EDA 수행 및 머신러닝 학습모델 만들기 기본 익히기

- 타이타닉 생존자 데이터 feature들간의 특성 파악하기
- 타이타닉 생존자 예측 학습모델 만들기
- 데이터 셋: 타이타닉 데이터셋 (출처: Kaggle.com)

```
import numpy as np
import pandas as pd
```

```
train = pd.read_csv('./titanic_clean.csv')
```

기초 통계 분석 및 EDA : 타이타닉 데이터셋 탐색하기

```
train.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3.0	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7
1	2	1	1.0	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71
2	3	1	3.0	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7
3	4	1	1.0	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53
4	5	0	3.0	Allen, Mr. William Henry	male	35.0	0	0	373450	8

```
#Pclass 정수로 type 변환
train['Pclass'] =train['Pclass'].astype(int)
```

```
train['Age'] = train['Age'].astype(int)
```

```
train.columns
```

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked', 'Sex_num',
       'Embarked_num', 'E_C', 'E_Q', 'E_S'],
      dtype='object')
```

- PassengerId: 승객 아이디
- Survived: 생존 여부, 1: 생존, 0: 사망
- Pclass: 등급
- Name: 성함
- Sex: 성별
- Age: 나이
- SibSp: 형제, 자매, 배우자 수
- Parch: 부모, 자식 수
- Ticket: 티켓번호
- Fare: 요금(운임)
- Cabin: 좌석번호
- Embarked: 탑승 항구

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 17 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           891 non-null    int64  
 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         891 non-null    object 
 11  Embarked      891 non-null    object 
 12  Sex_num       891 non-null    int64  
 13  Embarked_num  891 non-null    int64  
 14  E_C           891 non-null    int64  
 15  E_Q           891 non-null    int64  
 16  E_S           891 non-null    int64  
dtypes: float64(1), int64(11), object(5)
memory usage: 118.5+ KB
```

```
train.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.00
mean	446.000000	0.383838	2.303030	29.544332	0.523008	0.381594	32.20
std	257.353842	0.486592	0.833418	13.013778	1.102743	0.806057	49.69
min	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.00
25%	223.500000	0.000000	2.000000	22.000000	0.000000	0.000000	7.91
50%	446.000000	0.000000	3.000000	29.000000	0.000000	0.000000	14.45
75%	668.500000	1.000000	3.000000	35.000000	1.000000	0.000000	31.00
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.32

시각화 하여 살펴보기

```
import matplotlib.pyplot as plt
import seaborn as sns

df_eda = train.loc[:,'PassengerId':'Embarked']

# 데이터 프레임간의 correlation(상관관계)를 살펴봄.
df_eda.corr()['PassengerId':'Fare']
```

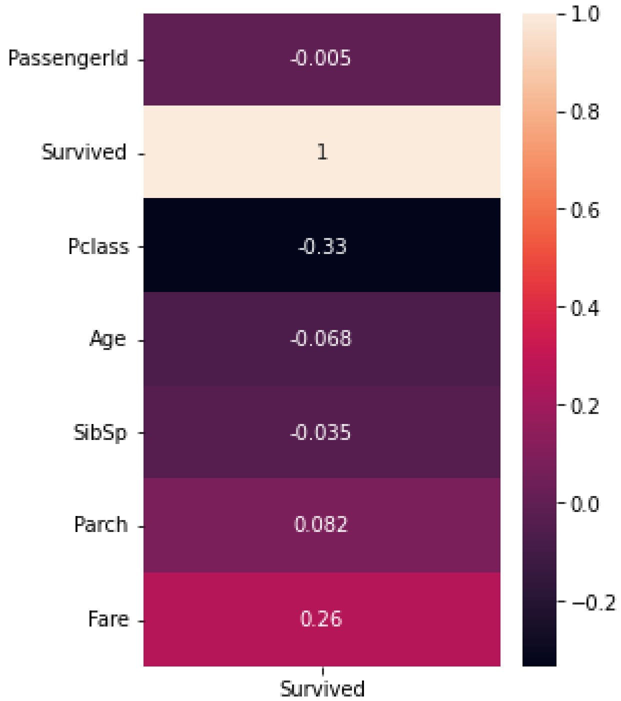
	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.025124	0.033741	-0.057527	-0.001652	0.01265
Survived	-0.005007	1.000000	-0.334241	-0.067809	-0.035322	0.081629	0.25730
Pclass	-0.025124	-0.334241	1.000000	-0.334923	0.082875	0.021693	-0.54798
Age	0.033741	-0.067809	-0.334923	1.000000	-0.232743	-0.176744	0.09385
SibSp	-0.057527	-0.035322	0.082875	-0.232743	1.000000	0.414838	0.15965
Parch	-0.001652	0.081629	0.021693	-0.176744	0.414838	1.000000	0.21622
Fare	0.012658	0.257307	-0.547980	0.093856	0.159651	0.216225	1.00000

```
# DataFrame의 corr() 메소드와 Seaborn의 heatmap() 메소드를 이용
fig = plt.figure(figsize=(10,10))
sns.heatmap(df_eda.corr(), annot=True)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fd41bda1ed0>


