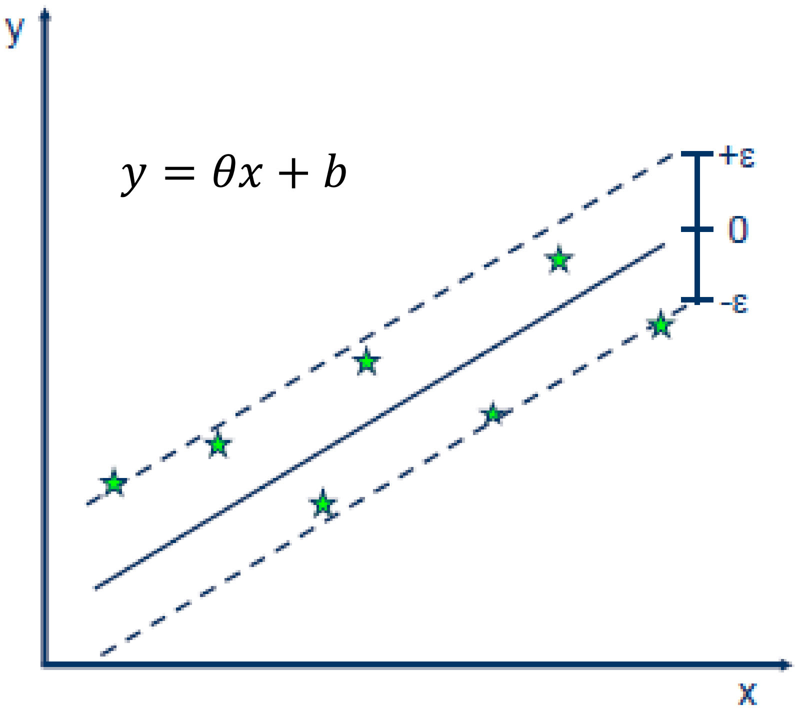
Regression model

Support Vector Regression (SVR)

Support vector regression



- Solution:

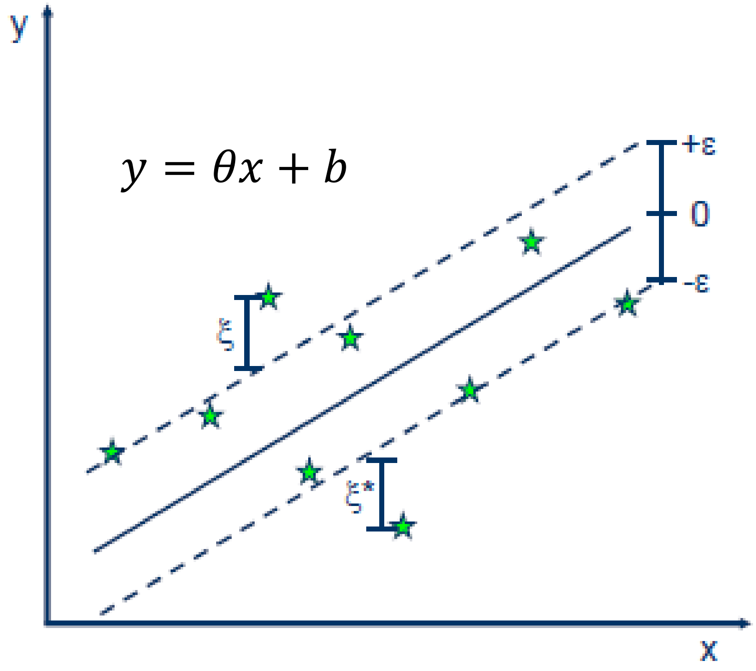
$$min\left(\frac{1}{2}\|\theta\|^2\right)$$

- Constraints:

$$y_i - \theta x_i - b \leq \varepsilon$$
$$\theta x_i + b - y_i \leq \varepsilon$$

- SVR is a regression model with **constraints**
  - Model estimation is performed inside the constraint space
    - Reduce outliers and noise effects
- Constraints
  - a margin of tolerance (ε)
- Cost of errors inside ε area
  - zero for all points that are inside the band.

Support vector regression



- Minimize:

$$min\left(\frac{1}{2}\|E\|^2 + C\sum_{i=1}^{N}\xi_i + \xi_i^*\right)$$

$$10^{-6} \leq C \leq 10^{6}$$

- Constraints:

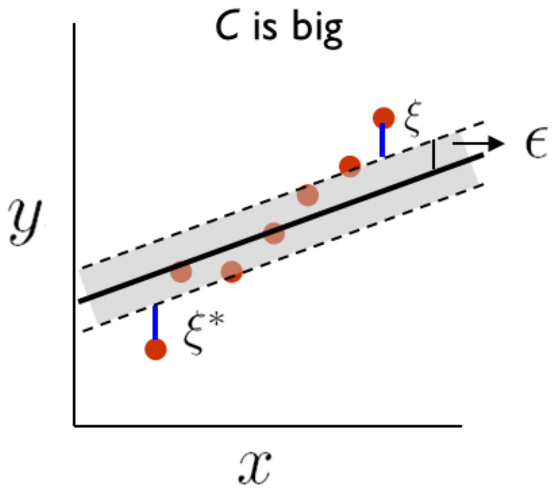
$$y_i - \theta x_i - b \leq \ \varepsilon + \xi_i$$

