

A background image of the RMS Titanic sailing on the ocean. The ship is shown from a low angle, emphasizing its massive scale. The two prominent red funnels with black tops are visible against a bright, slightly hazy sky. The ship's white hull and dark portholes are clearly visible on the right side of the frame.

타이타닉 생존자 예측

2019107003 고가현



목차

1. 소개
2. 가정
3. 분석
4. 예측
5. 결론



Getting Started Prediction Competition

Titanic: Machine Learning from Disaster

Start here! Predict survival on the Titanic and get familiar with ML basics



Kaggle · 11,727 teams · Ongoing · ID 3136

[Overview](#)[Data](#)[Kernels](#)[Discussion](#)[Leaderboard](#)[Rules](#)[Team](#)[...](#)[My Submissions](#)[Submit Predictions](#)

Overview

[Edit](#)

Description

1. 소개

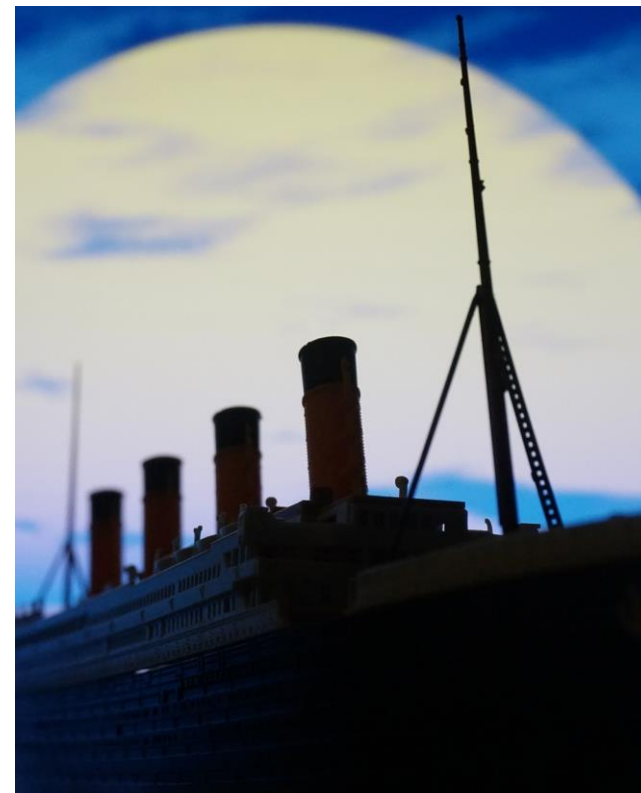
Titanic - Machine Learning from Disaster

소개

타이타닉 사건은 1912년 4월 15일, 빙산과의 충돌로 인해 일어난 일이에요. 불행하게도 구명보트가 충분하지 않았기 때문에 많은 사람들이 사망하게 되었습니다.

이 중에서 생존자의 목록을 보면 어느 특정 유형의 사람들이 생존 여부가 더 많았다고 합니다.

이 Kaggle challenge에서, “어떤 유형의 사람들이 생존했는지?”에 대한 답을 주는 예측 모델을 만드는 것이 목표입니다.



데이터셋

주어진 탑승객 데이터 변수들
(예: 나이, 성별, 등급석 등)은
왼쪽의 표와 같습니다.

Variable	Definition	Key
survival	생존 여부	0 = No, 1 = Yes
pclass	티켓 등급	1 = 1st, 2 = 2nd, 3 = 3rd
sex	성별	
Age	나이	
sibsp	같이 탑승한 형제 또는 배우자 수	
parch	같이 탑승한 부모 또는 자녀 수	
ticket	티켓 번호	
fare	지불한 요금	
cabin	선실 번호	
embarked	탑승한 곳	C = Cherbourg, Q = Queenstown, S = Southampton

데이터셋

'train.csv' 데이터셋과 'test.csv' 데이터셋,

그리고 test 데이터의 정답인

'gender_submission.csv' 데이터셋이 있습니다.

