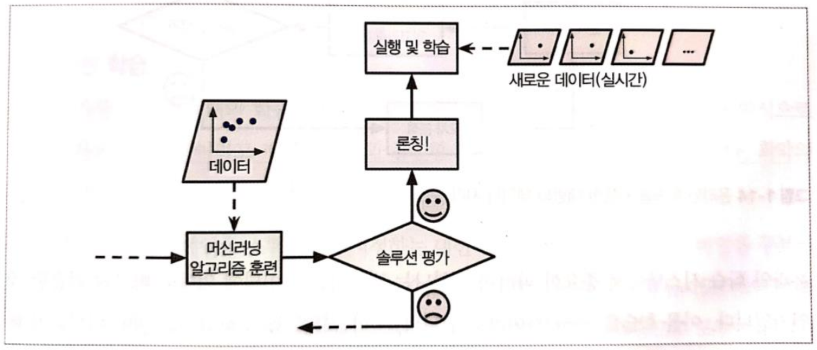
# On-line Learning

# What is on-line learning?

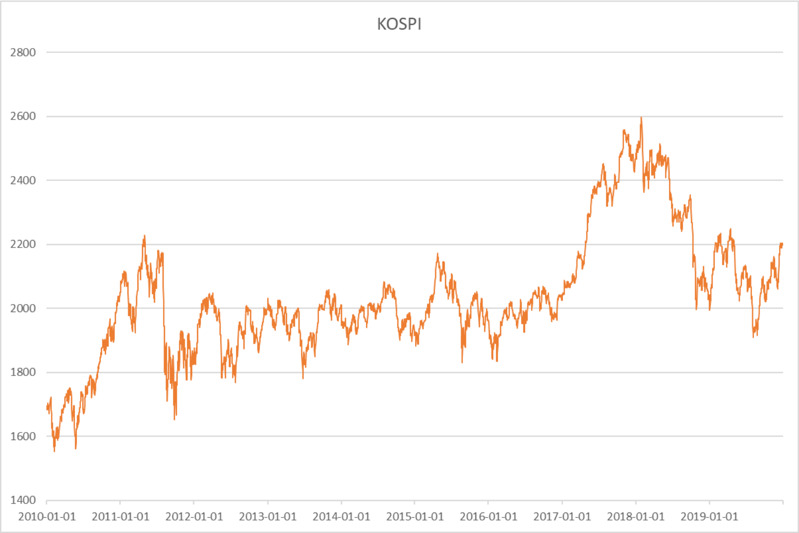
- On-line learning vs. batch learning
  - Data available in a sequential order
  - Not by learning on the entire training data set at once
- Learning method
  - Dynamically adapt to new patterns




출처: Aurelien Geron, “Hands-On Machine Learning with Scikit-Learn, Keras, and Ternsorflow

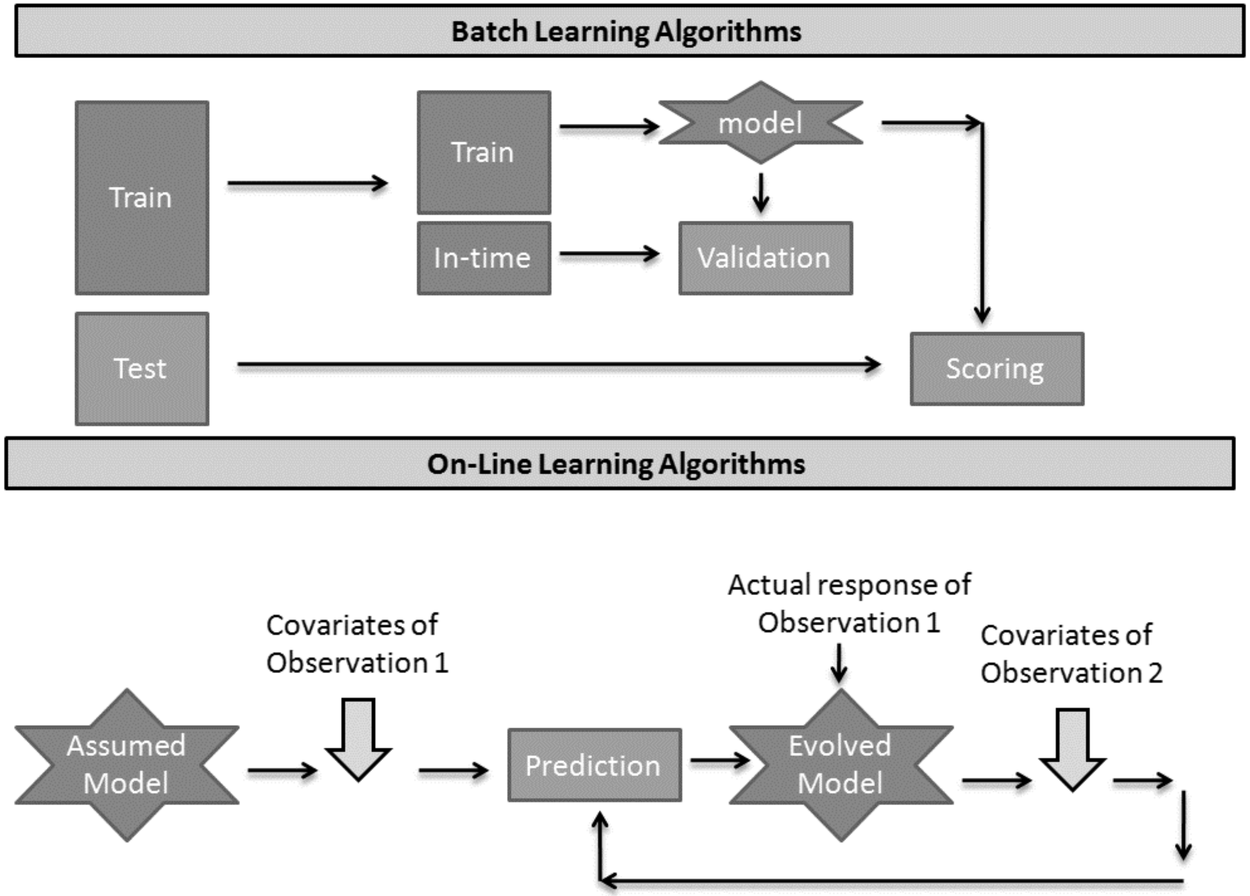
# Introduction

- Time series prediction models require new learning as data changes over time.
- For typical batch learning methods, learning is carried out using all the data so far for new learning.
- Batch learning methods require a long time and a large amount of storage space for learning, as they require all the data so far.



