

머신러닝2 과제 2025

(ver 2.0) - final

25. 6.13.

학번 : 1958015

이름 : 박시형

Preview

> 이전의 작업 요약

(ver 1.2)

1. 데이터 전처리

- process1_result.csv 사용, 피처: Sex, SibSp, Parch, Embarked, TicketNumeric
 - 결측치 처리: Age → Pclass×Sex별 중앙값; Embarked → 최빈값; Cabin 제외
-

2. Logistic Regression

- instance SGD(learning_rate=0.001) + Early Stopping → 최적 에폭스 = 9
 - 최적 k-fold = 18,
 - test1 결과: Acc 78.35%, F1 70.07%
-

3. Decision Tree

- Gini 불순도 기반 분할, max_depth 교차검증 (k=2→30 탐색)
 - 최적 k-fold = 16, max_depth = 15
 - test1 결과: Acc 78.69%, F1 70.64%
-

4. KNN

- 교차검증을 통한 k 탐색 (n_split(k-fold) 2→50 → 35-45 구간에서 수렴)
 - 최적 k-fold = 37, k = 20
 - test1 결과: Acc 79.79%, F1 69.70%
-

[PCA]

> 고차원 데이터(차원이 많은 데이터)가 있을 때,
차원을 줄이기 위해 사용하는 방법론

분산이 가장 큰 방향(축)을 찾고,

새 축 위로 데이터들을 투사시켜 상위 k개의 주성분만 남긴다.

(분산만을 사용, 공분산 행렬의 고유벡터들이 새 축이 되는 것이고, 이 고유값이 분산 크기를 나타낸다.)

보편적 관점

> 분산을 구하고, 누적 분산이 최대가 되는 지점을 선택하여 고유값을 이용해 차원을 줄인다.

pdf (MSE 관점)

> 보편적으로 고유치가 큰 방향을 선택했다는 것 = 원본 분산이 크고, 재구성 오차가 큰 비중을 차지하는 정보축을 최대한 살렸다는 뜻이다.

따라서 mean square error를 최소화하는 방식으로 사용한다.

두 방식의 수학적 차이는 없고,

보편적 분산 관점에서의 X_i 를 k차원으로 투사*복원할 때 생기는 재구성 오차를 최대한 줄이기 위해

$\frac{1}{n} \sum_i \|X_i - \hat{x}^i\|^2$ 를 직접 최소화 하기 위해 MSE(mean square error)의 관점을 사용한다.

mse의 관점에서 차원을 축소하면 "복원 오차가 최소" 라는 최적성을 명시적으로 보장하기 때문

- 연산 효율(covariance 생략, direct SVD)

- 수치 안정성

- Compression 최적성 보장

축을 바꾸는 이유

- 원본 feature 축이 서로 직교하지 않고,

의미적으로도 중복된 정보가 많을 수 있으므로

서로 직교하고 분산 순으로 정렬되어 있기 때문에 상위 몇개 만으로도 원본 정보의 대부분을 담을 수 있다.

따라서,

PCA로 차원을 축소하고,

이 결과에 대해

logistic, knn, decision tree를 진행하는 것.

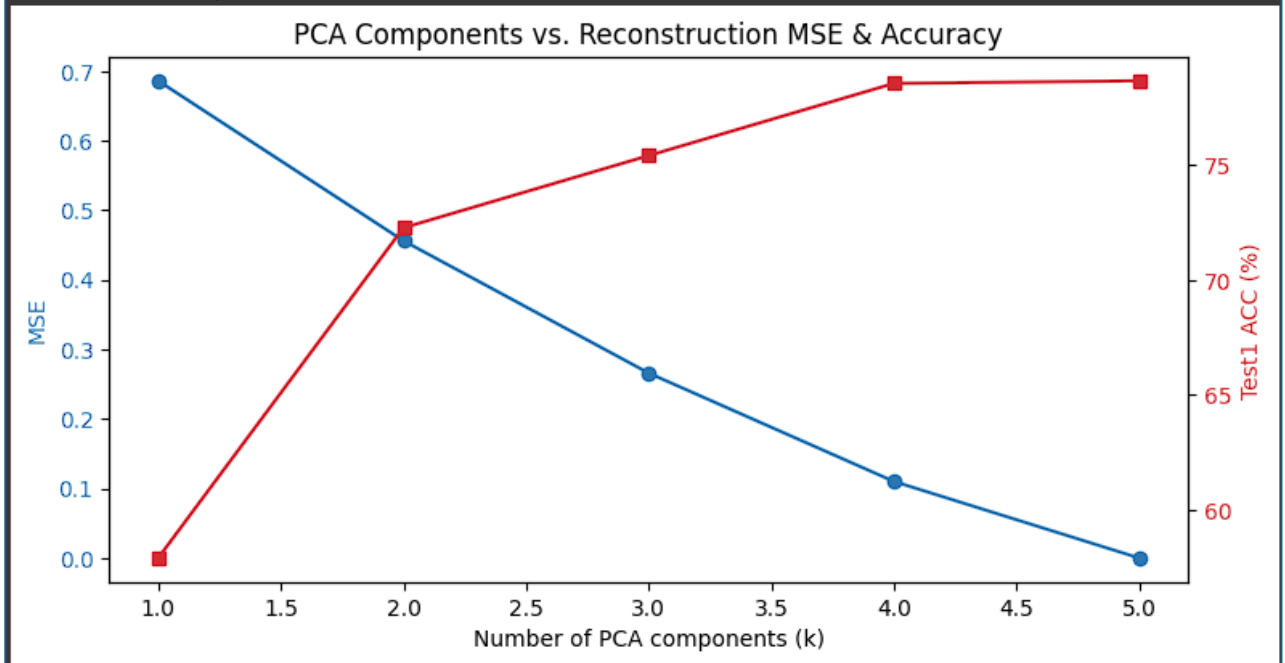
1. 데이터 zero centering
2. 공분산 행렬 계산
3. 고유분해
4. 고유값 내림차순 정렬
5. 상위 k주성분(고유벡터) 선택
6. 저차원 투사
7. 데이터 복원
8. 재구성 오차(MSE) 계산

이 값을 최소화하는 $V_k =$ 분산이 큰 축 순 으로 고른 주성분이라는 결론이 나온다.