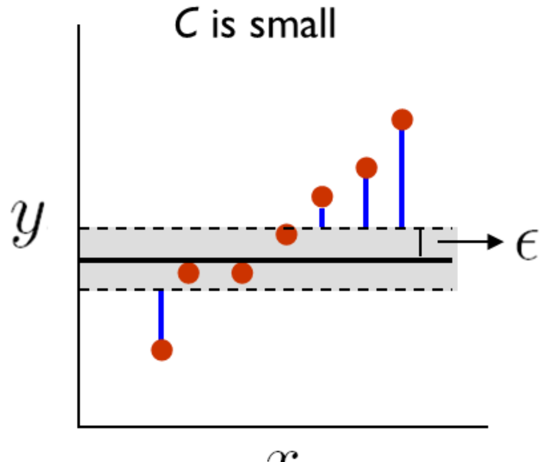
$$\theta x_i + b - y_i \leq \varepsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \geq 0$$

- SVR parameter estimation
  - #1: optimized parameters
  - #2: constraint effects
    - C: how much you want to avoid error?
    - ε: How large is constraint space?
    - ξ: (non-negative) Slag variables
      - allow regression errors to exist up to the value of ξ

Support vector regression





$$min\left(\frac{1}{2}\|E\|^2 + C\sum_{i=1}^{N}\xi_i + \xi_i^*\right)$$

- For **Large** values of **C**,
  - the optimization will choose a **smaller-margin (ξ) hyperplane**
  - if that hyperplane does a better job of getting all the training points classified correctly.
  - **C too large** -> Be careful for **Overfitting** model

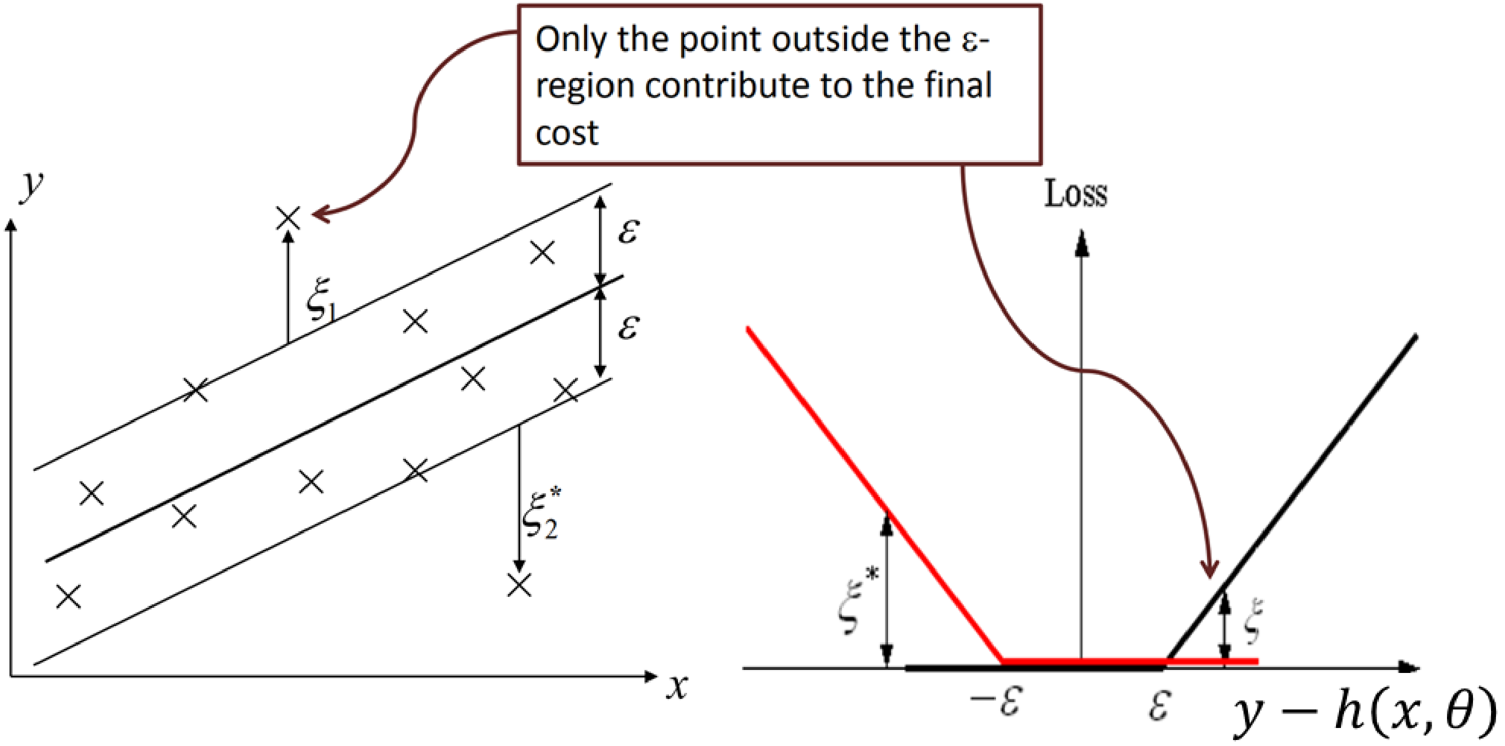
- For **Small** value of **C**
  - will cause the optimizer to look for a **larger-margin (ξ) hyperplane** even if that hyperplane causes error more points.
  - C too small -> Be careful for **Error** model