# Introduction

- Batch learning vs Online learning



# Logistic Regression example

```python
def calc_prob(b, x):
    dot = np.array(b).dot(np.array(x))
    if dot >= 7.6:
        prob = 1.0
    else:
        prob = np.exp(dot) / (1 + np.exp(dot))
    return prob

def calc_err(p, y):
    err = np.log2(np.power(p, y) * np.power((1-p),(1-y)))
    if err <= 1e-10:
        err = 1e+10
    return abs(err)

def relDiff(a, b):
    diff = abs(a - b) / (abs(a) + abs(b))
    return diff

def update(b, x, lr, y, p):
    for i in range(len(b)):
        b[i] +=  lr * x[i] * (y-p)
    return b

def OLR(x, y, threshold, epsilon, lr_i, delta, instances, B, W, fn, theta):
    instances.append((x, y)) # add x, y to the set of observed instances.
    H = sum(W) # w_i returns the weight value associated with parameter i
    rand_idx = int(np.random.choice(range(len(W)), 1, p = [i / H for i in W]))
    b = B[rand_idx].copy() # sample a random b_r, r is it's index in B
    loss_i = 0 # initial loss value
    loss_est = 1e+10 # infinite
    t = 0 # initial trial value
    prob = calc_prob(b, x) # calculate probability of potential

    if calc_err(prob, y) >= threshold: # The prediction makes mistake
        lr = lr_i / (1 + np.exp(1)/delta)
        b_est = b.copy() # initial a new parameter
        W[rand_idx] *= fn # update weight value of b_r
        err = 0
        if len(B) < theta:
            while relDiff(loss_est, loss_i) > epsilon:
                for (x_i, y_i) in instances:
                    p = calc_prob(b_est, x_i)
                    b_est = update(b, x_i, lr, y_i, p)
                    err += calc_err(p, y_i)
                loss_est = loss_i
                loss_i = -err
                t += 1
            B.append(b_est)
            W.append(1)

    return prob, B, W, instances
```

**Algorithm 3** OLR$(x, y, \epsilon, \varepsilon, \eta_0, \delta, \Omega, \mathcal{B}, W, \omega, \theta)$

**Input**:

- $x, y$ : Input vector $x$ and its true label $y$.
- $\epsilon$ : Threshold value $\epsilon \in \mathbb{R}, \epsilon > 0$
- $\varepsilon$ : Minimum relative error improvement $\varepsilon \in \mathbb{R}, \varepsilon > 0$
- $\eta_0$ : Initial learning rate $\eta_0 \in \mathbb{R}, \eta_0 > 0$
- $\delta$ : Annealing rate $\delta \in \mathbb{R}, \delta > 0$
- $\Omega$ : Set of observed instances
- $\mathcal{B}$ : Set of parameter vectors, $\mathcal{B} = \{\beta_1, \beta_2, \ldots, \beta_{|\mathcal{B}|}\}$
- $W$ : Set of associated weight of $\mathcal{B}$, $W = \{w_1, w_2, \ldots, w_{|\mathcal{B}|}\}$
- $\omega$ : A reduction function $\omega = e^{-\alpha}, \alpha > 0$
- $\theta$ : Maximum size of $\mathcal{B}$, $\theta \in \mathbb{R}, \theta > 0$

**Output**: $p, \mathcal{B}, W, \Omega$

$\Omega \leftarrow (x, y)$ /* add $x$, $y$ to the set of observed instances */
$H \leftarrow \sum_{i=1}^{|\mathcal{B}|} w_i$ /* $w_i$ returns the weight value associated with parameter $i$ */
$\beta_r \leftarrow \text{sampling}\left(\frac{w_1}{H}, \frac{w_2}{H}, \ldots, \frac{w_{|\mathcal{B}|}}{H}\right)$ /* sample a random $\beta_r$, $r$ is it's index in $\mathcal{B}$ */
$\ell \leftarrow 0$ /* initiate loss value */
$\hat{\ell} \leftarrow \infty$
$t \leftarrow 0$ /* initiate trial value */
$p \leftarrow \frac{e^{(\beta_r^T \cdot x_r)}}{1+e^{(\beta_r^T \cdot x_r)}}$ /* Calculate probability of potential */
**IF** $abs\left(\log_2\left(p^y(1-p)^{(1-y)}\right)\right) \geq \epsilon$ **THEN** /* The prediction makes mistake */
  $\eta_e \leftarrow \frac{\eta_0}{1+e/\delta}$
  $\hat{\beta} \leftarrow \beta_r$ /* initiate a new parameter */
  $w_r \leftarrow w_r \times \omega$ /* update weight value of $\beta_r$ */
  **IF** $|\mathcal{B}| < \theta$ **THEN**
    **WHILE** relDiff$(\hat{\ell}, \ell) > \varepsilon$ /* Define relDiff$(a, b) = \frac{abs(a-b)}{abs(a)+abs(b)}$ */
      **FOR** $i \leftarrow 1$ **TO** $|\Omega|$ **DO**
        $p_i = \frac{e^{\hat{\beta} \cdot x_i}}{1+e^{(\hat{\beta} \cdot x_i)}}$
        $\hat{\beta} \leftarrow \hat{\beta} + \eta_e x_i (\mathrm{I}(y_i = 1) - p_i)$
        /* The indicator function $\mathrm{I}(a = 1)$ returns 1 if $a = 1$, otherwise 0 */
      **ENDFOR**
      $\hat{\ell} \leftarrow \ell$
      $\ell \leftarrow -\sum_{i \leq |\Omega|} \log\left(p_i^{y_i}(1-p_i)^{(1-y_i)}\right)$
      $t \leftarrow t + 1$
    **ENDWHILE**
    $\mathcal{B} \leftarrow \hat{\beta}$
    $W \leftarrow 1$
  **ENDIF**
**ENDIF**
**RETURN** $p, \mathcal{B}, W, \Omega$