이 순서대로 진행해보겠다.

(log)

```
k= 1 → MSE=0.6858, Test1 ACC=57.91%  
k= 2 → MSE=0.4559, Test1 ACC=72.28%  
k= 3 → MSE=0.2658, Test1 ACC=75.42%  
k= 4 → MSE=0.1100, Test1 ACC=78.56%  
k= 5 → MSE=0.0000, Test1 ACC=78.68%
```



K = 피쳐의 차원

Pca를 통해 차원을 축소함에 따라, test1에 대한 acc가 점점 내려가는 것을 보이고 있고, MSE 또한 점점 커지고 있다.

따라서 차원축소를 최소한으로 하는 **k의 값을 4**로 지정한다.

(k=4임에 대해 log, knn, dt에서 동일한 조건이어야 함)

K=4일 때, log에서

Test1 : 78.56

Test2 : 76.794

F1 score(test1) : 70.57

(knn)

K=4일 때, knn에서

Test1 : 80.58

Test2 : 77.272

F1 score(test1) : 71.59

(dt)

K=4일 때, dt에서

Test1 : 94.16

Test2 : 72.009

F1 score(test1) : 92.02

[LDA]

>pca와 마찬가지로 차원을 축소하는 방법론.

그 방식에 있어서 차이가 있다.

축을 어떻게 골라야 하는가에 대해 고민한 방법.

클래스A와 B가 서로 최대한 멀리 떨어져 있으면서, 같은 클래스끼리 뭉쳐보일 수 있는지에 대해서 축을 선택한다.

이때, 단순한 거리가 가장 큰 것으로 축을 잡는다면, 구분되지 않는 경우가 있기 때문에

두 분산의 비율

$J(w) = \text{클래스 간 분산} / \text{클래스 내 분산}$

을 최대화하는 방향을 찾아 그것을 축으로 한다.

(따라서 FLD는 binary한 discriminant라고 할 수 있다.)

즉, 하나의 축만을 가진다는 것.

이 축이 가장 분리도가 큰 축인 것.

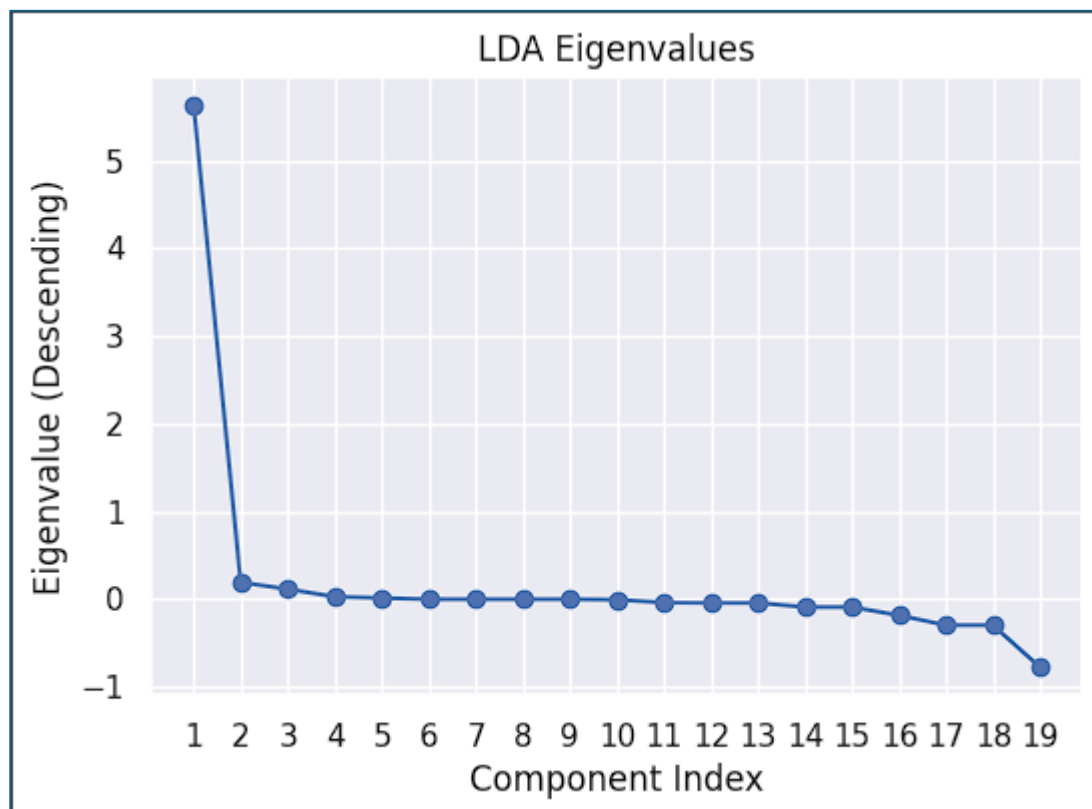
고유치를 통해 가장 분리가 잘 되는 축을 찾은 뒤에,

그 축에 대해서 고유벡터를 구하고

고유벡터에 대해 모든 데이터들을 투사시킨다.

