

# Reuters dataset

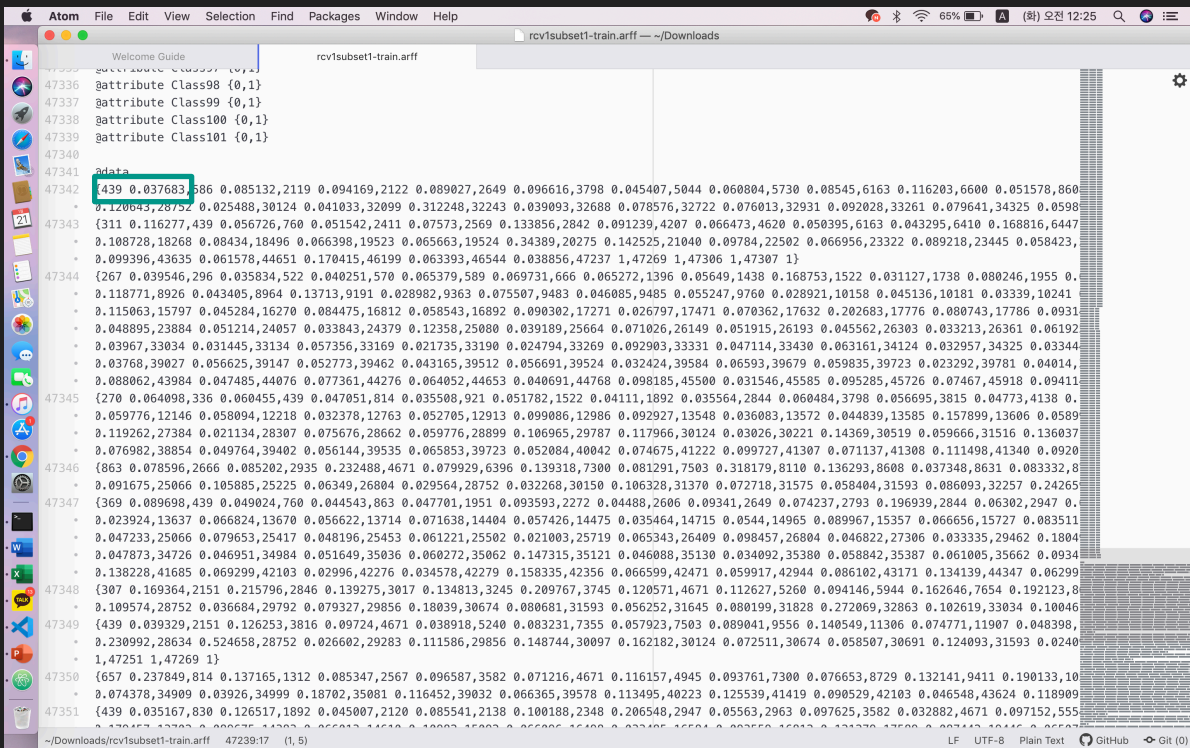
Team A

박성환, 남현지, 이고은, 송지현

# 1. 문제분석

{숫자 (공백) 숫자,  
숫자 (공백) 숫자,

{숫자 (공백) 숫자,  
숫자 (공백) 숫자,

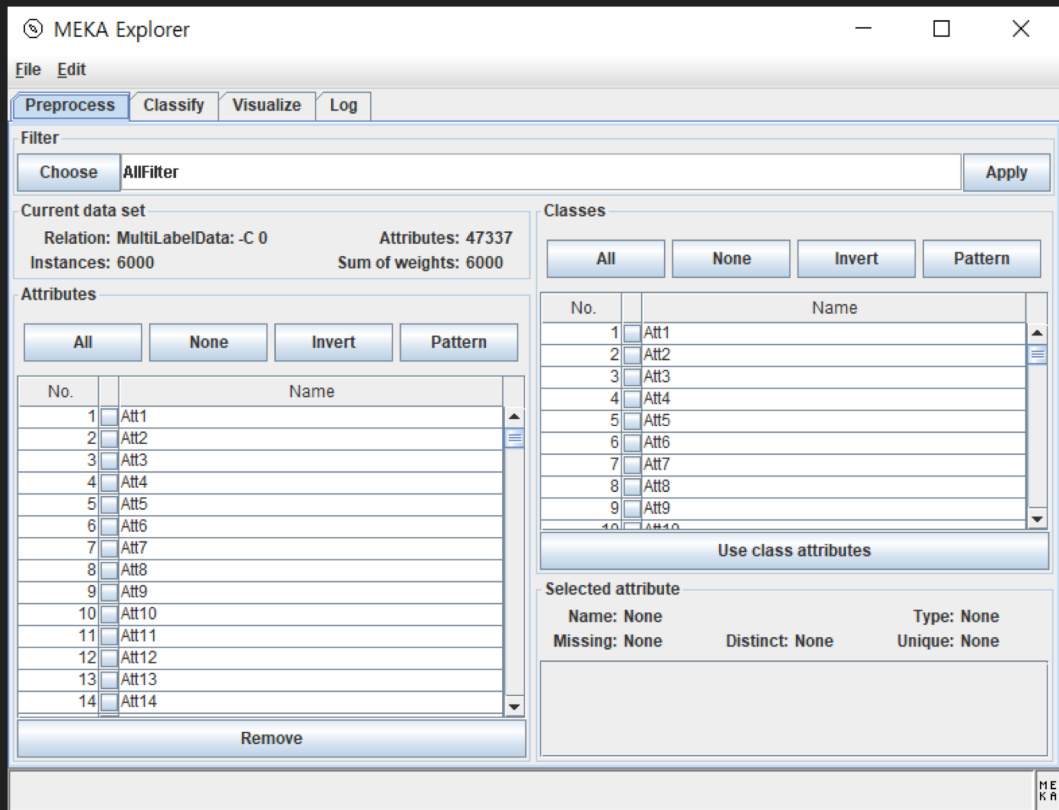
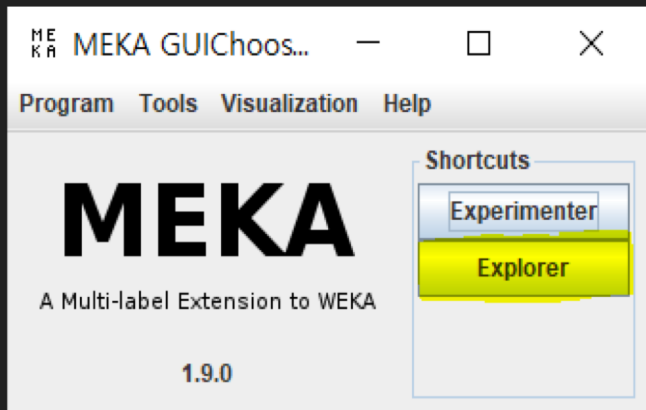


```
Atom File Edit View Selection Find Packages Window Help
rcv1subset1-train.arff
Welcome Guide
@attribute Class98 {0,1}
@attribute Class99 {0,1}
@attribute Class100 {0,1}
@attribute Class101 {0,1}
adata
439 0.037683 0.586 0.085132,2119 0.094169,2122 0.089027,2649 0.096616,3798 0.045407,5044 0.060804,5730 0.08545,6163 0.116203,6600 0.051578,860
0.120643,28752 0.025488,30124 0.041033,32099 0.312248,32243 0.039093,32688 0.078576,32722 0.076013,32931 0.092028,33261 0.079641,34325 0.0598
311 0.116277,439 0.056726,760 0.051542,2411 0.07573,2569 0.133856,2842 0.091239,4207 0.066473,4620 0.050395,6163 0.043295,6410 0.168816,6447
0.108728,18268 0.08434,18496 0.066398,19523 0.065663,19524 0.34389,20275 0.142525,21040 0.09784,22502 0.066956,23322 0.089218,23445 0.058423,
0.099396,43635 0.061578,44651 0.170415,46199 0.063393,46544 0.038856,47237 1,47269 1,47306 1,47307 1}
{267 0.039546,296 0.035834,522 0.040251,570 0.065379,589 0.069731,666 0.065272,1396 0.05649,1438 0.168753,1522 0.031127,1738 0.080246,1955 0.
0.118771,8926 0.043405,8964 0.13713,9191 0.028982,9363 0.075507,9483 0.046085,9485 0.055247,9760 0.028921,10158 0.045136,10181 0.03339,10241
0.115063,15797 0.045284,16270 0.084475,16812 0.058543,16892 0.090302,17271 0.026797,17471 0.078362,17632 0.202683,17776 0.080743,17786 0.0931
0.048895,23884 0.051214,24057 0.033843,24379 0.12358,25080 0.039189,25664 0.071026,26149 0.051915,26193 0.045562,26303 0.033213,26361 0.06192
0.03967,33034 0.031445,33134 0.057356,33169 0.021735,33190 0.024794,33269 0.092903,33331 0.047114,33430 0.063161,34124 0.032957,34325 0.03344
0.03768,39027 0.056625,39147 0.052773,39462 0.043165,39512 0.056691,39524 0.032424,39584 0.06593,39679 0.059835,39723 0.023292,39781 0.04041,
0.088062,43984 0.047485,44076 0.077361,44276 0.064052,44653 0.040691,44768 0.098185,45500 0.031546,45585 0.095285,45726 0.07467,45918 0.09411
{270 0.064098,336 0.060455,439 0.047051,814 0.035508,921 0.051782,1522 0.04111,1892 0.035564,2844 0.060484,3798 0.056695,3815 0.04773,4138 0.
0.059776,12146 0.058094,12218 0.032378,12763 0.052705,12913 0.099086,12986 0.092927,13548 0.036083,13572 0.044839,13585 0.157899,13606 0.0589
0.119262,27384 0.021134,28307 0.075676,28662 0.059776,28899 0.106965,29787 0.117966,30124 0.03026,30221 0.14369,30519 0.059666,31516 0.136037
0.076982,38854 0.049764,39402 0.056144,39535 0.065853,39723 0.052084,40042 0.074675,41222 0.099727,41307 0.071137,41308 0.111498,41340 0.0920
{863 0.078596,2666 0.085202,2935 0.232488,4671 0.079929,6396 0.139318,7300 0.081291,7503 0.318179,8110 0.136293,8608 0.037348,8631 0.083332,8
0.091675,25066 0.105885,25225 0.06349,26804 0.029564,28752 0.032268,30150 0.106328,31370 0.072718,31575 0.058404,31593 0.086093,32257 0.24265
{369 0.089698,439 0.049024,760 0.044543,863 0.047701,1951 0.093593,2272 0.044888,2606 0.09341,2649 0.074237,2793 0.196939,2844 0.06302,2947 0.