```
fig = plt.figure(figsize=(4, 6))
sns.heatmap(df_eda.corr()[['Survived']], annot=True)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fd413412250>



각 feature와 생존(Survived)과의 관계 시각화

```
df_eda.columns
```

df_eda.columns

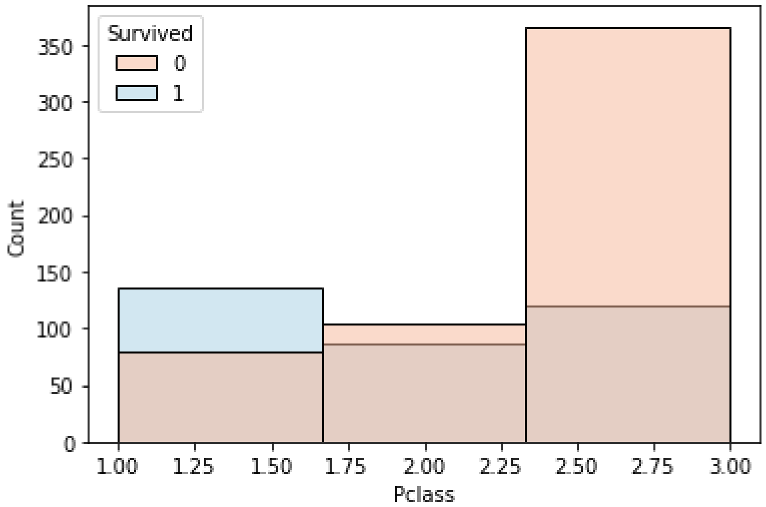
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

```
df_eda['Pclass'].value_counts()
```

```
3    485
1    215
2    191
Name: Pclass, dtype: int64
```

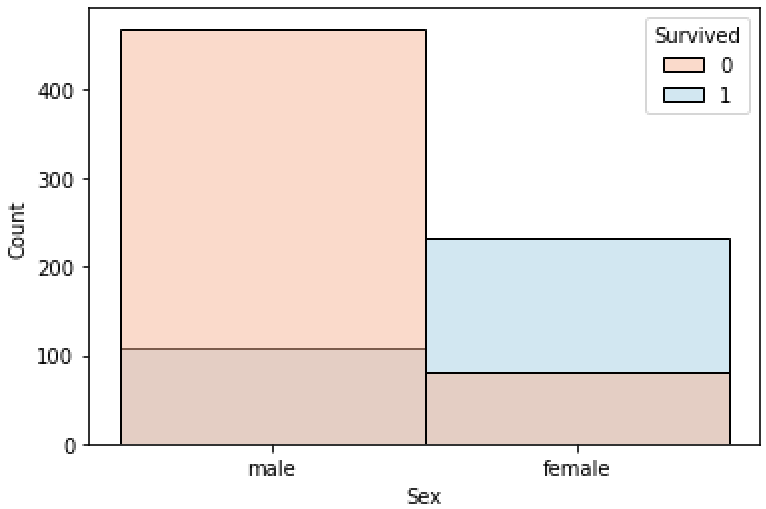
```
# Seaborn의 countplot() 및 histplot()을 사용하여 각 컬럼과 생존과의 관계를 시
sns.histplot(x='Pclass', data=df_eda, hue='Survived', palette='RdBu',bins=3)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd412f35b50>
```



