

모델 평가 및 개선

목차

- 1-1 데이터 불러오기
- 1-2 데이터 탐색하기- EDA
- 1-3 모델 만들고 제출해 보기

개발 환경

- 캐글에서 제공하는 개발환경 Notebook
- 만약 다른 환경에서 수행할 경우, 캐글 데이터 셋을 다운로드 (<https://www.kaggle.com/c/titanic/data>) 후, 수행.

1-1 데이터 불러오기

목차로 이동하기

데이터 경로 확인 및 초기 라이브러리 로드

```
In [24]: # 기본 라이브러리 불러오기
import numpy as np
import pandas as pd

## 캐글 노트북에서 실행 시, 파일의 경로 확인
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

/kaggle/input/titanic/train.csv
/kaggle/input/titanic/test.csv
/kaggle/input/titanic/gender_submission.csv
```

```
In [25]: # Kaggle Notebook 의 경우, 데이터가 있는 경로로 진행
train = pd.read_csv("/kaggle/input/titanic/train.csv")
test = pd.read_csv("/kaggle/input/titanic/test.csv")
```

1-2 데이터 탐색하기(EDA)

목차로 이동하기

EDA

```
In [26]: # 데이터 셋의 행과 열
print(train.shape)
print(test.shape)

# 데이터 셋의 컬럼명
```

```
print(train.columns)
print(test.columns)

(891, 12)
(418, 11)
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
Index(['PassengerId', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch',
       'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

데이터 셋의 자료형

In [27]: train.dtypes

Out[27]:

PassengerId	int64
Survived	int64
Pclass	int64
Name	object
Sex	object
Age	float64
SibSp	int64
Parch	int64
Ticket	object
Fare	float64
Cabin	object
Embarked	object

dtype: object

데이터 셋의 정보 확인

- 데이터 프레임의 구조, 열 이름, 데이터 유형, null 값의 수 및 메모리 사용량에 대한 요약 정보

In [28]:

```
print(train.info())
print()
print(test.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   PassengerId     891 non-null    int64
1   Survived        891 non-null    int64
2   Pclass          891 non-null    int64
3   Name            891 non-null    object
4   Sex             891 non-null    object
5   Age            714 non-null    float64
6   SibSp          891 non-null    int64
7   Parch          891 non-null    int64
8   Ticket         891 non-null    object
9   Fare           891 non-null    float64
10  Cabin          204 non-null    object
11  Embarked       889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
None
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   PassengerId     418 non-null    int64
1   Pclass          418 non-null    int64
2   Name            418 non-null    object
3   Sex             418 non-null    object
4   Age            332 non-null    float64
5   SibSp          418 non-null    int64
6   Parch          418 non-null    int64
7   Ticket         418 non-null    object
8   Fare           417 non-null    float64
9   Cabin          91 non-null     object
10  Embarked       418 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.1+ KB
None
```

데이터의 결측치 확인

```
In [29]: print( train.isnull().sum() )
print()
print( test.isnull().sum() )
```

```
PassengerId    0
Survived        0
Pclass         0
Name           0
Sex            0
Age           177
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin         687
Embarked       2
dtype: int64
```

```
PassengerId    0
Pclass         0
Name           0
Sex            0
Age            86
SibSp          0
Parch          0
Ticket         0
Fare            1
Cabin         327
Embarked       0
dtype: int64
```

데이터의 통계 정보 확인

```
In [30]: print("수치형 통계")
print( train.describe() )
print( )
print( test.describe() )

print("범주형 통계")
print( train.describe(include='O') )
print( )
print( test.describe(include='O') )
```

수치형 통계

	PassengerId	Survived	Pclass	Age	SibSp	W
count	891.000000	891.000000	891.000000	714.000000	891.000000	
mean	446.000000	0.383838	2.308642	29.699118	0.523008	
std	257.353842	0.486592	0.836071	14.526497	1.102743	
min	1.000000	0.000000	1.000000	0.420000	0.000000	
25%	223.500000	0.000000	2.000000	20.125000	0.000000	
50%	446.000000	0.000000	3.000000	28.000000	0.000000	
75%	668.500000	1.000000	3.000000	38.000000	1.000000	
max	891.000000	1.000000	3.000000	80.000000	8.000000	

	Parch	Fare
count	891.000000	891.000000
mean	0.381594	32.204208
std	0.806057	49.693429
min	0.000000	0.000000
25%	0.000000	7.910400
50%	0.000000	14.454200
75%	0.000000	31.000000
max	6.000000	512.329200

	PassengerId	Pclass	Age	SibSp	Parch	Fare
count	418.000000	418.000000	332.000000	418.000000	418.000000	417.000000
mean	1100.500000	2.265550	30.272590	0.447368	0.392344	35.627188
std	120.810458	0.841838	14.181209	0.896760	0.981429	55.907576
min	892.000000	1.000000	0.170000	0.000000	0.000000	0.000000
25%	996.250000	1.000000	21.000000	0.000000	0.000000	7.895800
50%	1100.500000	3.000000	27.000000	0.000000	0.000000	14.454200
75%	1204.750000	3.000000	39.000000	1.000000	0.000000	31.500000
max	1309.000000	3.000000	76.000000	8.000000	9.000000	512.329200

범주형 통계

	Name	Sex	Ticket	Cabin	Embarked
count	891	891	891	204	889
unique	891	2	681	147	3
top	Braund, Mr. Owen Harris	male	347082	B96 B98	S
freq	1	577	7	4	644

	Name	Sex	Ticket	Cabin	Embarked
count	418	418	418	91	418
unique	418	2	363	76	3
top	Kelly, Mr. James	male	PC 17608	B57 B59 B63 B66	S
freq	1	266	5	3	270

1-3 모델 만들고 제출해 보기

[목차로 이동하기](#)

KNN 분류 모델 활용한 모델 구축 및 학습

