Titanic Data 분석을 통한 생존자 예측

(2)

목차 - Titanic Data 분석 (2)

- 데이터 불러오기, 데이터 확인
- 결측치 확인 및 대체
- 데이터 시각화 (Survived, Sex, Embarked, Parch, SibSp)
- 파생변수 생성 (Family size, Alone)
- Feature Engineering (One hot Encoding)
- Feature Selection
- 분류 모델 정의 (Logistic Regression)
- 교차검증(Cross Validation)
- Train, Test Data Split
- Model Train (Logistic Regression)
- Model Evaluation
- Other Models (Decision Tree, RandomForest)
- Predict (나의 생존여부 예측)

(7) Classification Model 정의 – Logistic Regression

생존(1) 또는 **사망(0)**으로 분류할 모델 – **Logistic Regression** 사용

```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression(max_iter = 500) # 임의로 지정(max_iter)
```

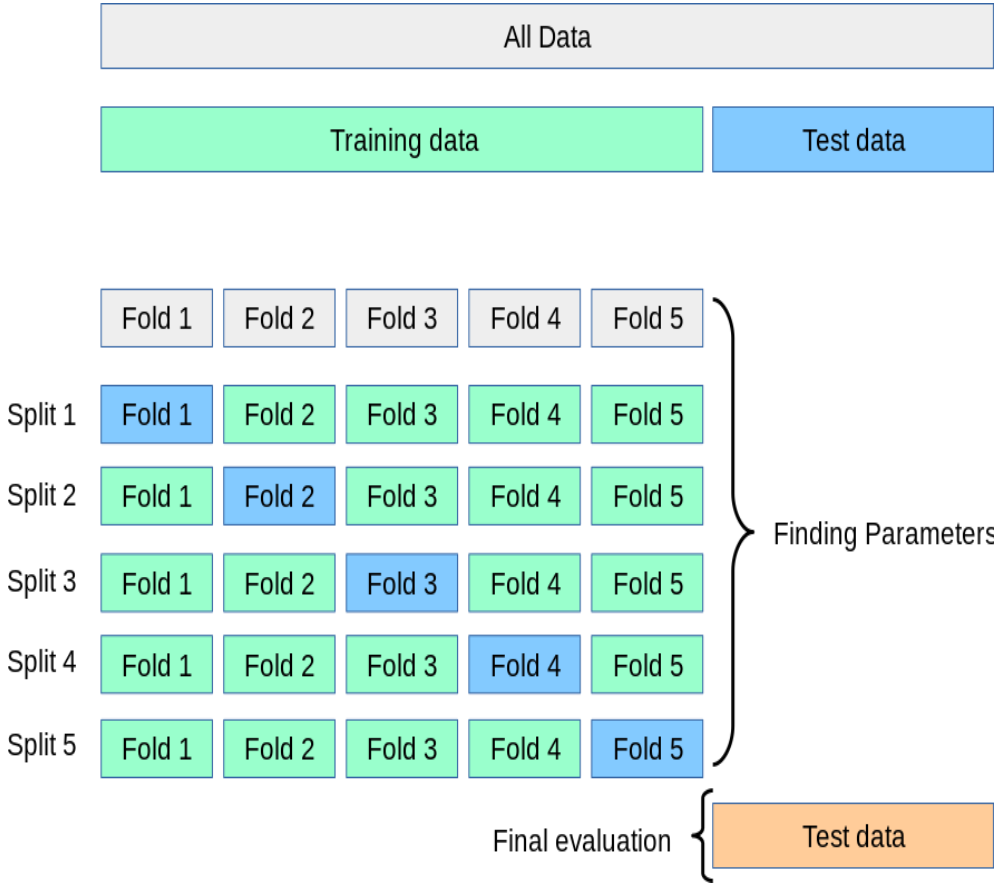
(8) Cross Validation - 교차검증

교차검증 :
Train data를 여러 개의 Fold로 나누어 한 개의 Fold는 검증, 나머지는 학습 데이터가 되어 학습 및 모델 평가를 진행하고, 검증 Fold를 바꾸며 이를 반복한다

Why 교차검증?
- 편향적인 Model (일부 data에만 예측성능이 뛰어남)의 여부를 파악할 수 있다

```
from sklearn.model_selection import cross_val_score

cv = cross_val_score(lr, X, y, cv = 5)
print(cv)
```

```
[0.78212291 0.79213483 0.79775281 0.78089888 0.83146067]
```