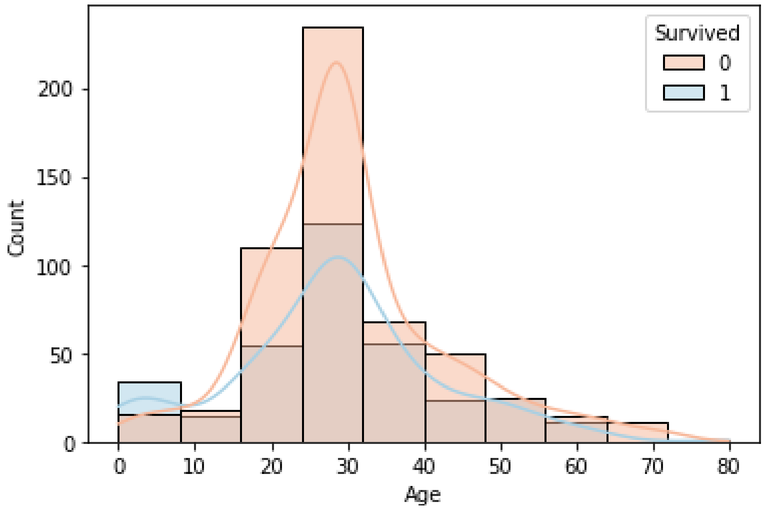

```
sns.histplot(x='Sex', data=df_eda, hue='Survived', palette='RdBu',bins=2)
```

```
<AxesSubplot:xlabel='Sex', ylabel='Count'>
```




```
sns.histplot(x='Age', data=df_eda, hue='Survived', palette='RdBu',bins=10, kd
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fd411bdae50>




```
sns.histplot(x='Fare', data=df_eda, hue='Survived', palette='RdBu', bins=8, k
```

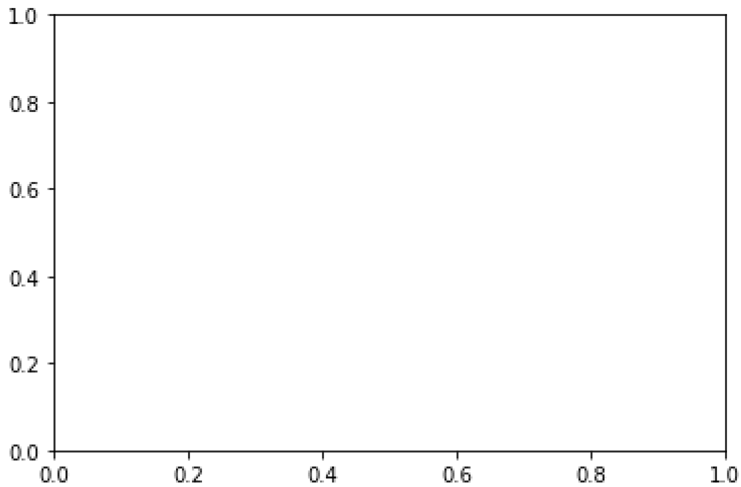
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-25-91fe3a318319> in <module>()
----> 1 sns.histplot(x='Fare', data=df_eda, hue='Survived', camp='RdBu', bins=8, kde=True)
```

6 frames

```
/usr/local/lib/python3.7/dist-packages/matplotlib/artist.py in _update_property(self, k, v)
   1000                 if not callable(func):
   1001                     raise AttributeError('{!r} object has no property {!r}'
-> 1002                                          .format(type(self).__name__, k))
   1003                 return func(v)
   1004 

AttributeError: 'Rectangle' object has no property 'camp'
```