# Logistic Regression example

- Data – Telecom-CostomerChunk.csv (7032 rows, 16 columns)

```python
# experiments
x = data.iloc[:, :-1]
y = data.iloc[:, -1]
B = [np.zeros_like(x.iloc[0].to_numpy())]
W = [1]
lr_i = 0.01
instances = []
threshold = 2
epsilon = 0.1
delta = 2
fn = np.exp(-0.5)
theta = 800

probs = list()
for i in range(x.index.size):
    p, B, W, instances = OLR(x.iloc[i].to_numpy(), y.iloc[i], threshold, epsilon,
                             lr_i, delta, instances, B, W, fn, theta)
    probs.append(p)
```

| | |
|---|---|
| 0 | 0 |
| 0 | 0 |
| 0 | -0.10253 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | -0.10253 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | -0.0118343 |
| 0 | -0.000130736 |

parameter vector updated
(1 step)

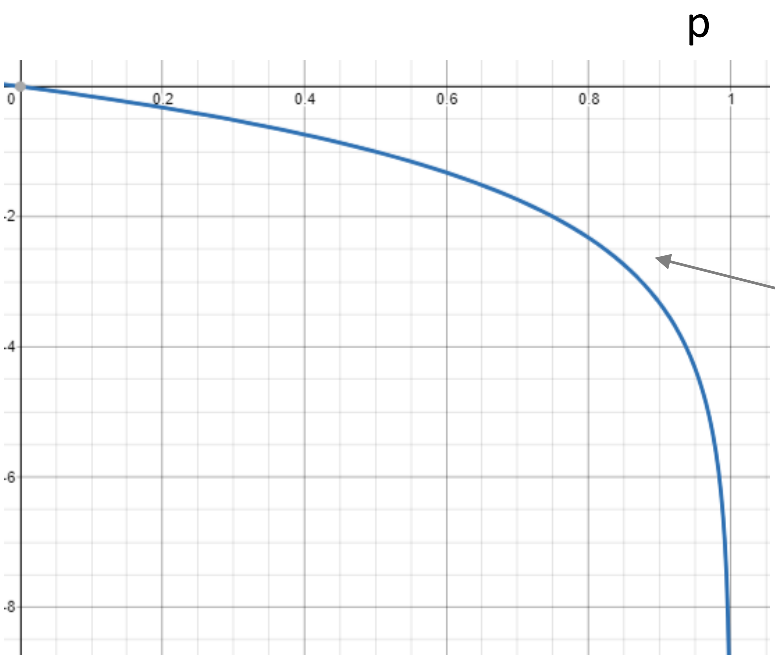
```
[0.6065306597126334, 1]
```

parameter weight vector
updated (1 step)

# Logistic Regression example

- Compare to TLR (Traditional Logistic Regression)
  - OLR acc : 0.7436
  - TLR acc : 0.7996

- The greater theta, the better the acc
  - theta 200 : 0.7463
  - theta 500 : 0.7571
  - theta 800 : 0.7651




$$abs(log_2(p^y(1-p)^{(1-y)}))$$

when y = 0, the err value according to p

threshold = 2 means when y = 0 and p >= 0.7
(y = 1 , p <= 0.25)

# LSTM example

**Algorithm** OLL

**INPUT :**

- x, y : Input vector x and true value y
- model : pretrained LSTM model
- $m$ : margin of error
- $\eta$ : initial learning rate
- $n$ : online learning epochs