- Usually a **good guess** is $0.01 \le C \le 100$

https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel

Support vector regression

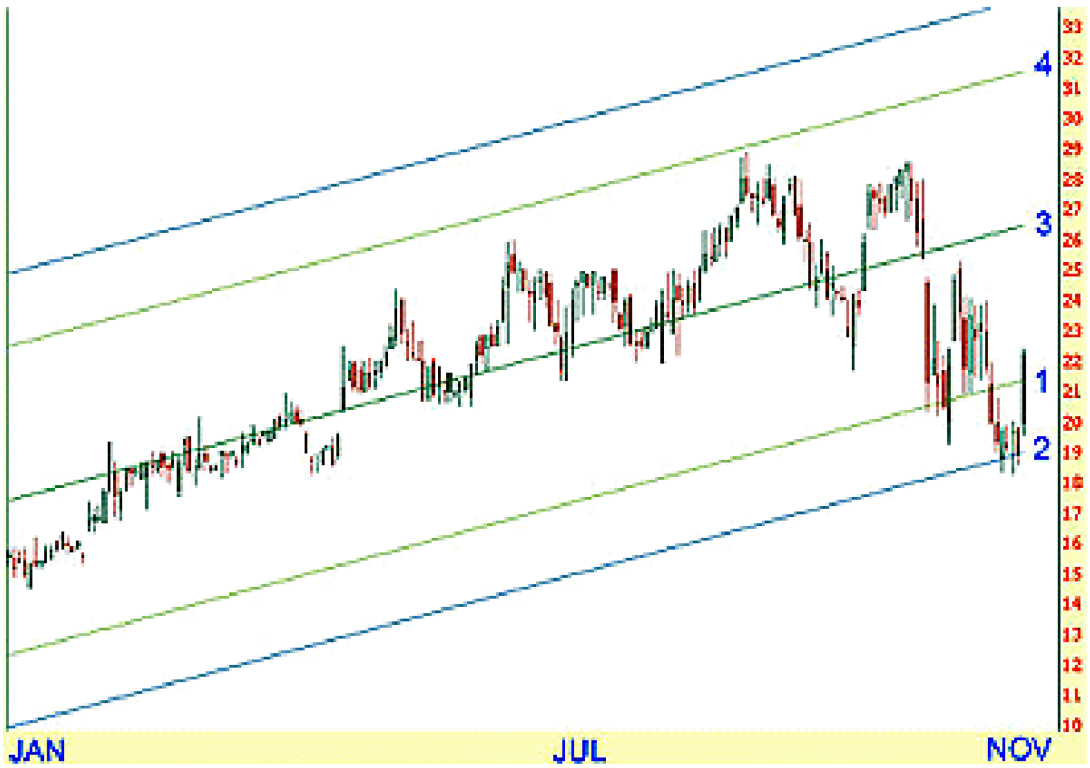
Assume linear parameterization $h(x, \theta) = \theta x + b$



$$L_\varepsilon\big(y, h(x,\theta)\big) = max(|y - h(x,\theta) - \varepsilon, 0|)$$