SEARCH STACK OVERFLOW

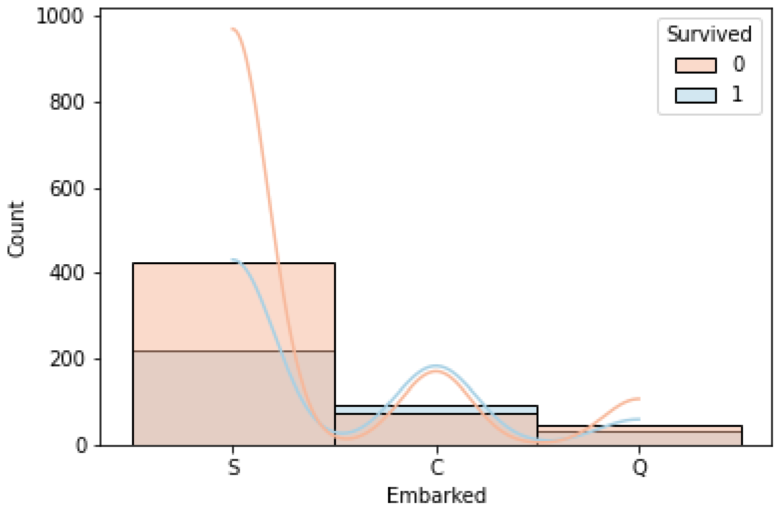



```
df_eda['Embarked'].value_counts()
```

```
S    646
C    168
Q     77
Name: Embarked, dtype: int64
```

```
sns.histplot(x='Embarked', data=df_eda, hue='Survived', palette='RdBu', bins=
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd4134a3a10>
```

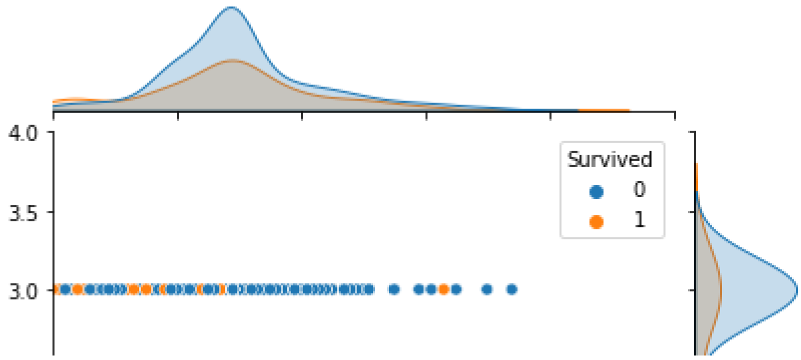



```
df_eda.info()
```

```
<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    int32  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          891 non-null    int32  
 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        891 non-null    object 
 11  Embarked     891 non-null    object 
dtypes: float64(1), int32(2), int64(4), object(5)
memory usage: 76.7+ KB
```

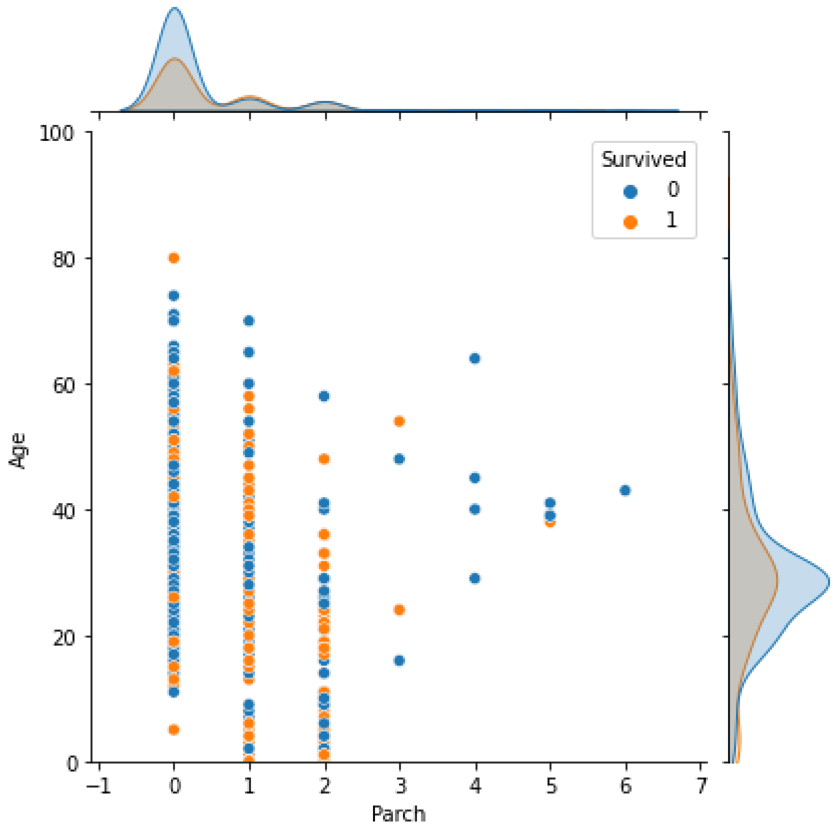
```
sns.jointplot(x='Age', y='Pclass', data=df_eda, hue='Survived', xlim=(0,100),
```

```
<seaborn.axisgrid.JointGrid at 0x1f7b9f25a60>
```