◆ **'train.csv'** (1~891명의 탑승객 정보 /생존 여부 확실)

-> 머신러닝 모델 학습용도

◆ **'test.csv'** -> (892~1309명의 탑승객 정보/ 생존 여부 표시 없음)

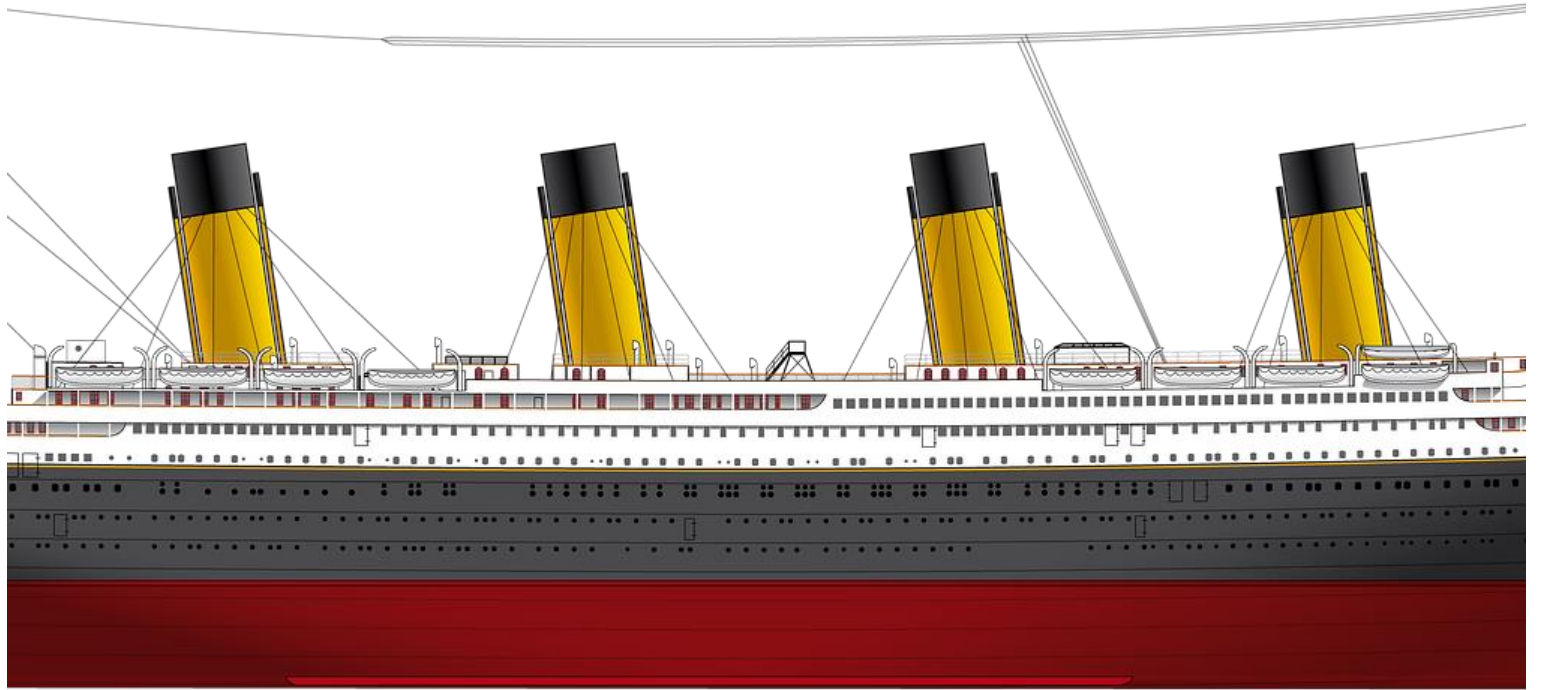
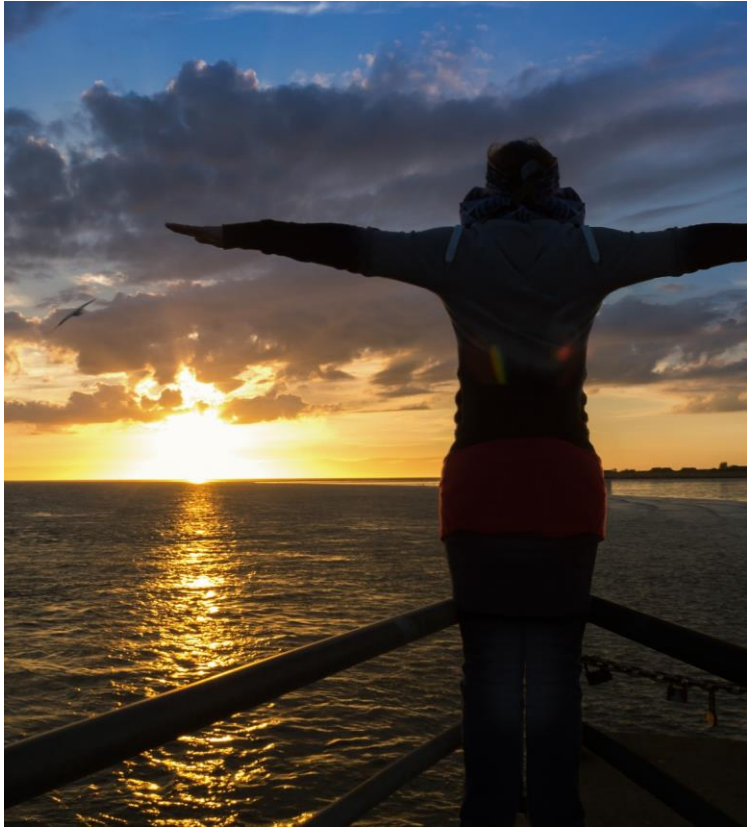
-> 생존 여부를 예측하는 것이 목표입니다.

```
train_df.head(6)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q