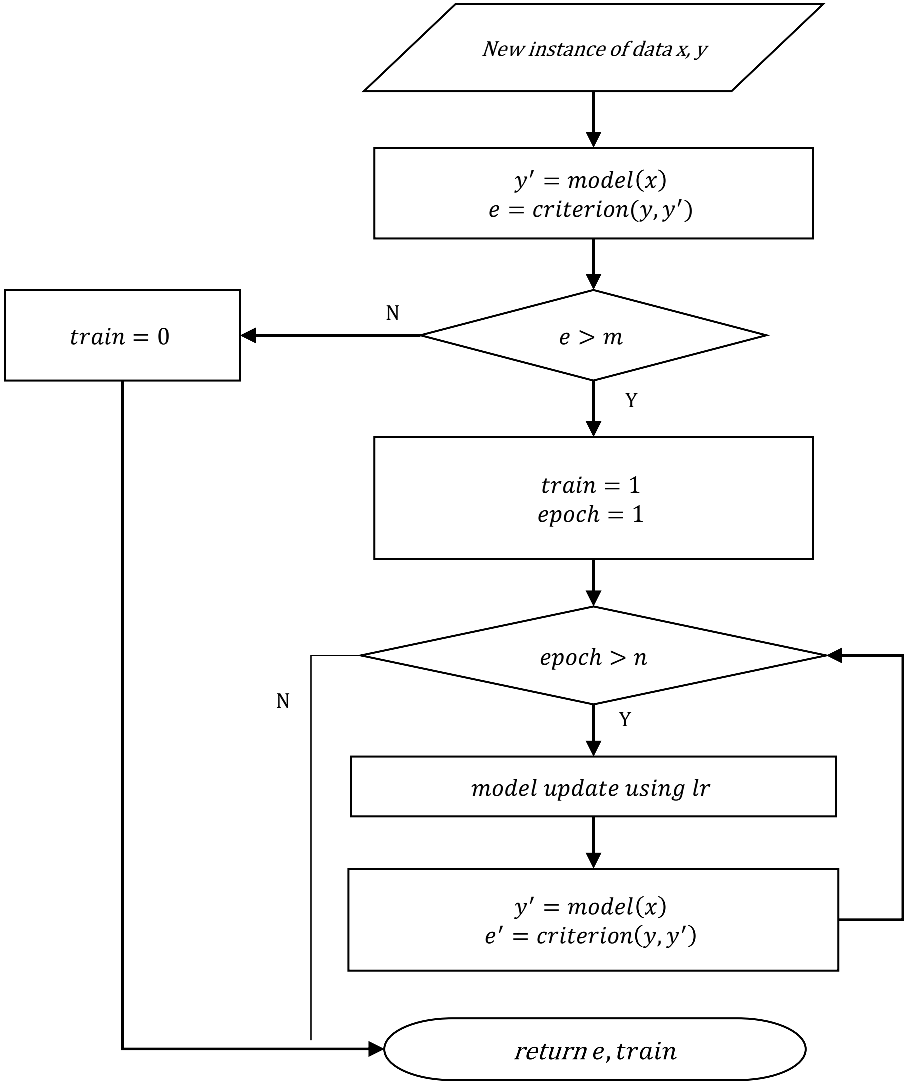
**OUTPUT** : *e, e', train*

*y' = model(x)*
*e = mae(y, y')*

**IF** *e > m* **THEN**
    *train = 1*
    *epoch = 0*

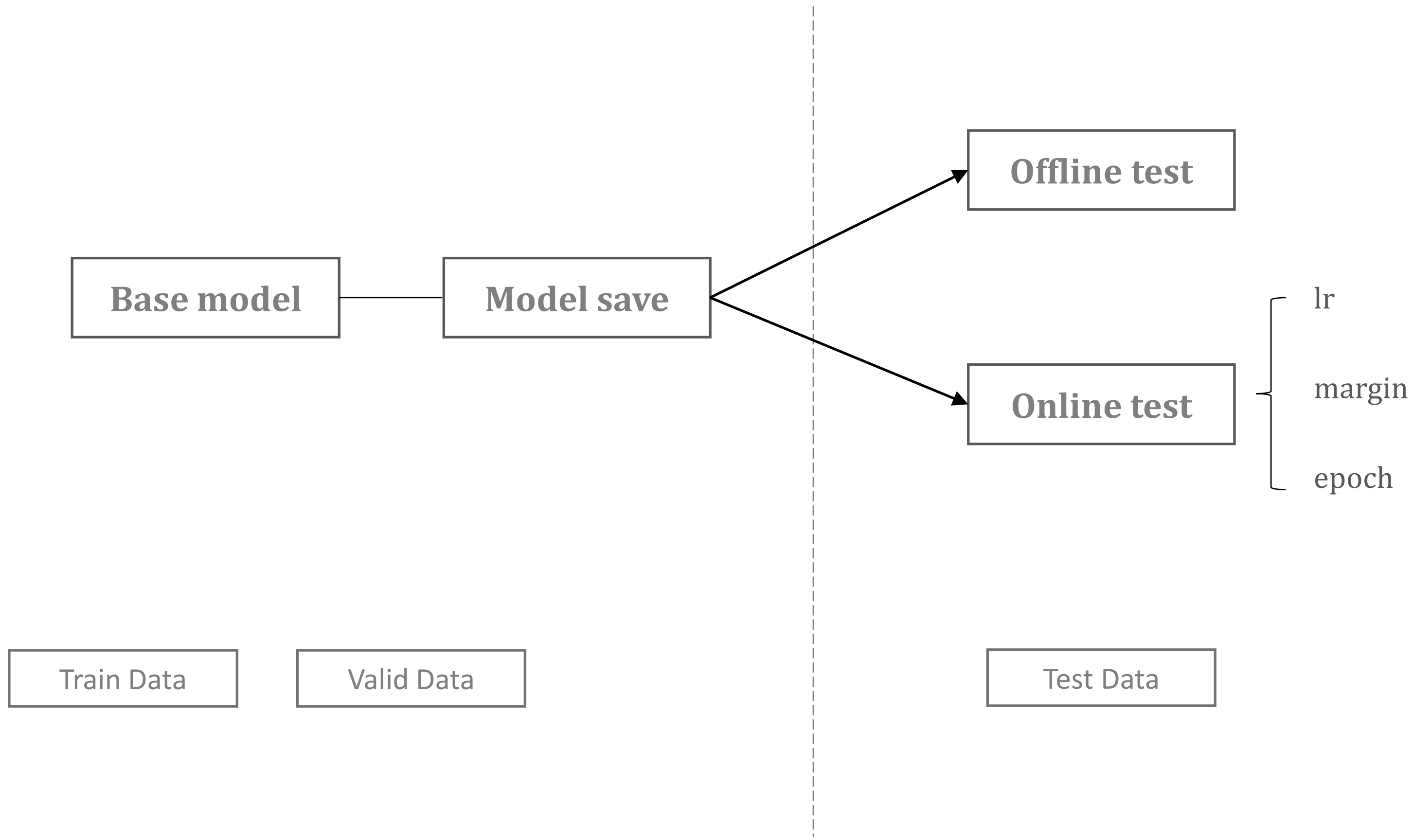
    **WHILE** *epoch > n*
        *model update using η*
        *y' = model(x)*
        *e' = mae(y, y')*
**ELSE**
    *train = 0*