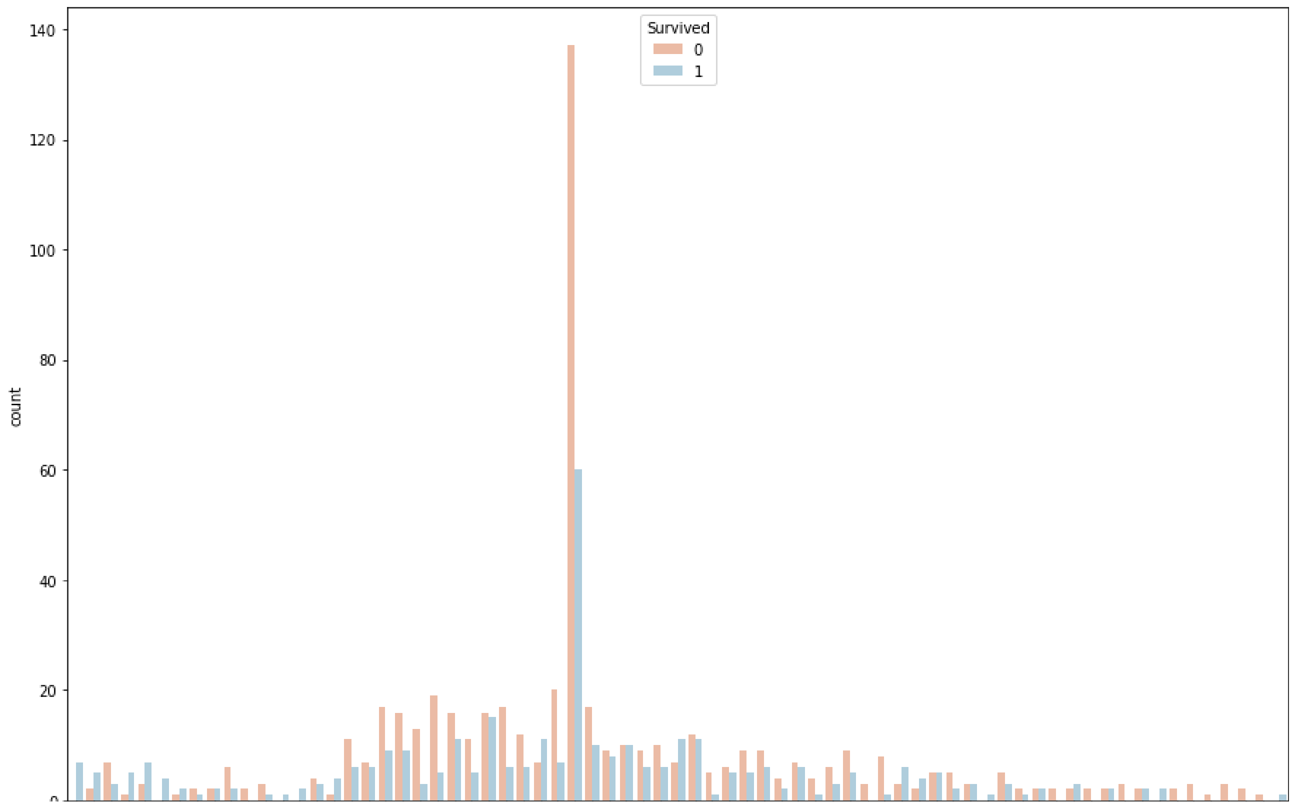



```
sns.jointplot(x='Parch', y='Age', data=df_eda, hue='Survived', ylim=(0,100))
```

```
<seaborn.axisgrid.JointGrid at 0x1f7ba0458b0>
```




```
plt.figure(figsize=(15,10)) #sns보다 위쪽라인에 있어야 함.
plt.xticks(rotation=90)
sns.countplot(x='Age', data=df_eda, hue='Survived', palette='RdBu')
plt.show()
```