```
test_df.head(6)
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
5	897	3	Svensson, Mr. Johan Cervin	male	14.0	0	0	7538	9.2250	NaN	S



2. 가정

가설

1. 노약자를 우대하지 않았을까?

- 나이에 따른 생존 여부에서 어린이와 노인 등이 생존율이 더 클 것이다
- 성별에 따른 생존 여부에서 여자의 생존율이 더 클 것이다.

2. 선실 위치에 따라 다르지 않을까?

- 탑승객이 어디에서 탑승했는지,
- 탑승객이 어느 선실에 있었는지에 따라 다를 것이다.

가정

필요 없는 변수

- 'Name' (이름)과
- 'Ticket' (티켓 번호)는
- 고유키에 해당하므로 분석에 필요 없다 판단하여 제거했습니다.

결측값(NULL)에 대해

- 'Cabin' (선실 번호)는 데이터 셋에서 절반 이상이 null값이므로 Cabin으로 분석이 힘들다고 판단하여 제거했습니다.

```
train_df.drop(['Name'], axis = 1, inplace = True)
train_df.drop(['Ticket'], axis = 1, inplace = True)
train_df.drop(['Cabin'], axis = 1, inplace = True)
test_df.drop(['Name'], axis = 1, inplace = True)
test_df.drop(['Ticket'], axis = 1, inplace = True)
test_df.drop(['Cabin'], axis = 1, inplace = True)
```

가정

```
train_df.isnull().sum()    # 각 열마다 null값이 몇개 있는지 세기
```

```
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
```

```
test_df.isnull().sum()    # 각 열마다 null값이 몇개 있는지 세기
```

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

가정

결측값(NULL)에 대해

- train과 test 데이터를 합친 데이터셋에서
- 'Age' (나이) NULL 값을 중앙값으로 치환하고
- 'Fare' (승선 요금) NULL 값을 중앙값으로 치환할 것입니다.
- 'Embarked' (탑승한 곳)는 가장 많은 S도시로 치환할 것입니다.

```
train_df.drop(['Survived'], axis = 1, inplace = True)
```

```
merge_df = pd.concat([train_df, test_df], keys = ['train', 'test'])
merge_df
```

```
show_median = merge_df.median()
print(show_median)
```