```
In [31]: from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
In [32]: train.columns
```

```
Out[32]: Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
               'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
              dtype='object')
```

특징 선택 및 데이터 나누기(입력, 출력)

- 결측치 없는 특징 위주로 선택

```
In [35]: # 데이터 준비 - 빠른 모델 생성을 위해 처리 없이 가능한 변수만 선택
# 'Survived'를 제외 ,
# 'Embarked', 'Sex', 'Name', 'Ticket' => 문자포함
# 'Age'는 결측치 존재
sel = ['PassengerId', 'Pclass', 'SibSp', 'Parch' ]

# 학습에 사용될 데이터 준비 X_train, y_train
X = train[sel]
y = train['Survived']

# 예측 및 평가에 사용되는 데이터 셋
X_test = test[sel]
```

```
In [36]: # 훈련 데이터를 학습용과 검증용으로 나누기 (80% 학습, 20% 검증)
X_train, X_val, y_train, y_val = train_test_split(X, y,
                                                    test_size=0.2, random_state=42)
```

```
In [49]: # 모델 선택
model = KNeighborsClassifier()

# 모델 학습
model.fit(X_train, y_train)

# 검증 데이터로 예측 및 정확도 측정
knn_val_pred = model.predict(X_val)
accuracy = accuracy_score(y_val, knn_val_pred)
print(f"검증 데이터에 대한 모델 정확도: {accuracy:.4f}")
```

검증 데이터에 대한 모델 정확도: 0.5754

```
In [43]: from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
```

```
In [44]: # 2. 랜덤 포레스트 모델
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
rf_val_predictions = rf_model.predict(X_val)
rf_accuracy = accuracy_score(y_val, rf_val_predictions)
print(f"랜덤 포레스트 모델 검증 데이터 정확도: {rf_accuracy:.4f}")
```

```
# 3. 로지스틱 회귀 모델
lr_model = LogisticRegression(max_iter=1000, random_state=42)
lr_model.fit(X_train, y_train)
lr_val_predictions = lr_model.predict(X_val)
lr_accuracy = accuracy_score(y_val, lr_val_predictions)
print(f"로지스틱 회귀 모델 검증 데이터 정확도: {lr_accuracy:.4f}")
```

랜덤 포레스트 모델 검증 데이터 정확도: 0.6592

로지스틱 회귀 모델 검증 데이터 정확도: 0.7151

```
In [45]: # 학습 완료된 모델을 활용한 테스트 데이터 예측
pred = lr_model.predict(X_test)
pred[:15]
```

```
Out[45]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1])
```

```
In [50]: # 각 모델의 상세 분류 보고서 생성
print("\n===== KNN 분류 보고서 =====")
print(classification_report(y_val, knn_val_pred))
```

```
print("\n===== 랜덤 포레스트 분류 보고서 =====")
print(classification_report(y_val, rf_val_predictions))

print("\n===== 로지스틱 회귀 분류 보고서 =====")
print(classification_report(y_val, lr_val_predictions))
```

```
===== KNN 분류 보고서 =====
              precision    recall  f1-score   support

         0       0.61      0.75      0.68      105
         1       0.48      0.32      0.39       74

    accuracy          0.58      179
   macro avg       0.55      0.54      0.53      179
weighted avg       0.56      0.58      0.56      179
```

```
===== 랜덤 포레스트 분류 보고서 =====
              precision    recall  f1-score   support

         0       0.70      0.72      0.71      105
         1       0.59      0.57      0.58       74

    accuracy          0.66      179
   macro avg       0.65      0.65      0.65      179
weighted avg       0.66      0.66      0.66      179
```

```
===== 로지스틱 회귀 분류 보고서 =====
              precision    recall  f1-score   support

         0       0.72      0.84      0.78      105
         1       0.70      0.54      0.61       74

    accuracy          0.72      179
   macro avg       0.71      0.69      0.69      179
weighted avg       0.71      0.72      0.71      179
```

실제 제공된 제출용 파일 불러오기

```
In [46]: sub = pd.read_csv("/kaggle/input/titanic/gender_submission.csv")
sub.head(15)
```

Out[46]:

	PassengerId	Survived
0	892	0
1	893	1
2	894	0
3	895	0
4	896	1
5	897	0
6	898	1
7	899	0
8	900	1
9	901	0
10	902	0
11	903	0
12	904	1
13	905	0
14	906	1

```
In [47]: sub['Survived'] = pred
sub.head(15)
```

Out[47]:

	PassengerId	Survived
0	892	0
1	893	0
2	894	0
3	895	0
4	896	0
5	897	0
6	898	0
7	899	0
8	900	0
9	901	0
10	902	0
11	903	1
12	904	1
13	905	0
14	906	1

파일 생성 후, 제출


```
In [41]: sub.to_csv("lr_second_model.csv", index=False)
```