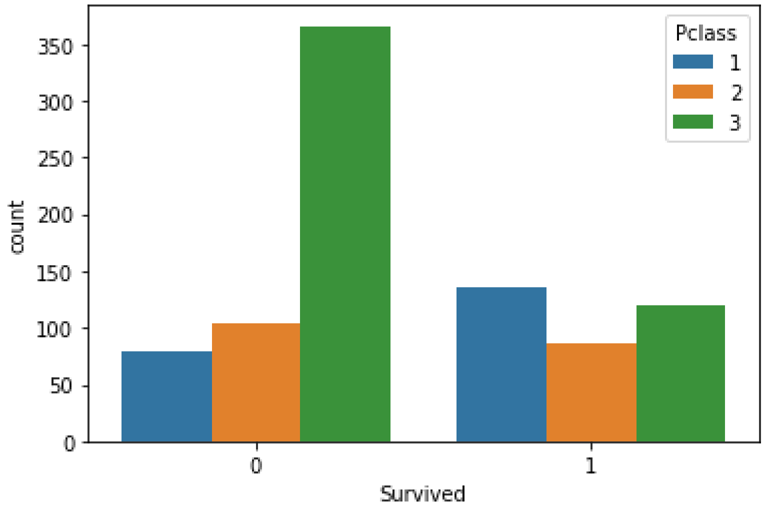



```
df_eda['Embarked'].value_counts()
```

```
S    646
C    168
Q     77
Name: Embarked, dtype: int64
```

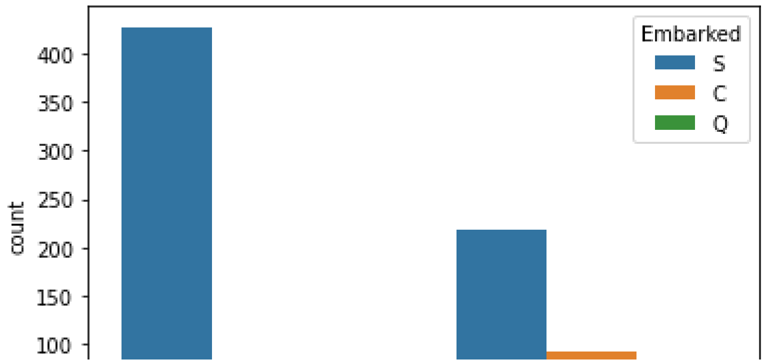
```
sns.countplot(x='Survived', data=df_eda, hue='Pclass', hue_order=[1, 2, 3])
```

<AxesSubplot:xlabel='Survived', ylabel='count'>




```
sns.countplot(x='Survived', data=df_eda, hue='Embarked', hue_order=['S', 'C',
```

<AxesSubplot:xlabel='Survived', ylabel='count'>



전처리: 학습 데이터(feature)와 정답 데이터(label) 구분

Survived

1. feature(X) 와 label(y) 정의하기
2. feature, label을 정의했으면, 적절한 비율로 train / validation set 나누기

```
train.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0	PC 17599	71.
2	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.
3	4	1	1	Futrelle, Mrs. Jacques	female	35	1	0	113803	53.