**RETURN** *e, train*

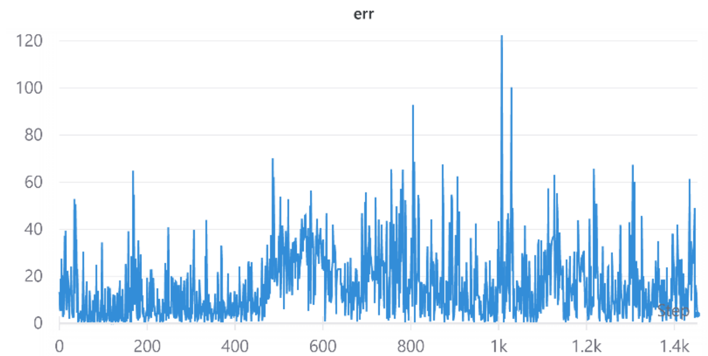


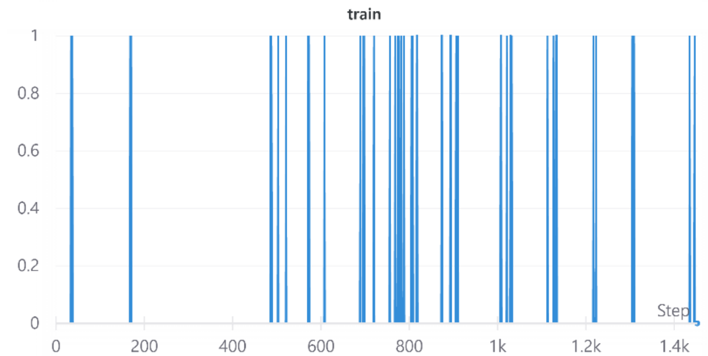
# Experiment Result

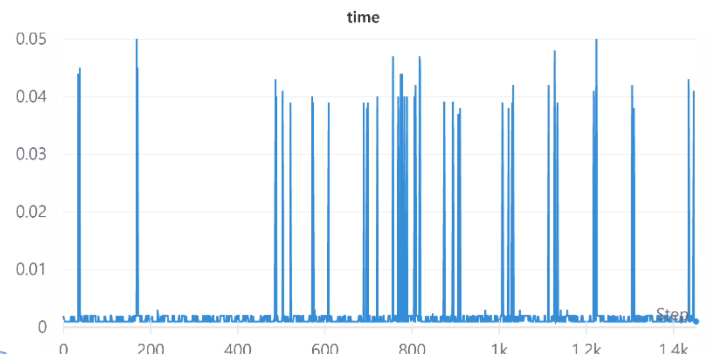
- Experiment Overview



# Proposed method



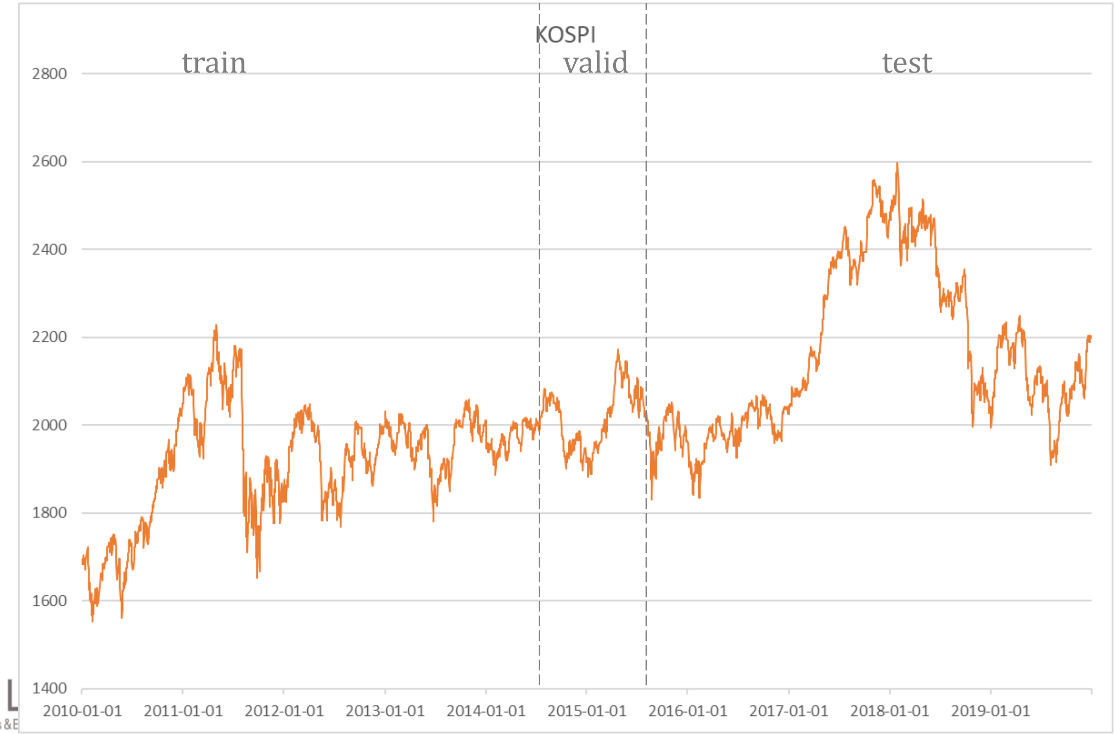

- MAE
- Train count
- Train time

# Experiment Result

- Experiment Data
  - KOSPI
  - 10 years, 3652 days

| | mean | std | min | max |
|---|---|---|---|---|
| KOSPI | 2036.37 | 192.63 | 1552.79 | 2598.19 |

- Train : Valid : Test = 0.5 : 0.1 : 0.4 (1826, 365, 1461)
- window len : 7,  target len: 1