1. 데이터 중앙화*스케일링
2. 클래스별 평균 벡터 계산
3. 클래스 내 산포 행렬 S_w
4. 클래스 간 산포 행렬 S_b
5. 판별 벡터 w 계산 최대화
6. 판별 벡터 w 정규화
7. 새로운 축에 투사

Process1에 대한 LDA eigenvalue의 그래프



(log)

Test1 : 78.90

Test2 : 77.033

F1 score(test1) : 71.08

(dt)

Test1 : 92.14

Test2 : 67.224

F1 score(test1) : 89.14

(knn)

Test1 : 79.46

Test2 : 77.511

F1 score(test1) : 70.72

(conclusion)

PCA 사용 : knn의 test1, test2 개선 (acc상승, f1score 하락)

FLD 사용 : DT의 f1score 개선, knn의 test2 개선

[MLP]

다층 퍼셉트론, 은닉층 + 뉴런

(full feature)

```
mlp_simple = MLPClassifier(  
    hidden_layer_sizes=(32,),  
    activation='relu',  
    solver='adam',  
    alpha=1e-3,  
    learning_rate='constant',  
    learning_rate_init=0.0005,  
    max_iter=200,  
    batch_size=32,  
    shuffle=True,  
    random_state=42,  
    early_stopping=True,  
    validation_fraction=0.1,  
    n_iter_no_change=10
```

은닉층 : 1, 뉴런 : 16 32 64 128

```
=== 최종 결과 요약 ===  
은닉층 노드 16개 → Test1 Accuracy = 79.89%  
은닉층 노드 32개 → Test1 Accuracy = 80.45%  
은닉층 노드 64개 → Test1 Accuracy = 81.01%  
은닉층 노드 128개 → Test1 Accuracy = 80.45%
```

은닉층 : 2, 뉴런 : (16,16), (16,32), ' '

```
=== 상위 5개 조합 (h1, h2, Accuracy) ===  
(128, 64) → Test1 Accuracy = 80.45%  
(32, 16) → Test1 Accuracy = 79.89%  
(64, 32) → Test1 Accuracy = 79.89%  
(128, 16) → Test1 Accuracy = 79.89%  
(16, 16) → Test1 Accuracy = 79.33%
```

은닉층 : 3, 뉴런 : (16,16,16), ' '

```
=== 상위 5개 조합 (h1, h2, h3, Accuracy) ===  
(16, 16, 16) → Test1 Accuracy = 80.45%  
(64, 16, 16) → Test1 Accuracy = 80.45%  
(128, 32, 16) → Test1 Accuracy = 80.45%  
(128, 128, 32) → Test1 Accuracy = 80.45%  
(64, 32, 16) → Test1 Accuracy = 79.89%
```

정확도가 더 이상 늘어나지 않는 것을 확인,
과적합을 피하기 위해 최대한 은닉층을 줄여야 함.

따라서 은닉층 1, 노드는 32로 정한다.

그 주변의 데이터들을 살펴봐야 하는데, 27~37까지의 범위로 test1 결과를 다시
내보면

```
=== 탐색 결과 요약 ===  
노드 27개 → Test1 Accuracy = 80.45%  
노드 28개 → Test1 Accuracy = 81.01%  
노드 29개 → Test1 Accuracy = 79.33%  
노드 30개 → Test1 Accuracy = 81.56%  
노드 31개 → Test1 Accuracy = 80.45%  
노드 32개 → Test1 Accuracy = 80.45%  
노드 33개 → Test1 Accuracy = 81.56%  
노드 34개 → Test1 Accuracy = 81.01%  
노드 35개 → Test1 Accuracy = 81.01%  
노드 36개 → Test1 Accuracy = 81.01%  
노드 37개 → Test1 Accuracy = 81.01%
```

따라서 노드는 30으로 지정.

학습률 – 너무 크면 노이즈에 민감, 작으면 학습이 느려짐

```
=== 학습률별 Test1 Accuracy 요약 ===  
lr=0.0001 → 81.01%  
lr=0.0005 → 81.01%  
lr=0.0010 → 81.01%  
lr=0.0050 → 81.56%  
lr=0.0100 → 81.56%
```

lr = 0.005로 지정

alpha – 값이 클수록 가중치가 0에 가깝게 수렴하려 하므로 모델 복잡도가 낮음

0.01: 81.01

0.001 : 80.45

0.0005 : 80.45

0.0001 : 81.56

Alpha = 0.0001로 지정

Max_iter – 반복

300 : 81.56

500 : 81.56

1000 : 81.56

Max_iter = 300으로 지정

(final)