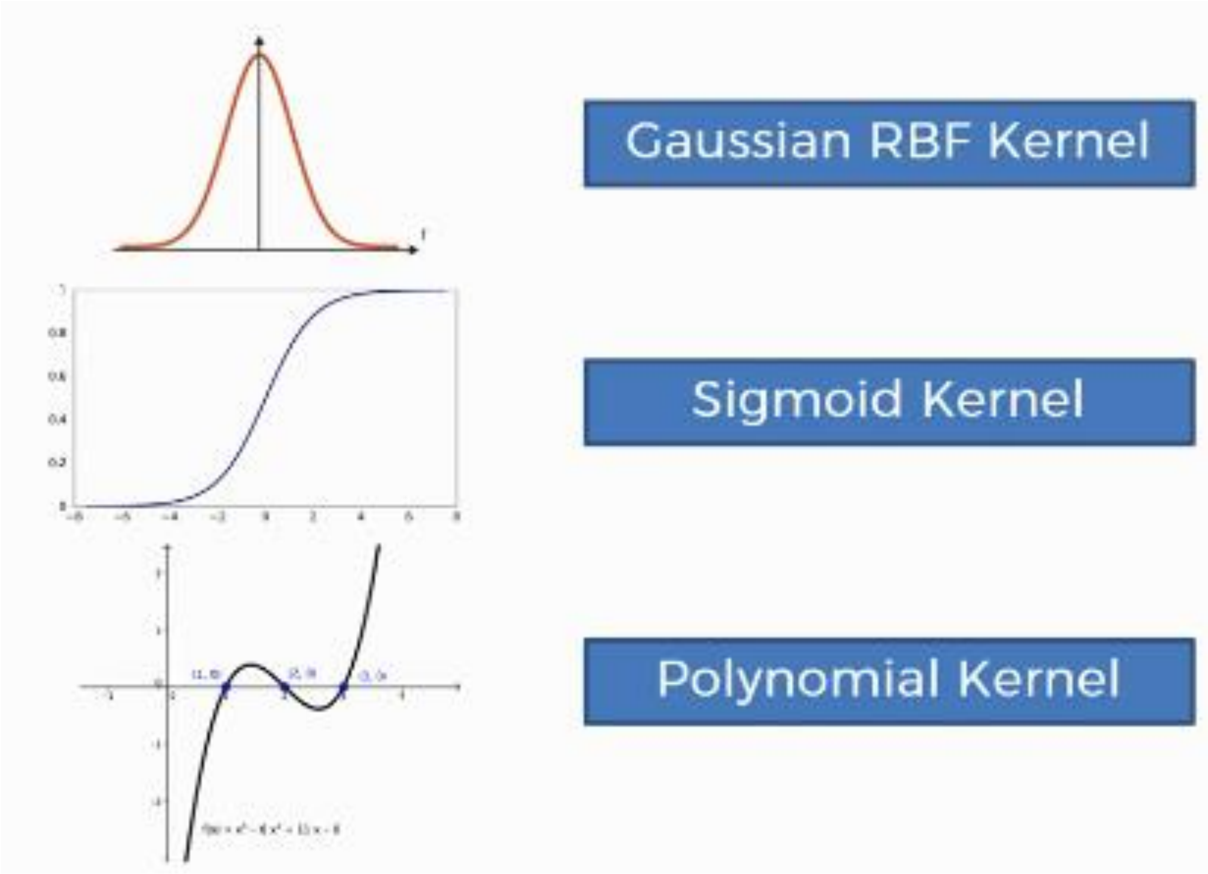
Support vector regression

- Stock price prediction



SVR: kernels

SVR: kernels



Gaussian kernel

$$k(x_i, x_j) = exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

Gaussian radial basis function (RBF)

$$k(x_i, x_j) = exp\left(-\gamma\|x_i - x_j\|^2\right)$$

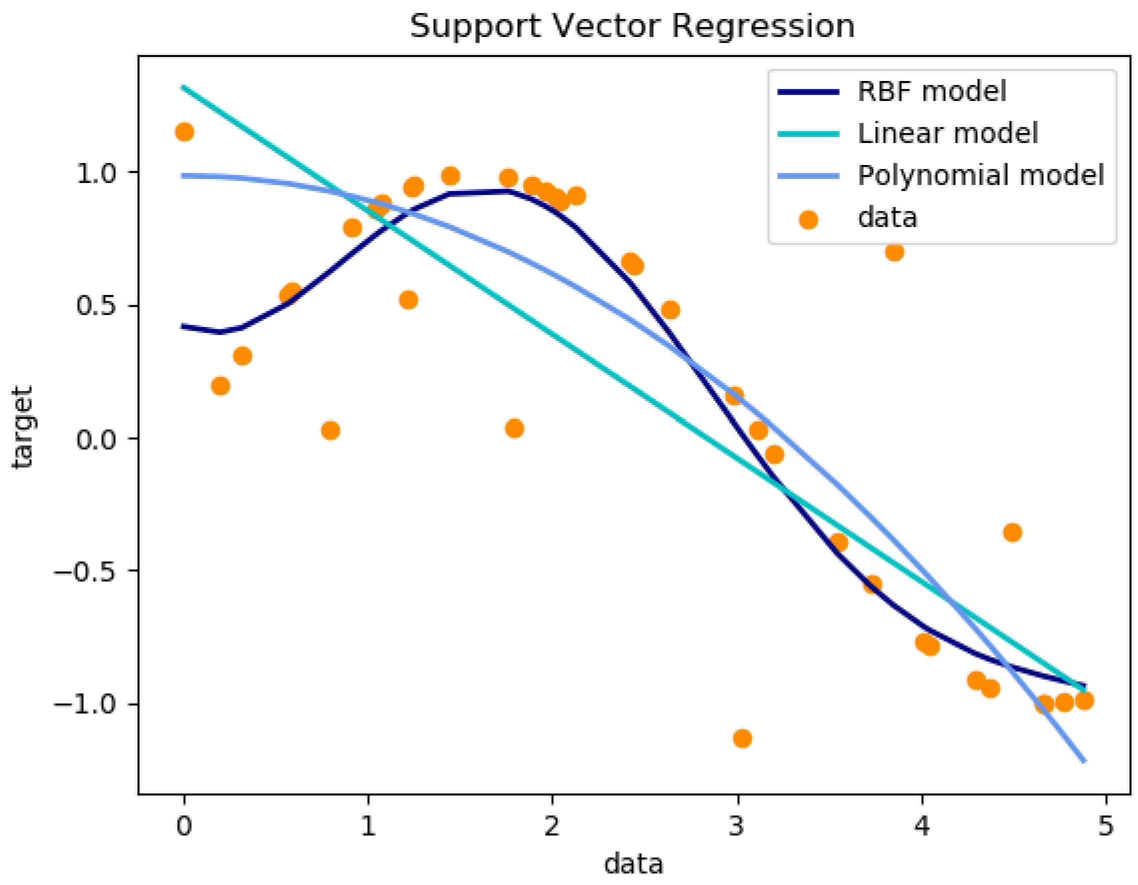
Polynomial kernel

$$k(x_i, x_j) = (x_i x_j)^d$$

Sigmoid kernel

$$k(x_i, x_j) = tanh(\alpha {x_i}^T x_j + c)$$

SVR: kernels



Gaussian kernel

$$k(x_i, x_j) = exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

Gaussian radial basis function (RBF)