부산대학교 PUSAN NATIONAL UNIVERSITY

# Experiment Result

- Online Learning Test example

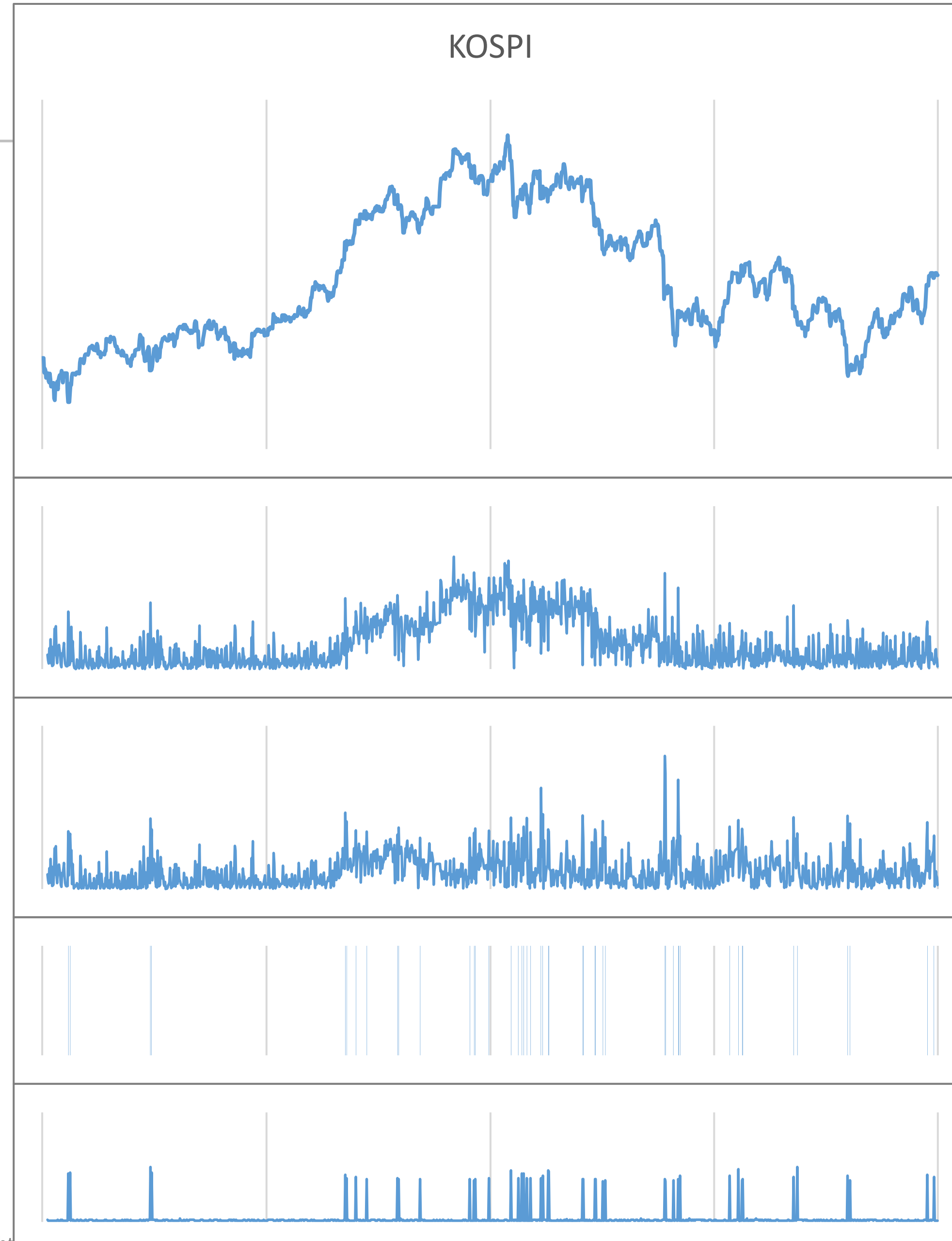
KOSPI Index

Batch Learning MAE

Online Learning MAE

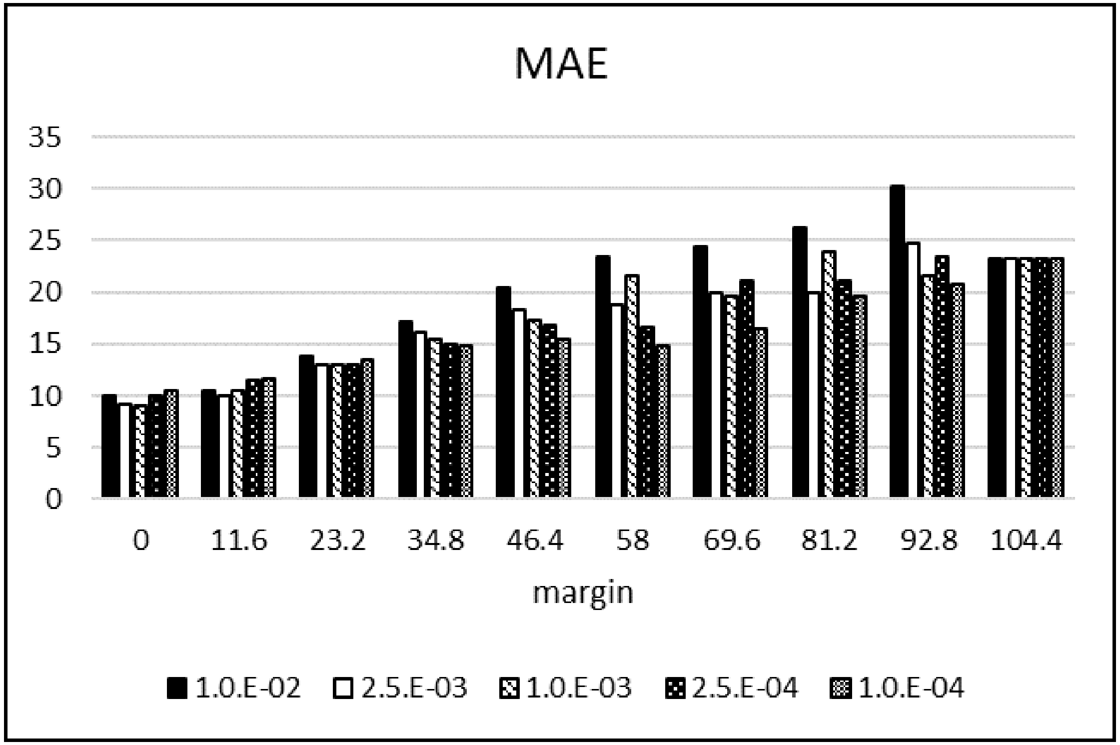
Train

