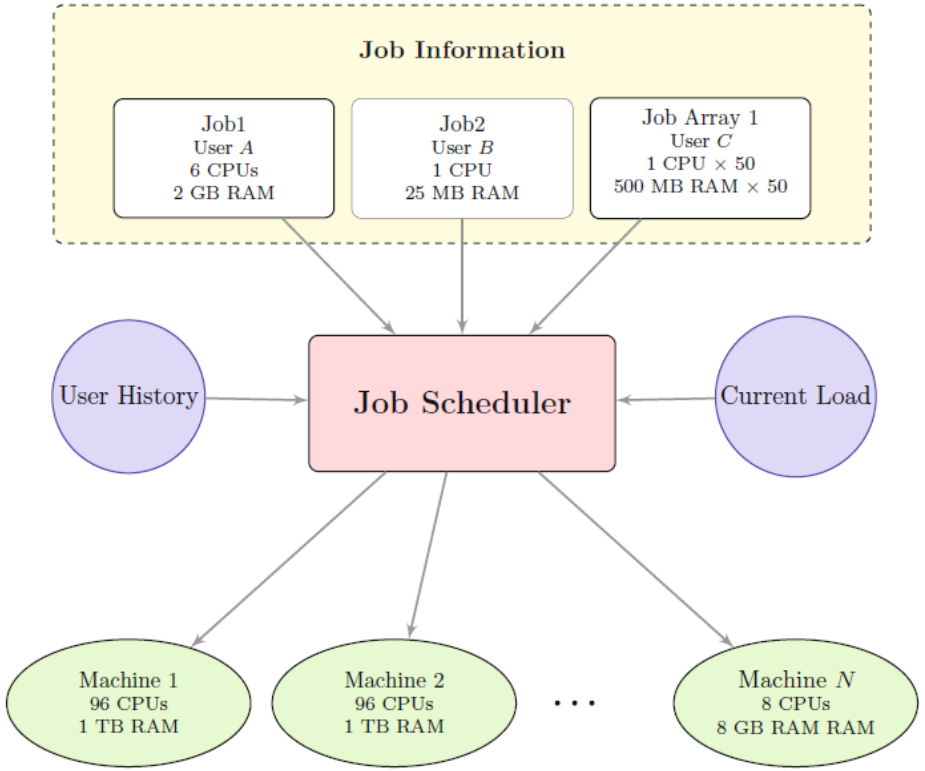
Job Scheduling
Multi-Class Classification Problem

1. 문제 소개
2. 탐색적 자료 분석 · 전처리
3. 모델 튜닝
4. 최종 성능 평가

2019.01.25

3조 김동욱 · 김용호 · 김형면 · 백성훈 · 차주영

1. 문제 소개 – Job Scheduling



Job Scheduler

Multi-Class Classification

- Very Fast
- Fast
- Moderate
- Long

- 사용자는 대용량 처리 연산을 위해 고성능 컴퓨팅(HPC) 환경을 사용
- 사용자가 요구하는 연산은 각기 다른 자원 사용량을 요구
- ‘Job Scheduler’는 각각의 연산에 대해 자원 사용량(작업 속도)을 예측하여 HPC에 작업을 할당해야 함.

그림 출처 : Max Kuhn · Kjell Johnson, *Applied Predictive Modeling ch.17*, 2013.

1. 문제 소개 – Type Ⅰ · Ⅱ Misclassification Error

Confusion Matrix		실제값			
		VF	F	M	L
예측값	VF				
	F				
	M				
	L				

Cost Matrix		실제값			
		VF	F	M	L
예측값	VF	0	1	5	10
	F	1	0	5	5
	M	1	1	0	1
	L	1	1	1	0

Type Ⅰ Error

- 실제로는 빠른 연산 작업을 느린 작업으로 분류한 경우
- 컴퓨팅 자원의 낭비가 발생하지만 심각한 오류는 아님.

Type Ⅱ Error

- 실제로는 느린 연산 작업을 빠른 작업으로 분류한 경우
- System 전체를 느리게 만들 수 있는 심각한 오류

- 정확도(Accuracy) 뿐만 아니라 비용 함수에 따라서 최적 모델 선택이 필요
- Confusion Matrix와 Cost Matrix를 바탕으로 각 모델 별 Cost 도출 후 모델 평가

자료 출처 : Max Kuhn · Kjell Johnson, *Applied Predictive Modeling ch.17*, 2013.

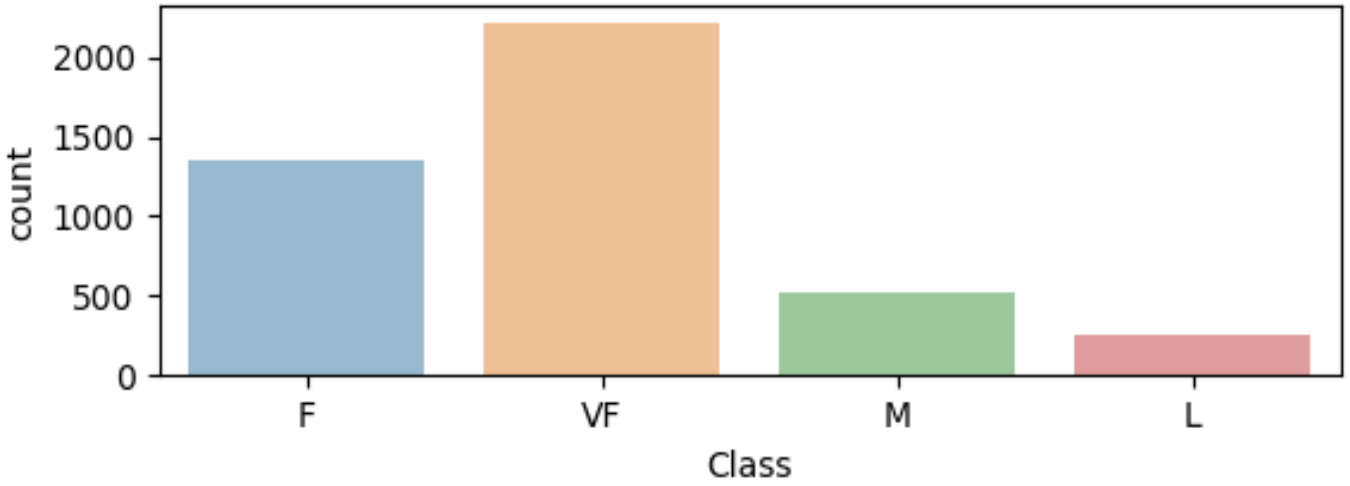
2. 탐색적 자료분석 · 전처리 – 자료 및 변수 설명

데이터셋	데이터 설명	제약 회사가 HPC 환경에서 행한 작업 정보
	데이터 크기	8개 변수에 대한 4331개 데이터 (4331, 8)

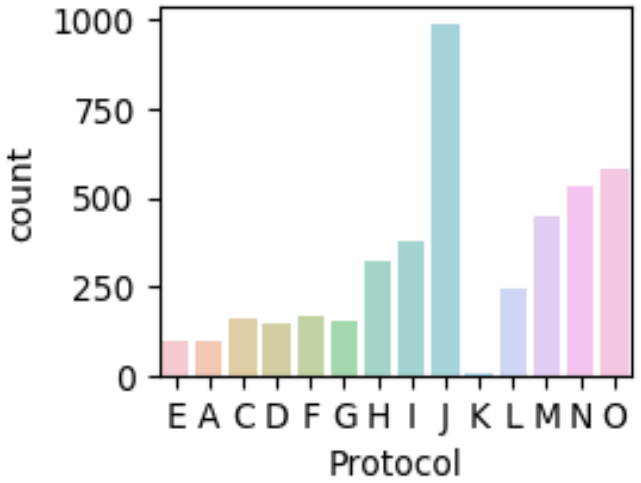
변수명	내용	데이터 타입	비고
Protocol	화합물의 결과를 계산하는 여러 가지 분석 기술	Object	A ~ O (B 제외)
Compounds	작업 과정에 사용되는 화합물 수	Int64	
InputFields	분석 목적에 따른 데이터 입력 항목의 수	Int64	
Iterations	프로토콜 반복 횟수	Int64	
NumPending	지연 작업 횟수 (연산 지시 당시 작업 환경의 부하)	Int64	
Hour	일이 시작된 시각	Float64	
Day	작업이 실행된 요일	Object	월요일 ~ 일요일
Class	작업 속도	Object	VF, F, M, L

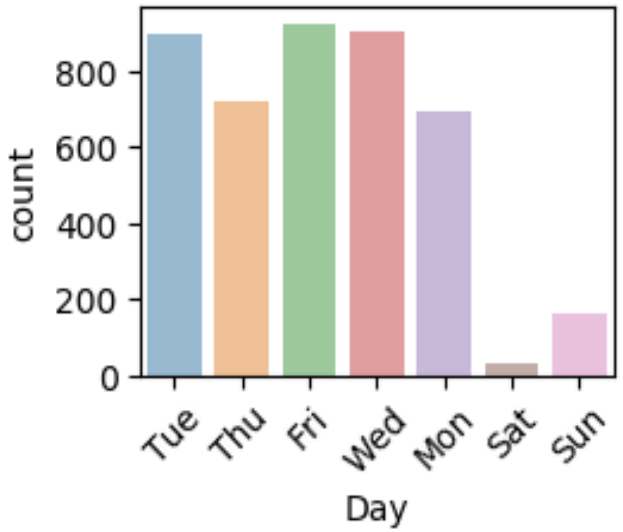
자료 출처 : Max Kuhn · Kjell Johnson, *Applied Predictive Modeling ch.17,* 2013.

2. 탐색적 자료분석 · 전처리 – 범주형 변수



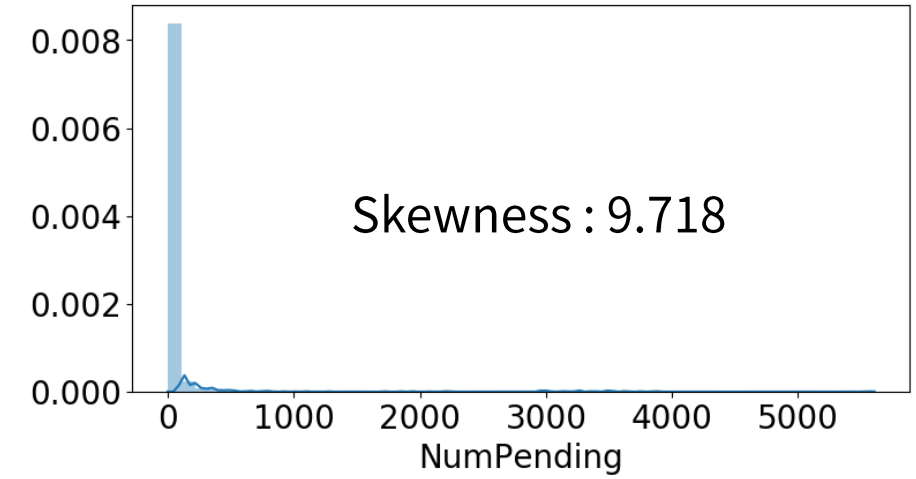
- 순서형 변수
- 빠르기에 따라 맵핑 (1~4)

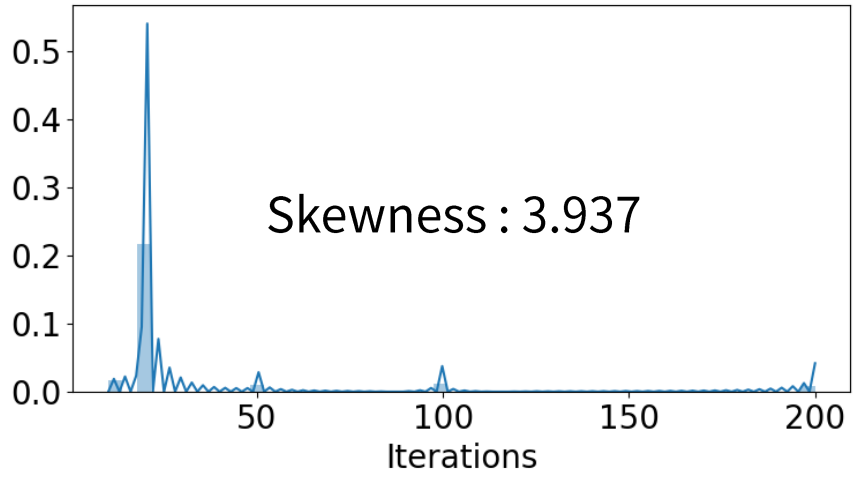


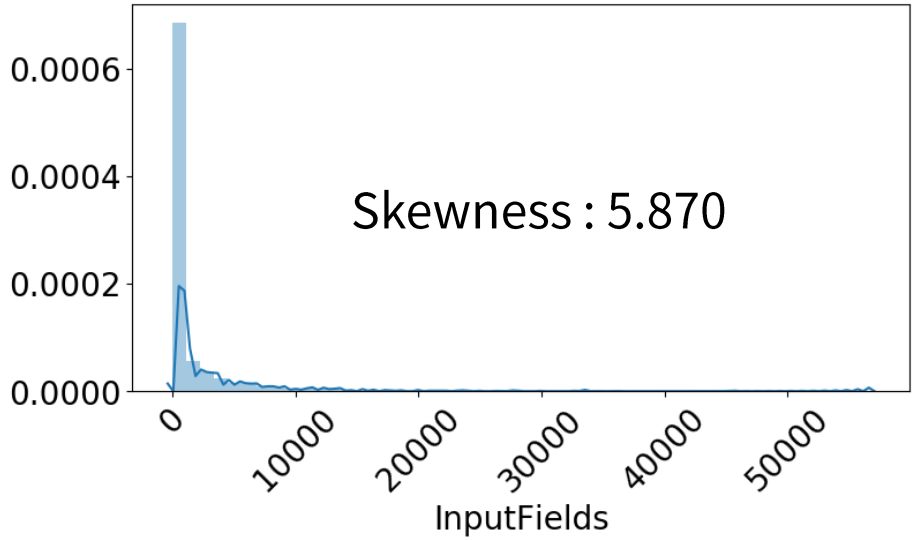


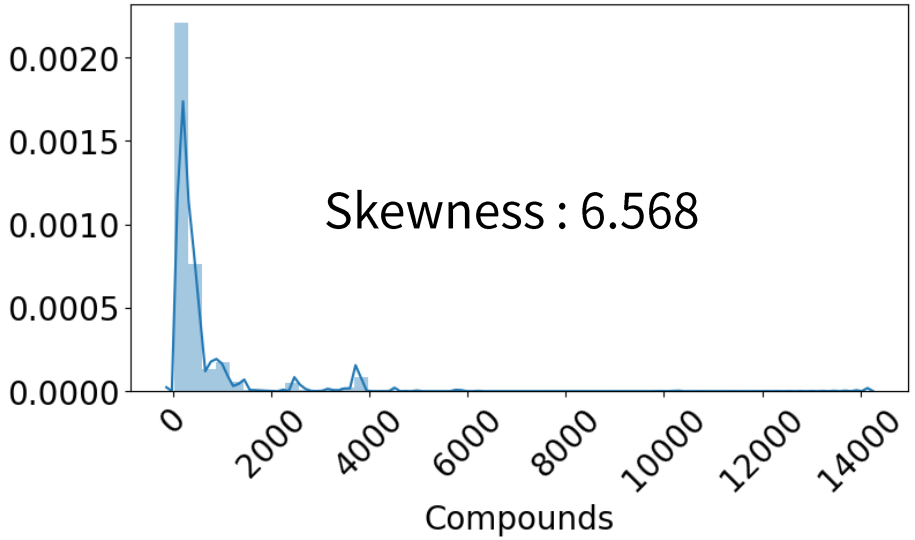
- 범주형 변수
- 가변수화

2. 탐색적 자료분석 · 전처리 – 숫자형 변수(1)



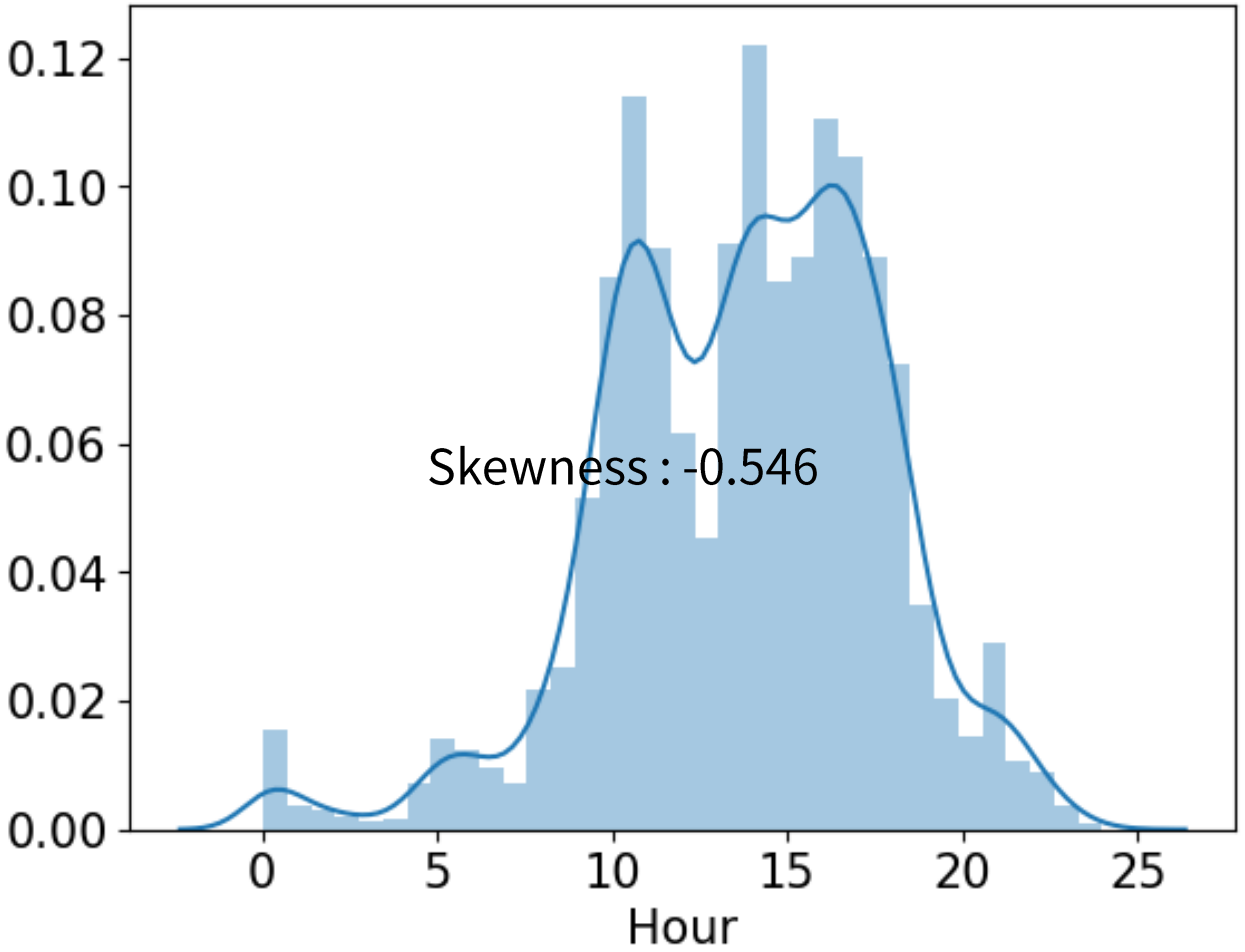






- 오른쪽으로 긴 꼬리를 가지는 숫자형 변수에 대해 log 변환

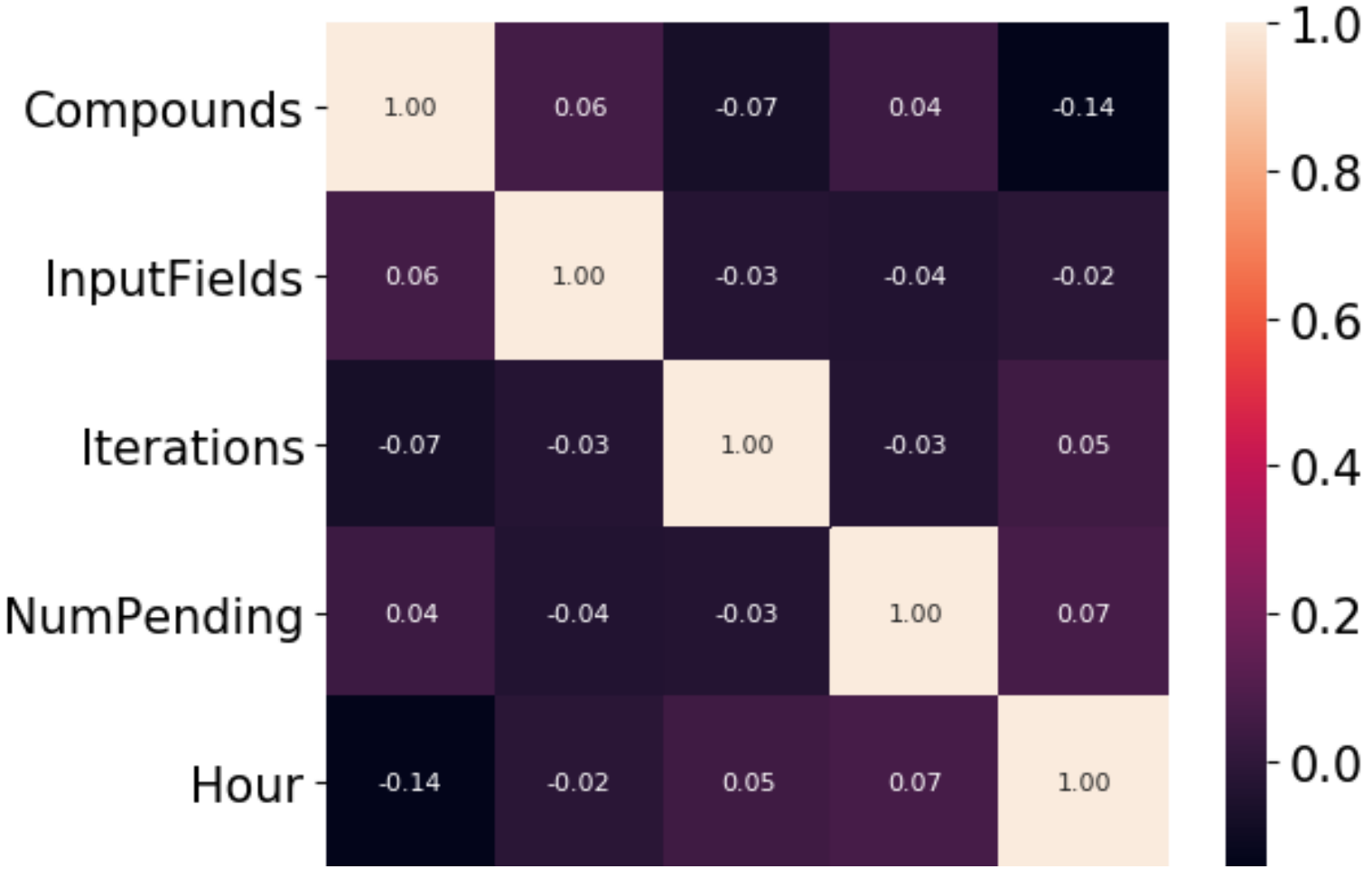
2. 탐색적 자료분석 · 전처리 – 숫자형 변수(2)



- ‘Hour’ 변수에 대한 log 변환은 불필요한 것으로 보임.

자료 출처 : Max Kuhn · Kjell Johnson, *Applied Predictive Modeling ch.17,* 2013.

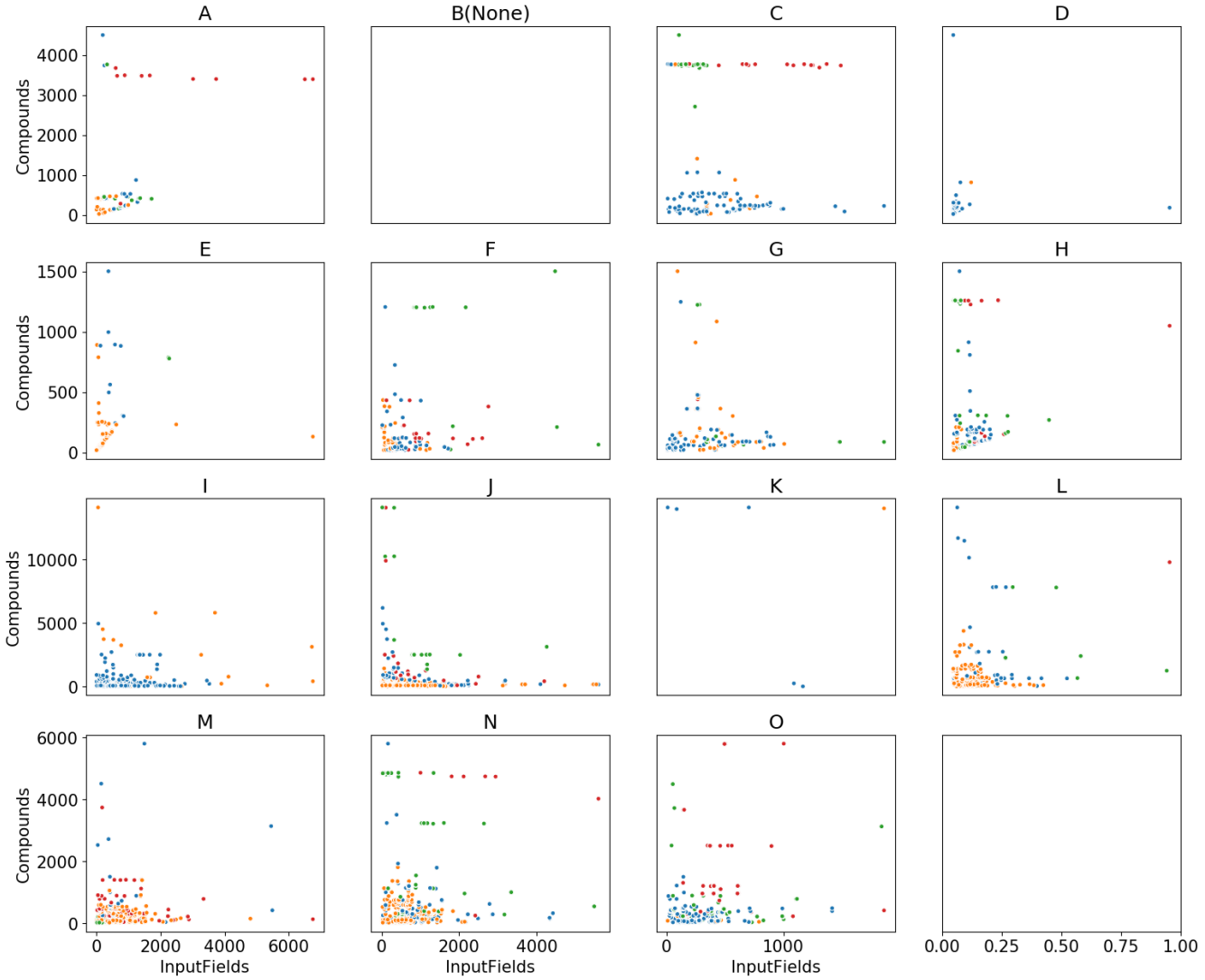
2. 탐색적 자료분석 · 전처리 – 상관관계 확인



- 숫자형 변수들 간에 큰 상관관계는 나타나지 않음.

2. 탐색적 자료분석 · 전처리 – 상관관계 확인

<‘Protocol’에 따른 ‘InputFields’와 ‘Compounds’의 산점도>



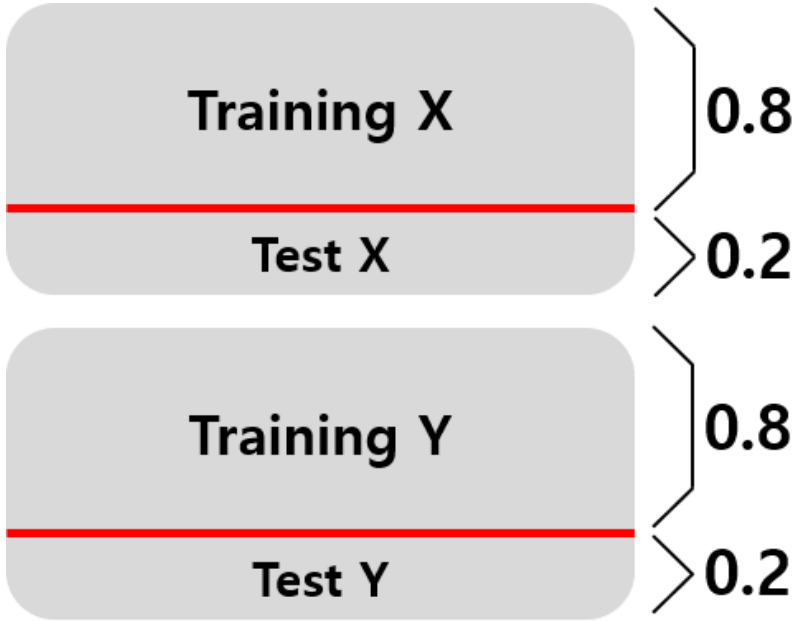
- 제약회사의 자료이므로 특정 변수 간의 상관관계 확인
- 특징적인 연관성을 발견하기 어려운 것으로 나타남.

2. 탐색적 자료분석 · 전처리 – Null 값 확인

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4331 entries, 0 to 4330
Data columns (total 8 columns):
Protocol        4331 non-null object
Compounds       4331 non-null int64
InputFields     4331 non-null int64
Iterations      4331 non-null int64
NumPending      4331 non-null int64
Hour            4331 non-null float64
Day             4331 non-null object
Class           4331 non-null object
dtypes: float64(1), int64(4), object(3)
memory usage: 270.8+ KB
```

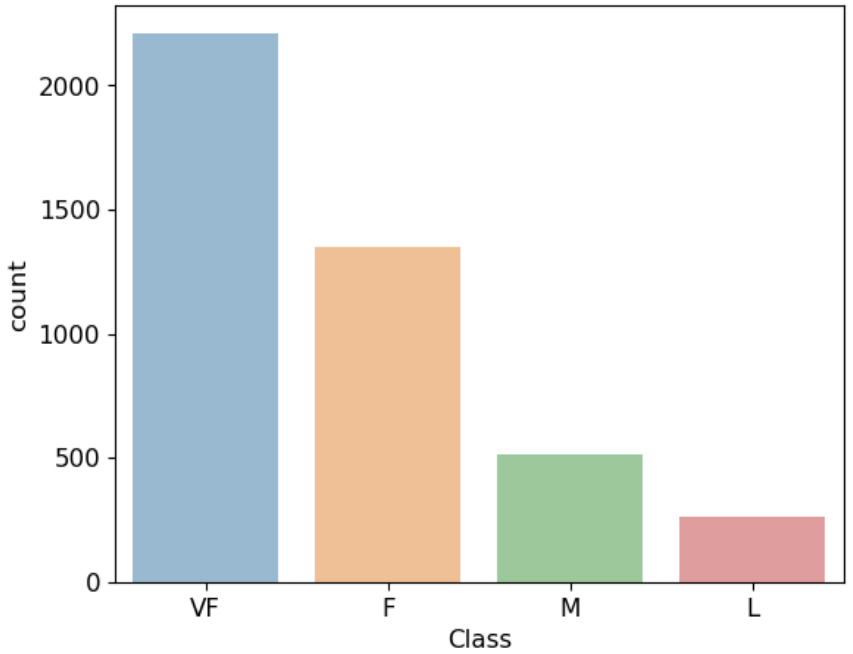
- 결측값이 없으므로 이에 따른 처리는 하지 않았음.

2. 탐색적 자료분석 · 전처리 – 자료 분할



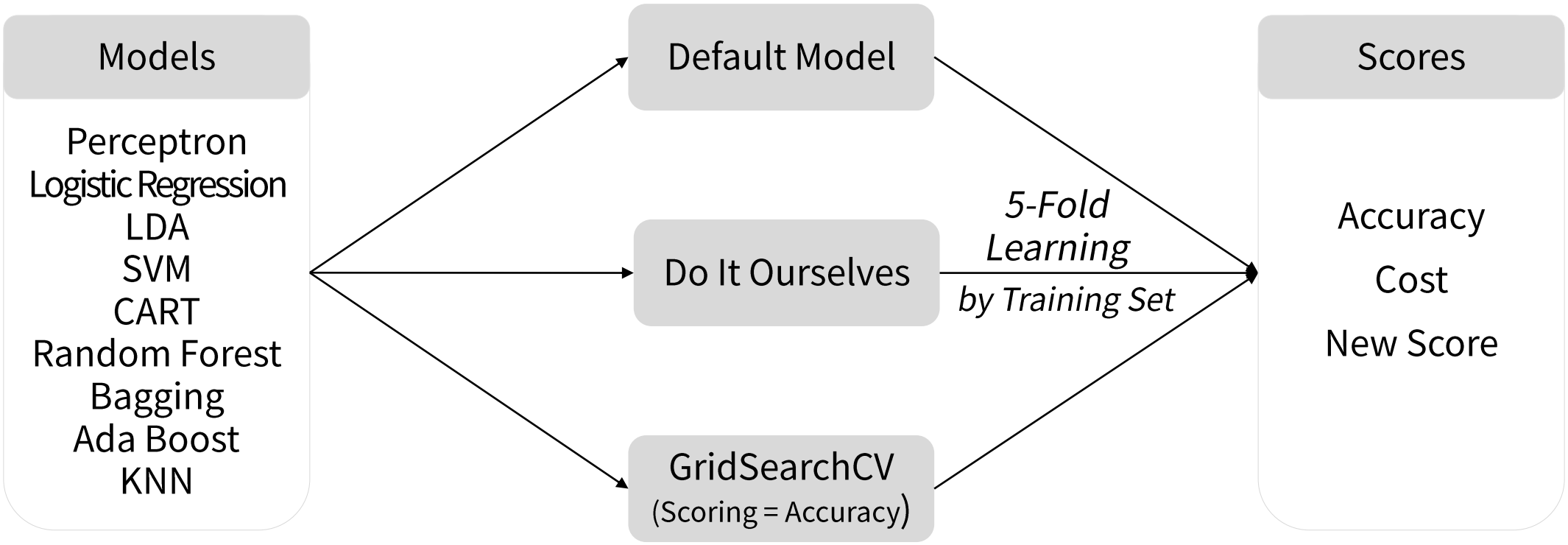

```
print(y.mean())
print(y_train.mean())
print(y_test.mean())

>>> Class   1.727776
      dtype: float64
      Class   1.727483
      dtype: float64
      Class   1.72895
      dtype: float64
```



- Training Set 80% / Test Set 20%로 자료 분할
- 종속변수 y인 ‘Class’ 변수는 클래스 별로 빈도에 차이가 존재하므로 y에 대해 층화 추출

3. 모델 튜닝



- 다양한 모델에 대해 GridSeachCV Method, Default값 등으로 파라미터를 조정
- Training Set으로 5-Fold Learning을 실시하였고, 성능에 대한 평가는 'Accuracy'와 새롭게 정의한 'Cost'를 활용
- 'Accuracy'와 'Cost'의 성능을 한 번에 평가하기 위해 'New Score'를 생성하여 성능 평가

3. 모델 튜닝 – Cost 함수

Confusion Matrix		실제			
		VF	F	M	L
예측	VF	208	5	21	36
	F	3	41	7	1
	M	31	6	62	4
	L	49	0	1	392

Cost Matrix					
		VF	F	M	L
	VF	0	1	5	10
	F	1	0	5	5
	M	1	1	0	1
	L	1	1	1	0

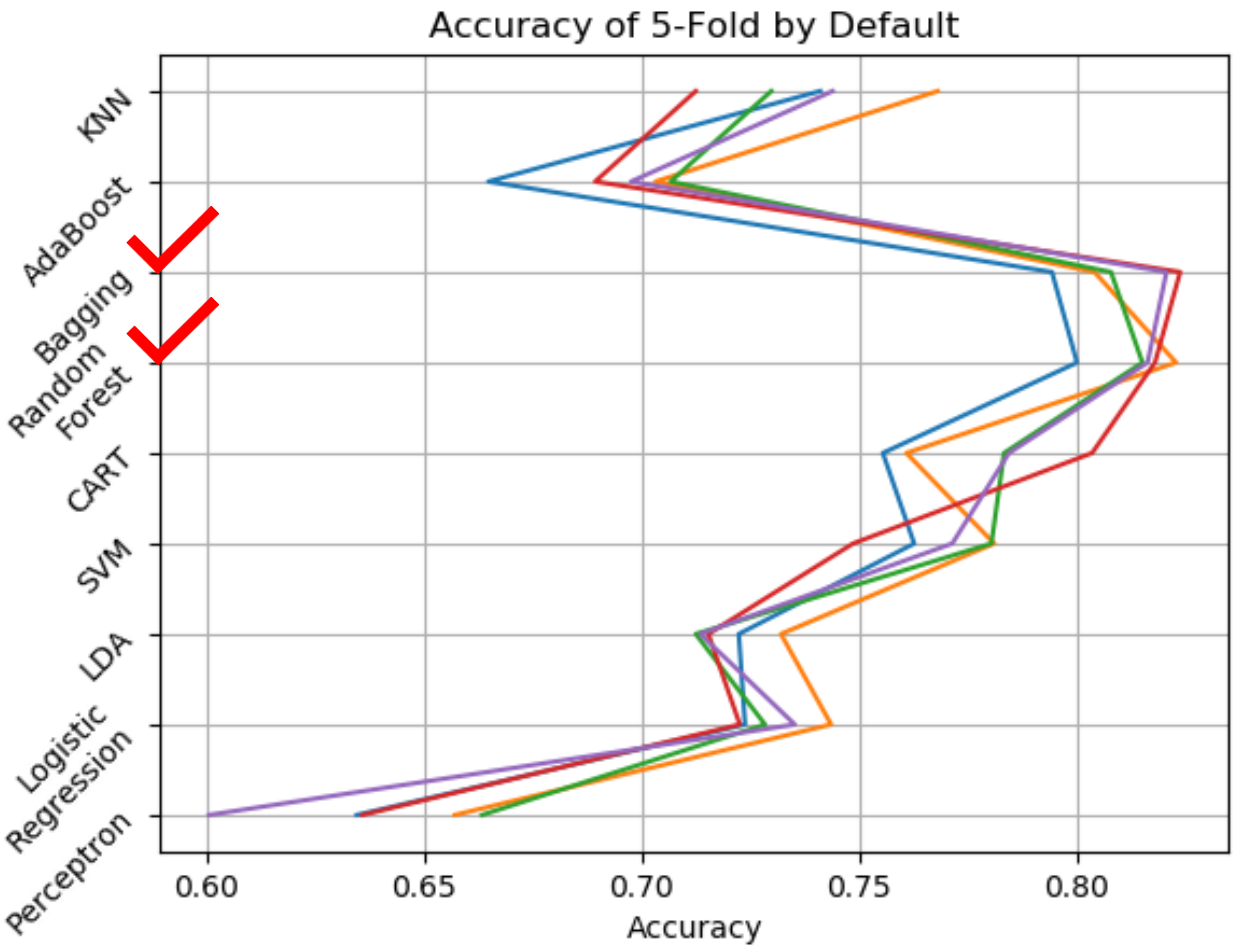
$$Total\ Cost = \sum_{i=1}^{4}\sum_{j=1}^{4} cfm_{ij} \cdot cm_{ij} = 208 \cdot 0 + 5 \cdot 1 + 21 \cdot 5 + 36 \cdot 10 + \cdots + 392 \cdot 0$$

```
def cost_score(x, y):
    model.fit(x, y) #지정한 x,y (X_train_std, y_train)으로 모델 적합
    y_pred = model.predict(x) #X_train_std를 통해 y의 예측값인 y_pred 계산
    Cfm_a = confusion_matrix(y, y_pred)  #실제 y값인 y_train과 예측값 y_pred로 confusion matrix 생성
    return np.multiply(Cfm_a, cost_matrix).sum() #사전에 정의한 cost matrix와 같은 위치의 값끼리 곱하여 모두 더함.

my_score = make_scorer(cost_score, greater_is_better=True) #위에서 정의한 Cost 값을 반환하는 함수를 score로 지정
```

- Confusion Matrix와 Cost Matrix에 대해 같은 행, 열에 위치한 원소들끼리 곱한 후 모두 더하여 Total Cost 도출

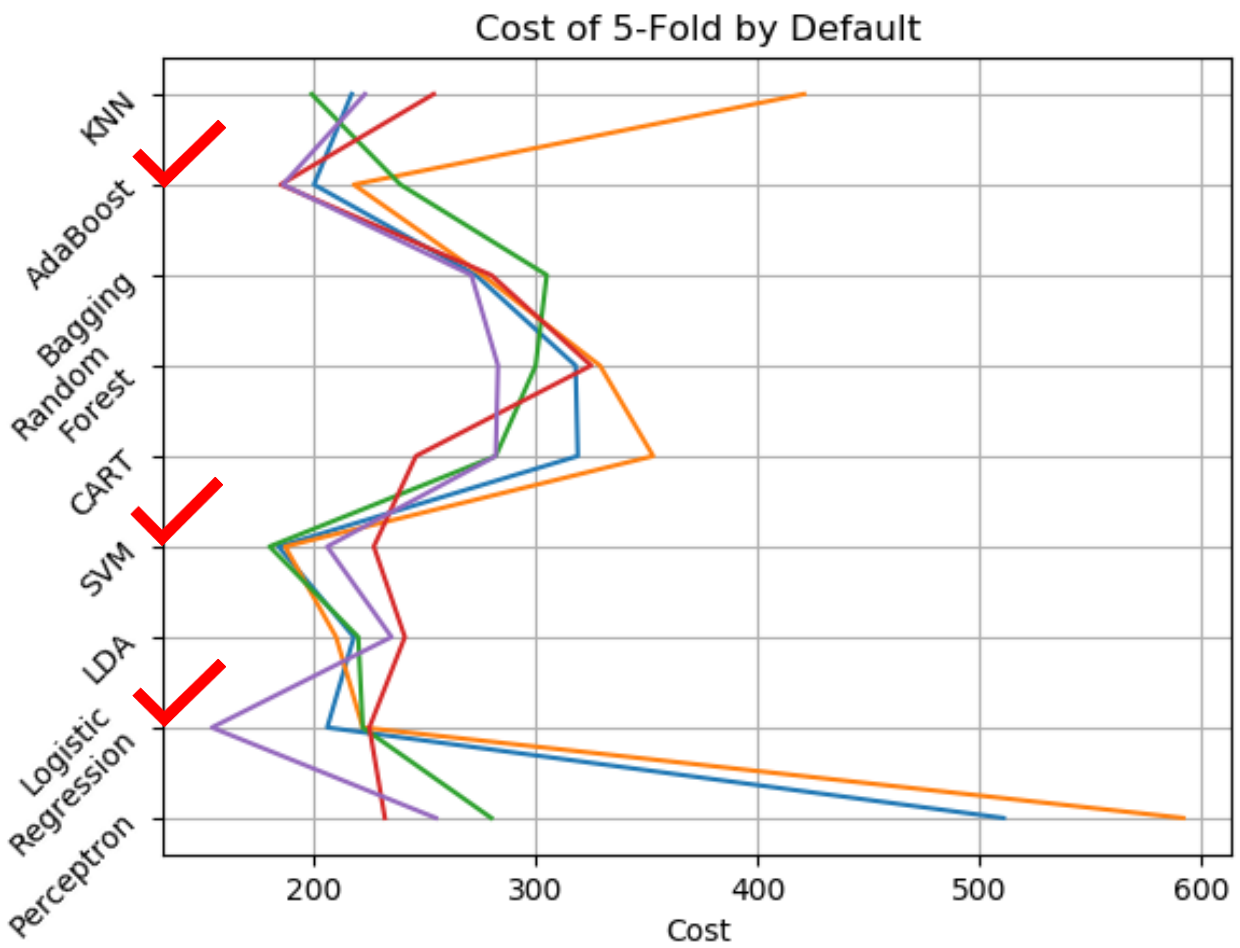
3. 모델 튜닝 – Default Models_Accuracy




```
<Mean & Standard Deviation of Accuracy>
-------------------------------------------------
KNN : 0.7390 / 0.0182
-------------------------------------------------
AdaBoost : 0.6923 / 0.0150
-------------------------------------------------
Bagging : 0.8101 / 0.0108
-------------------------------------------------
Random
Forest : 0.8144 / 0.0077
-------------------------------------------------
CART : 0.7775 / 0.0174
-------------------------------------------------
SVM : 0.7688 / 0.0121
-------------------------------------------------
LDA : 0.7191 / 0.0073
-------------------------------------------------
Logistic
Regression : 0.7307 / 0.0078
-------------------------------------------------
Perceptron : 0.6383 / 0.0220
```

- Accuracy에 대해서는 Bagging · Random Forest 모델의 성능이 좋은 것으로 나타남.

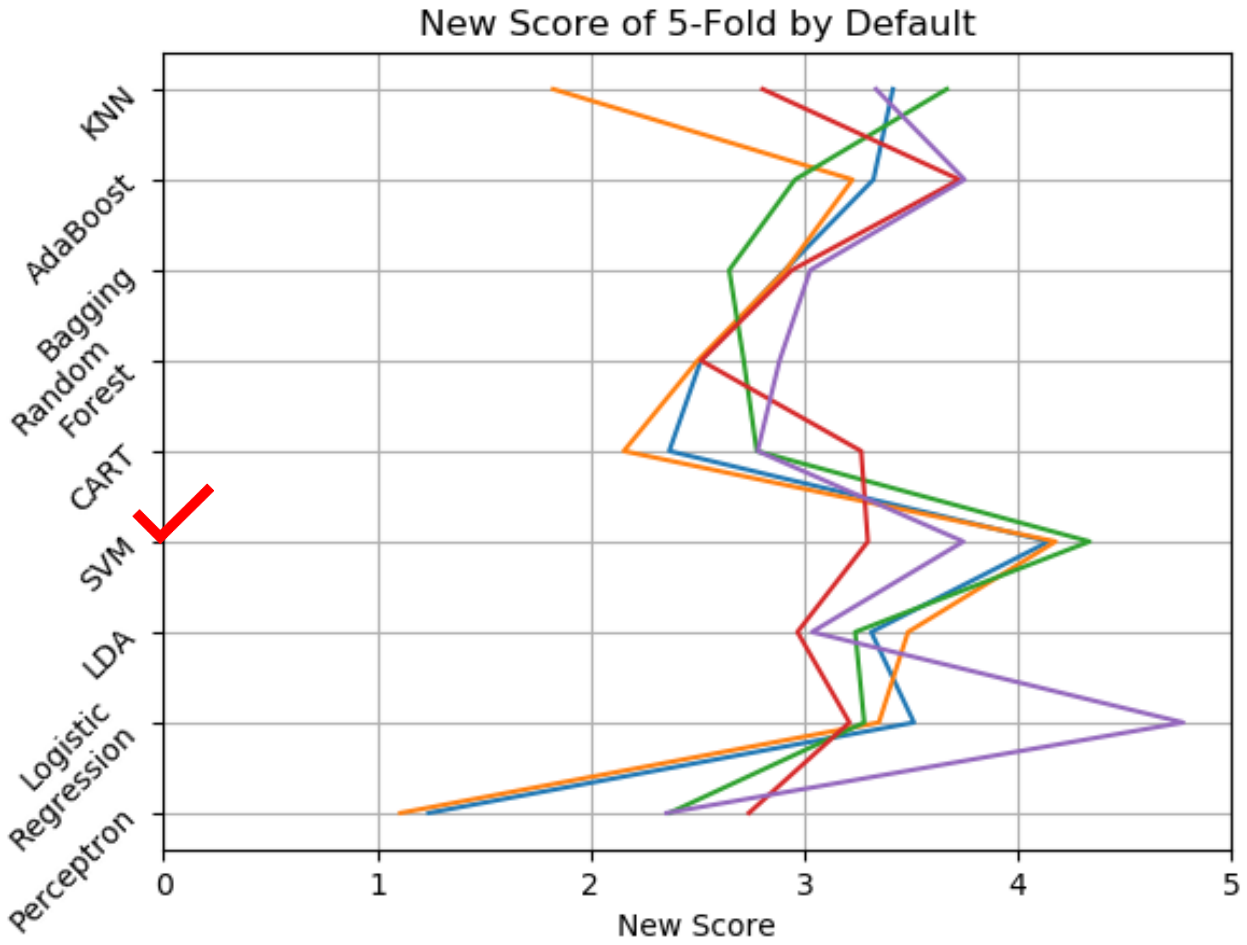
3. 모델 튜닝 – Default Models_Cost




```
<Mean & Standard Deviation of Cost>
-------------------------------------------------
KNN : 262.8 / 81.0639
-------------------------------------------------
AdaBoost : 205.6 / 20.5387
-------------------------------------------------
Bagging : 281.0 / 12.3774
-------------------------------------------------
Random
Forest : 311.0 / 17.1697
-------------------------------------------------
CART : 296.4 / 36.5218
-------------------------------------------------
SVM : 196.8 / 17.5431
-------------------------------------------------
LDA : 224.8 / 11.4438
-------------------------------------------------
Logistic
Regression : 205.8 / 26.7462
-------------------------------------------------
Perceptron : 374.0 / 147.9554
```

- Cost에 대해서는 AdaBoost · SVM · Logistic Regression 모델의 성능이 좋은 것으로 나타남.
- Accuracy가 높게 나타난 모델이 Cost에서의 좋은 성능을 담보하지는 않음.

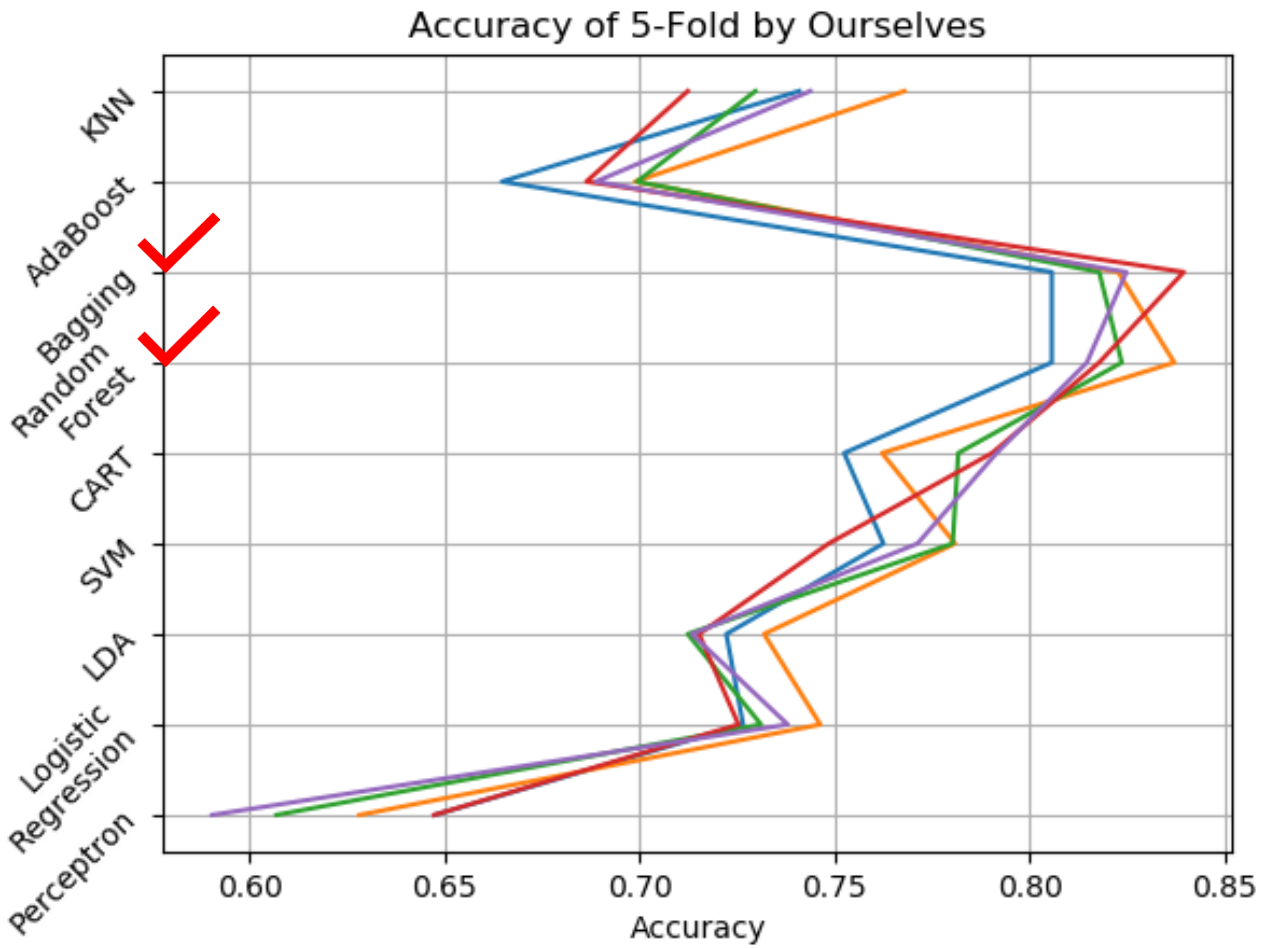
3. 모델 튜닝 – Default Models_New Score




```
<Mean & Standard Deviation of New Score>
-------------------------------------------------
KNN : 3.0093 / 0.6557
-------------------------------------------------
AdaBoost : 3.3964 / 0.3039
-------------------------------------------------
Bagging : 2.8881 / 0.1272
-------------------------------------------------
Random
Forest : 2.6268 / 0.1514
-------------------------------------------------
CART : 2.6697 / 0.3835
-------------------------------------------------
SVM : 3.9396 / 0.3755
-------------------------------------------------
LDA : 3.2083 / 0.1877
-------------------------------------------------
Logistic
Regression : 3.6257 / 0.5827
-------------------------------------------------
Perceptron : 1.9633 / 0.6590
```

- Accuracy와 Cost를 동시에 판단하기 위해 New Score를 정의함. (New Score = Accuracy / (Cost/1,000))
- 그 결과 SVM 모델이 고르게 좋은 성능을 내는 것으로 나타남.

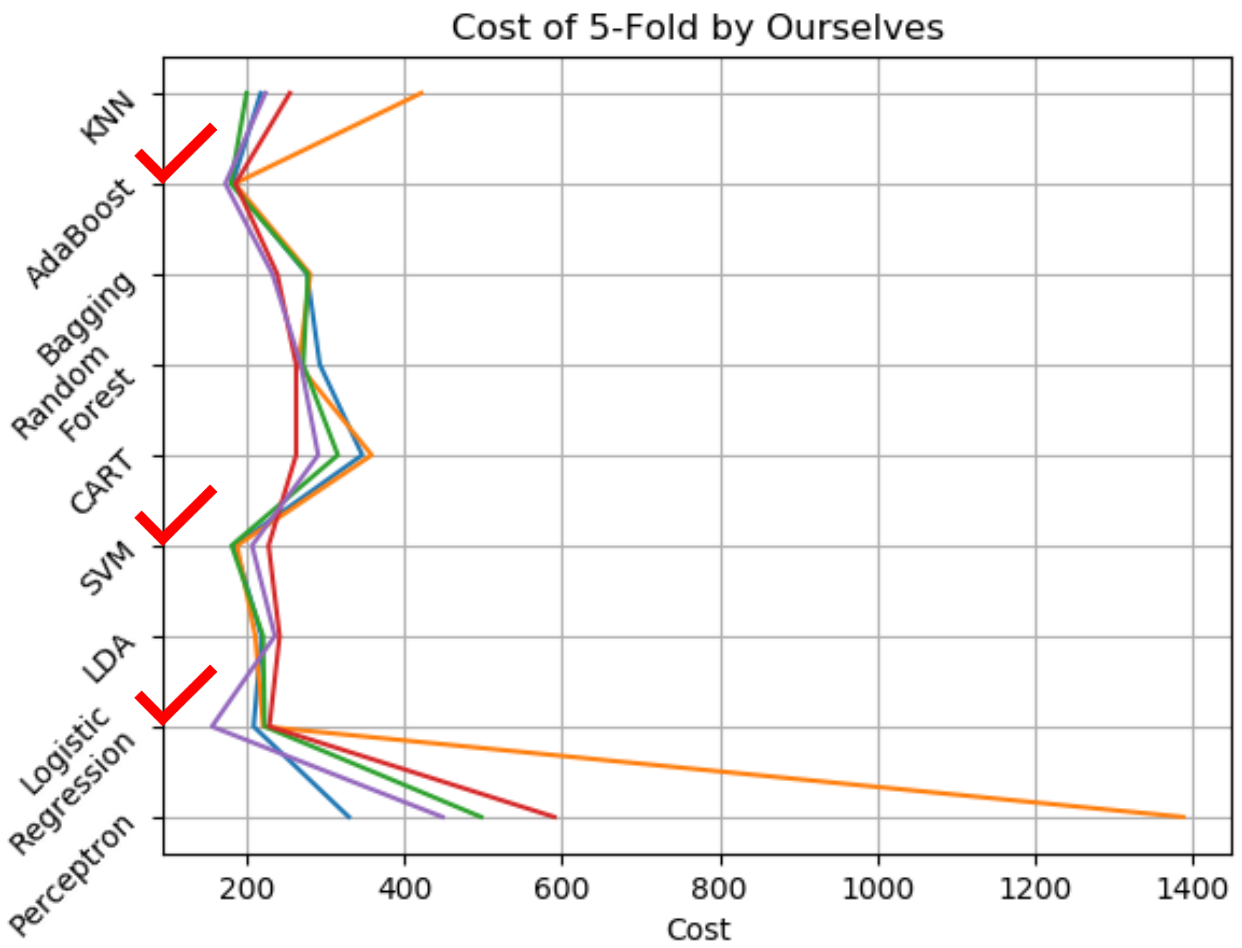
3. 모델 튜닝 – Do It Ourselves_Accuracy




```
<Mean & Standard Deviation of Accuracy>
-------------------------------------------------
KNN : 0.7390 / 0.0182
-------------------------------------------------
AdaBoost : 0.6877 / 0.0126
-------------------------------------------------
Bagging : 0.8222 / 0.0109
-------------------------------------------------
Random
Forest : 0.8199 / 0.0104
-------------------------------------------------
CART : 0.7757 / 0.0157
-------------------------------------------------
SVM : 0.7688 / 0.0121
-------------------------------------------------
LDA : 0.7191 / 0.0073
-------------------------------------------------
Logistic
Regression : 0.7335 / 0.0078
-------------------------------------------------
Perceptron : 0.6241 / 0.0225
```

- 직접 Parameter 값을 조정하여 모델링한 결과 또한 Default Models와 거의 유사한 결과가 나타남.

3. 모델 튜닝 – Do It Ourselves_Cost




```
<Mean & Standard Deviation of Cost>
-------------------------------------------------
KNN : 262.8 / 81.0639
-------------------------------------------------
AdaBoost : 180.2 / 4.6217
-------------------------------------------------
Bagging : 260.6 / 21.0295
-------------------------------------------------
Random
Forest : 271.6 / 10.6320
-------------------------------------------------
CART : 314.2 / 35.3293
-------------------------------------------------
SVM : 196.8 / 17.5431
-------------------------------------------------
LDA : 224.8 / 11.4438
-------------------------------------------------
Logistic
Regression : 206.6 / 26.6053
-------------------------------------------------
Perceptron : 650.6 / 378.6748
```

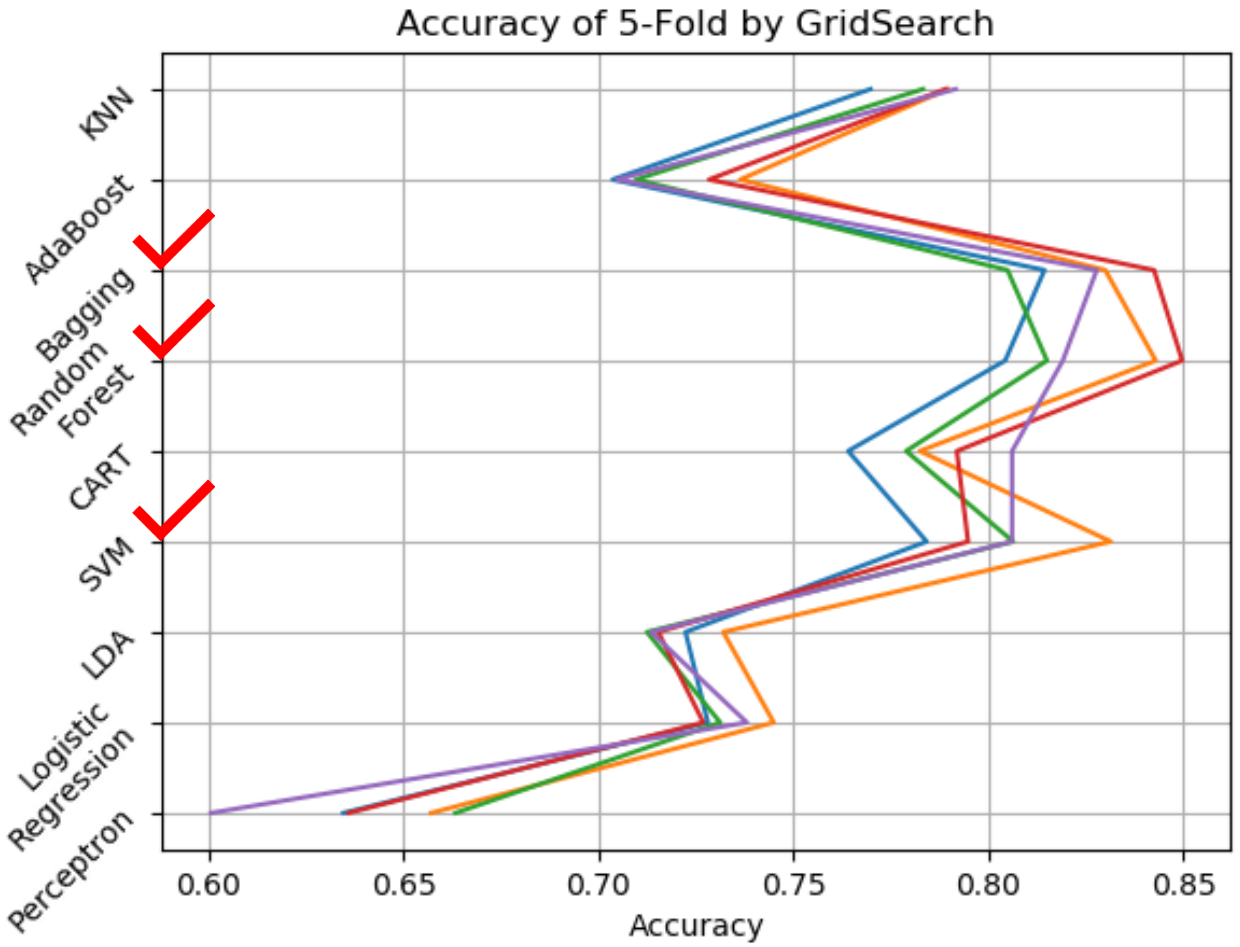
- 직접 Parameter 값을 조정하여 모델링한 결과 또한 Default Models와 거의 유사한 결과가 나타남.

3. 모델 튜닝 – Do It Ourselves_New Score


```
<Mean & Standard Deviation of New Score>
----------------------------------------------------
KNN : 3.0093 / 0.6557
----------------------------------------------------
AdaBoost : 3.8187 / 0.1233
----------------------------------------------------
Bagging : 3.1788 / 0.2966
----------------------------------------------------
Random
Forest : 3.0240 / 0.1402
----------------------------------------------------
CART : 2.5065 / 0.3357
----------------------------------------------------
SVM : 3.9396 / 0.3755
----------------------------------------------------
LDA : 3.2083 / 0.1877
----------------------------------------------------
Logistic
Regression : 3.6246 / 0.5778
----------------------------------------------------
Perceptron : 1.2114 / 0.4844
```

- New Score 또한 Default Model과 유사한 결과
- Ada Boost의 성능은 Default Model에 비해 개선된 것으로 보임.

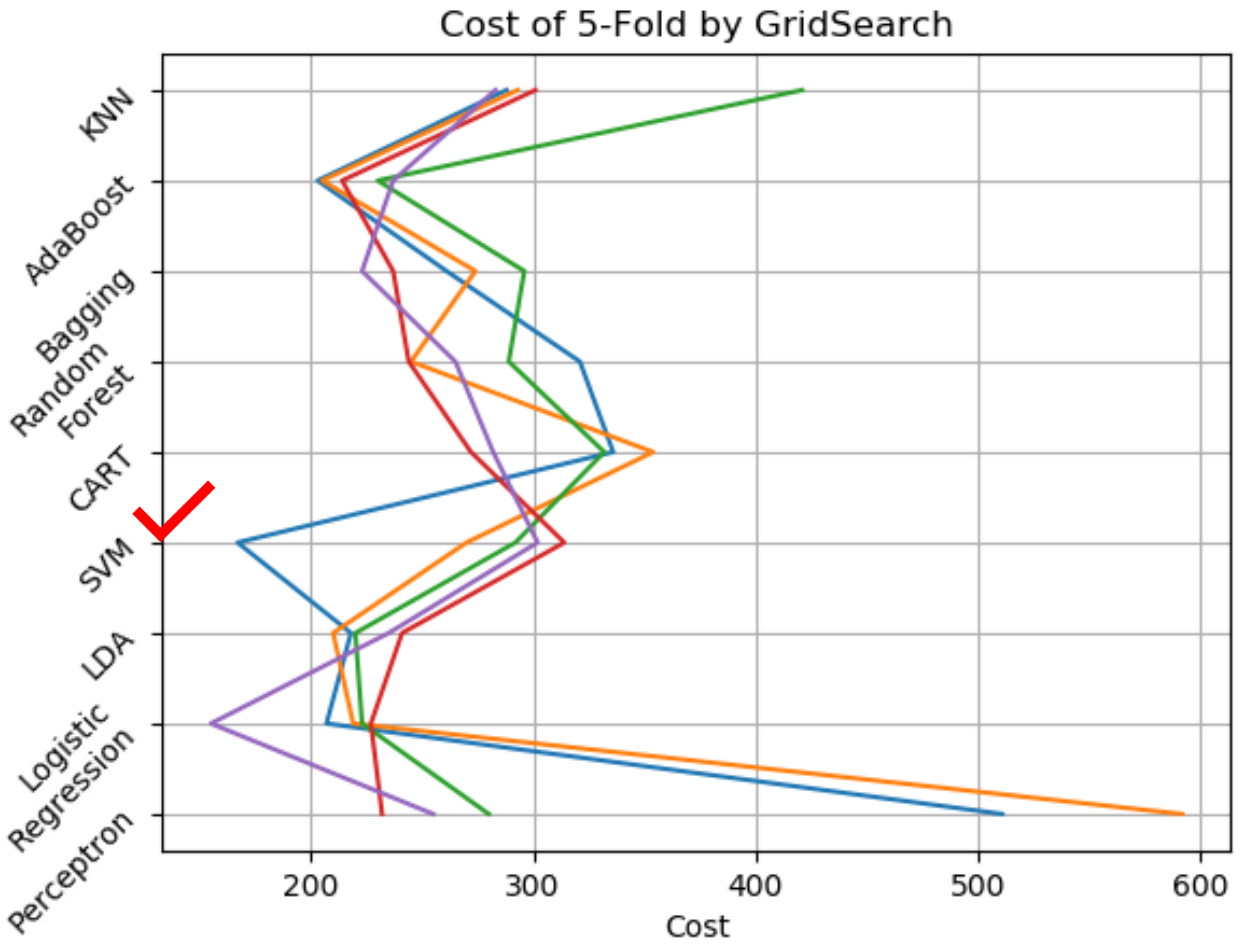
3. 모델 튜닝 – GridSearchCV_Accuracy




```
<Mean & Standard Deviation of Accuracy>
-------------------------------------------------
KNN : 0.7847 / 0.0079
-------------------------------------------------
AdaBoost : 0.7165 / 0.0133
-------------------------------------------------
Bagging : 0.8239 / 0.0130
-------------------------------------------------
Random
Forest : 0.8262 / 0.0172
-------------------------------------------------
CART : 0.7847 / 0.0140
-------------------------------------------------
SVM : 0.8046 / 0.0157
-------------------------------------------------
LDA : 0.7191 / 0.0073
-------------------------------------------------
Logistic
Regression : 0.7338 / 0.0068
-------------------------------------------------
Perceptron : 0.6383 / 0.0220
```

- SVM의 성능이 많이 개선되었고, 다른 모델 또한 다소 개선된 점을 보임.

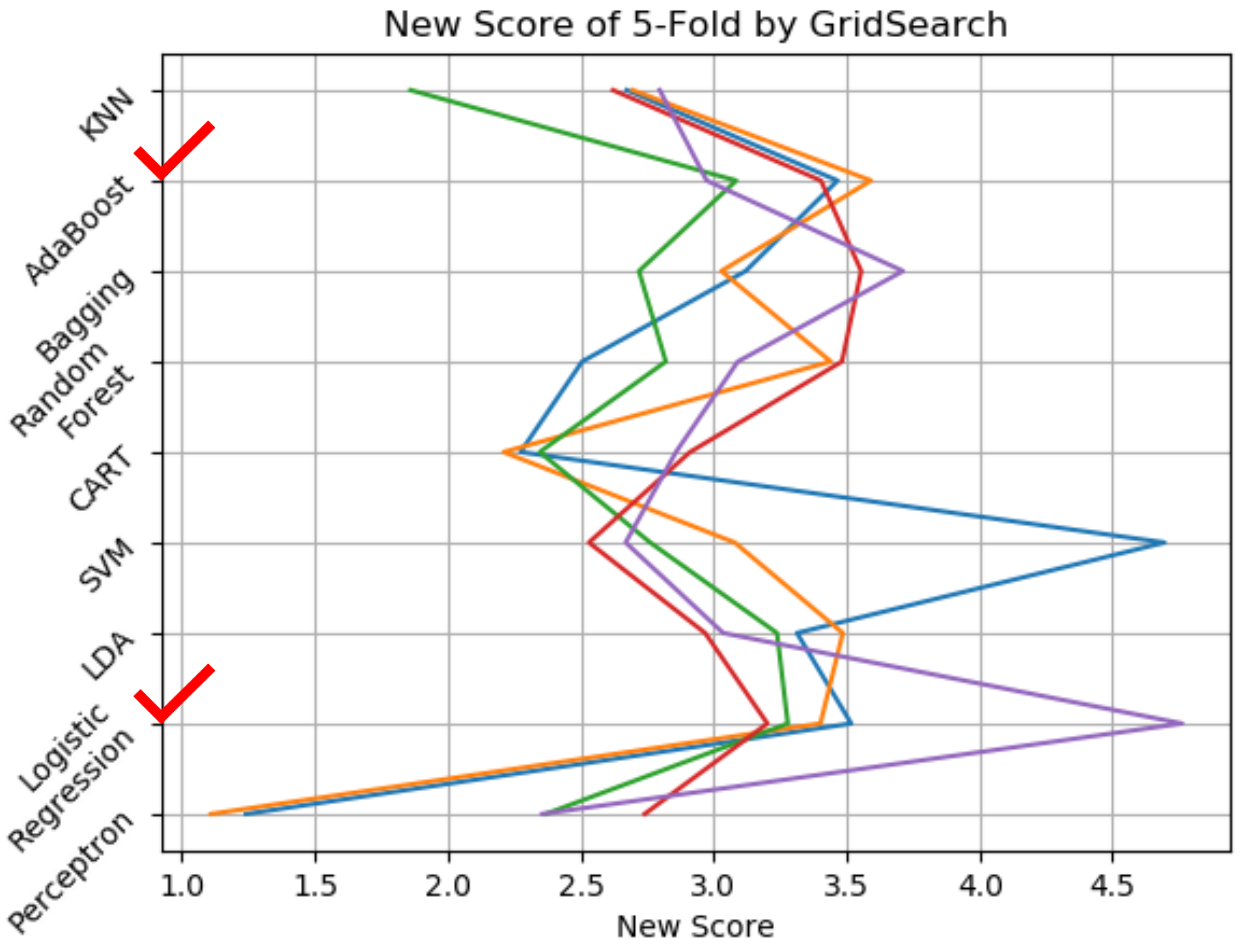
3. 모델 튜닝 – GridSearchCV_Cost




```
<Mean & Standard Deviation of Cost>
-------------------------------------------------
KNN : 317.2000 / 52.2394
-------------------------------------------------
AdaBoost : 217.8000 / 13.5263
-------------------------------------------------
Bagging : 258.2000 / 25.9800
-------------------------------------------------
Random
Forest : 272.8000 / 29.1506
-------------------------------------------------
CART : 315.2000 / 32.2143
-------------------------------------------------
SVM : 269.0000 / 53.0057
-------------------------------------------------
LDA : 224.8000 / 11.4438
-------------------------------------------------
Logistic
Regression : 206.2000 / 26.4605
-------------------------------------------------
Perceptron : 374.0000 / 147.9554
```

- GridSearch 시에 Scoring 기준을 Accuracy로 잡아 오히려 전체적인 Cost는 상승한 것으로 나타남.

3. 모델 튜닝 – GridSearchCV_New Score




```
<Mean & Standard Deviation of New Score>
-------------------------------------------------
KNN : 2.5294 / 0.3393
-------------------------------------------------
AdaBoost : 3.3040 / 0.2349
-------------------------------------------------
Bagging : 3.2271 / 0.3608
-------------------------------------------------
Random
Forest : 3.0680 / 0.3711
-------------------------------------------------
CART : 2.5200 / 0.3015
-------------------------------------------------
SVM : 3.1474 / 0.7949
-------------------------------------------------
LDA : 3.2083 / 0.1877
-------------------------------------------------
Logistic
Regression : 3.6323 / 0.5748
-------------------------------------------------
Perceptron : 1.9633 / 0.6590
```

- New Score 또한 오히려 전체적으로 감소한 것으로 나타남.

4. 최종 성능 평가

```
<Mean & Standard Deviation of New Score>
----------------------------------------------------
KNN : 3.0093 / 0.6557
----------------------------------------------------
AdaBoost : 3.8187 / 0.1233
----------------------------------------------------
Bagging : 3.1788 / 0.2966
----------------------------------------------------
Random
Forest : 3.0240 / 0.1402
----------------------------------------------------
CART : 2.5065 / 0.3357
----------------------------------------------------
SVM : 3.9396 / 0.3755
----------------------------------------------------
LDA : 3.2083 / 0.1877
----------------------------------------------------
Logistic
Regression : 3.6246 / 0.5778
----------------------------------------------------
Perceptron : 1.2114 / 0.4844
```

```
<Mean & Standard Deviation of New Score>
----------------------------------------------------
KNN : 2.5294 / 0.3393
----------------------------------------------------
AdaBoost : 3.3040 / 0.2349
----------------------------------------------------
Bagging : 3.2271 / 0.3608
----------------------------------------------------
Random
Forest : 3.0680 / 0.3711
----------------------------------------------------
CART : 2.5200 / 0.3015
----------------------------------------------------
SVM : 3.1474 / 0.7949
----------------------------------------------------
LDA : 3.2083 / 0.1877
----------------------------------------------------
Logistic
Regression : 3.6323 / 0.5748
----------------------------------------------------
Perceptron : 1.9633 / 0.6590
```

- ‘Do It Ourselves’와 ‘GridSearchCV’ 모델 중 성능이 좋게 나타난 5개 모델을 선정하여 Test Set으로 최종 성능 평가

4. 최종 성능 평가

```
<SVM>
Confusion Matrix
[[398  43   1   0]
 [ 61 200   9   0]
 [  7  50  45   1]
 [  1  12  12  27]]
Accuracy : 0.7728
Cost : 237
New Score : 3.2607
```

```
<Bagging>
Confusion Matrix
[[401  36   5   0]
 [ 42 207  18   3]
 [  4  30  64   5]
 [  0   8   7  37]]
Accuracy : 0.8178
Cost : 262
New Score : 3.1212
```

```
<Logistic Regression>
Confusion Matrix
[[409  32   1   0]
 [ 82 174  12   2]
 [ 11  64  17  11]
 [  2  15   1  34]]
Accuracy : 0.7313
Cost : 293
New Score :2.4958
```

```
<Ada Boost>
Confusion Matrix
[[351  89   2   0]
 [ 61 199  10   0]
 [  4  67  31   1]
 [  0  27  10  15]]
Accuracy : 0.6874
Cost : 319
New Score : 2.1549
```

```
<LDA>
Confusion Matrix
[[399  37   6   0]
 [ 82 156  25   7]
 [ 11  54  22  16]
 [  2  19   4  27]]
Accuracy : 0.6967
Cost : 415
New Score : 1.6787
```

- Test Set을 통한 평가 결과, SVM 모델의 New Score 성능이 가장 좋은 것으로 나타남.
- 본 문제는 컴퓨팅 환경에서 작업을 할당하는 시스템에 대한 문제이므로 Classifier 자체의 분류 속도 또한 빨라야 함.
- 따라서, 정확도와 비용 측면 뿐만 아니라 각 모델의 연산 속도에 대한 고려가 필요할 것으로 보임.

끝