$$k(x_i, x_j) = exp\left(-\gamma\|x_i - x_j\|^2\right)$$

Polynomial kernel

$$k(x_i, x_j) = (x_i x_j)^d$$

Sigmoid kernel

$$k(x_i, x_j) = tanh(\alpha {x_i}^T x_j + c)$$

Model Preparation

Example svr in sklearn.svm

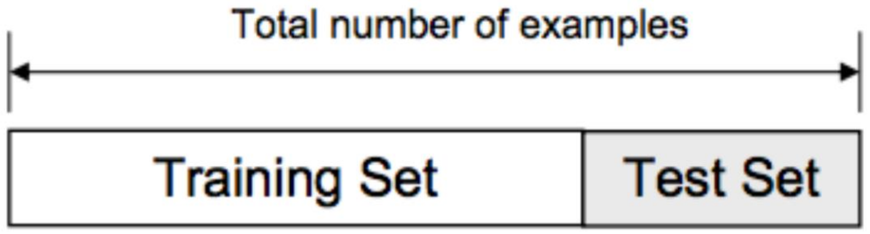
```
###############################################################################
# Fit regression model
from sklearn.svm import SVR

svr_rbf = SVR(kernel='rbf', C=1e4, gamma=0.1)
svr_lin = SVR(kernel='linear', C=1e4)
svr_poly = SVR(kernel='poly', C=1e4, degree=2)
y_rbf = svr_rbf.fit(X, y).predict(X)
y_lin = svr_lin.fit(X, y).predict(X)
y_poly = svr_poly.fit(X, y).predict(X)

###############################################################################
# look at the results
import pylab as pl
pl.scatter(X, y, c='k', label='data')
pl.hold('on')
pl.plot(X, y_rbf, c='g', label='RBF model')
pl.plot(X, y_lin, c='r', label='Linear model')
pl.plot(X, y_poly, c='b', label='Polynomial model')
pl.xlabel('data')
pl.ylabel('target')
pl.title('Support Vector Regression')
pl.legend()
pl.show()
```

$$X = X_{train} = X_{test}$$

Train/test split

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3)
```

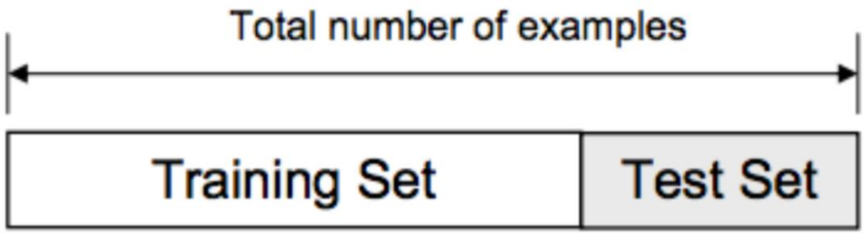
- ❑ Qantitiy Split (ปริมาณ train/test data)
  - Partition Dataset into
    - Model tolerance evaluation: XTest
      - Test_size = 0.3 = 30%
    - Model selection: using XTrain
      - Traing_size = 0.7 = 70% Training
        - Evaluate on various model learning parameters



$X_{train}\ for\ fitting\ model$
$X_{test}\ for\ prediction\ test$

Train/test splitting

- ❑ Selection Pattern
  - ▪ Ramdom (default: np.random() : pseudo random)
    - ✓ Random_state = initial seed
      - ○ For repeated sequence of random number
      - ○ default seed = current system time
    - ✓ Mathematical function of seed input
      - ○ Np.random() -> PCG64(seed)

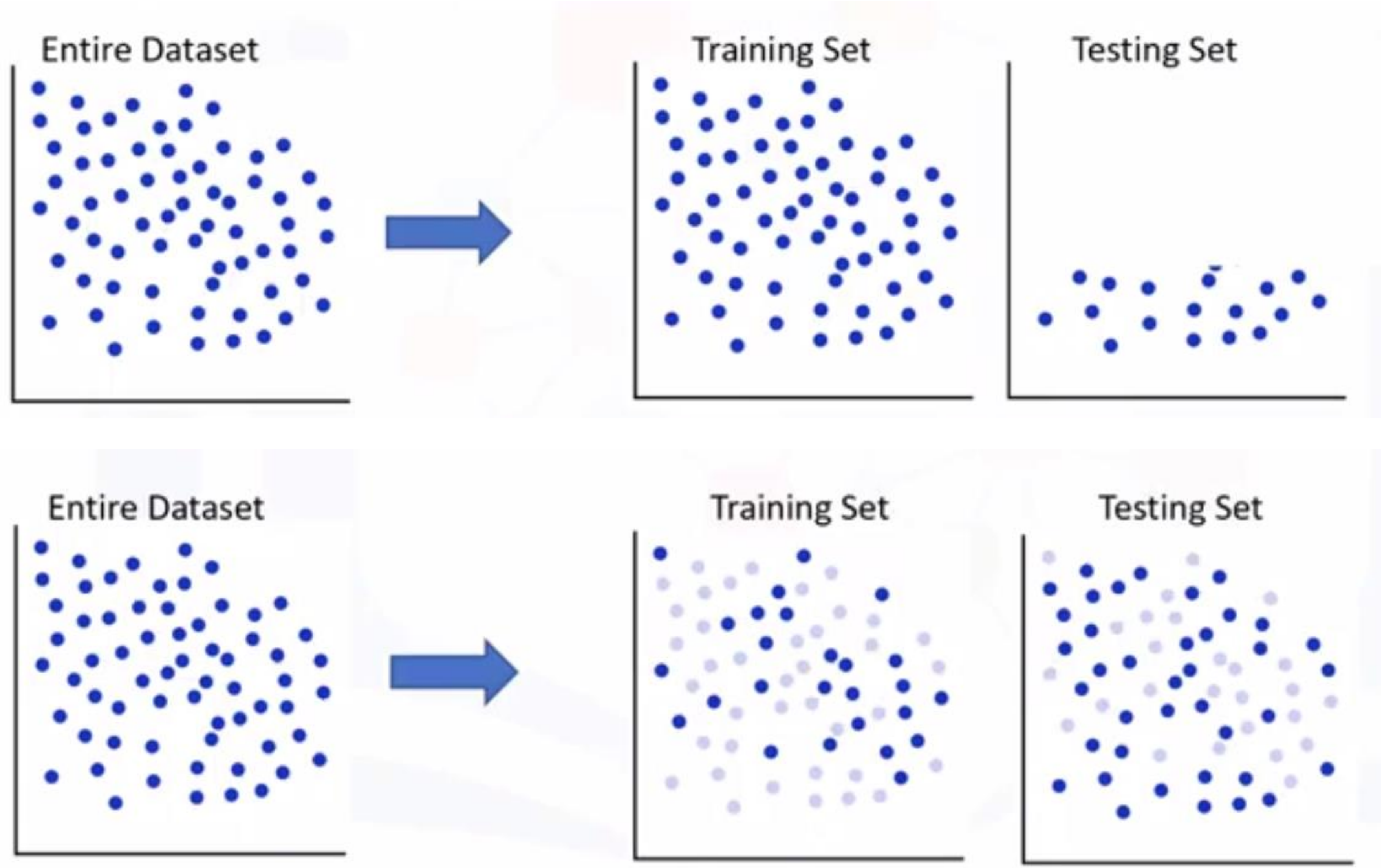


X_{train} *for fitting model*
X_{test} *for prediction test*

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.33, random_state=42)
```