0.023924,13637 0.066824,13670 0.056622,13714 0.071638,14404 0.057426,14475 0.035464,14715 0.0544,14965 0.089967,15357 0.066656,15727 0.083511
0.047233,25066 0.079653,25417 0.048196,25453 0.061221,25502 0.021003,25719 0.065343,26409 0.098457,26804 0.046822,27306 0.033335,29462 0.1804
0.047873,34726 0.046951,34984 0.051649,35036 0.060272,35062 0.147315,35121 0.046088,35130 0.034092,35380 0.058842,35387 0.061005,35662 0.0934
0.138228,41685 0.069299,42103 0.02996,42276 0.034578,42279 0.158335,42356 0.066599,42471 0.059917,42944 0.086102,43171 0.134139,44347 0.06299
{307 0.169364,2151 0.215796,2846 0.139275,3017 0.134813,3245 0.209767,3745 0.120571,4671 0.112627,5260 0.094146,5944 0.162646,7654 0.192123,8
0.109574,28752 0.036684,29792 0.079327,29856 0.18039,30674 0.080681,31593 0.056252,31645 0.080199,31828 0.272069,32863 0.102619,33034 0.10046
{439 0.039329,2151 0.126253,3816 0.09724,4671 0.038918,5240 0.083231,7355 0.057923,7503 0.089041,9556 0.140549,11306 0.074771,11907 0.048398,
0.230992,28634 0.524658,28752 0.026602,29283 0.111586,29856 0.148744,30097 0.162182,30124 0.072511,30674 0.058507,30691 0.124093,31593 0.0240
1,47251 1,47269 1}
{657 0.237849,814 0.137165,1312 0.085347,2567 0.096587,3582 0.072126,4671 0.116157,4945 0.093761,7300 0.076653,8729 0.132141,9411 0.190133,10
0.074378,34909 0.03926,34999 0.18702,35081 0.116452,39032 0.066365,39578 0.113495,40223 0.125539,41419 0.090529,42103 0.046548,43624 0.118909
{439 0.035167,830 0.126517,1892 0.045007,2120 0.085541,2138 0.100188,2348 0.206548,2947 0.05563,2963 0.097625,3582 0.032882,4671 0.097152,555
0.120643,28752 0.025488,30124 0.041033,32099 0.312248,32243 0.039093,32688 0.078576,32722 0.076013,32931 0.092028,33261 0.079641,34325 0.0598
311 0.116277,439 0.056726,760 0.051542,2411 0.07573,2569 0.133856,2842 0.091239,4207 0.066473,4620 0.050395,6163 0.043295,6410 0.168816,6447
0.108728,18268 0.08434,18496 0.066398,19523 0.065663,19524 0.34389,20275 0.142525,21040 0.09784,22502 0.066956,23322 0.089218,23445 0.058423,
0.099396,43635 0.061578,44651 0.170415,46199 0.063393,46544 0.038856,47237 1,47269 1,47306 1,47307 1}
{267 0.039546,296 0.035834,522 0.040251,570 0.065379,589 0.069731,666 0.065272,1396 0.05649,1438 0.168753,1522 0.031127,1738 0.080246,1955 0.
0.118771,8926 0.043405,8964 0.13713,9191 0.028982,9363 0.075507,9483 0.046085,9485 0.055247,9760 0.028921,10158 0.045136,10181 0.03339,10241
0.115063,15797 0.045284,16270 0.084475,16812 0.058543,16892 0.090302,17271 0.026797,17471 0.078362,17632 0.202683,17776 0.080743,17786 0.0931
0.048895,23884 0.051214,24057 0.033843,24379 0.12358,25080 0.039189,25664 0.071026,26149 0.051915,26193 0.045562,26303 0.033213,26361 0.06192
0.03967,33034 0.031445,33134 0.057356,33169 0.021735,33190 0.024794,33269 0.092903,33331 0.047114,33430 0.063161,34124 0.032957,34325 0.03344
0.03768,39027 0.056625,39147 0.052773,39462 0.043165,39512 0.056691,39524 0.032424,39584 0.06593,39679 0.059835,39723 0.023292,39781 0.04041,
0.088062,43984 0.047485,44076 0.077361,44276 0.064052,44653 0.040691,44768 0.098185,45500 0.031546,45585 0.095285,45726 0.07467,45918 0.09411
{270 0.064098,336 0.060455,439 0.047051,814 0.035508,921 0.051782,1522 0.04111,1892 0.035564,2844 0.060484,3798 0.056695,3815 0.04773,4138 0.
0.059776,12146 0.058094,12218 0.032378,12763 0.052705,12913 0.099086,12986 0.092927,13548 0.036083,13572 0.044839,13585 0.157899,13606 0.0589
0.119262,27384 0.021134,28307 0.075676,28662 0.059776,28899 0.106965,29787 0.117966,30124 0.03026,30221 0.14369,30519 0.059666,31516 0.136037
0.076982,38854 0.049764,39402 0.056144,39535 0.065853,39723 0.052084,40042 0.074675,41222 0.099727,41307 0.071137,41308 0.111498,41340 0.0920
{863 0.078596,2666 0.085202,2935 0.232488,4671 0.079929,6396 0.139318,7300 0.081291,7503 0.318179,8110 0.136293,8608 0.037348,8631 0.083332,8
0.091675,25066 0.105885,25225 0.06349,26804 0.029564,28752 0.032268,30150 0.106328,31370 0.072718,31575 0.058404,31593 0.086093,32257 0.24265
{369 0.089698,439 0.049024,760 0.044543,863 0.047701,1951 0.093593,2272 0.044888,2606 0.09341,2649 0.074237,2793 0.196939,2844 0.06302,2947 0.
0.023924,13637 0.066824,13670 0.056622,13714 0.071638,14404 0.057426,14475 0.035464,14715 0.0544,14965 0.089967,15357 0.066656,15727 0.083511
0.047233,25066 0.079653,25417 0.048196,25453 0.061221,25502 0.021003,25719 0.065343,26409 0.098457,26804 0.046822,27306 0.033335,29462 0.1804
0.047873,34726 0.046951,34984 0.051649,35036 0.060272,35062 0.147315,35121 0.046088,35130 0.034092,35380 0.058842,35387 0.061005,35662 0.0934
0.138228,41685 0.069299,42103 0.02996,42276 0.034578,42279 0.158335,42356 0.066599,42471 0.059917,42944 0.086102,43171 0.134139,44347 0.06299
{307 0.169364,2151 0.215796,2846 0.139275,3017 0.134813,3245 0.209767,3745 0.120571,4671 0.112627,5260 0.094146,5944 0.162646,7654 0.192123,8
0.109574,28752 0.036684,29792 0.079327,29856 0.18039,30674 0.080681,31593 0.056252,31645 0.080199,31828 0.272069,32863 0.102619,33034 0.10046
{439 0.039329,2151 0.126253,3816 0.09724,4671 0.038918,5240 0.083231,7355 0.057923,7503 0.089041,9556 0.140549,11306 0.074771,11907 0.048398,
0.230992,28634 0.524658,28752 0.026602,29283 0.111586,29856 0.148744,30097 0.162182,30124 0.072511,30674 0.058507,30691 0.124093,31593 0.0240
1,47251 1,47269 1}
{657 0.237849,814 0.137165,1312 0.085347,2567 0.096587,3582 0.072126,4671 0.116157,4945 0.093761,7300 0.076653,8729 0.132141,9411 0.190133,10
0.074378,34909 0.03926,34999 0.18702,35081 0.116452,39032 0.066365,39578 0.113495,40223 0.125539,41419 0.090529,42103 0.046548,43624 0.118909
{439 0.035167,830 0.126517,1892 0.045007,2120 0.085541,2138 0.100188,2348 0.206548,2947 0.05563,2963 0.097625,3582 0.032882,4671 0.097152,555
0.120643,28752 0.025488,30124 0.041033,32099 0.312248,32243 0.039093,32688 0.078576,32722 0.076013,32931 0.092028,33261 0.079641,34325 0.0598
311 0.116277,439 0.056726,760 0.051542,2411 0.07573,2569 0.133856,2842 0.091239,4207 0.066473,4620 0.050395,6163 0.043295,6410 0.168816,6447
0.108728,18268 0.08434,18496 0.066398,19523 0.065663,19524 0.34389,20275 0.142525,21040 0.09784,22502 0.066956,23322 0.089218,23445 0.058423,
0.099396,43635 0.061578,44651 0.170415,46199 0.063393,46544 0.038856,47237 1,47269 1,47306 1,47307 1}
{267 0.039546,296 0.035834,522 0.040251,570 0.065379,589 0.069731,666 0.065272,1396 0.05649,1438 0.168753,1522 0.031127,1738 0.080246,1955 0.
0.118771,8926 0.043405,8964 0.13713,9191 0.028982,9363 0.075507,9483 0.046085,9485 0.055247,9760 0.028921,10158 0.045136,10181 0.03339,10241
0.115063,15797 0.045284,16270 0.084475,16812 0.058543,16892 0.090302,17271 0.026797,17471 0.078362,17632 0.202683,17776 0.080743,17786 0.0931
0.048895,23884 0.051214,24057 0.033843,24379 0.12358,25080 0.039189,25664 0.071026,26149 0.051915,26193 0.045562,26303 0.033213,26361 0.06192
0.03967,33034 0.031445,33134 0.057356,33169 0.021735,33190 0.024794,33269 0.092903,33331 0.047114,33430 0.063161,34124 0.032957,34325 0.03344
0.03768,39027 0.056625,39147 0.052773,39462 0.043165,39512 0.056691,39524 0.032424,39584 0.06593,39679 0.059835,39723 0.023292,39781 0.04041,
0.088062,43984 0.047485,44076 0.077361,44276 0.064052,44653 0.040691,44768 0.098185,45500 0.031546,45585 0.095285,45726 0.07467,45918 0.09411
{270 0.064098,336 0.060455,439 0.047051,814 0.035508,921 0.051782,1522 0.04111,1892 0.035564,2844 0.060484,3798 0.056695,3815 0.04773,4138 0.
0.059776,12146 0.058094,12218 0.032378,12763 0.052705,12913 0.099086,12986 0.092927,13548 0.036083,13572 0.044839,13585 0.157899,13606 0.0589
0.119262,27384 0.021134,28307 0.075676,28662 0.059776,28899 0.106965,29787 0.117966,30124 0.03026,30221 0.14369,30519 0.059666,31516 0.136037
0.076982,38854 0.049764,39402 0.056144,39535 0.065853,39723 0.052084,40042 0.074675,41222 0.099727,41307 0.071137,41308 0.111498,41340 0.0920
{863 0.078596,2666 0.085202,2935 0.232488,4671 0.079929,6396 0.139318,7300 0.081291,7503 0.318179,8110 0.136293,8608 0.037348,8631 0.083332,8
0.091675,25066 0.105885,25225 0.06349,26804 0.029564,28752 0.032268,30150 0.106328,31370 0.072718,31575 0.058404,31593 0.086093,32257 0.24265
{369 0.089698,439 0.049024,760 0.044543,863 0.047701,1951 0.093593,2272 0.044888,2606 0.09341,2649 0.074237,2793 0.196939,2844 0.06302,2947 0.
0.023924,13637 0.066824,13670 0.056622,13714 0.071638,14404 0.057426,14475 0.035464,14715 0.0544,14965 0.089967,15357 0.066656,15727 0.083511
0.047233,25066 0.079653,25417 0.048196,25453 0.061221,25502 0.021003,25719 0.065343,26409 0.098457,26804 0.046822,27306 0.033335,29462 0.1804
0.047873,34726 0.046951,34984 0.051649,35036 0.060272,35062 0.147315,35121 0.046088,35130 0.034092,35380 0.058842,35387 0.061005,35662 0.0934
0.138228,41685 0.069299,42103 0.02996,42276 0.034578,42279 0.158335,42356 0.066599,42471 0.059917,42944 0.086102,43171 0.134139,44347 0.06299
{307 0.169364,2151 0.215796,2846 0.139275,3017 0.134813,3245 0.209767,3745 0.120571,4671 0.112627,5260 0.094146,5944 0.162646,7654 0.192123,8
0.109574,28752 0.036684,29792 0.079327,29856 0.18039,30674 0.080681,31593 0.056252,31645 0.080199,31828 0.272069,32863 0.102619,33034 0.10046
{439 0.039329,2151 0.126253,3816 0.09724,4671 0.038918,5240 0.083231,7355 0.057923,7503 0.089041,9556 0.140549,11306 0.074771,11907 0.048398,
0.230992,28634 0.524658,28752 0.026602,29283 0.111586,29856 0.148744,30097 0.162182,30124 0.072511,30674 0.058507,30691 0.124093,31593 0.0240
1,47251 1,47269 1}
{657 0.237849,814
```

	Att1	Att2	Att3	Att4	...	Class98	Class99	Class100	Class101
0	0	0.0	0.0	0.0	...	0	0	0	0
1	0	0.0	0.0	0.0	...	0	0	0	0
2	0	0.0	0.0	0.0	...	0	0	0	0
3	0	0.0	0.0	0.0	...	0	0	0	0
4	0	0.0	0.0	0.0	...	0	0	0	0
...	...	...	...	...	...	...	...	...	...
5995	0	0.0	0.0	0.0	...	0	0	0	0
5996	0	0.0	0.0	0.0	...	0	0	0	0
5997	0	0.0	0.0	0.0	...	0	0	0	0
5998	0	0.0	0.0	0.0	...	0	0	0	0
5999	0	0.0	0.0	0.0	...	0	0	0	0

[6000 rows x 47337 columns]

instance : 6000, numeric : 47236, label : 101



```
1 class_sum_list=[]
2 for i in range(1,102):
3     class_name = "Class"+str(i)
4     class_data = pandas_file[class_name]
5     class_sum = class_data.sum()
6     class_sum_list.append(class_sum)
7 print(class_sum_list)
8 print(class_sum_list[4])
```

[120, 197, 196, 527, 764, 328, 285, 247, 140,  
764

```
1 top_10_index=[]
2 for _ in range (10):
3     Maxnum = max(class_sum_list)
4     Maxnum_index = class_sum_list.index(Maxnum)
5     top_10_index.append(Maxnum_index)
6     class_sum_list[Maxnum_index]=0
7 print(top_10_index)
```

[32, 33, 70, 69, 4, 98, 59, 3, 97, 58]

# LogReg

```
1 import torch
2
3 class CustomDataset(torch.utils.data.Dataset):
4     def __init__(self, filename, index):
5         data = pd.read_csv(filename, sep=',')
6         data = data.to_numpy()
7         |
8         self.x_data=data[:, :47236]
9         self.y_data = data[:, 47236+index]
10        self.length = len(data)
11
12    def __getitem__(self, index):
13        return self.x_data[index], self.y_data[index]
14
15    def __len__(self):
16        return self.length
```

```
1 from sklearn.model_selection import GridSearchCV
2 from sklearn.linear_model import LogisticRegression
3 from sklearn.preprocessing import MinMaxScaler
4
5 train_dataset_1 = CustomDataset('drive/My Drive/rcv1subset1-train.csv', 33)
6 test_dataset_1 = CustomDataset('drive/My Drive/rcv1subset1-test.csv', 33)
7 |
8 X_tr = train_dataset.x_data
9 y_tr = train_dataset.y_data
10 X_ts = test_dataset.x_data
11 y_ts = test_dataset.y_data
```

```
1 from sklearn.metrics import (accuracy_score, precision_score, recall_score, f1_score, log_loss)
2
3 normalizer = MinMaxScaler(feature_range=(0,1))
4 normalizer.fit(X_tr)
5 X_tr_normalized = normalizer.transform(X_tr)
6 X_ts_normalized = normalizer.transform(X_ts)
7
8 clf = LogisticRegression(max_iter = 5000)
9 parameters = {'penalty':['l2'],
10              'C': [10e-5, 10e-3, 10e-2, 10e-1, 10e0, 10e1, 10e2, 10e3, 10e5]}
11 gridsearch = GridSearchCV(clf, parameters, scoring = 'accuracy', cv = 5)
12 gridsearch.fit(X_tr_normalized, y_tr)
13 print(f'gridsearch.best_params_={gridsearch.best_params_}')
14
15 best_clf = gridsearch.best_estimator_
16
17 y_pred = best_clf.predict(X_ts_normalized)
18 test_acc = accuracy_score(y_ts, y_pred)
19 print(f'test_acc={test_acc}')
```

```
➞ gridsearch.best_params_={'C': 1000.0, 'penalty': 'l2'}
   test_acc=0.5603333333333333
```

# Random Forest

```
1 # Instantiate a model object
2 clf = RandomForestClassifier()
3
4 # Set a search range
5 parameters = {'n_estimators': [100, 150, 200],
6               'criterion': ['gini', 'entropy']}
7 # Find the best hyperparameters using GridSearchCV
8 gridsearch = GridSearchCV(clf, parameters, scoring = 'accuracy', cv = 5)
9 gridsearch.fit(X_tr, y_tr)
10
11 # Show the best hyperparameters
12 print(f'gridsearch.best_params_ = {gridsearch.best_params_}')
13
14 # The best model is stored in 'best_clf'
15 best_clf = gridsearch.best_estimator_
16 best_clf
```

```
1 y_pred = best_clf.predict(X_ts)
2 test_acc = accuracy_score(y_ts, y_pred)
3 print(f'test_acc = {test_acc}')
```



[32] test\_acc = 0.056

[33] test\_acc = 0.604

[70] test\_acc = 0.780333

[69] test\_acc = 0.713666

[4] test\_acc = 0.95

[98] test\_acc = 0.94866

[59] test\_acc = 0.947333

[3] test\_acc = 0.834666

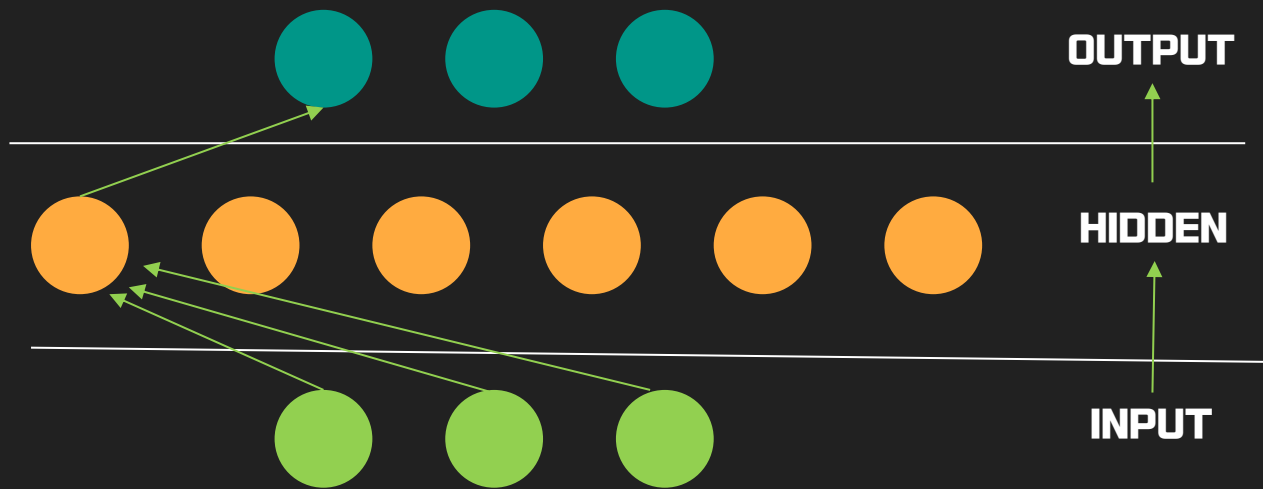
[97] test\_acc = 0.878

[58] test\_acc = 0.863

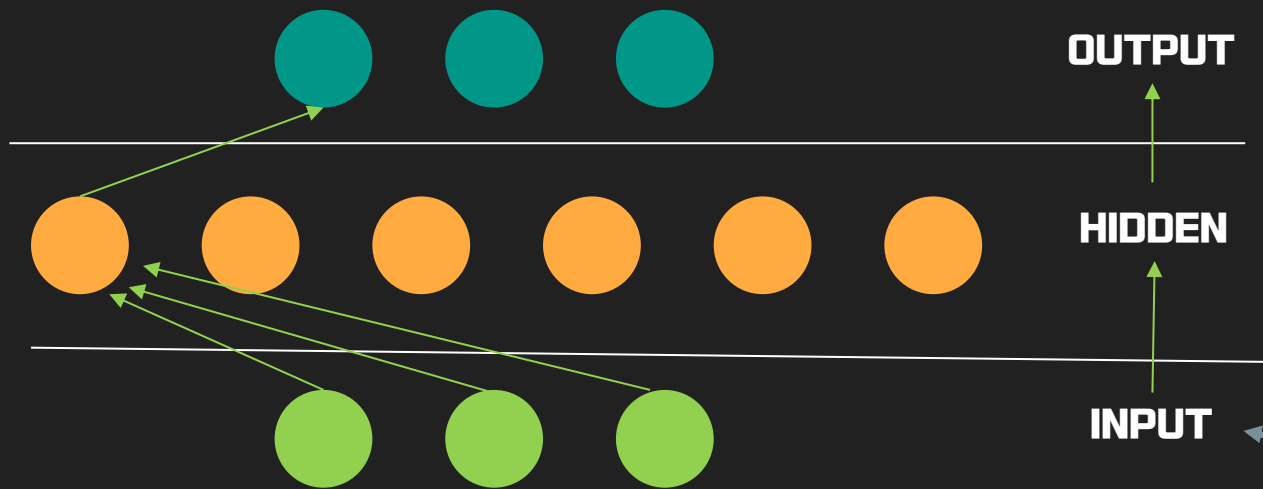
**MLP**

**MLP = multi layer perceptron**

# MLP = multi layer perceptron

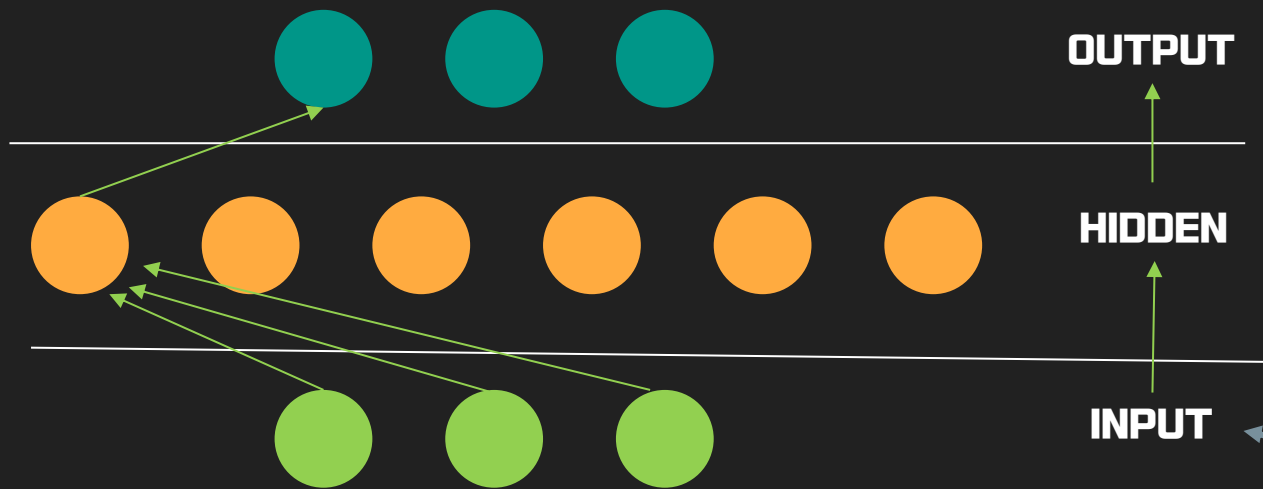


# MLP = multi layer perceptron



```
1 input_size = 47236
2 num_classes = 2
3 batch_size = 100
4 learning_rate = 0.001
5
```

# MLP = multi layer perceptron



```
1 input_size = 47236
2 num_classes = 2
3 batch_size = 100
4 learning_rate = 0.001
5
```

input size  
x  
256  
x  
128  
x  
num\_classes

```
82 class FFNet(nn.Module):
83     def __init__(self, input_size, num_classes):
84         super(FFNet, self).__init__()
85         self.fc1 = nn.Linear(input_size, 256)
86         self.relu = nn.ReLU()
87         # self.fc2 = nn.Linear(512, 512)
88         # self.fc3 = nn.Linear(512, 256)
89         self.fc4 = nn.Linear(256, 128)
90         self.fc5 = nn.Linear(128, num_classes)
91
92     def forward(self, x):
93         out = self.fc1(x)
94         out = self.relu(out)
95         # out = self.fc2(out)
96         # out = self.relu(out)
97         # out = self.fc3(out)
98         # out = self.relu(out)
99         out = self.fc4(out)
100        out = self.relu(out)
101        out = self.fc5(out)
102        return out
```

```

1 class CustomDataset(torch.utils.data.Dataset):
2
3 # 우리가 class 생성시에 해줄것은
4 # self.len 에 길이를 넣어주고
5 # self.x_data에 x 데이터를
6 # self.y_data에 y 데이터를 넣어주고
7 # 인덱스에 따라 i 번째 데이터를 잘 리턴해주면 됩니다.
8
9 def __init__(self, filename, index):
10     # 1. initialize file paths or a list of file names
11     # 데이터셋의 전처리를 해주는 부분
12
13     data = pd.read_csv(filename, sep=',')
14     data = data.to_numpy()
15
16     x_data=data[:, :47236]
17     self.x_data = torch.from_numpy(x_data.astype('float'))
18     y_data = data[:, 47236+index]
19     self.y_data = torch.from_numpy(y_data.astype('int64'))
20     self.length = len(data)
21
22 def __getitem__(self, index):
23     return self.x_data[index], self.y_data[index]
24
25 def __len__(self):
26     return self.length
27

```

csv.file

dataframe(pandas)

numpy

tensor



Model 1(32)

Accuracy of the network on the 1000 test inputs: 56.03333333333333%

---

Model 2(33)

Accuracy of the network on the 1000 test inputs: 57.26666666666666%

Model 3(70)

Accuracy of the network on the 1000 test inputs: 75.7%

---

# xgboost

: Extreme gradient boosting

## DT → RF → Boosting

Tree구조

앙상블

연속 앙상블

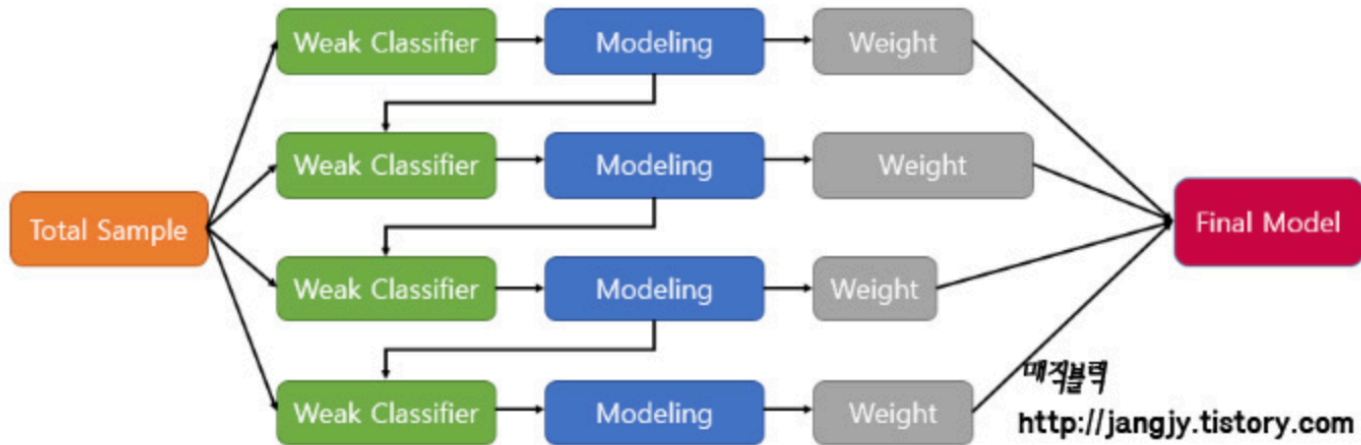


그림 2. boosting의 system flow

```
1 xg_reg = xgb.XGBRegressor(objective = 'reg:linear', colsample_bytree = 0.3, learning_rate = 0.1,  
2                             max_depth = 5, alpha = 10, n_estimators = 10)
```

```
1 xg_reg.fit(X_tr, y_tr)  
2  
3 preds = xg_reg.predict(X_ts)
```

```
1 rmse = np.sqrt(mean_squared_error(y_ts, preds))  
2 print("RMSE: %f" % (rmse))
```

RMSE: 0.560028

**[parameter]**

**n-estimator**

**max\_depth**

**learning rate**