Time



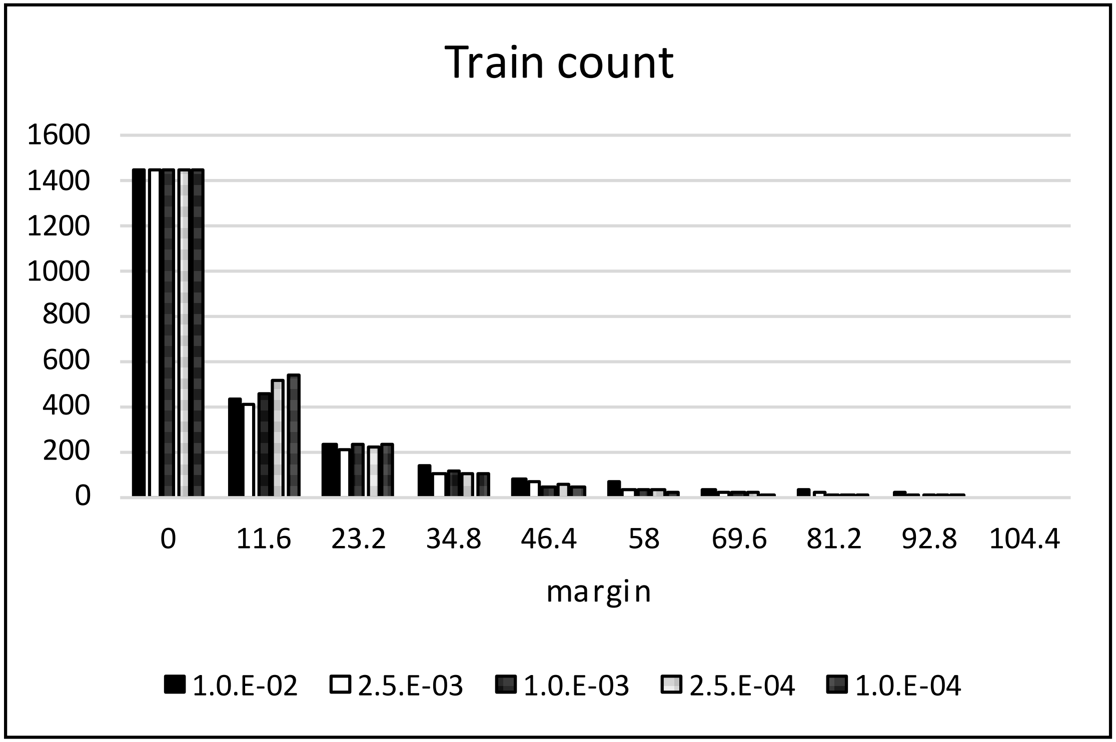
# Experiment Result

- Online Learning Test Result (epoch = 25)



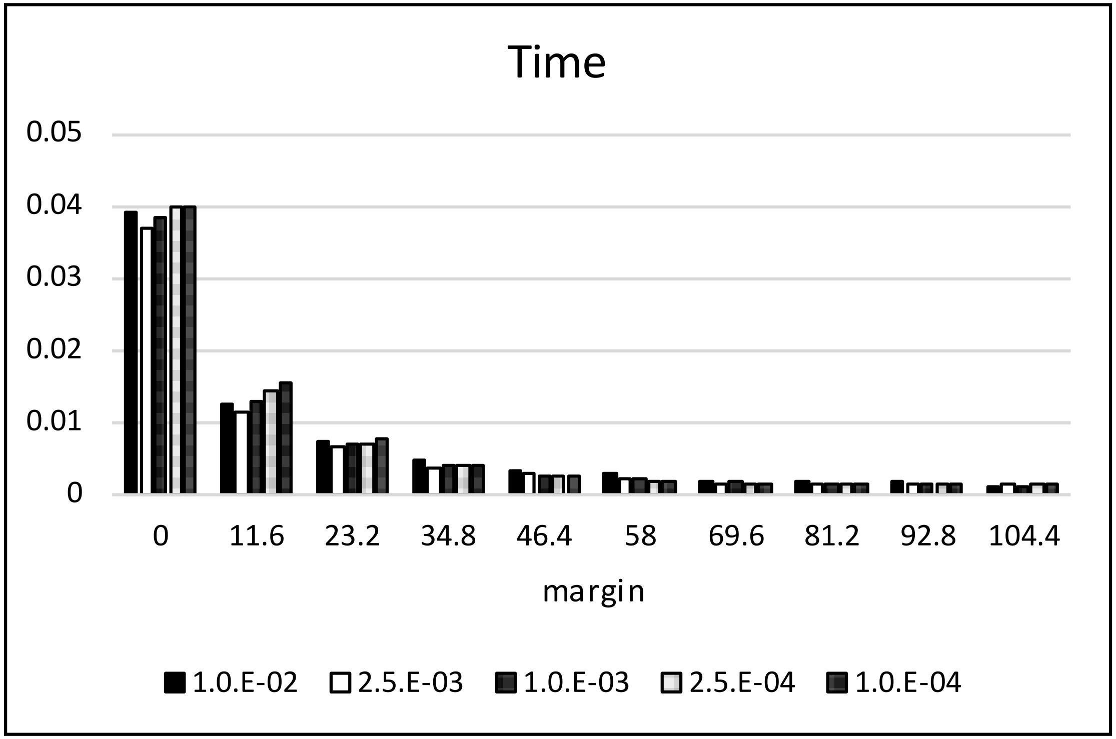
# Experiment Result

- Online Learning Test Result (epoch = 25)