은닉층 : 1

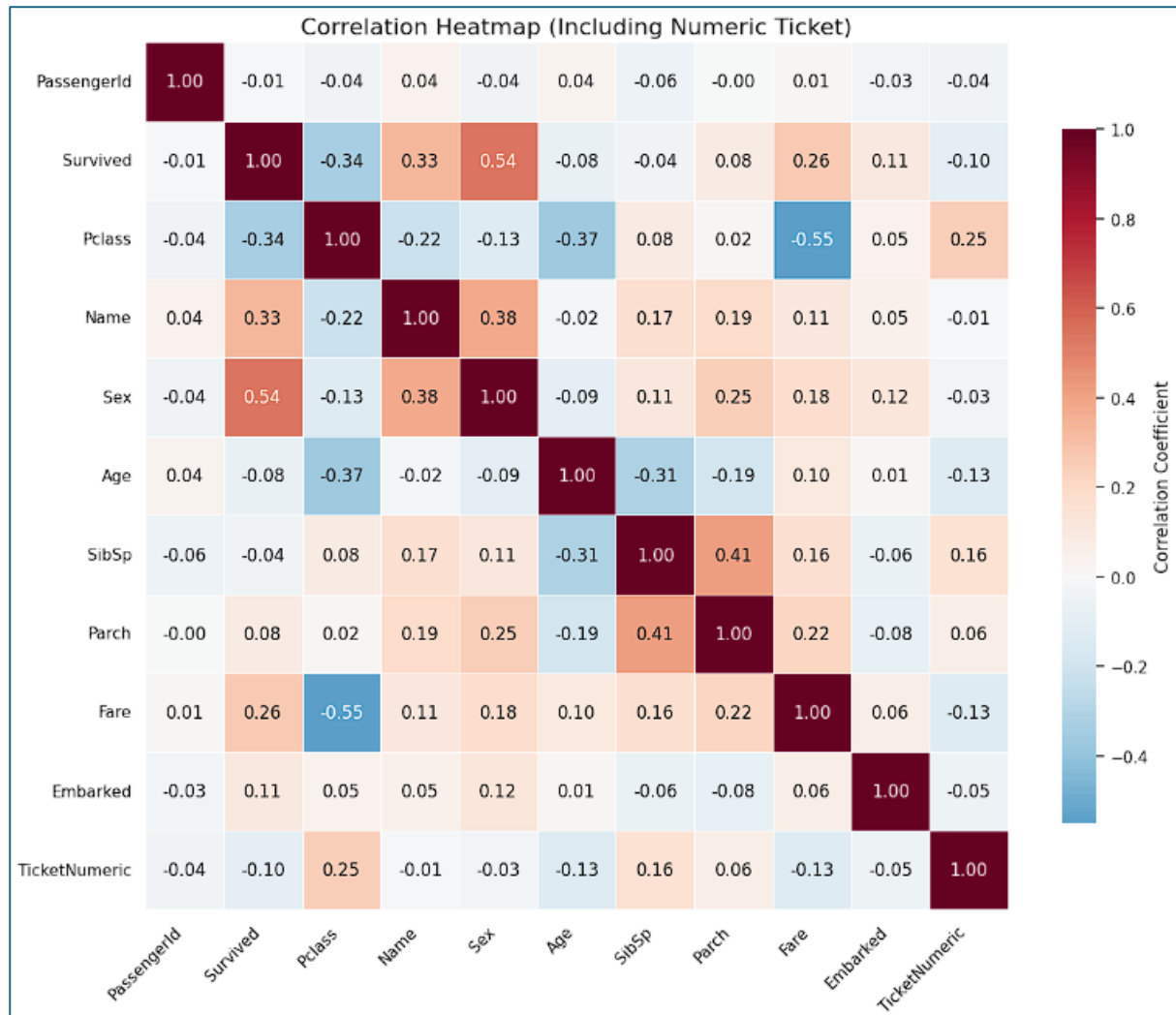
노드 : 32

학습률 : 0.005

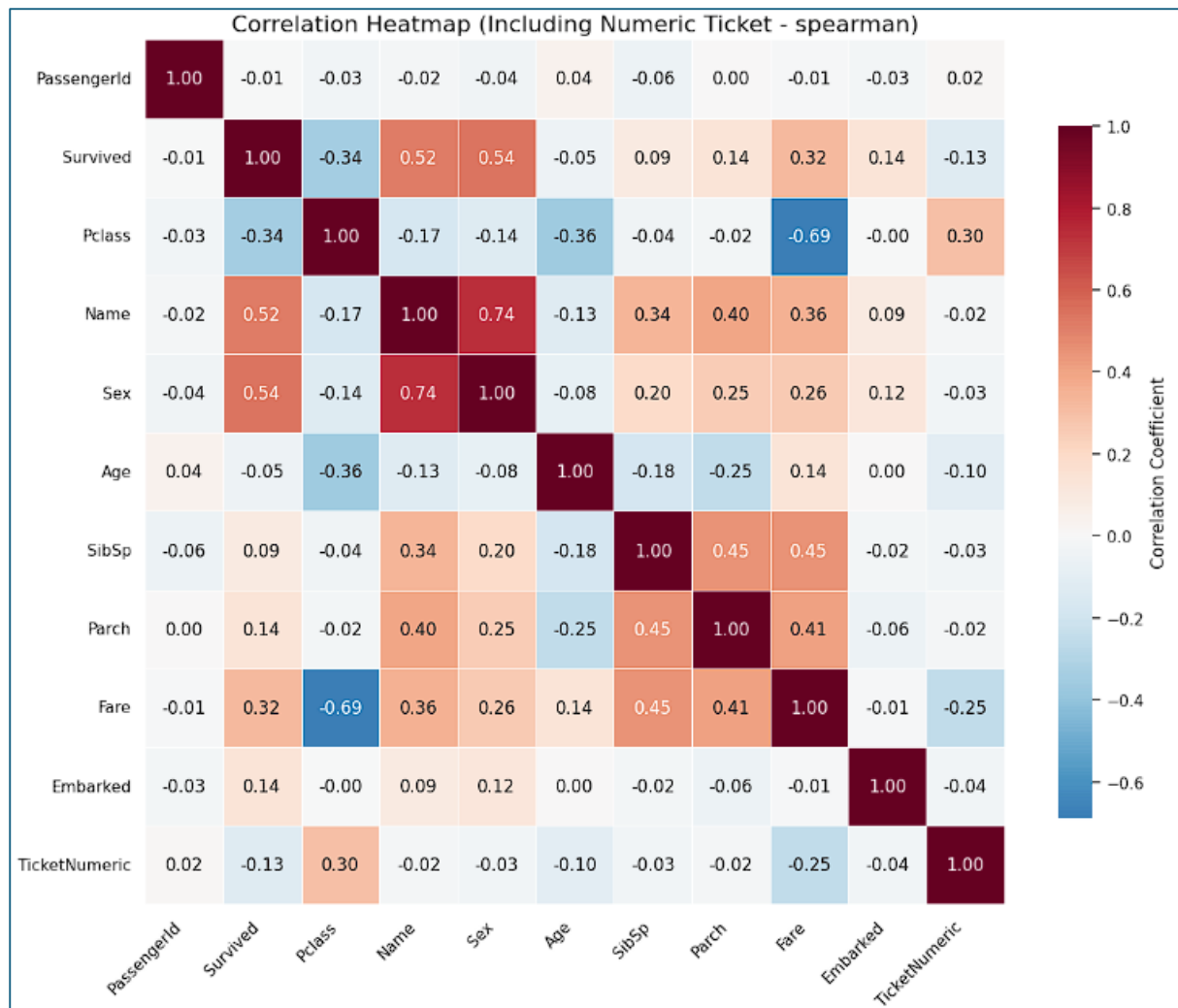
Alpha : 0.0001

Maxiter: 300

모든 타이틀에 대한 상관관계도(피어슨)



