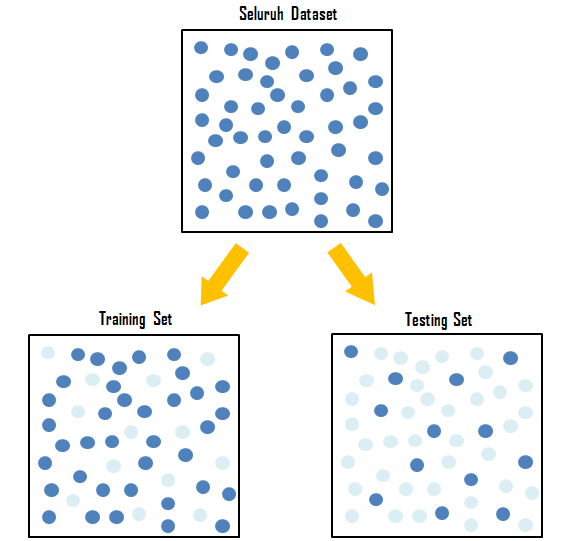
Train/test split adalah salah satu metode yang dapat digunakan untuk mengevaluasi performa model machine learning. Metode evaluasi model ini membagi dataset menjadi dua bagian yakni bagian yang digunakan untuk training data dan untuk testing data dengan proporsi tertentu.

**Train data** digunakan untuk *fit* model machine learning, sedangkan **test data** digunakan untuk mengevaluasi hasil *fit* model tersebut.

Train/test split dapat digunakan untuk problem regresi maupun klasifikasi.

**Evaluasi Model Train/test Split**

Evaluasi model machine learning dengan train/test split cocok digunakan untuk dataset yang berukuran besar. Seperti yang kita ketahui, train/test split membagi dataset menjadi *train set* dan *test set*,  atau dengan kata lain, data yang digunakan untuk proses *training* dan *testing* merupakan kumpulan data yang berbeda.



Metode train/test split ini akan memberikan hasil prediksi yang lebih akurat untuk *new data* atau data yang belum pernah di-*train*.

Karena data testing tidak digunakan untuk melatih model, maka model tidak mengetahui *outcome* dari data tersebut. Ini yang disebut dengan *out-of-sample testing*.

Suatu model dikatakan bagus jika memiliki akurasi yang tinggi atau bagus untuk data *out-of-sample*, karena tujuan utama dibuatnya sebuah model tentunya adalah untuk memprediksi dengan benar data yang belum diketahui *outcome*-nya.

**Train/test split di Python**

Python memiliki *library* yang dapat mengimplementasikan train/test split dengan mudah yaitu [Scikit-Learn](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html). Untuk menggunakannya, kita perlu mengimport Scikit-Learn terlebih dahulu, kemudian setelah itu kita dapat menggunakan fungsi train\_test\_split().

from sklearn.model\_selection import train\_test\_split

Setelah itu kita definisikan data yang menjadi *source*-nya (X)  dan juga data targetnya (y). Misalnya data *source*-nya adalah semua kolom kecuali kolom terakhir yang di ujung sebelah kanan dataset **df**, sedangkan data targetnya adalah kolom paling ujung kanan dengan nama kolom “Class”, maka kita dapat mendefinisikannya seperti di bawah ini.

X = df.iloc[:, :-1]

y = df['Class']

Setelah didefinisikan, kita dapat langsung mengimplementasikan train/test split.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

* **X\_train**: Untuk menampung *data source* yang akan dilatih.
* **X\_test**: Untuk menampung *data target* yang akan dilatih.
* **y\_train**: Untuk menampung *data source* yang akan digunakan untuk testing.
* **y\_test**: Untuk menampung *data target* yang akan digunakan untuk testing.

X dan y adalah nama variabel yang digunakan saat mendefinisikan *data source* dan *data target*. Parameter test\_size digunakan untuk mendefinisikan ukuran data testing. Dalam contoh di atas, test\_size=0.2 berarti data yang digunakan sebagai data testing adalah sebesar 20% dari keseluruhan dataset.

Perlu diketahui bahwa metode ini akan membagi *train set* dan *test set* secara random atau acak. Jadi, jika kita mengulang proses *running*, maka tentunya hasil yang didapat akan berubah-ubah. Untuk mengatasinya, kita dapat menggunakan parameter random\_state seperti contoh di bawah ini.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=3)

Angka yang didefinisikan pada parameter random\_state boleh angka berapa saja yang berupa *integer*.

**Menerapkan Hasil Train/test Split Dalam Pemodelan**

Selanjutnya, kita dapat menggunakannya untuk pemodelan dengan algoritma tertentu misalnya disini menggunakan Linear Regression seperti contoh berikut.

from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_absolute\_error  
  
# fit the model  
model = LinearRegression()  
model.fit(X\_train, y\_train)  
  
# memprediksi model dengan test set  
pred = model.predict(X\_test)  
  
# mengevaluasi model dengan MAE  
mae = mean\_absolute\_error(y\_test, pred)  
print("Mean Absolute Error (MAE): ", mae)

LinearRegression() adalah fungsi untuk mengimplementasikan algoritma Linear Regression di Python. Fungsi fit() digunakan untuk melatih model, predict() digunakan untuk memprediksi hasil model, sedangkan mean\_absolute\_error() adalah fungsi untuk men-generate nilai Mean Absolute Error (MAE) yang merupakan salah satu metode untuk mengukur seberapa baik hasil prediksi untuk kasus regresi.

Mean Absolute Error (MAE) adalah dua diantara banyak metode untuk mengukur tingkat keakuratan suatu model peramalan. Nilai MAE merepresentasikan rata – rata kesalahan (error) absolut antara hasil peramalan dengan nilai sebenarnya.

MAE menyatakan kesalahan prediksi model rata-rata dalam unit variabel yang diminati. Kedua metrik dapat berkisar dari 0 hingga ∞ dan tidak berbeda dengan arah kesalahan. Mereka adalah skor berorientasi negatif, yang berarti nilai yang lebih rendah lebih baik.