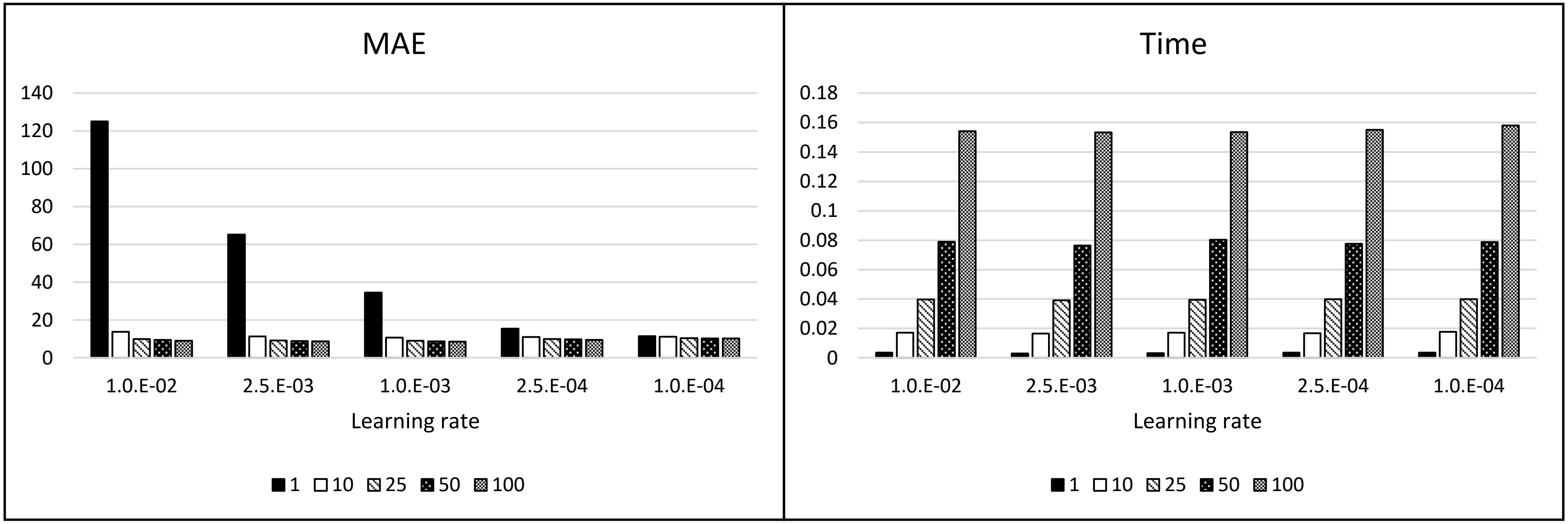
## Experiment Result

- Online Learning Test Result (epoch = 25)



# Experiment Result

- Online Learning Test Result (margin = 0)



# On-line Learning pseudo-code

**Algorithm 3** OLR$(x, y, \epsilon, \varepsilon, \eta_0, \delta, \Omega, \mathcal{B}, W, \omega, \theta)$

**Input**:

- $x, y$ : Input vector $x$ and its true label $y$.
- $\epsilon$ : Threshold value $\epsilon \in \mathbb{R}, \epsilon > 0$
- $\varepsilon$ : Minimum relative error improvement $\varepsilon \in \mathbb{R}, \varepsilon > 0$
- $\eta_0$ : Initial learning rate $\eta_0 \in \mathbb{R}, \eta_0 > 0$
- $\delta$ : Annealing rate $\delta \in \mathbb{R}, \delta > 0$
- $\Omega$ : Set of observed instances
- $\mathcal{B}$ : Set of parameter vectors, $\mathcal{B} = \{\beta_1, \beta_2, \dots, \beta_{|\mathcal{B}|}\}$
- $W$ : Set of associated weight of $\mathcal{B}$, $W = \{w_1, w_2, \dots, w_{|\mathcal{B}|}\}$
- $\omega$ : A reduction function $\omega = e^{-\alpha}, \alpha > 0$
- $\theta$ : Maximum size of $\mathcal{B}$, $\theta \in \mathbb{R}, \theta > 0$

**Output**: $p, \mathcal{B}, W, \Omega$

$\Omega \longleftarrow (x, y)$ /* add $x, y$ to the set of observed instances */
$H \longleftarrow \sum_{i=1}^{|\mathcal{B}|} w_i$ /* $w_i$ returns the weight value associated with parameter $i$ */
$\beta_r \longleftarrow \text{sampling}\left(\frac{w_1}{H}, \frac{w_2}{H}, \dots, \frac{w_{|\mathcal{B}|}}{H}\right)$ /* sample a random $\beta_r$, $r$ is it's index in $\mathcal{B}$ */
$\ell \longleftarrow 0$ /* initiate loss value */
$\hat{\ell} \longleftarrow \infty$
$t \longleftarrow 0$ /* initiate trial value */
$p \longleftarrow \frac{e^{(\beta_r^T \cdot x_r)}}{1+e^{(\beta_r^T \cdot x_r)}}$ /* Calculate probability of potential */
**IF** $abs\left(\log_2\left(p^y(1-p)^{(1-y)}\right)\right) \geq \epsilon$ **THEN** /* The prediction makes mistake */
$\eta_e \longleftarrow \frac{\eta_0}{1+e/\delta}$
$\hat{\beta} \longleftarrow \beta_r$ /* initiate a new parameter */
$w_r \longleftarrow w_r \times \omega$ /* update weight value of $\beta_r$*/
**IF** $|\mathcal{B}| < \theta$ **THEN**
**WHILE** relDiff$(\hat{\ell}, \ell) > \varepsilon$ /* Define relDiff$(a, b) = \frac{abs(a-b)}{abs(a)+abs(b)}$ */
**FOR** $i \longleftarrow 1$ **TO** $|\Omega|$ **DO**
$p_i = \frac{e^{\hat{\beta} \cdot x_i}}{1+e^{(\hat{\beta} \cdot x_i)}}$
$\hat{\beta} \longleftarrow \hat{\beta} + \eta_e x_i(\mathrm{I}(y_i = 1) - p_i)$
/* The indicator function I($a$ = 1) returns 1 if $a$ = 1, otherwise 0 */
**ENDFOR**
$\hat{\ell} \longleftarrow \ell$
$\ell \longleftarrow -\sum_{i \leq |\Omega|} \log\left(p_i^{y_i}(1-p_i)^{(1-y_i)}\right)$
$t \longleftarrow t + 1$
**ENDWHILE**
$\mathcal{B} \longleftarrow \hat{\beta}$
$W \longleftarrow 1$
**ENDIF**
**ENDIF**
**RETURN** $p, \mathcal{B}, W, \Omega$

Source: Dinh Thanh Cong, "Mining potential iddeas for new product development in crowdsourcing application", Master thesis of PNU

Prof. Hyerim Bae (hrbae@pusan.ac.kr)