https://machinelearningmastery.com/how-to-generate-random-numbers-in-python/

Equally distributed Tran/test



Random

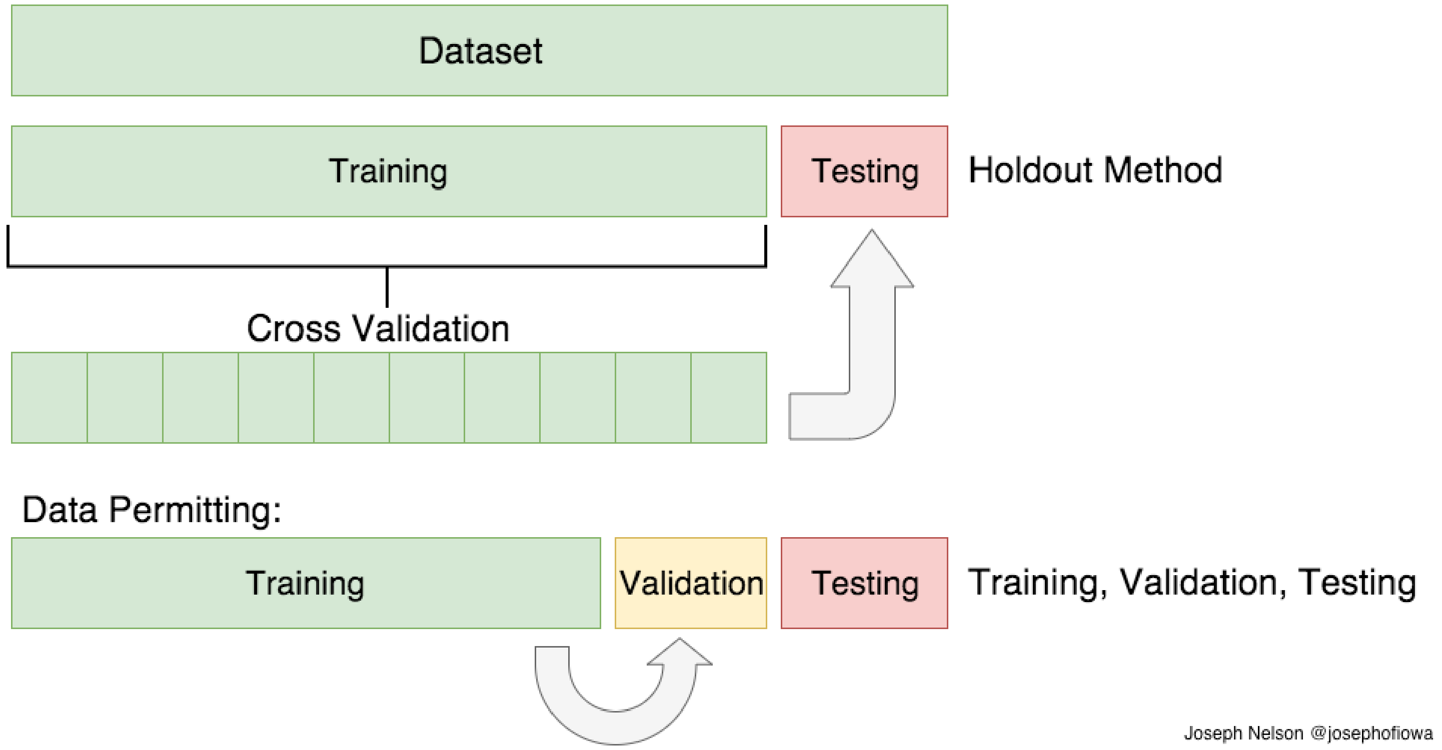
Shuffle

Stratify=y

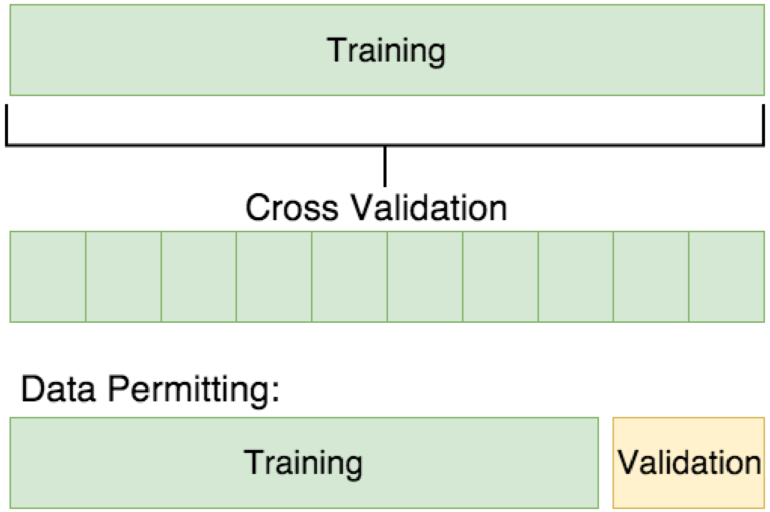
Model Validation

Cross validation (CV)

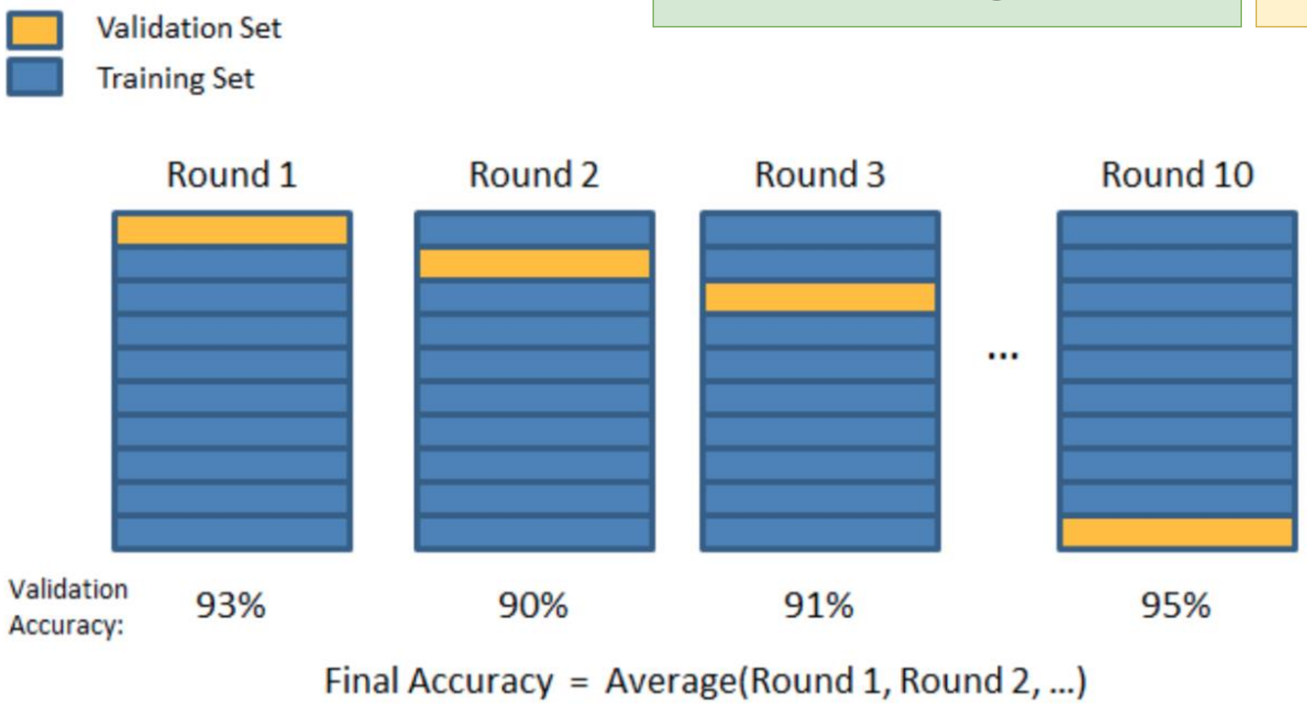
- Partition Dataset into
  - Cross Validation (CV)
    - K-Fold CV
    - Leave one out (LOOCV)
    - Shuffle Split CV



K-fold cv



- K-Fold Cross Validation
  - Partition Training into k subsets
    - For i=1:k iteration
      - Select ith subset as test data
      - k-1 subsets are used to train



https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

K-fold cv

- K-Fold Cross Validation
  - Partition Training into k subsets
    - For i=1:k iteration
      - Select ith subset as test data
      - k-1 subsets are used to train

```
from sklearn import model_selection
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.33, random_state=42)
```

```
seed = 7
kfold = model_selection.KFold(n_splits=10, Shuffle = True, random_state=seed)
score = model_selection.cross_val_score(model, Xtrain, Ytrain, cv=kfold)
```