스피어만



두 방식에서 추출한 상관관계상수를 이용하여

1. 모든 title에 대해 log, dt, knn, mlp 적용. and pca, lda
2. Sex&age 를 통해 생존에 대한 survived score TITLE 생성. 이후 적용


```

=== Test1 (Hold-out) Accuracy (Full Feature) ===
Logistic      : 97.77%
KNN (k=20)    : 96.09%
DecisionT     : 97.21%
MLP           : 96.65%

Saved files:
- submission_logistic_full133.csv
- submission_knn_full133.csv
- submission_dt_full133.csv
- submission_mlp_full133.csv

```

process1과 process2에 있는 겹치는 타이틀 중에 survived와 passengerid 를 제외
모든 타이틀을 사용(cabin 제외),

전처리 과정에서 소수점이나 nan값이 생긴 부분이있어서

- 그 부분을 해당 열로 채워넣고, 모든 열 별로 평균과 표준편차를 이용해서 scale 된 데이터를 넣었다.(편차가 큰 데이터들을 normalize 하기 위함)

전처리 사용 타이틀 = full title									
Train Data / Test1 Data					Train Data / Test2 Data				
	Logistic R	Decision T	KNN	MLP		Logistic R	Decision T	KNN	MLP
full feature	97.77	97.21	96.09	96.65	full feature	73.684	73.205	74.162	75.358
PCA					PCA				
FLD					FLD				

Test1에서의 결과는 모두 90% 이상

Test2에서 70%대. 하락 혹은 현상 유지

- 과적합 중.

test2의 결과를 올리기 위한 코드들, test1의 정확도는 어느정도 포기하고, test2의 결과를 최대한 끌어 올려야 한다.

사용하고 있는 피쳐

```
> ['Sex', 'SibSp', 'Parch', 'Embarked', 'TicketNumeric']
```

사용하는 이유는, 모든 타이틀에 대해서 상관관계를 구한 뒤에, survival rate를 구하고 threshold를 통해 survived를 나눈 뒤 submission 작성, 이후 test2의 acc가 가장 높은 5개에 대해서

test2의 결과가 가장 좋았던 피쳐를 골랐던 것이다.

2025에는

여기에서 판별식을 바꿔보거나,

피쳐의 차원을 축소하거나,

k-fold를 진행해서 결과를 뽑았다.

결과적으로 test1보다 test2에서 acc가 낮은 모습을 보여줬고(full feature 기준),

이는 과적합이라고 판단할 수 있다.

log, dt, knn에서 과적합을 피하기 위해 일정 수렴하는 값에 도달하기 이전에 멈춘다거나,

dt의 depth를 줄인다거나 여러가지를 동원해봤으나 유의미한 결과가 나오진 않았다.

또한. pca와 fld의 차원 축소로 나오는 eigenvalue의 그래프를 그렸을 때, 차원을 축소하지 않았을 때의 값이 가장 좋았기 때문에 근본적으로 피쳐를 고르는데에

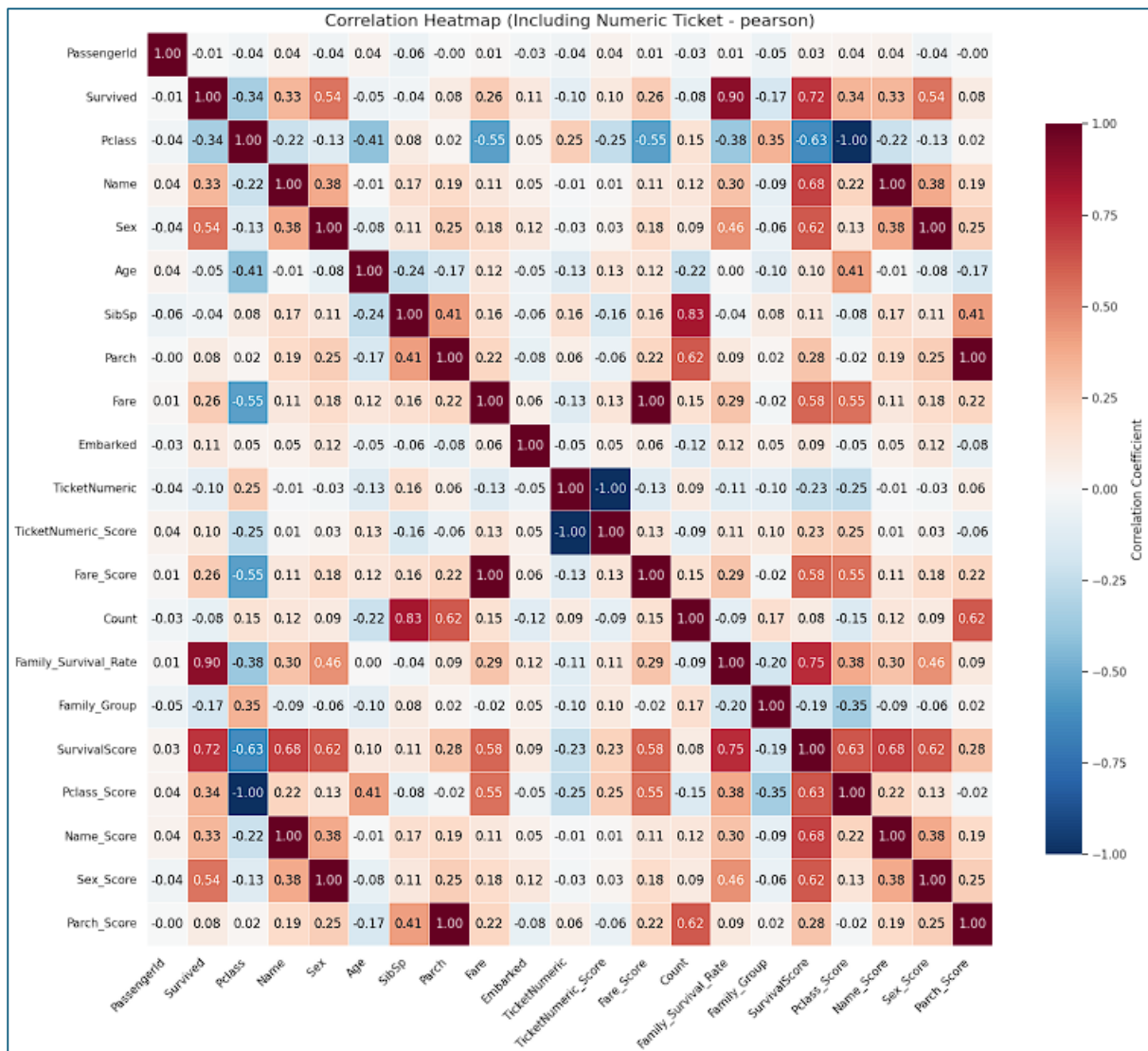
포커스를 두어야 한다고 생각한다.

과적합을 피한다 = train데이터에 과하게 학습되어 있는 데이터를 그렇지 않게 만들어야 한다.

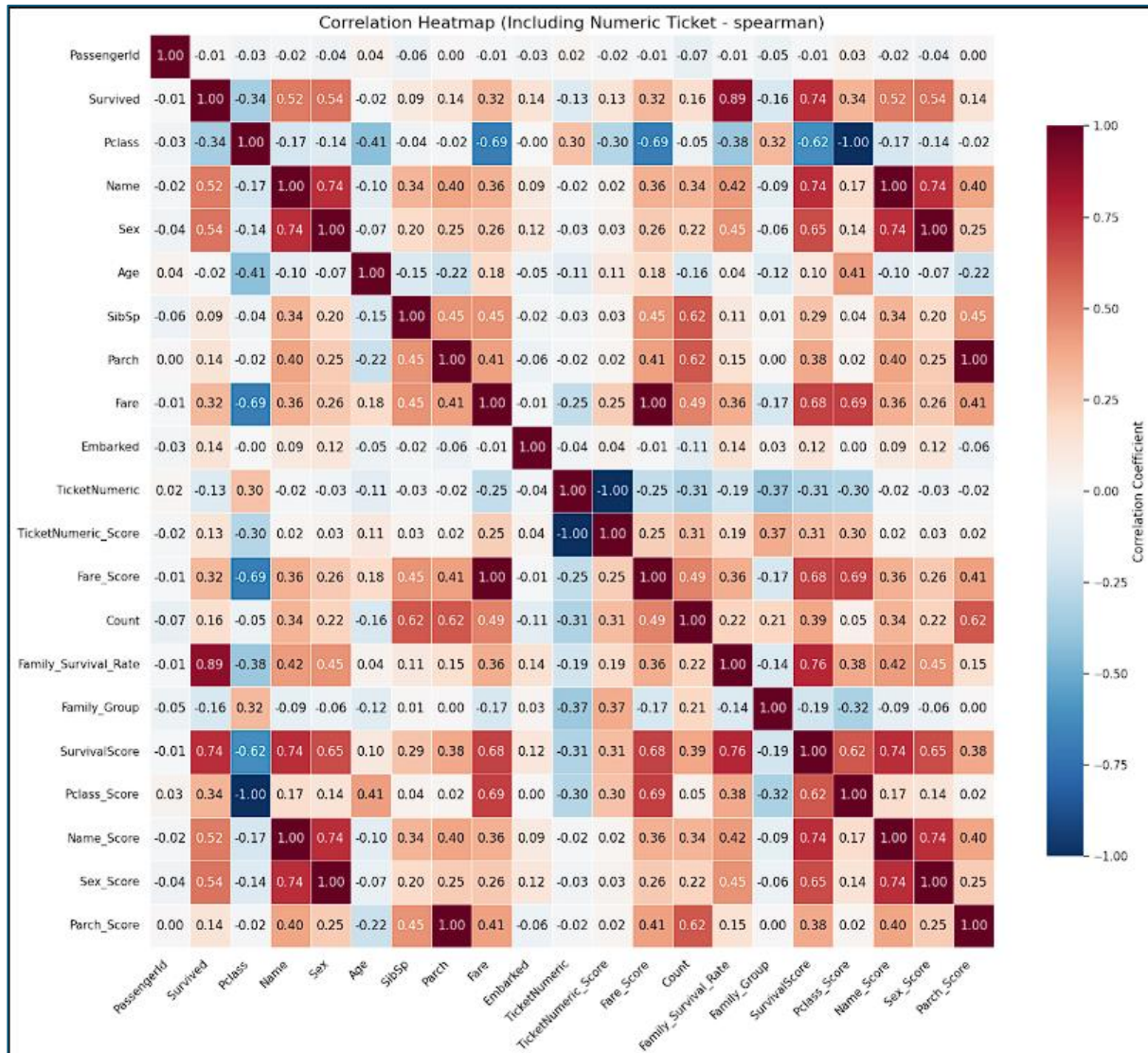
>> k-fold를 통해, train 데이터에 100% 학습시키는게 아닌, k-fold(5)로 나온 acc가 가장 높다는 의미는 20%씩 학습시킨 5개의 average를 구하는 것이기 때문에 과적합을 피할 수 있음을 의미한다.

따라서 k-fold 의 acc가 가장 높은 방법으로 진행한다.

데이터 선택을 함에 있어서, 상관계수가 높은 순으로 정리해야 함.



➤ 전처리 과정에서 생긴 모든 타이틀에 대해 상관계수를 구한 모습 - Pearson



➤ 전처리 과정에서 생긴 모든 타이틀에 대해 상관계수를 구한 모습 - Spearman

상관계수가 겹치는 변수는 normalizing을 하며 생긴 타이틀이기 때문,

따라서 Count < 숫자를 셀 때 사용했던 타이틀이므로

이보다 상관계수가 낮은 모든 타이틀을 사용해서

결과가 가장 좋은 부분 집합을 사용한다.

상관계수가 가장 높은 FamilyRate, SurvivalScore는 Survived를 사용하여 만들어낸 변수
이므로 사용해선 안된다.

(pearson)

Family_Survival_Rate = 0.90

SurvivalScore = 0.72

Sex = 0.54

Sex_Score = 0.54

Pclass = -0.34

Pclass_Score = 0.34

Name = 0.33

Name_Score = 0.33

Family_Group = -0.17

embarked = 0.11

Count = -0.08

(count 는 숫자를 셀 때 사용한 타이틀임, 이보다 낮은 타이틀과 count는 필요 없다고 판단)

$2^5 - 1 = 31$

Pearson - ['Sex','Pclass','Name', 'Family_Group', 'Embarked']

(spearman)

Family_Survival_Rate = 0.89

SurvivalScore = 0.74

Sex = 0.54

Sex_Score = -0.54

Name = 0.52

Name_Score = 0.52

Pclass = -0.34

Pclass_Score = 0.34

Count = 0.16

(count 는 숫자를 셀 때 사용한 타이틀임, 이보다 낮은 타이틀과 count는 필요 없다고 판단)

$2^3 - 1 = 7$

Spearman - ['Sex', 'Name', 'Pclass']

따라서 사용하는 피쳐는 각각 아래처럼 된다.

Pearson - ['Sex','Pclass','Name', 'Family_Group', 'Embarked']

Spearman - 'Sex', 'Name', 'Pclass']

(Pearson 피쳐를 쓸 때)

k-fold = 5

피쳐 5개에 대한 부분집합 2^5-1 의 경우의 수를 모두 돌렸을 때,

```
Best subset: ['Pclass', 'Name', 'Embarked']  
Model: dt CV score: 0.8260435628648546  
Saved final_submission.csv
```

82% 정도로 dt가 가장 좋은 성능을 보이고 있다.

But, 이는 depth가 15인 과적합 상태라고 판단하여

depths = [3, 5, 7, 10, 15] 을 각각 넣었을 때 k-fold 값이 가장 좋은 피쳐와 depth를 선택한다.

```
▶ Best subset: ['Pclass', 'Name', 'Embarked']  
▶ Best max_depth: 10  
▶ Best CV score: 0.8260435628648546
```

Feature = pclass, name, embarked

Dt(depth = 10)

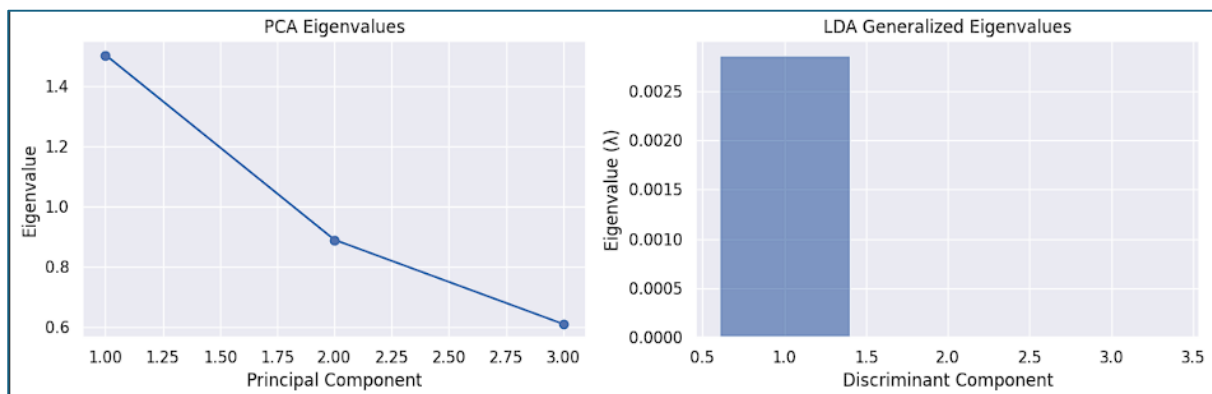
➤ Test1 : 82.604%

따라서,

피어슨 상관계수를 따를 때,

Full feature = pclass, name, embarked 이다.

여기에 대해 pca, LDA를 진행하여 eigenvalue를 시각화하면,



이렇게 나오고

따라서 pca의 차원축소는 3 => 2

LDA의 차원축소는 3 => 1 로 하겠다.

최종 test1의 acc를 엑셀로 정리하면

### ----- NEW FEATURE ----- ###				
PEARSON = pclass, name, embarked				
K-FOLD	Train Data / Test1 Data			
	Logistic R	Decision T	KNN	MLP
full feature	73.51453	82.60436	79.91149	78.00577
PCA	73.85161	82.71734	79.68615	72.27983
FLD	72.27858	82.49262	79.57253	72.95524

최종적으로 결과가 제일 좋을 때는

Feature=(pclass,name,embarked), k-fold=5, use PCA, dt(depth=10)

Test1 : 82.71734%

Test2 : 77.272%



submission_pca_dt10.csv
Complete · now

0.77272

(Spearman 피처를 쓸 때)

k-fold = 5

피처 5개에 대한 부분집합 $2^5 - 1$ 의 경우의 수를 모두 돌렸을 때,

```
Best subset: ['Sex', 'Name']  
Model: dt CV score: 0.7923608059757704  
Saved final_submission.csv
```

79% 정도로 dt가 가장 좋은 성능을 보이고 있다.

Dt(depth = 10)

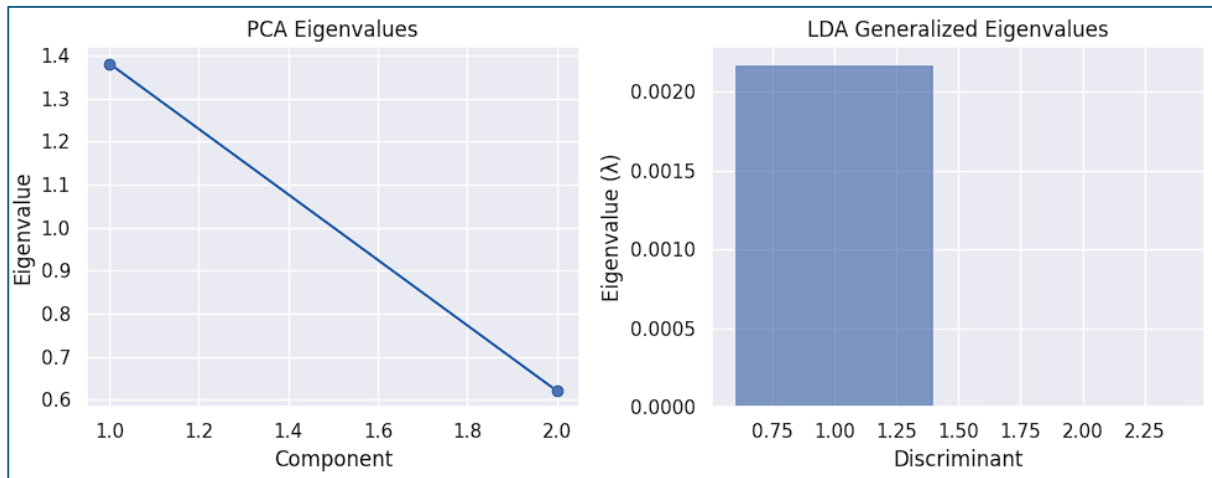
➤ Test1 : 79.236%

따라서,

스피어만 상관계수를 따를 때,

Full feature = Sex, Name 이다.

여기에 대해 pca, LDA를 진행하여 eigenvalue를 시각화하면,



이렇게 나오고

따라서 pca의 차원축소는 2 => 2 (변화없음)

LDA의 차원축소는 2 => 1 로 하겠다.

최종 test1의 acc를 엑셀로 정리하면


### ----- NEW FEATURE ----- ###				
Spearman = Sex, Name				
K-FOLD	Train Data / Test1 Data			
	Logistic R	Decision T	KNN	MLP
full feature	78.22798	79.23608	75.32107	74.98588
PCA	78.22798	79.01262	75.09635	78.11562
FLD	78.56381	79.34781	78.56381	78.56381

최종적으로 결과가 제일 좋을 때는

Feature=(Sex, Name), k-fold=5, use FLD, dt(depth=10)

Test1 : 79.34781%

Test2 : 75.358%

	submission_lda_dt10.csv Complete · now	0.75358
---	---	---------

[Conclusion]

- 상관계수가 높은 피쳐만을 뽑아서 사용하는 것이 오히려 떨어지는 결과를 냈음.

기존 : 77.511 (5-feature, mlp, k-fold)

이후 : 77.272 (3-feature, dt, k-fold) (약 -0.3%)

따라서 둘의 submission을 합친 후

1) And/or 방식

두 submission에서 survived = 1 이라면 생존으로 한다.

submission_ensemble_or.csv Complete · 13m ago	0.77511
submission_ensemble_and.csv Complete · 14m ago	0.77511

- 77.511로 기존과 동일한 결과,


```
Identical predictions: 370/418 (88.52%)
PCA+DT survivors : 123
5-feat MLP survivors: 145
AND intersection survivors: 110
Disagreements (up to 10 rows):
  PassengerId  Survived_pca  Survived_5feat
1           893            0             1
2           894            1             0
16          908            1             0
18          910            0             1
32          924            0             1
33          925            0             1
36          928            0             1
37          929            0             1
49          941            0             1
80          972            1             0
```

- and 연산임에도 결과가 똑같기에, submission을 출력하여 비교한 모습

두 submission은 88% 동일하지만 test2에서의 and 연산 결과는 77.511로 완전히 같은 결과를 내고 있다. (**and/or 채택 불가능**)

2) 가중치 방식(채택)

두 submission에 가중치를 0.5씩 두고 투표하는 방식 이후에 threshold를 통해 0.5 이상인 survival score라면 survived에 1을 채워넣는다.

 submission_weighted_vote.csv Complete · 10m ago	0.77511
--	---------

결과는 변화없음

두 submission의 test2 acc가 다르므로

mlp에서 0.3% 더 높음을 이용해

최종 predict에 미세하게 영향을 끼치게 한다.

```
w_mlp = 0.51  
w_pca = 0.49  
votes = w_pca * sub_pca['Survived'].values + w_mlp * sub5['Survived'].values
```

Mlp+0.01

Pca-0.01

이후, test2에 대한 캐글 결과

submission_weighted_0.49_0.51.csv Complete · 7m ago	0.77751
--	---------

Test2 : 77.751%

+0.2%