```
# feature(X)의 항목 list
feature = ['Pclass', 'Sex_num', 'Age', 'Fare', 'E_C', 'E_Q', 'E_S']
X = train[feature]
```

Henry

```
# label(y)의 항목
label = 'Survived'
y = train[label]
y
```

```
0    0
1    1
2    1
```

```
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

```
train.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0	PC 17599	71.
2	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.
4	5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.

train / validation 세트 나누기

from sklearn.model_selection import train_test_split

```
from sklearn.model_selection import train_test_split
```

- **test_size**: validation set에 할당할 비율 (20% -> 0.2)
- **shuffle**: 셔플 옵션 (기본 True)
- **random_state**: 랜덤 시드값

```
# return받는 데이터의 순서 꼭 지키기, random_state=10, shuffle=True와 False인
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ran
```

```
# 학습 데이터 shape 확인
X_train.shape, y_train.shape
```

    ((668, 7), (668,))

```
# 테스트 데이터 shape 확인
X_test.shape, y_test.shape
```

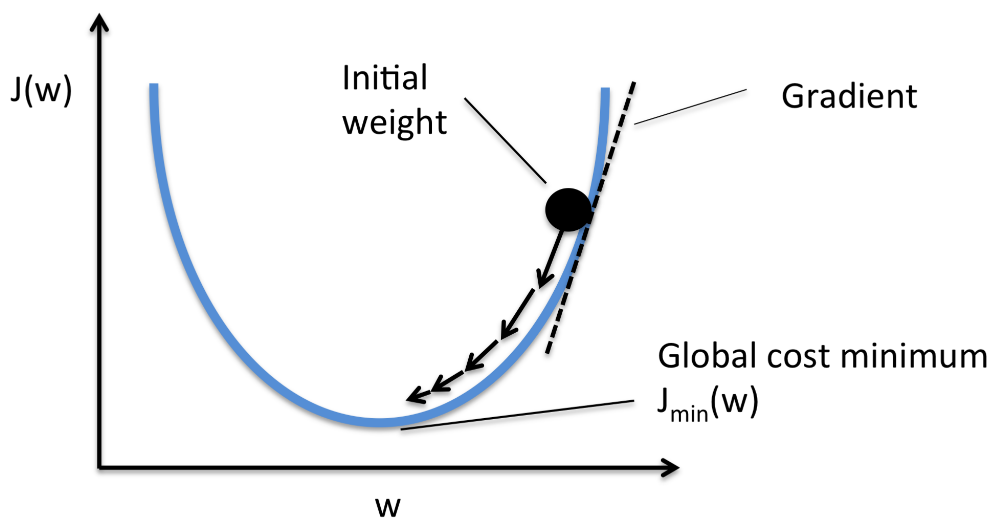
    ((223, 7), (223,))

학습하기

stochastic gradient descent (SGD) : 확률적 경사 하강법

[SGDClassifier scikit-learn 문서]()

from sklearn.linear_model import SGDClassifier




```
# 모델 학습을 위한 모듈 import
from sklearn.linear_model import SGDClassifier
```

```
# 모델 객체 생성,SGD(Stochastic Gradient Descent)
model_sgd = SGDClassifier(random_state=0)
model_sgd
```

    SGDClassifier(random_state=0)

```
# 모델 학습
model_sgd.fit(X_train, y_train)
```

    SGDClassifier(random_state=0)

예측하기

- 학습 모델이 없으면 예측도 없음

```
# 테스트 데이터를 넣어서 예측결과 확인
pred = model_sgd.predict(X_test)
pred
```

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

```
y_test
```

```
590    0
131    0
628    0
195    1
230    1
      ..
12     0
203    0
84     1
886    0
759    1
Name: Survived, Length: 223, dtype: int64
```

```
# 실제값과 예측값을 맞춘 평균 비율
(pred == y_test).mean()
```

```
0.6681614349775785
```

성능 평가하기

```
from sklearn.metrics import classification_report
```

```
# Predict를 수행하고 classification_report() 결과 출력하기
print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support
```

```
           0       0.67      0.97      0.79       147
           1       0.58      0.09      0.16        76

    accuracy                           0.67       223
   macro avg       0.63      0.53      0.48       223
weighted avg       0.64      0.67      0.58       223
```

SGD 모델 계수로 상관성 파악하기

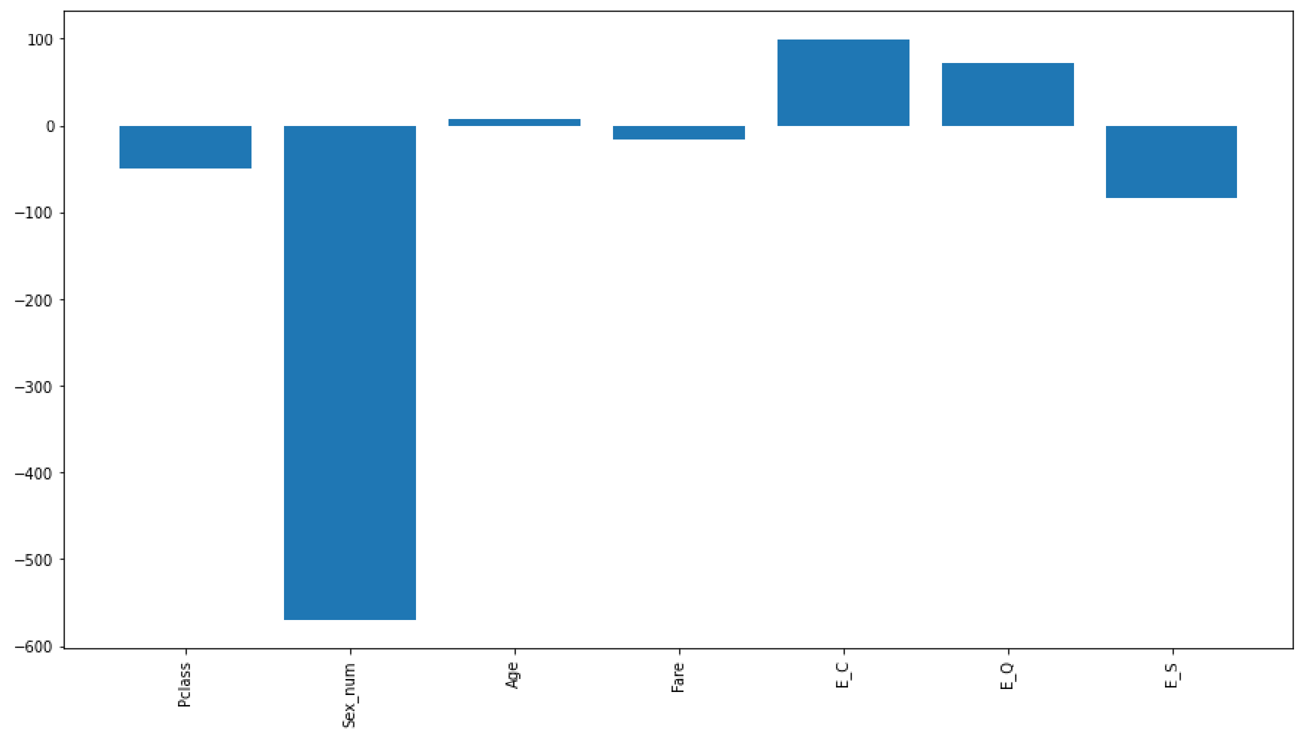
```
model_sgd.classes_
```

```
array([0, 1])
```

```
model_sgd.coef_.shape
```

```
(1, 7)
```

```
# sgd 모델의 coef_ 속성을 plot하기
fig = plt.figure(figsize=(15,8))
plt.bar(X.columns, model_sgd.coef_[0,:])
plt.xticks(rotation=90)
plt.show()
```



[문제해결] 결정트리 학습모델 만들기, 예측하기, 성능평가

- SGD 모델과 성능을 비교해 보세요.

```
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

dt_clf = DecisionTreeClassifier()
dt_clf.fit(X_train, y_train)
y_pred = dt_clf.predict(X_test)

print('예측 정확도: %.2f' % accuracy_score(y_test, y_pred))
```

```
예측 정확도: 0.79
```

```
# Predict를 수행하고 classification_report() 결과 출력하기
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.86      0.85       147
           1       0.71      0.67      0.69        76

    accuracy                           0.79       223
   macro avg       0.77      0.76      0.77       223
weighted avg       0.79      0.79      0.79       223
```