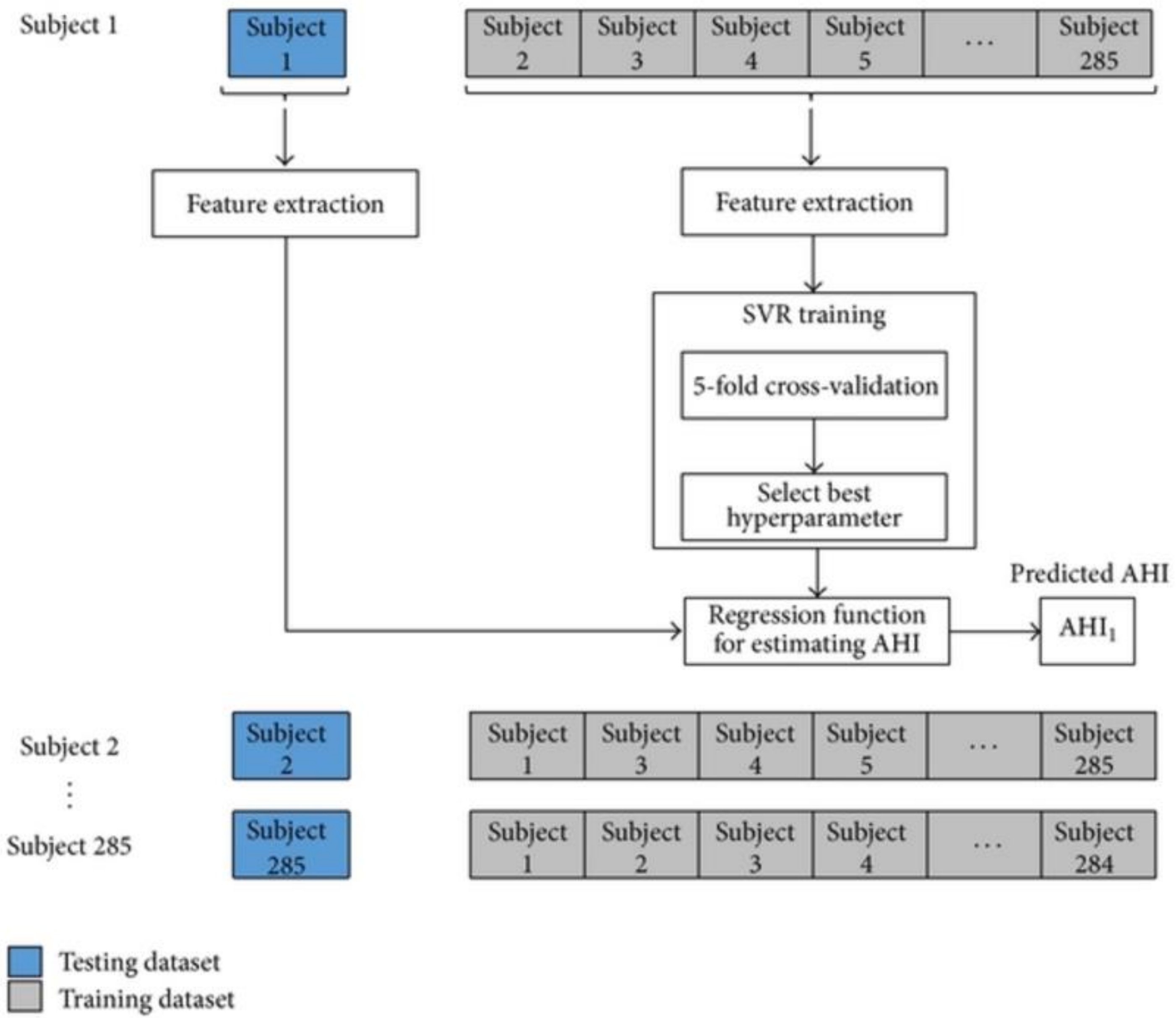
Note: Shuffle = True เพื่อ random data before K-Fold partition

score = array of k-accuracy
score.mean() = Average Accuracy
score.std() = Std Accuracy

https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

Loocv

- Leave-One-Out Cross Validation
  - For i=1:N iteration
    - Select ith record as test data
    - N-1 subsets are used to train



Loocv

- Leave-One-Out Cross Validation
  - For i=1:N iteration
    - Select ith subset as test data
    - N-1 subsets are used to train

```
from sklearn import model_selection

from sklearn.model_selection import train_test_split

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

loocv = model_selection.LeaveOneOut()

score = model_selection.cross_val_score(model, Xtrain, Ytrain, cv=loocv)
```

score = array of k-accuracy
score.mean() = Average Accuracy
score.std() = Std Accuracy

https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

Model evaluation

Regression metrics

- **Measure regression performance**
  - Mean absolute error (MAE) -> **ค่าสัมบูรณ์ความผิดพลาดเฉลี่ย** (matrics. mean_abosolute_error())
  - Mean squared error (MSE) -> **ค่าความผิดพลาดกำลังสองเฉลี่ย** (matrics. mean_squared_error())
  - Explained variance score -> **ค่าความเกาะกลุ่มของผลการทำนายกับข้อมูลจริง** (**ดูในเชิงสถิติ**)
  - R^2 score -> how **well** future samples a

$$explained\_variance(y, \hat{y}) = 1 - \frac{Var\{y - \hat{y}\}}{Var\{y\}}$$

https://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics