KCSE 2019 Conference

빅데이터

Dept. of Computer Engineering,
Korea Polytechnic University

2019. 05. 29.

김홍준, 신제우

CONTENTS

01

데이터 파일

01 데이터 파일

1.1 칼럼

Basic columns

거래 날짜, 종목 명, 종목 코드 , 시가, 고가, 저가, 종가, 거래량

Add columns

cv_diff_value, cv_diff_rate, cv_maN_value, cv_maN_rate, ud_Nd, cvNd_diff_rate

Option columns

vv_diff_value, vv_diff_rate, vv_maN_value, vv_maN_rate

02

데이터 모델

02 데이터 모델

2.1 N일의 설정

N = 3

263

N = 4

2

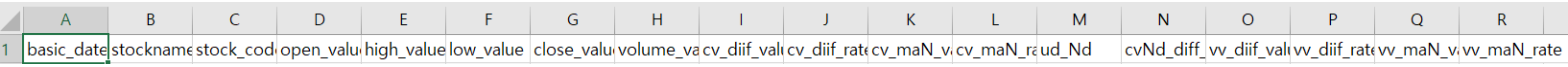
N = 5

0

02 데이터 모델

2.2 독립변수 , 종속변수

총 칼럼 수 : 18개



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	basic_date	stockname	stock_cod	open_valu	high_value	low_value	close_valu	volume_va	cv_diif_valu	cv_diif_rate	cv_maN_v	cv_maN_ra	ud_Nd	cvNd_diff_	vv_diif_valu	vv_diif_rate	vv_maN_v	vv_maN_rate

독립변수 모델에서 제외된 칼럼

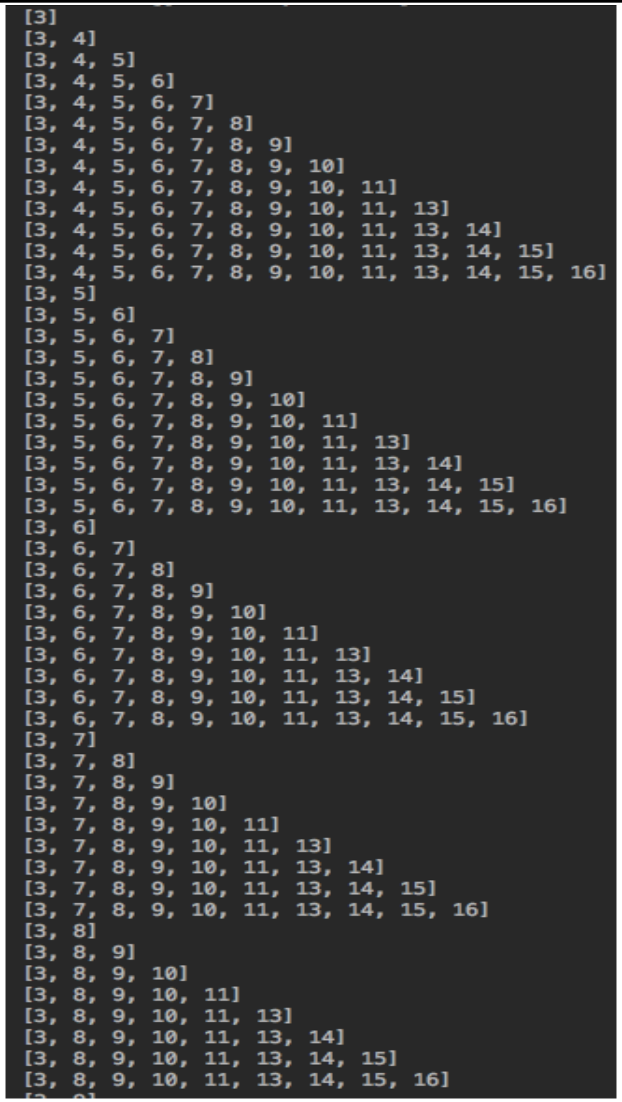
udNd (종속변수), 거래날짜, 종목 명, 종목 코드

사용한 칼럼 수 15개

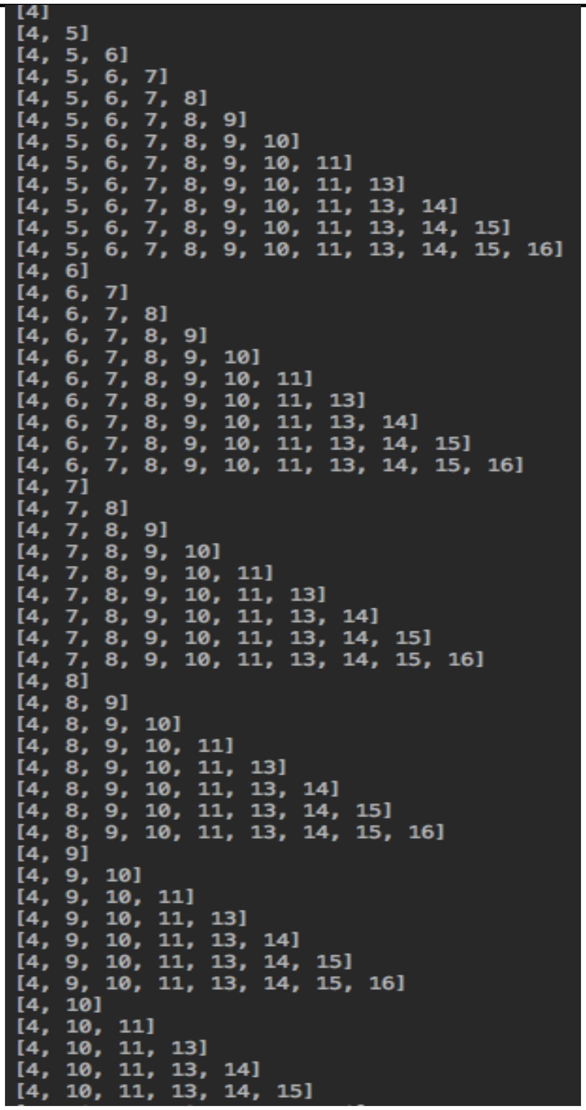
경우의 수 : 469 * (K값) 5 = 2345 개

02 데이터 모델

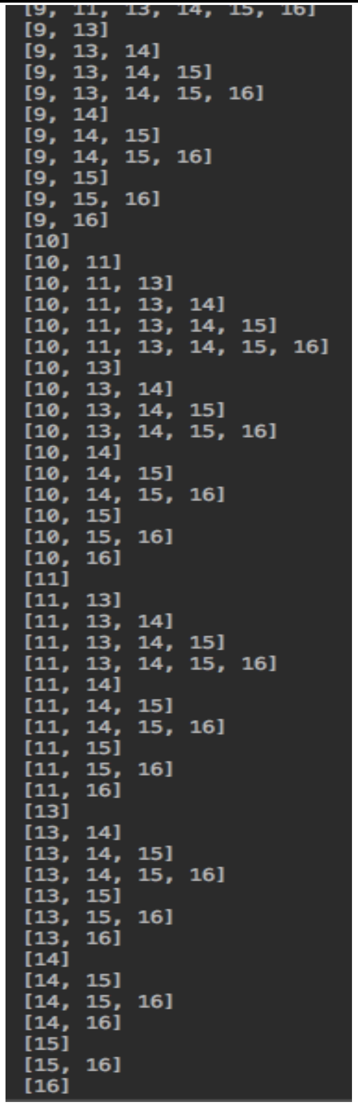
2.2 독립변수 경우의 수



```
[3]
[3, 4]
[3, 4, 5]
[3, 4, 5, 6]
[3, 4, 5, 6, 7]
[3, 4, 5, 6, 7, 8]
[3, 4, 5, 6, 7, 8, 9]
[3, 4, 5, 6, 7, 8, 9, 10]
[3, 4, 5, 6, 7, 8, 9, 10, 11]
[3, 4, 5, 6, 7, 8, 9, 10, 11, 13]
[3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14]
[3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15]
[3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16]
[3, 5]
[3, 5, 6]
[3, 5, 6, 7]
[3, 5, 6, 7, 8]
[3, 5, 6, 7, 8, 9]
[3, 5, 6, 7, 8, 9, 10]
[3, 5, 6, 7, 8, 9, 10, 11]
[3, 5, 6, 7, 8, 9, 10, 11, 13]
[3, 5, 6, 7, 8, 9, 10, 11, 13, 14]
[3, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15]
[3, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16]
[3, 6]
[3, 6, 7]
[3, 6, 7, 8]
[3, 6, 7, 8, 9]
[3, 6, 7, 8, 9, 10]
[3, 6, 7, 8, 9, 10, 11]
[3, 6, 7, 8, 9, 10, 11, 13]
[3, 6, 7, 8, 9, 10, 11, 13, 14]
[3, 6, 7, 8, 9, 10, 11, 13, 14, 15]
[3, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16]
[3, 7]
[3, 7, 8]
[3, 7, 8, 9]
[3, 7, 8, 9, 10]
[3, 7, 8, 9, 10, 11]
[3, 7, 8, 9, 10, 11, 13]
[3, 7, 8, 9, 10, 11, 13, 14]
[3, 7, 8, 9, 10, 11, 13, 14, 15]
[3, 7, 8, 9, 10, 11, 13, 14, 15, 16]
[3, 8]
[3, 8, 9]
[3, 8, 9, 10]
[3, 8, 9, 10, 11]
[3, 8, 9, 10, 11, 13]
[3, 8, 9, 10, 11, 13, 14]
[3, 8, 9, 10, 11, 13, 14, 15]
[3, 8, 9, 10, 11, 13, 14, 15, 16]
```



```
[4]
[4, 5]
[4, 5, 6]
[4, 5, 6, 7]
[4, 5, 6, 7, 8]
[4, 5, 6, 7, 8, 9]
[4, 5, 6, 7, 8, 9, 10]
[4, 5, 6, 7, 8, 9, 10, 11]
[4, 5, 6, 7, 8, 9, 10, 11, 13]
[4, 5, 6, 7, 8, 9, 10, 11, 13, 14]
[4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15]
[4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16]
[4, 6]
[4, 6, 7]
[4, 6, 7, 8]
[4, 6, 7, 8, 9]
[4, 6, 7, 8, 9, 10]
[4, 6, 7, 8, 9, 10, 11]
[4, 6, 7, 8, 9, 10, 11, 13]
[4, 6, 7, 8, 9, 10, 11, 13, 14]
[4, 6, 7, 8, 9, 10, 11, 13, 14, 15]
[4, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16]
[4, 7]
[4, 7, 8]
[4, 7, 8, 9]
[4, 7, 8, 9, 10]
[4, 7, 8, 9, 10, 11]
[4, 7, 8, 9, 10, 11, 13]
[4, 7, 8, 9, 10, 11, 13, 14]
[4, 7, 8, 9, 10, 11, 13, 14, 15]
[4, 7, 8, 9, 10, 11, 13, 14, 15, 16]
[4, 8]
[4, 8, 9]
[4, 8, 9, 10]
[4, 8, 9, 10, 11]
[4, 8, 9, 10, 11, 13]
[4, 8, 9, 10, 11, 13, 14]
[4, 8, 9, 10, 11, 13, 14, 15]
[4, 8, 9, 10, 11, 13, 14, 15, 16]
[4, 9]
[4, 9, 10]
[4, 9, 10, 11]
[4, 9, 10, 11, 13]
[4, 9, 10, 11, 13, 14]
[4, 9, 10, 11, 13, 14, 15]
[4, 9, 10, 11, 13, 14, 15, 16]
[4, 10]
[4, 10, 11]
[4, 10, 11, 13]
[4, 10, 11, 13, 14]
[4, 10, 11, 13, 14, 15]
```



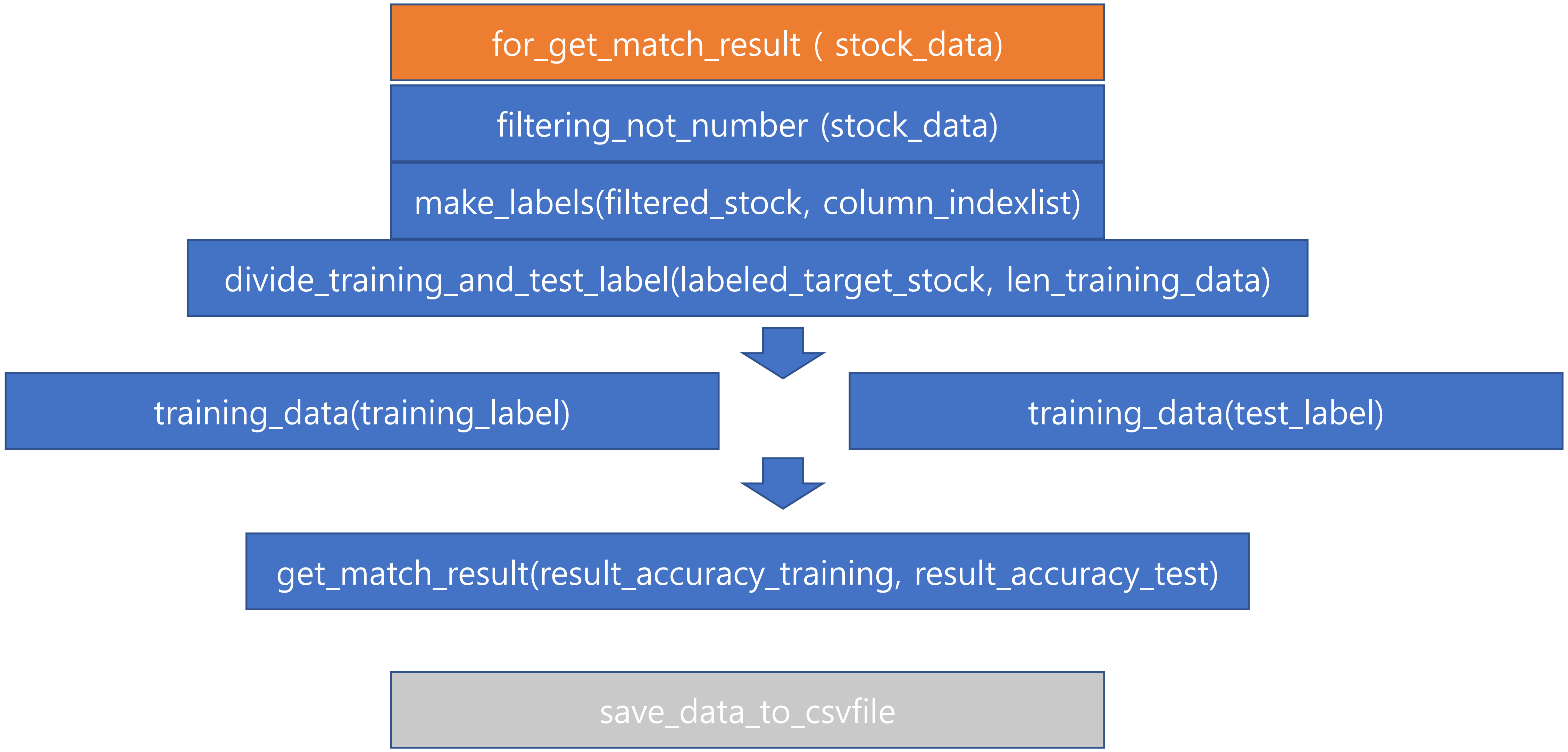
```
[9, 11, 13, 14, 15, 16]
[9, 13]
[9, 13, 14]
[9, 13, 14, 15]
[9, 13, 14, 15, 16]
[9, 14]
[9, 14, 15]
[9, 14, 15, 16]
[9, 15]
[9, 15, 16]
[9, 16]
[10]
[10, 11]
[10, 11, 13]
[10, 11, 13, 14]
[10, 11, 13, 14, 15]
[10, 11, 13, 14, 15, 16]
[10, 13]
[10, 13, 14]
[10, 13, 14, 15]
[10, 13, 14, 15, 16]
[10, 14]
[10, 14, 15]
[10, 14, 15, 16]
[10, 15]
[10, 15, 16]
[10, 16]
[11]
[11, 13]
[11, 13, 14]
[11, 13, 14, 15]
[11, 13, 14, 15, 16]
[11, 14]
[11, 14, 15]
[11, 14, 15, 16]
[11, 15]
[11, 15, 16]
[11, 16]
[13]
[13, 14]
[13, 14, 15]
[13, 14, 15, 16]
[13, 15]
[13, 15, 16]
[13, 16]
[14]
[14, 15]
[14, 15, 16]
[14, 16]
[15]
[15, 16]
[16]
```

03

분석 결과 설명

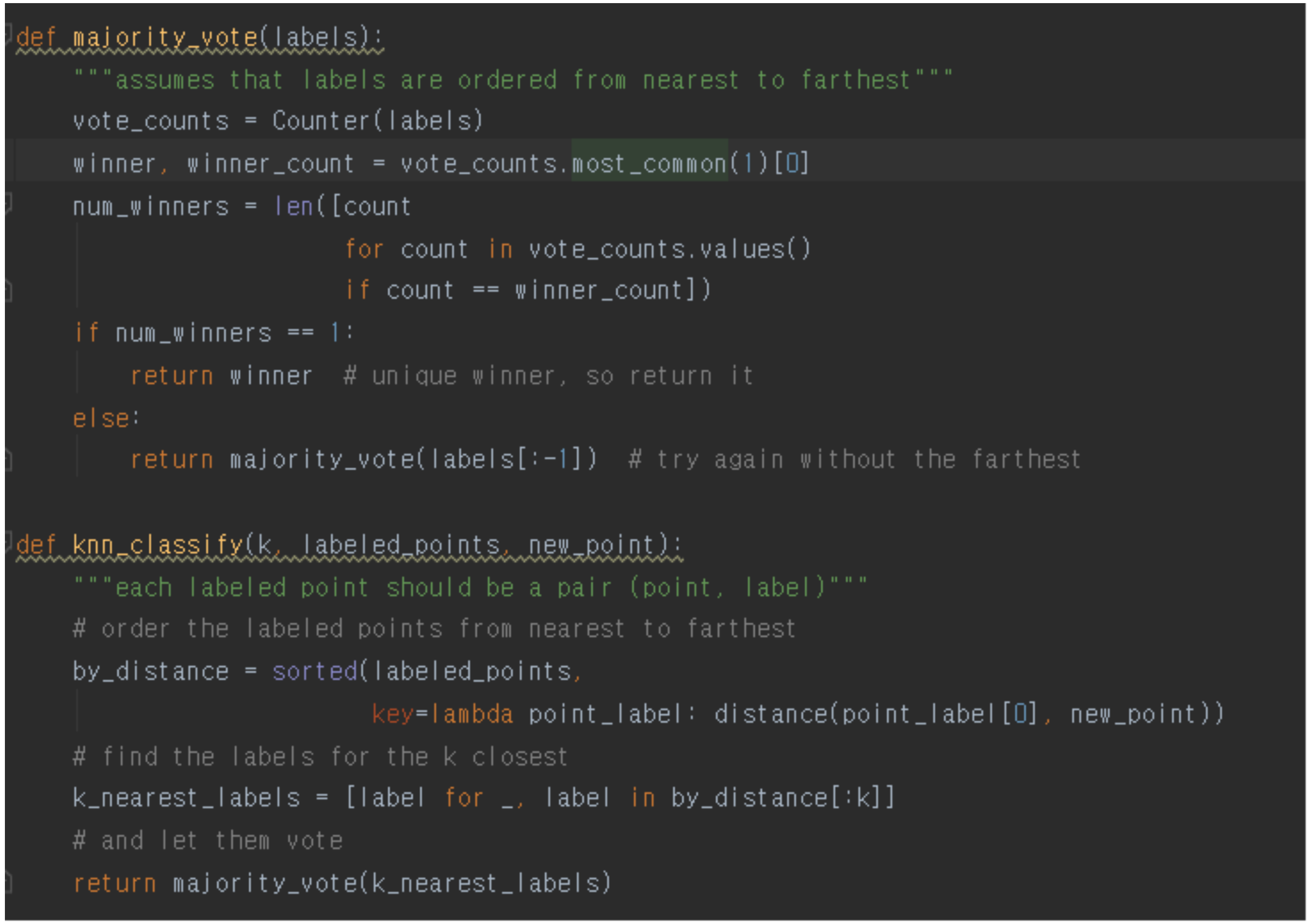
03 분석 결과 설명

시스템 수행 시나리오


for_get_match_result (stock_data)
filtering_not_number (stock_data)
make_labels(filtered_stock, column_indexlist)
divide_training_and_test_label(labeled_target_stock, len_training_data)
training_data(training_label)
training_data(test_label)
get_match_result(result_accuracy_training, result_accuracy_test)
save_data_to_csvfile

03 분석 결과 설명

시스템 수행 시나리오

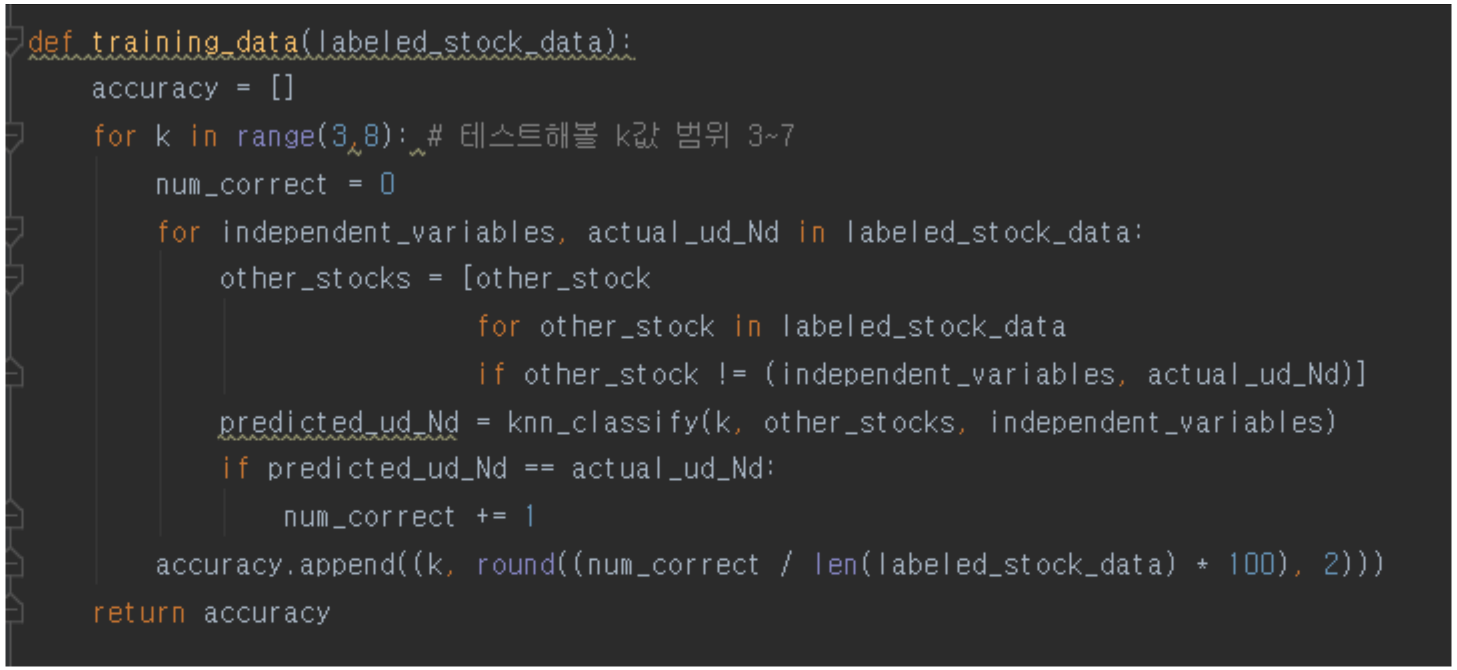


```python
def majority_vote(labels):
    """assumes that labels are ordered from nearest to farthest"""
    vote_counts = Counter(labels)
    winner, winner_count = vote_counts.most_common(1)[0]
    num_winners = len([count
                       for count in vote_counts.values()
                       if count == winner_count])
    if num_winners == 1:
        return winner  # unique winner, so return it
    else:
        return majority_vote(labels[:-1])  # try again without the farthest

def knn_classify(k, labeled_points, new_point):
    """each labeled point should be a pair (point, label)"""
    # order the labeled points from nearest to farthest
    by_distance = sorted(labeled_points,
                         key=lambda point_label: distance(point_label[0], new_point))
    # find the labels for the k closest
    k_nearest_labels = [label for _, label in by_distance[:k]]
    # and let them vote
    return majority_vote(k_nearest_labels)
```

03 분석 결과 설명

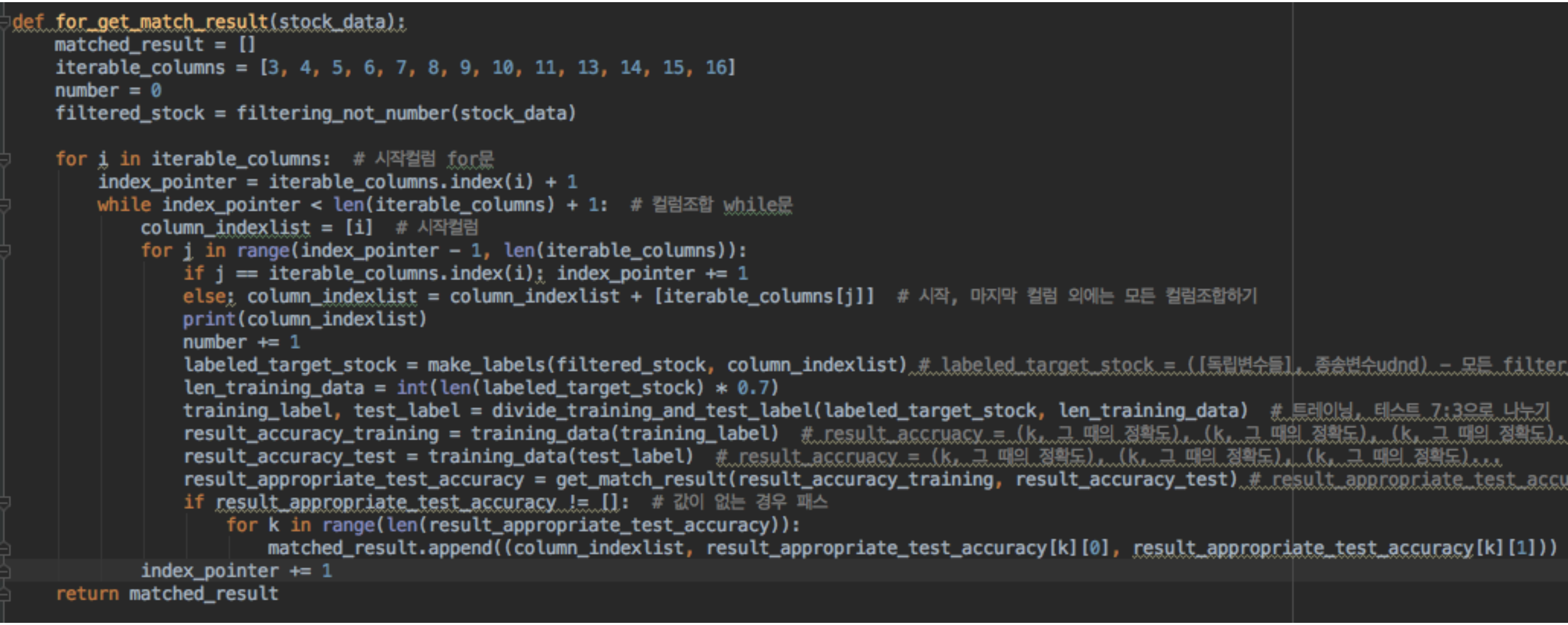
시스템 수행 시나리오



```
def training_data(labeled_stock_data):
    accuracy = []
    for k in range(3,8): # 테스트해볼 k값 범위 3~7
        num_correct = 0
        for independent_variables, actual_ud_Nd in labeled_stock_data:
            other_stocks = [other_stock
                            for other_stock in labeled_stock_data
                            if other_stock != (independent_variables, actual_ud_Nd)]
            predicted_ud_Nd = knn_classify(k, other_stocks, independent_variables)
            if predicted_ud_Nd == actual_ud_Nd:
                num_correct += 1
        accuracy.append((k, round((num_correct / len(labeled_stock_data) * 100), 2)))
    return accuracy
```

03 분석 결과 설명

시스템 수행 시나리오

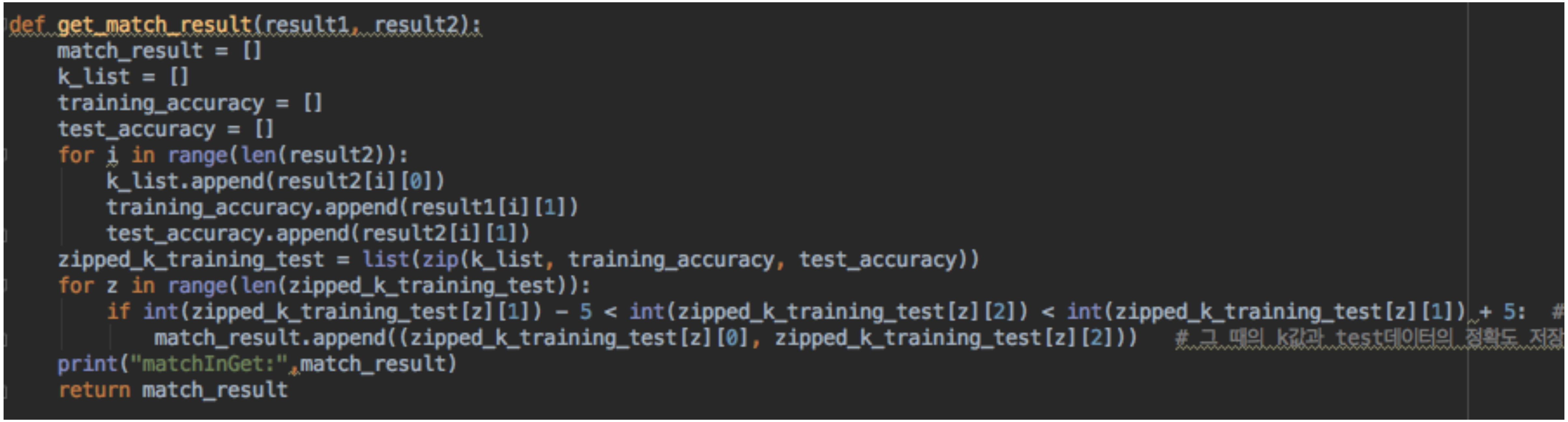


```python
def for_get_match_result(stock_data):
    matched_result = []
    iterable_columns = [3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16]
    number = 0
    filtered_stock = filtering_not_number(stock_data)

    for i in iterable_columns:  # 시작컬럼 for문
        index_pointer = iterable_columns.index(i) + 1
        while index_pointer < len(iterable_columns) + 1:  # 컬럼조합 while문
            column_indexlist = [i]  # 시작컬럼
            for j in range(index_pointer - 1, len(iterable_columns)):
                if j == iterable_columns.index(i): index_pointer += 1
                else: column_indexlist = column_indexlist + [iterable_columns[j]]  # 시작, 마지막 컬럼 외에는 모든 컬럼조합하기
                print(column_indexlist)
                number += 1
                labeled_target_stock = make_labels(filtered_stock, column_indexlist) # labeled_target_stock = ([독립변수들], 종속변수udnd) - 모든 filter
                len_training_data = int(len(labeled_target_stock) * 0.7)
                training_label, test_label = divide_training_and_test_label(labeled_target_stock, len_training_data)  # 트레이닝, 테스트 7:3으로 나누기
                result_accuracy_training = training_data(training_label)  # result_accruacy = (k, 그 때의 정확도), (k, 그 때의 정확도), (k, 그 때의 정확도),
                result_accuracy_test = training_data(test_label)  # result_accruacy = (k, 그 때의 정확도), (k, 그 때의 정확도), (k, 그 때의 정확도)...
                result_appropriate_test_accuracy = get_match_result(result_accuracy_training, result_accuracy_test) # result_appropriate_test_accu
                if result_appropriate_test_accuracy != []:  # 값이 없는 경우 패스
                    for k in range(len(result_appropriate_test_accuracy)):
                        matched_result.append((column_indexlist, result_appropriate_test_accuracy[k][0], result_appropriate_test_accuracy[k][1]))
            index_pointer += 1
    return matched_result
```

03 분석 결과 설명

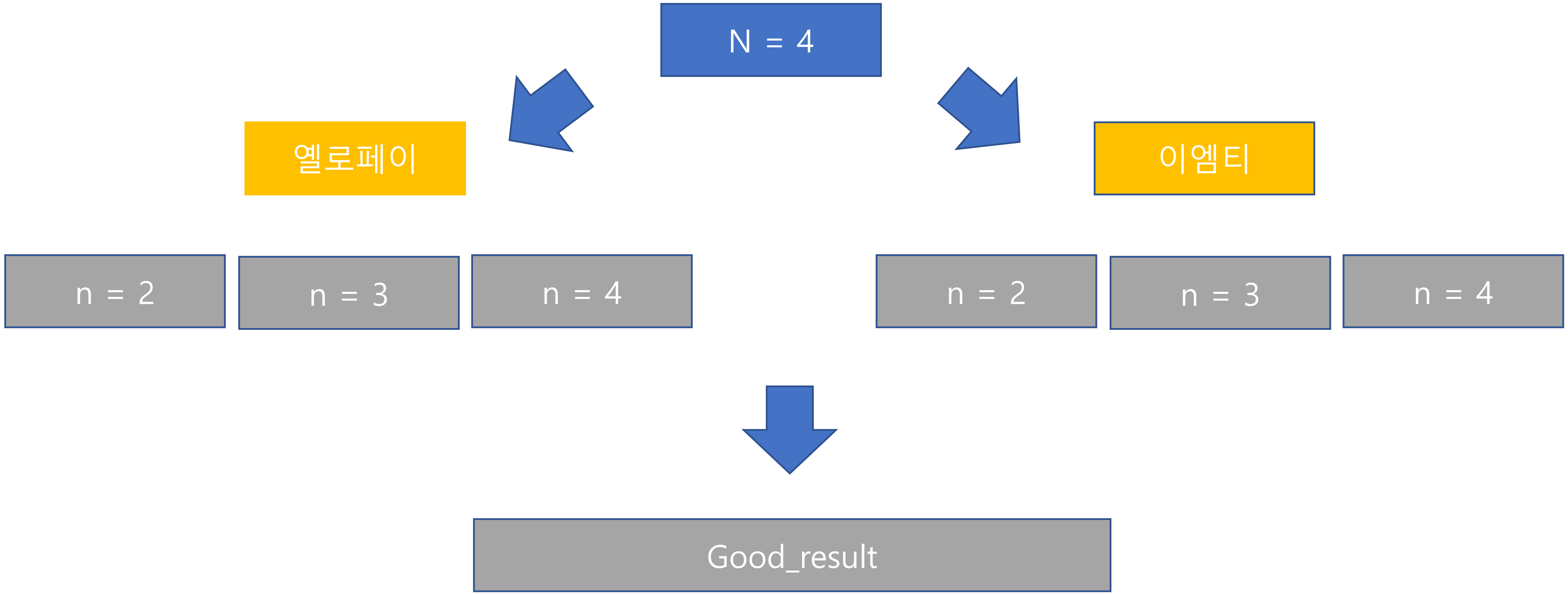
시스템 수행 시나리오



```python
def get_match_result(result1, result2):
    match_result = []
    k_list = []
    training_accuracy = []
    test_accuracy = []
    for i in range(len(result2)):
        k_list.append(result2[i][0])
        training_accuracy.append(result1[i][1])
        test_accuracy.append(result2[i][1])
    zipped_k_training_test = list(zip(k_list, training_accuracy, test_accuracy))
    for z in range(len(zipped_k_training_test)):
        if int(zipped_k_training_test[z][1]) - 5 < int(zipped_k_training_test[z][2]) < int(zipped_k_training_test[z][1]) + 5:   #
            match_result.append((zipped_k_training_test[z][0], zipped_k_training_test[z][2]))   # 그 때의 k값과 test데이터의 정확도 저장
    print("matchInGet:",match_result)
    return match_result
```

03 분석 결과 설명

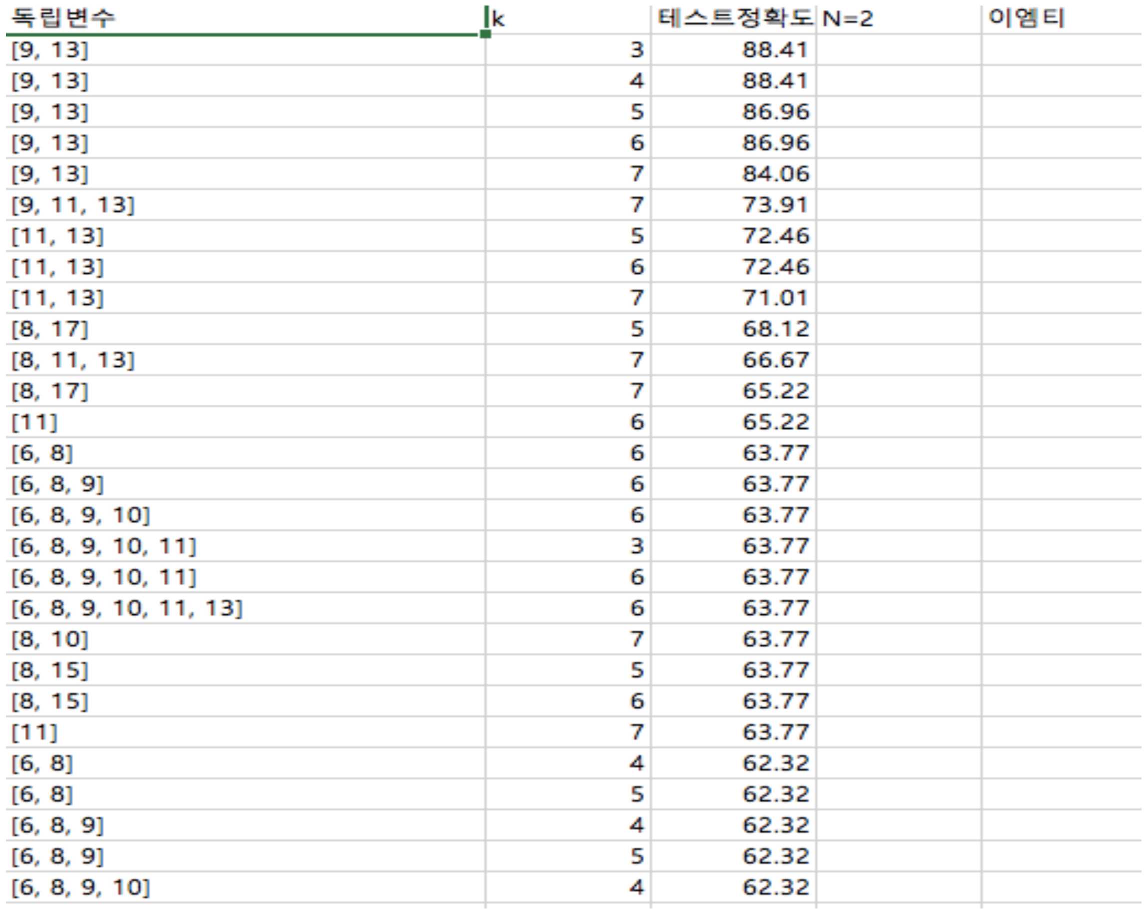
분석 시나리오



03 분석 결과 설명

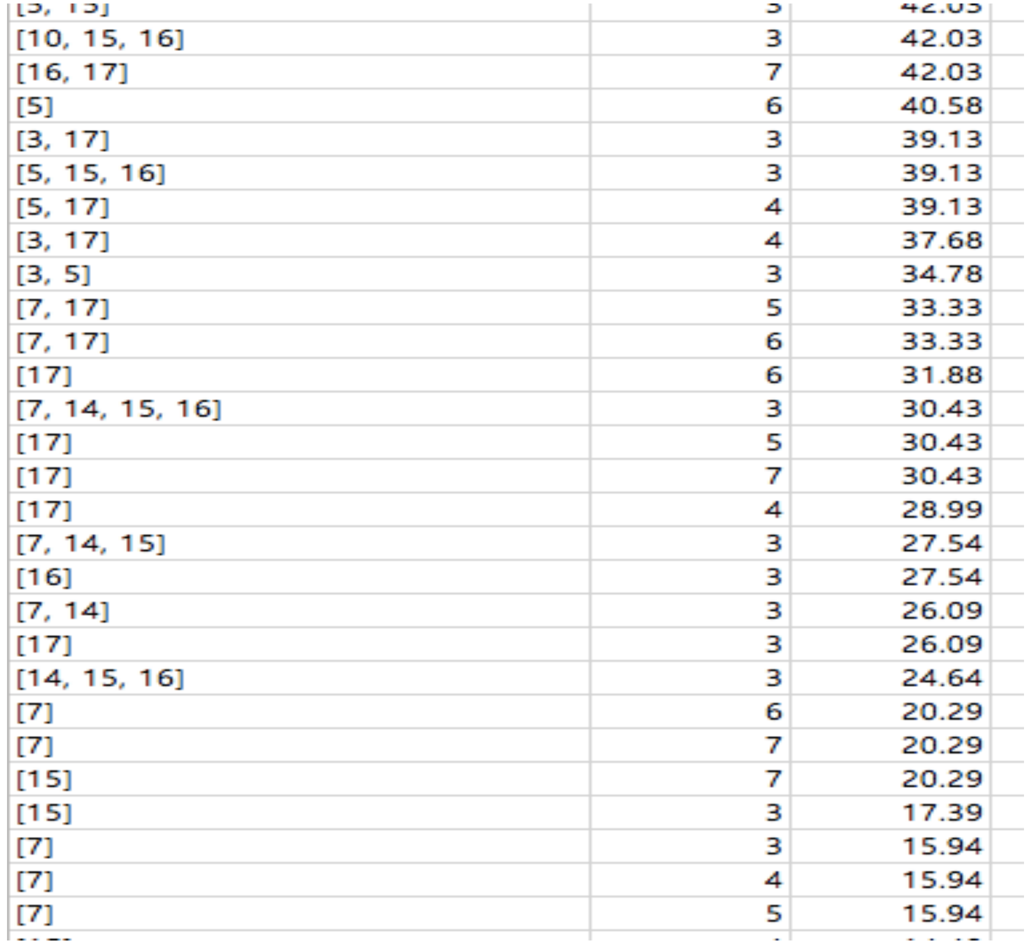
이엠티 (N – 2)

좋은 결과값



독립변수	k	테스트정확도	N=2	이엠티
[9, 13]	3	88.41		
[9, 13]	4	88.41		
[9, 13]	5	86.96		
[9, 13]	6	86.96		
[9, 13]	7	84.06		
[9, 11, 13]	7	73.91		
[11, 13]	5	72.46		
[11, 13]	6	72.46		
[11, 13]	7	71.01		
[8, 17]	5	68.12		
[8, 11, 13]	7	66.67		
[8, 17]	7	65.22		
[11]	6	65.22		
[6, 8]	6	63.77		
[6, 8, 9]	6	63.77		
[6, 8, 9, 10]	6	63.77		
[6, 8, 9, 10, 11]	3	63.77		
[6, 8, 9, 10, 11]	6	63.77		
[6, 8, 9, 10, 11, 13]	6	63.77		
[8, 10]	7	63.77		
[8, 15]	5	63.77		
[8, 15]	6	63.77		
[11]	7	63.77		
[6, 8]	4	62.32		
[6, 8]	5	62.32		
[6, 8, 9]	4	62.32		
[6, 8, 9]	5	62.32		
[6, 8, 9, 10]	4	62.32		

안좋은 결과 값

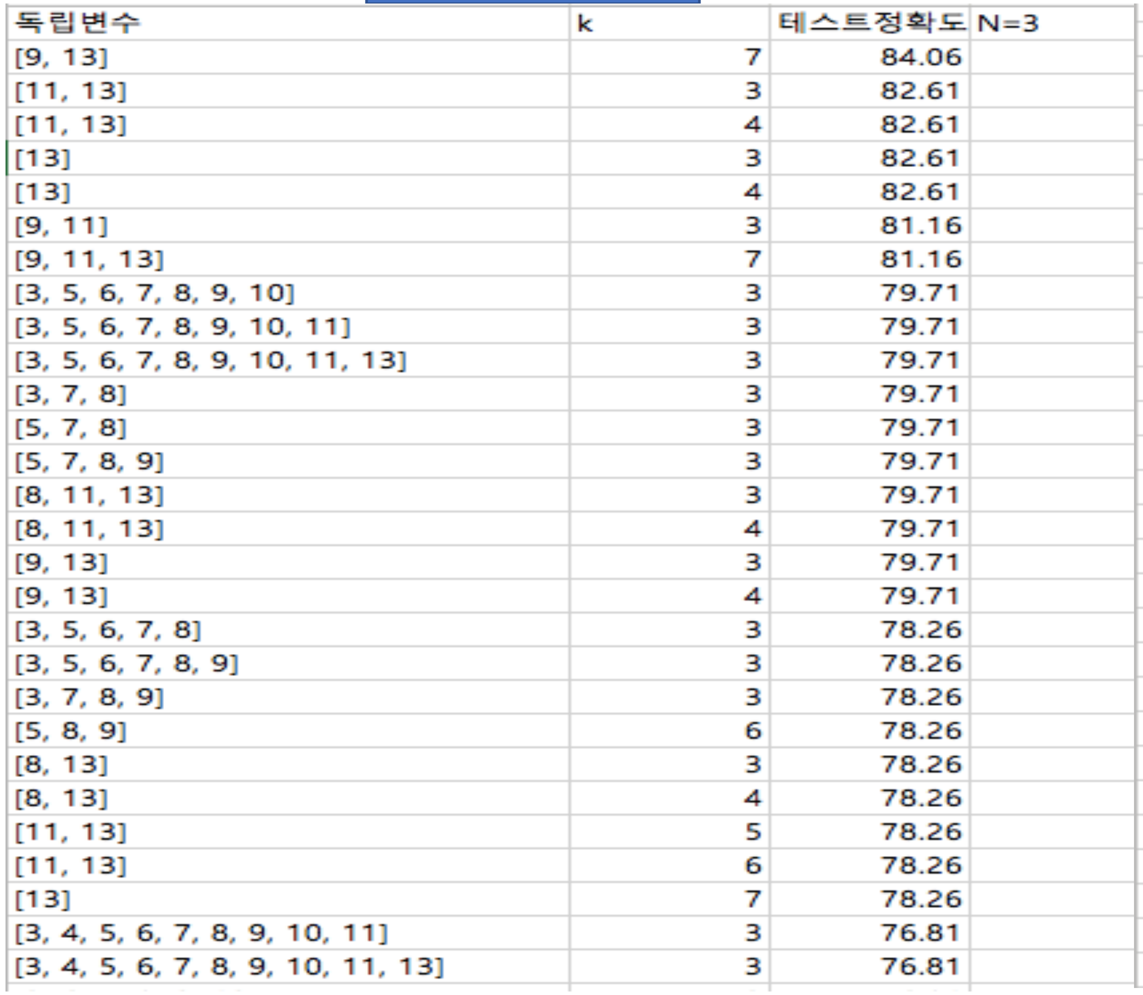


[10, 15, 16]	3	42.03
[16, 17]	7	42.03
[5]	6	40.58
[3, 17]	3	39.13
[5, 15, 16]	3	39.13
[5, 17]	4	39.13
[3, 17]	4	37.68
[3, 5]	3	34.78
[7, 17]	5	33.33
[7, 17]	6	33.33
[17]	6	31.88
[7, 14, 15, 16]	3	30.43
[17]	5	30.43
[17]	7	30.43
[17]	4	28.99
[7, 14, 15]	3	27.54
[16]	3	27.54
[7, 14]	3	26.09
[17]	3	26.09
[14, 15, 16]	3	24.64
[7]	6	20.29
[7]	7	20.29
[15]	7	20.29
[15]	3	17.39
[7]	3	15.94
[7]	4	15.94
[7]	5	15.94

03 분석 결과 설명

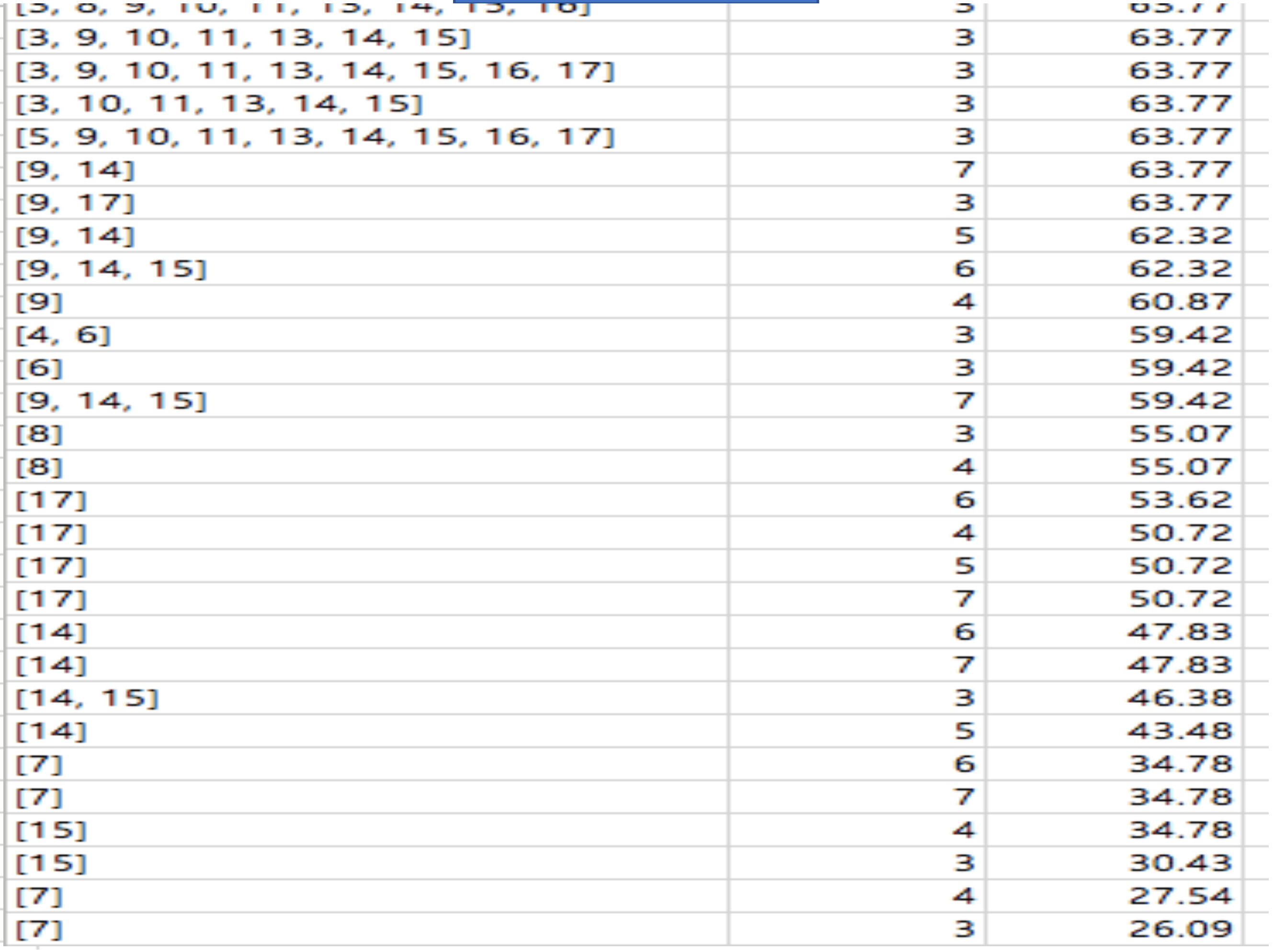
이엠티 (N – 3)

좋은 결과값



독립변수	k	테스트정확도 N=3
[9, 13]	7	84.06
[11, 13]	3	82.61
[11, 13]	4	82.61
[13]	3	82.61
[13]	4	82.61
[9, 11]	3	81.16
[9, 11, 13]	7	81.16
[3, 5, 6, 7, 8, 9, 10]	3	79.71
[3, 5, 6, 7, 8, 9, 10, 11]	3	79.71
[3, 5, 6, 7, 8, 9, 10, 11, 13]	3	79.71
[3, 7, 8]	3	79.71
[5, 7, 8]	3	79.71
[5, 7, 8, 9]	3	79.71
[8, 11, 13]	3	79.71
[8, 11, 13]	4	79.71
[9, 13]	3	79.71
[9, 13]	4	79.71
[3, 5, 6, 7, 8]	3	78.26
[3, 5, 6, 7, 8, 9]	3	78.26
[3, 7, 8, 9]	3	78.26
[5, 8, 9]	6	78.26
[8, 13]	3	78.26
[8, 13]	4	78.26
[11, 13]	5	78.26
[11, 13]	6	78.26
[13]	7	78.26
[3, 4, 5, 6, 7, 8, 9, 10, 11]	3	76.81
[3, 4, 5, 6, 7, 8, 9, 10, 11, 13]	3	76.81

안좋은 결과 값

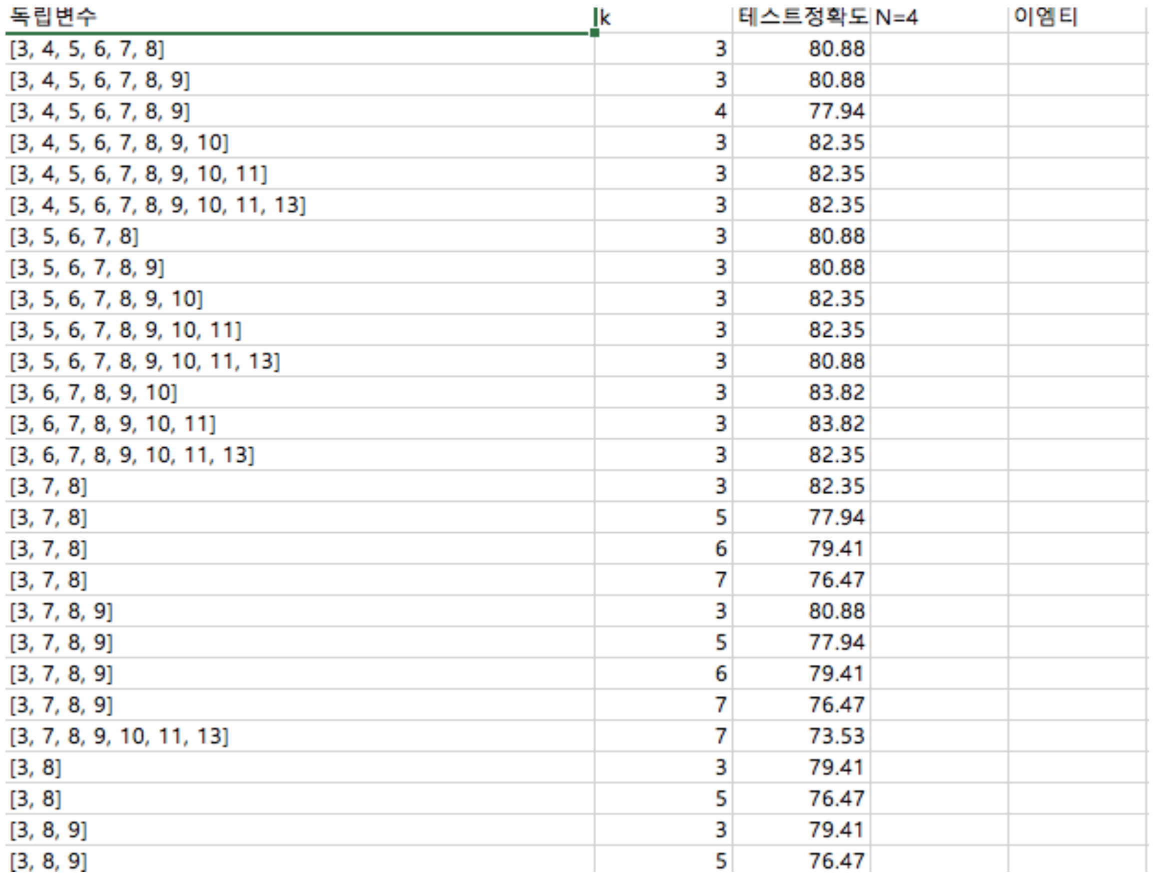


[3, 8, 9, 10, 11, 13, 14, 15, 16]	3	63.77
[3, 9, 10, 11, 13, 14, 15]	3	63.77
[3, 9, 10, 11, 13, 14, 15, 16, 17]	3	63.77
[3, 10, 11, 13, 14, 15]	3	63.77
[5, 9, 10, 11, 13, 14, 15, 16, 17]	3	63.77
[9, 14]	7	63.77
[9, 17]	3	63.77
[9, 14]	5	62.32
[9, 14, 15]	6	62.32
[9]	4	60.87
[4, 6]	3	59.42
[6]	3	59.42
[9, 14, 15]	7	59.42
[8]	3	55.07
[8]	4	55.07
[17]	6	53.62
[17]	4	50.72
[17]	5	50.72
[17]	7	50.72
[14]	6	47.83
[14]	7	47.83
[14, 15]	3	46.38
[14]	5	43.48
[7]	6	34.78
[7]	7	34.78
[15]	4	34.78
[15]	3	30.43
[7]	4	27.54
[7]	3	26.09

03 분석 결과 설명

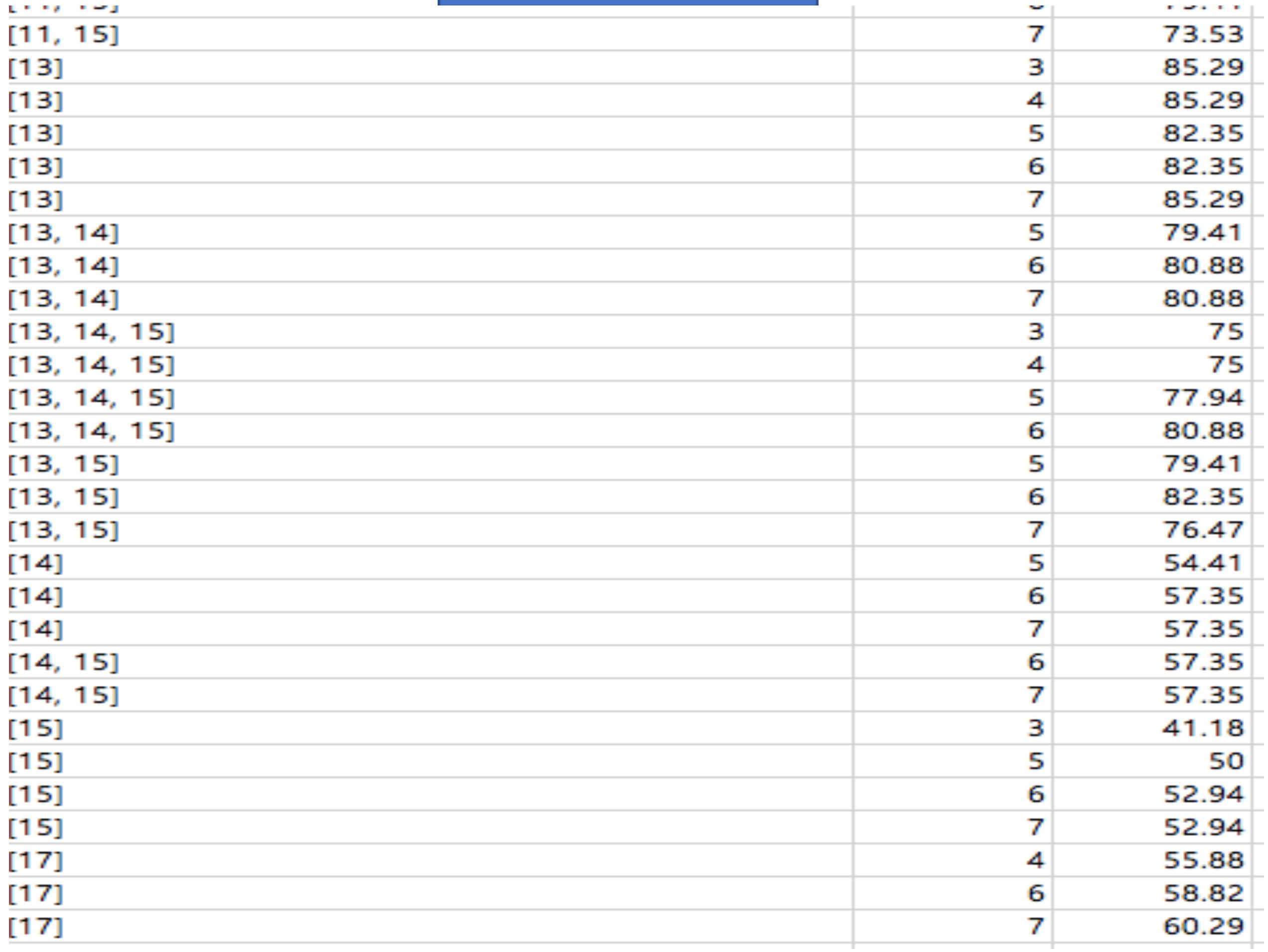
이엠티 (N – 4)

좋은 결과값



독립변수	k	테스트정확도	N=4	이엠티
[3, 4, 5, 6, 7, 8]	3	80.88		
[3, 4, 5, 6, 7, 8, 9]	3	80.88		
[3, 4, 5, 6, 7, 8, 9]	4	77.94		
[3, 4, 5, 6, 7, 8, 9, 10]	3	82.35		
[3, 4, 5, 6, 7, 8, 9, 10, 11]	3	82.35		
[3, 4, 5, 6, 7, 8, 9, 10, 11, 13]	3	82.35		
[3, 5, 6, 7, 8]	3	80.88		
[3, 5, 6, 7, 8, 9]	3	80.88		
[3, 5, 6, 7, 8, 9, 10]	3	82.35		
[3, 5, 6, 7, 8, 9, 10, 11]	3	82.35		
[3, 5, 6, 7, 8, 9, 10, 11, 13]	3	80.88		
[3, 6, 7, 8, 9, 10]	3	83.82		
[3, 6, 7, 8, 9, 10, 11]	3	83.82		
[3, 6, 7, 8, 9, 10, 11, 13]	3	82.35		
[3, 7, 8]	3	82.35		
[3, 7, 8]	5	77.94		
[3, 7, 8]	6	79.41		
[3, 7, 8]	7	76.47		
[3, 7, 8, 9]	3	80.88		
[3, 7, 8, 9]	5	77.94		
[3, 7, 8, 9]	6	79.41		
[3, 7, 8, 9]	7	76.47		
[3, 7, 8, 9, 10, 11, 13]	7	73.53		
[3, 8]	3	79.41		
[3, 8]	5	76.47		
[3, 8, 9]	3	79.41		
[3, 8, 9]	5	76.47		

안좋은 결과 값

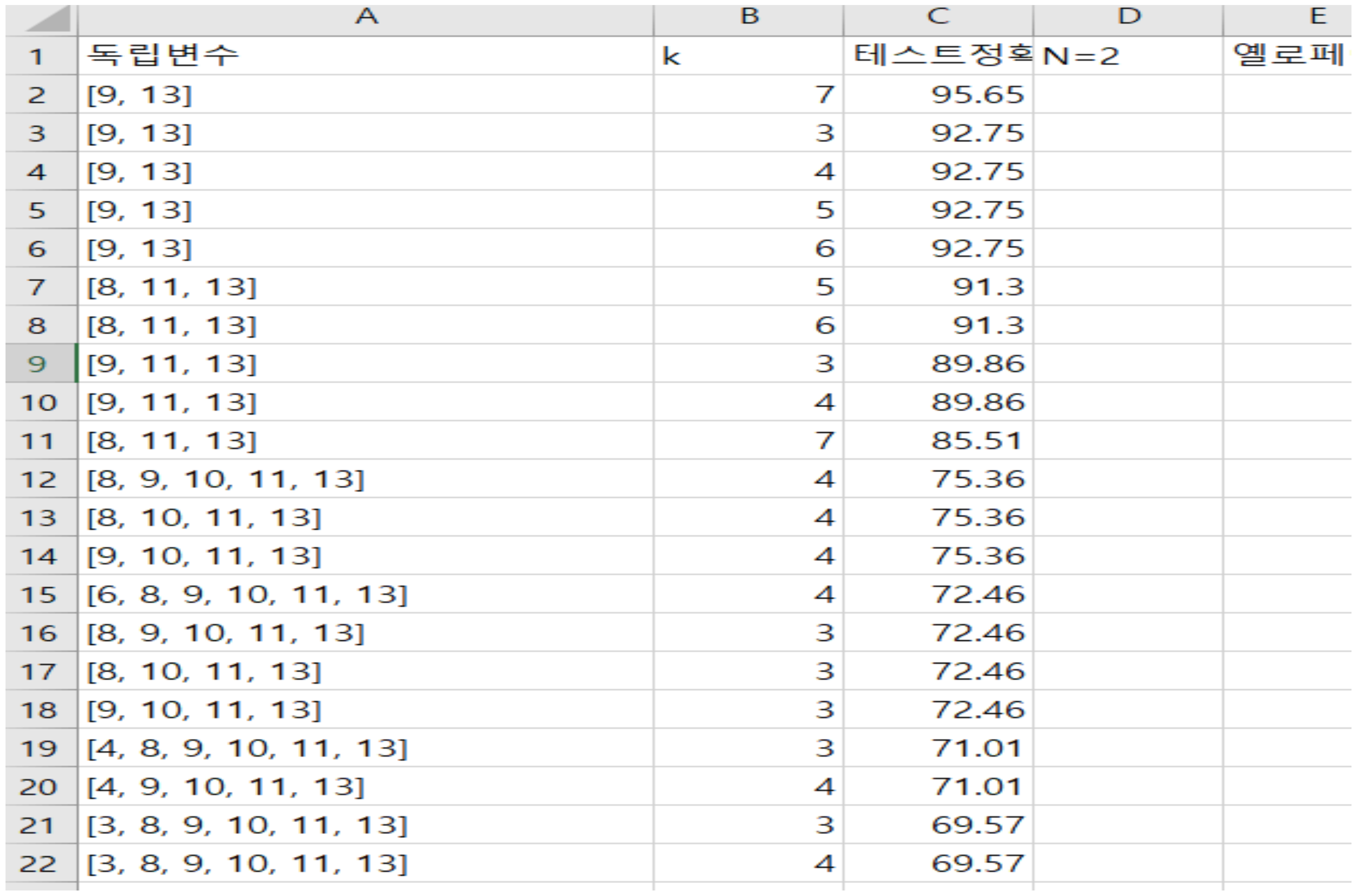


[11, 15]	7	73.53
[13]	3	85.29
[13]	4	85.29
[13]	5	82.35
[13]	6	82.35
[13]	7	85.29
[13, 14]	5	79.41
[13, 14]	6	80.88
[13, 14]	7	80.88
[13, 14, 15]	3	75
[13, 14, 15]	4	75
[13, 14, 15]	5	77.94
[13, 14, 15]	6	80.88
[13, 15]	5	79.41
[13, 15]	6	82.35
[13, 15]	7	76.47
[14]	5	54.41
[14]	6	57.35
[14]	7	57.35
[14, 15]	6	57.35
[14, 15]	7	57.35
[15]	3	41.18
[15]	5	50
[15]	6	52.94
[15]	7	52.94
[17]	4	55.88
[17]	6	58.82
[17]	7	60.29

03 분석 결과 설명

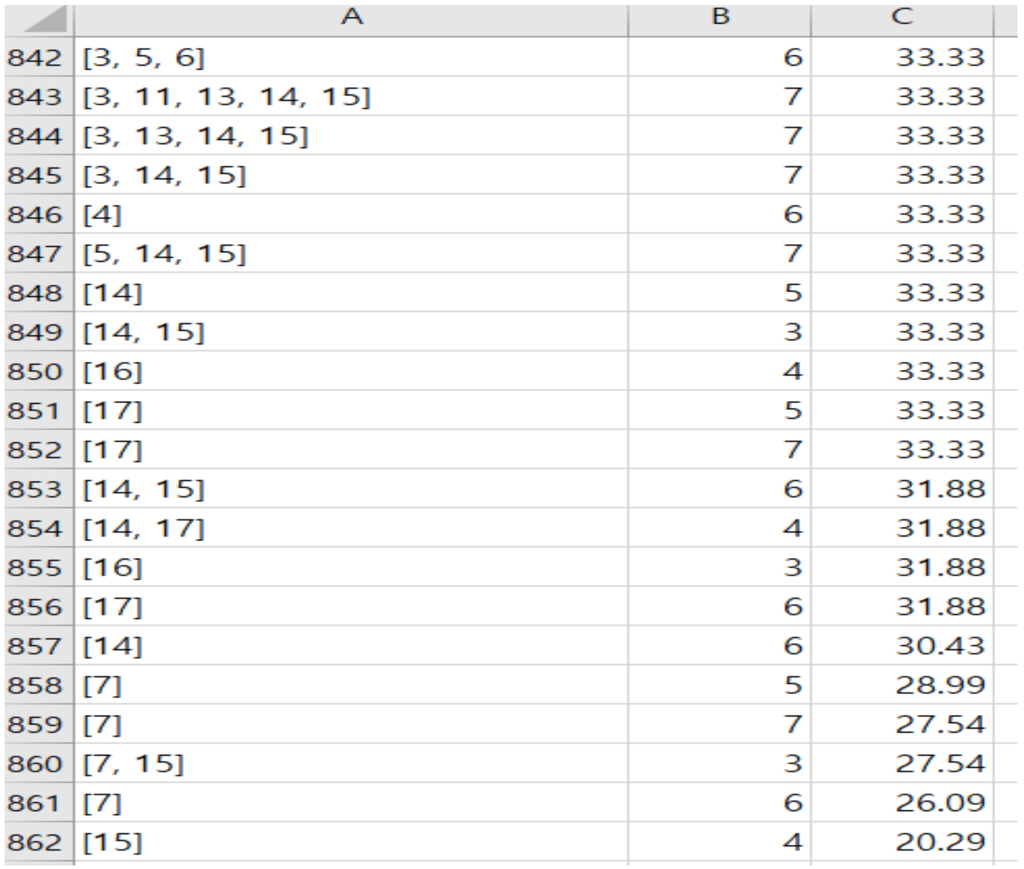
옐로페이 (N – 2)

좋은 결과값



	A	B	C	D	E
1	독립변수	k	테스트정확	N=2	옐로페
2	[9, 13]	7	95.65		
3	[9, 13]	3	92.75		
4	[9, 13]	4	92.75		
5	[9, 13]	5	92.75		
6	[9, 13]	6	92.75		
7	[8, 11, 13]	5	91.3		
8	[8, 11, 13]	6	91.3		
9	[9, 11, 13]	3	89.86		
10	[9, 11, 13]	4	89.86		
11	[8, 11, 13]	7	85.51		
12	[8, 9, 10, 11, 13]	4	75.36		
13	[8, 10, 11, 13]	4	75.36		
14	[9, 10, 11, 13]	4	75.36		
15	[6, 8, 9, 10, 11, 13]	4	72.46		
16	[8, 9, 10, 11, 13]	3	72.46		
17	[8, 10, 11, 13]	3	72.46		
18	[9, 10, 11, 13]	3	72.46		
19	[4, 8, 9, 10, 11, 13]	3	71.01		
20	[4, 9, 10, 11, 13]	4	71.01		
21	[3, 8, 9, 10, 11, 13]	3	69.57		
22	[3, 8, 9, 10, 11, 13]	4	69.57		

안좋은 결과 값

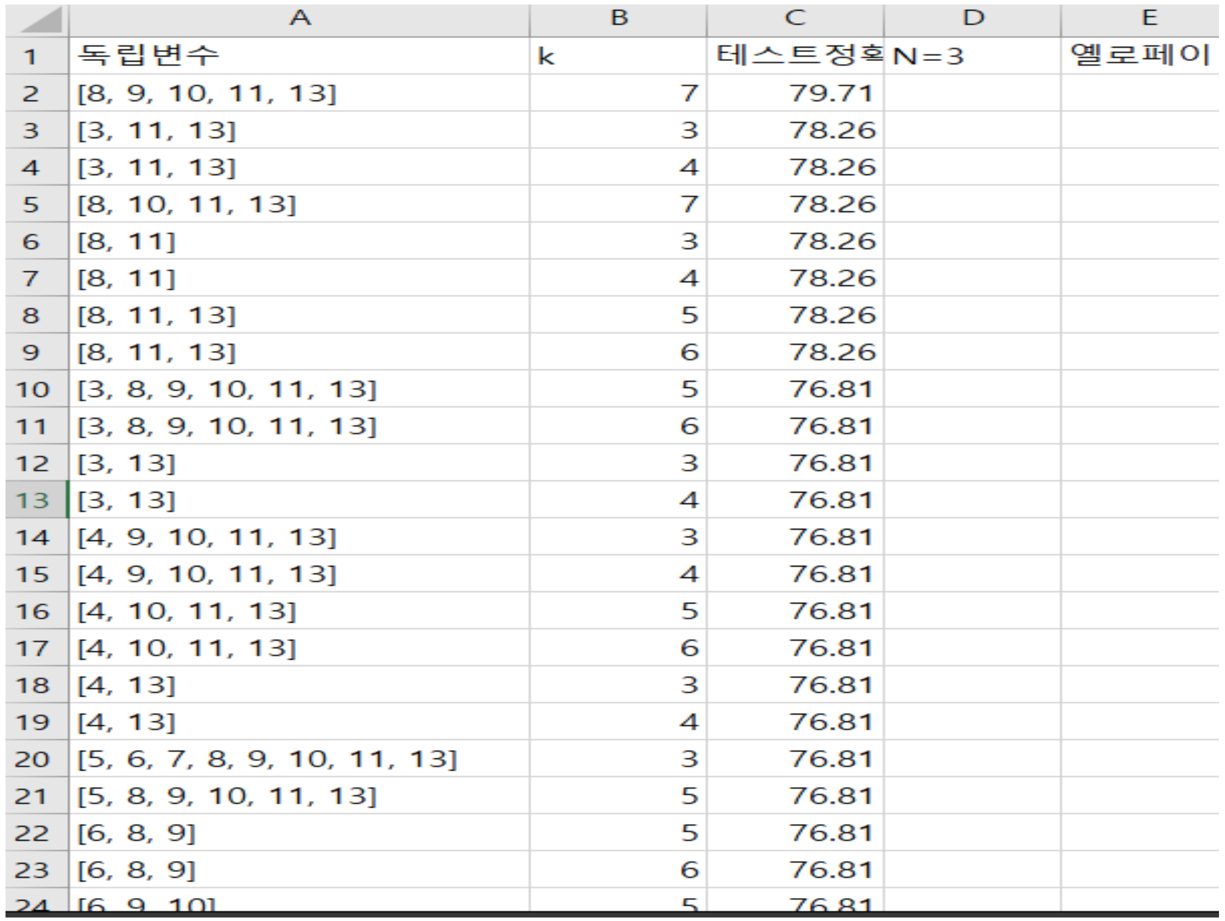


	A	B	C
842	[3, 5, 6]	6	33.33
843	[3, 11, 13, 14, 15]	7	33.33
844	[3, 13, 14, 15]	7	33.33
845	[3, 14, 15]	7	33.33
846	[4]	6	33.33
847	[5, 14, 15]	7	33.33
848	[14]	5	33.33
849	[14, 15]	3	33.33
850	[16]	4	33.33
851	[17]	5	33.33
852	[17]	7	33.33
853	[14, 15]	6	31.88
854	[14, 17]	4	31.88
855	[16]	3	31.88
856	[17]	6	31.88
857	[14]	6	30.43
858	[7]	5	28.99
859	[7]	7	27.54
860	[7, 15]	3	27.54
861	[7]	6	26.09
862	[15]	4	20.29

03 분석 결과 설명

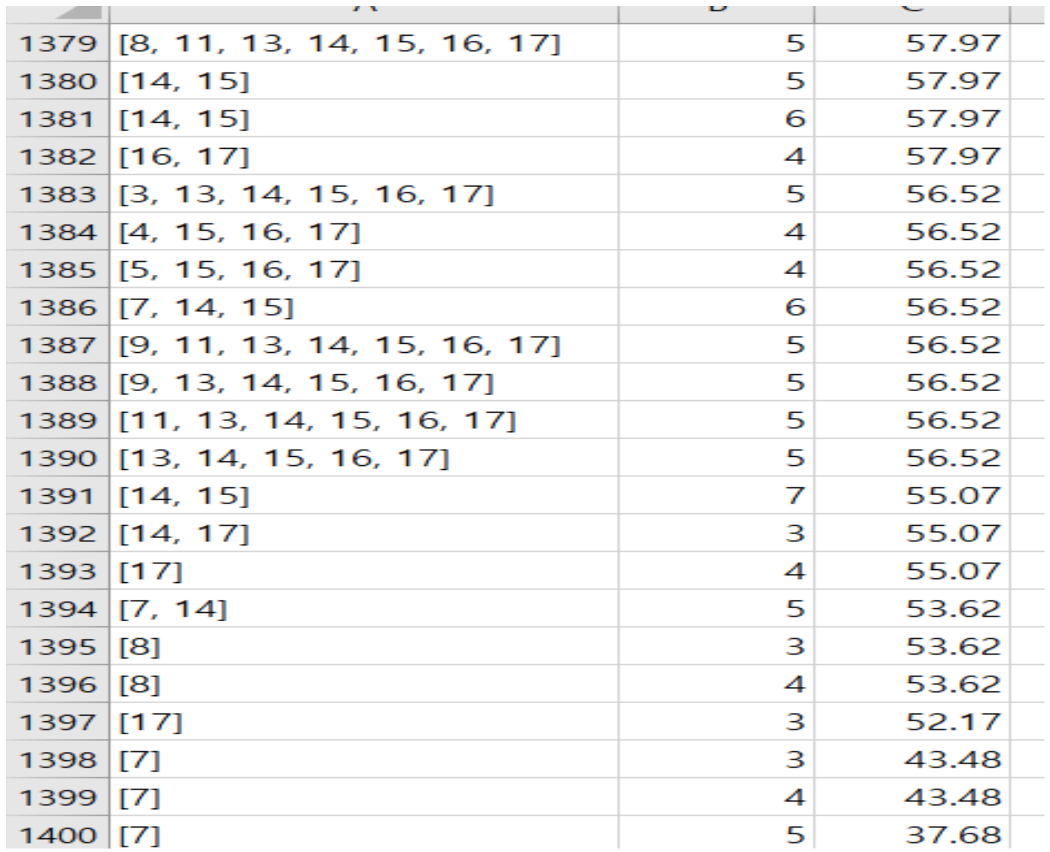
옐로페이 (N – 3)

좋은 결과값



	A	B	C	D	E
1	독립변수	k	테스트정확	N=3	옐로페이
2	[8, 9, 10, 11, 13]	7	79.71		
3	[3, 11, 13]	3	78.26		
4	[3, 11, 13]	4	78.26		
5	[8, 10, 11, 13]	7	78.26		
6	[8, 11]	3	78.26		
7	[8, 11]	4	78.26		
8	[8, 11, 13]	5	78.26		
9	[8, 11, 13]	6	78.26		
10	[3, 8, 9, 10, 11, 13]	5	76.81		
11	[3, 8, 9, 10, 11, 13]	6	76.81		
12	[3, 13]	3	76.81		
13	[3, 13]	4	76.81		
14	[4, 9, 10, 11, 13]	3	76.81		
15	[4, 9, 10, 11, 13]	4	76.81		
16	[4, 10, 11, 13]	5	76.81		
17	[4, 10, 11, 13]	6	76.81		
18	[4, 13]	3	76.81		
19	[4, 13]	4	76.81		
20	[5, 6, 7, 8, 9, 10, 11, 13]	3	76.81		
21	[5, 8, 9, 10, 11, 13]	5	76.81		
22	[6, 8, 9]	5	76.81		
23	[6, 8, 9]	6	76.81		
24	[6, 9, 10]	5	76.81		

안좋은 결과 값

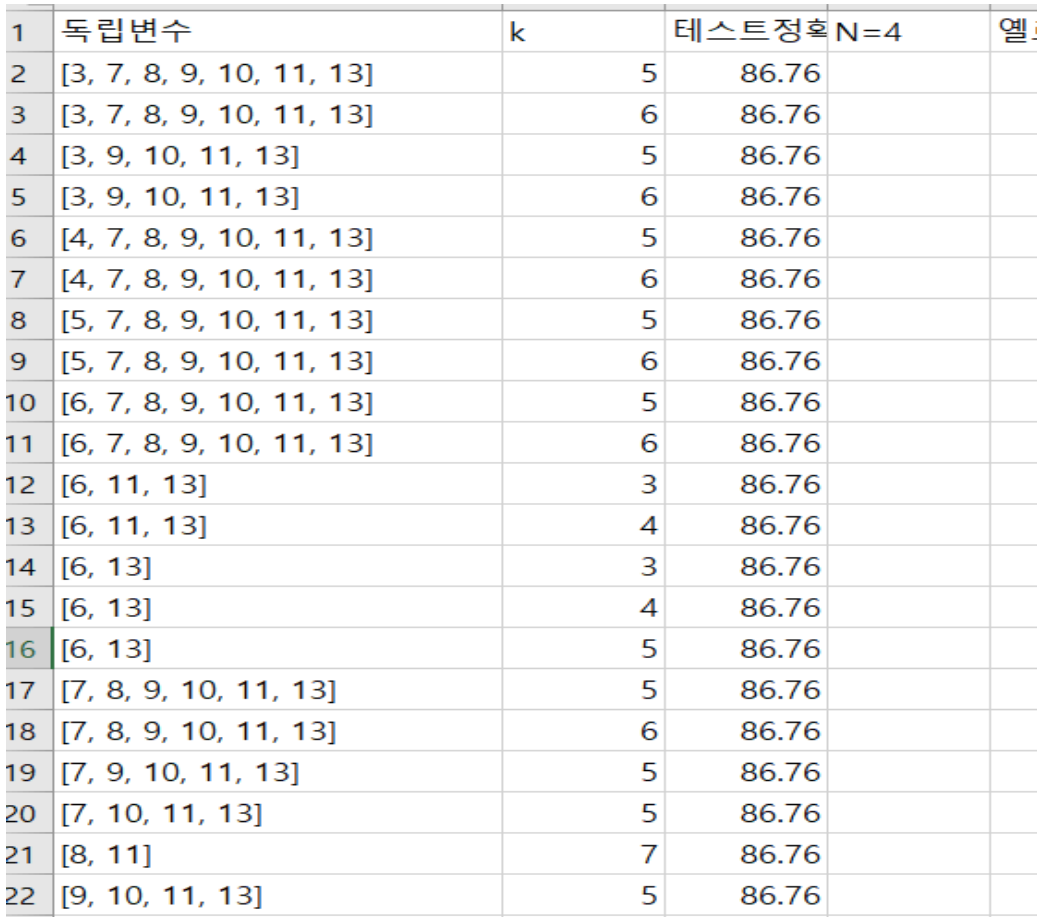


1379	[8, 11, 13, 14, 15, 16, 17]	5	57.97
1380	[14, 15]	5	57.97
1381	[14, 15]	6	57.97
1382	[16, 17]	4	57.97
1383	[3, 13, 14, 15, 16, 17]	5	56.52
1384	[4, 15, 16, 17]	4	56.52
1385	[5, 15, 16, 17]	4	56.52
1386	[7, 14, 15]	6	56.52
1387	[9, 11, 13, 14, 15, 16, 17]	5	56.52
1388	[9, 13, 14, 15, 16, 17]	5	56.52
1389	[11, 13, 14, 15, 16, 17]	5	56.52
1390	[13, 14, 15, 16, 17]	5	56.52
1391	[14, 15]	7	55.07
1392	[14, 17]	3	55.07
1393	[17]	4	55.07
1394	[7, 14]	5	53.62
1395	[8]	3	53.62
1396	[8]	4	53.62
1397	[17]	3	52.17
1398	[7]	3	43.48
1399	[7]	4	43.48
1400	[7]	5	37.68

03 분석 결과 설명

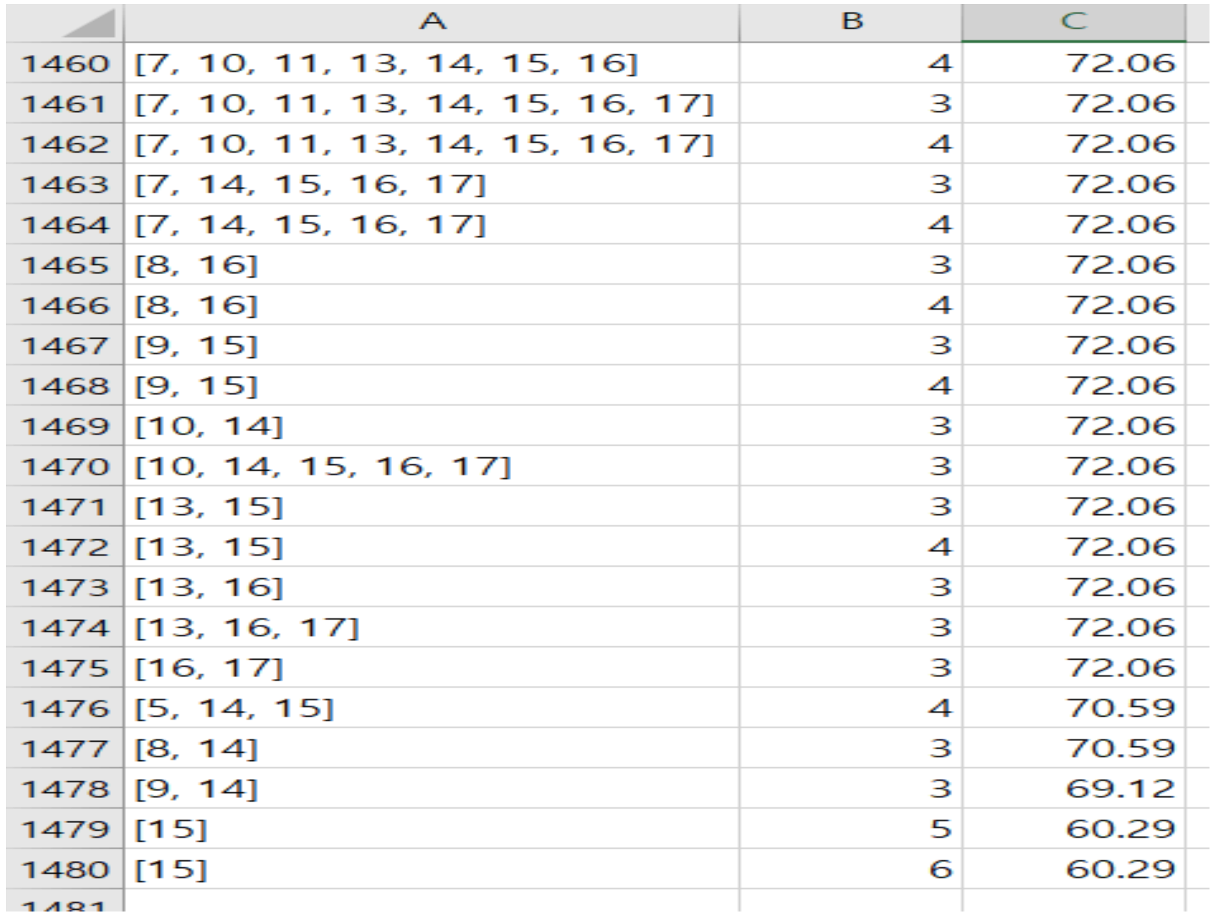
옐로페이 (N – 4)

좋은 결과값



1	독립변수	k	테스트정확	N=4	옐
2	[3, 7, 8, 9, 10, 11, 13]	5	86.76		
3	[3, 7, 8, 9, 10, 11, 13]	6	86.76		
4	[3, 9, 10, 11, 13]	5	86.76		
5	[3, 9, 10, 11, 13]	6	86.76		
6	[4, 7, 8, 9, 10, 11, 13]	5	86.76		
7	[4, 7, 8, 9, 10, 11, 13]	6	86.76		
8	[5, 7, 8, 9, 10, 11, 13]	5	86.76		
9	[5, 7, 8, 9, 10, 11, 13]	6	86.76		
10	[6, 7, 8, 9, 10, 11, 13]	5	86.76		
11	[6, 7, 8, 9, 10, 11, 13]	6	86.76		
12	[6, 11, 13]	3	86.76		
13	[6, 11, 13]	4	86.76		
14	[6, 13]	3	86.76		
15	[6, 13]	4	86.76		
16	[6, 13]	5	86.76		
17	[7, 8, 9, 10, 11, 13]	5	86.76		
18	[7, 8, 9, 10, 11, 13]	6	86.76		
19	[7, 9, 10, 11, 13]	5	86.76		
20	[7, 10, 11, 13]	5	86.76		
21	[8, 11]	7	86.76		
22	[9, 10, 11, 13]	5	86.76		

안좋은 결과 값



	A	B	C
1460	[7, 10, 11, 13, 14, 15, 16]	4	72.06
1461	[7, 10, 11, 13, 14, 15, 16, 17]	3	72.06
1462	[7, 10, 11, 13, 14, 15, 16, 17]	4	72.06
1463	[7, 14, 15, 16, 17]	3	72.06
1464	[7, 14, 15, 16, 17]	4	72.06
1465	[8, 16]	3	72.06
1466	[8, 16]	4	72.06
1467	[9, 15]	3	72.06
1468	[9, 15]	4	72.06
1469	[10, 14]	3	72.06
1470	[10, 14, 15, 16, 17]	3	72.06
1471	[13, 15]	3	72.06
1472	[13, 15]	4	72.06
1473	[13, 16]	3	72.06
1474	[13, 16, 17]	3	72.06
1475	[16, 17]	3	72.06
1476	[5, 14, 15]	4	70.59
1477	[8, 14]	3	70.59
1478	[9, 14]	3	69.12
1479	[15]	5	60.29
1480	[15]	6	60.29
1481			

03 분석 결과 설명

결과

좋은 칼럼

[8]종가 일간 변화량, [9]종가 일간 변화율,
[11]종가의 N일 이동평균의 일간 변화율, [13]N일 간의 종가 상승률

안좋은 칼럼

[7]거래량, [14]거래량 일간 변화량, [15]거래량 일간 변화율

좋은 조합의 칼럼

시작 가 , 종가의 N일 이동 평균의 일간 (변화율, 일간 변화량), N일 간의 종가 상승 률

03 분석 결과 설명

결과

결론 1

N 값이 커질 수록, 트레이닝 데이터와 테스트 데이터의 정확도가 비슷한 경우가 많아짐

결론 2

N 값이 클 수록 최대정확도와 최소정확도의 편차가 작고, 각 정확도마다의 표준편차도 작아짐

결론 3

K값에 따라 트레이닝 데이터와 테스트 데이터의 편차가 달라짐

종속변수udND의 데이터 모델: [cv_diff_value, cv_diff_rate, cv_maN_rate, cvNd_diff_rate]
N=3, k=3

Thank you.