(9) Train Test Data Split – 훈련, 테스트 데이터 분리

전체 Data => 학습(Train)과 평가(Test)에 사용할 Data로 **random**으로 분리

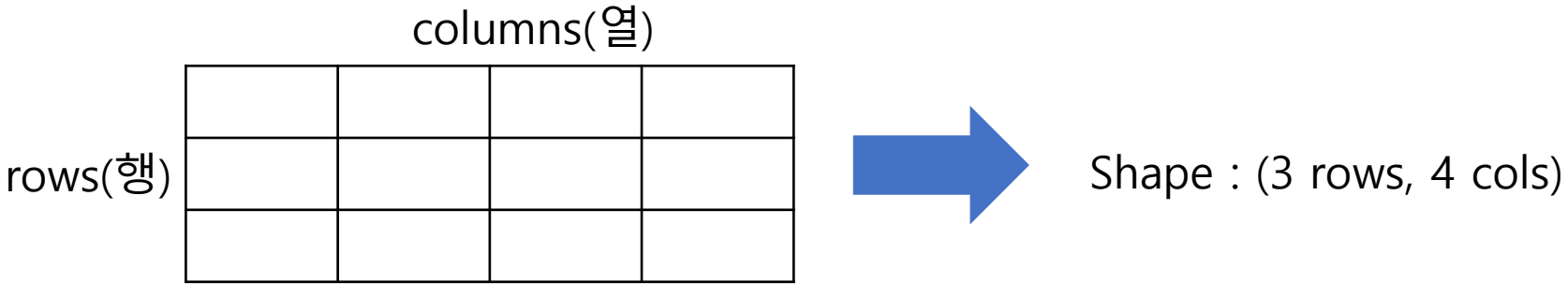
X_train : **학습** 데이터의 **독립변수**
y_train : **학습** 데이터의 **종속변수**
X_test : **평가** 데이터의 **독립변수**
y_test : **평가** 데이터의 **종속변수**

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)

print('X_train.shape:', X_train.shape)
print('y_train.shape:', y_train.shape)
print('X_test.shape:', X_test.shape)
print('y_test.shape:',y_test.shape)
```

Train Data 70%
Test Data 30%

```
X_train.shape: (623, 14)
y_train.shape: (623,)
X_test.shape: (268, 14)
y_test.shape: (268,)
```

Shape? 데이터의 구조(행과 열의 수)을 표현



(10) Model Train – 모델 학습

Logistic Regression Model이 Train data(X, y)를 학습

-> X를 이용하여 Y를 잘 예측하는 모델이 되는 과정

```
lr.fit(X_train, y_train) # train data 학습
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=500,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

(11) Model Evaluation – 모델 평가

y_pred: Model이 X_test를 예측한 결과

y_pred(예측 결과), y_test(실제 결과)를 대조

-> **Model**의 **정확도(Accuracy) 평가**

```
from sklearn.metrics import accuracy_score

y_pred = lr.predict(X_test)
print('정확도:', accuracy_score(y_test, y_pred))
```

```
정확도: 0.8059701492537313
```

```
y_pred # 승선객들에 대한 생존여부 예측 [0: 사망, 1: 생존]
```

```
array([0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0,
       0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0,
       1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0,
       0, 0, 0, 0], dtype=int64)
```

분류모델 평가 지표

Accuracy (정확도), Precision (정밀도), Recall (재현율), F1 Score

<Confusion Matrix>

실제 / 예측	Positive	Negative
Positive	**True Positive (TP)**	False Positive (FP)
Negative	False Negative (FN)	**True Negative (TN)**

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Pos: positive / Neg: Negative

True Positive(TP) : 실제 Pos인 정답을 Pos라고 예측 (정답)
False Positive(FP) : 실제 Neg인 정답을 Pos라고 예측 (오답)
False Negative(FN) : 실제 Pos인 정답을 Neg라고 예측 (오답)
True Negative(TN) : 실제 Neg인 정답을 Neg라고 예측 (정답)

Accuracy : 전체 중 정답(**TP, TN**)을 예측한 확률
Precision : Pos라고 예측한 것 중 실제 Pos인 확률
Recall : 실제 Pos인 것 중 Pos라고 예측한 확률
F1 Score : precision과 recall의 조화평균
(Label 개수가 불균형일 때 효과적)

(12) Other Models – DecisionTree, RandomForest

다른 분류모델 사용 – DecisionTree, RandomForest

```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
y_pred = dt.predict(X_test)
print('정확도:', accuracy_score(y_test, y_pred))
```

```
정확도: 0.7388059701492538
```

Decision Tree(의사 결정트리)

데이터를 분석하여 특정 규칙들과 패턴을 찾아 올바르게 예측/분류하도록 학습하는 모델 (스무고개와 유사)

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
print('정확도:', accuracy_score(y_test, y_pred))
```

```
정확도: 0.7798507462686567
```

Random Forest

다수의 결정트리(Decision Tree)를 사용하여 평균예측치를 출력하여 정확도를 높인 모델

(13) Predict - 나의 생존 여부 예측

내가 Titanic에 있었다면 ????

필요한 나의 Data :

Age	SibSp	Parch	Fare	FamilySize	Alone	1	2	3	female	male	C	Q	S

```
# my data 생성

# Age:23 SibSp: 1 Parch: 2 Fare: 100 FamilySize: 4 Alone: 0
                                # Pclass: 2(0,1,0) Sex: male(0,1) Emb: C(1,0,0)
my = [[23, 1, 2, 100, 4, 0, 0, 1, 0, 0, 1, 1, 0, 0]]
```

```
pred = lr.predict(my)

if pred:
    print('생존!')
else:
    print('사망.')
```

사망.

Summary

- 데이터 불러오기, 데이터 확인 (info, head, tail)
- 결측치 확인 및 대체 (isnull.sum, mean, mode)
- 데이터 시각화 (Matplotlib Seaborn)
- 파생변수 생성 (Family size, Alone)
- 명목변수 인코딩 (One Hot Encoding)
- 변수 선택 (drop)
- 분류 모델 정의 (Logistic Regression)
- 교차검증 (cross_val_score)
- Train, Test data split (train_test_split)
- Model Train (Logistic Regression)
- Model Evaluation (accuracy Score)
- Other Models (Decision Tree, RandomForest)
- Predict (나의 생존여부 예측)

수고하셨습니다!