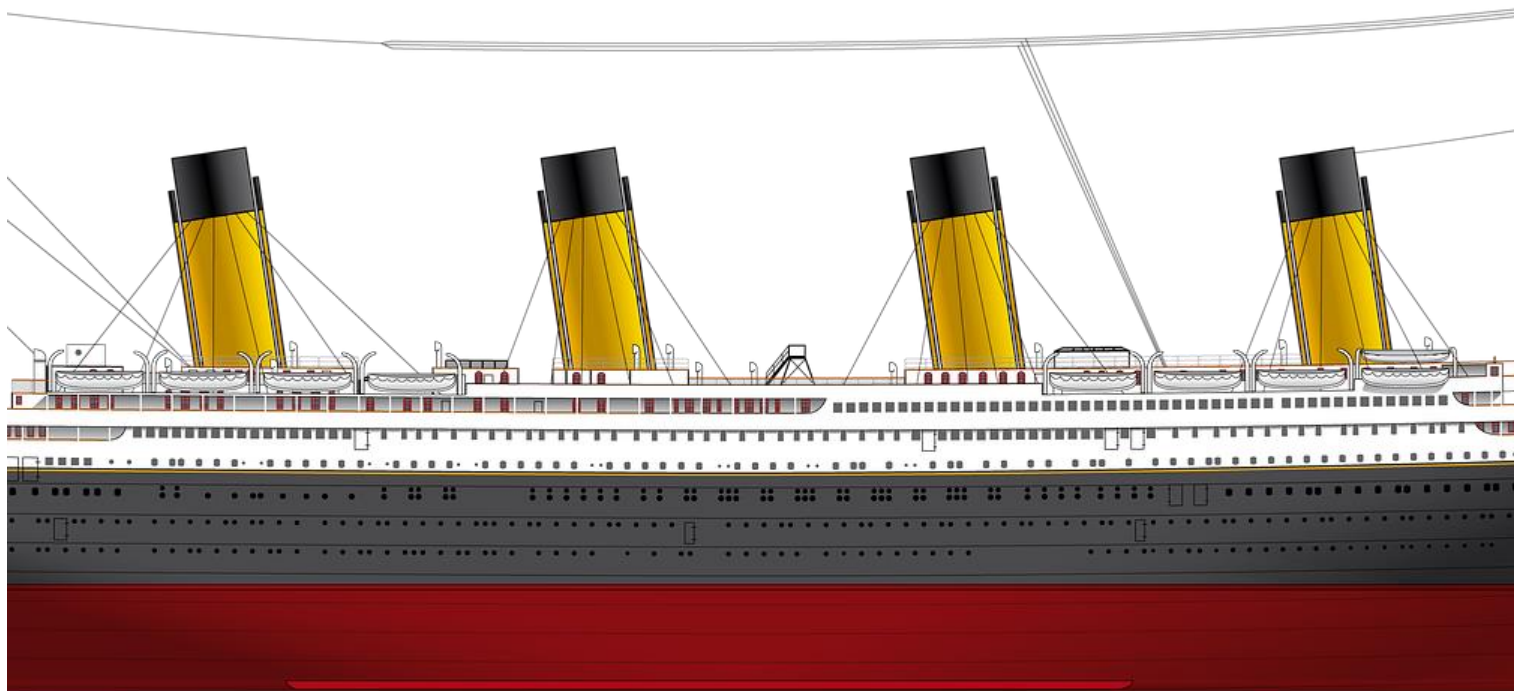
```
PassengerId    655.0000
Pclass          3.0000
Age            28.0000
SibSp           0.0000
Parch           0.0000
Fare           14.4542
dtype: float64
```

```
merge_df['Age'] = merge_df['Age'].fillna(merge_df['Age'].median()) # age열에 null값을 중앙값으로 치환
merge_df['Fare'] = merge_df['Fare'].fillna(merge_df['Fare'].median()) # age열에 null값을 중앙값으로 치환
```

```
merge_df['Embarked'].value_counts() # 각 승선도시에서의 사람 수 세기
```

```
S    914
C    270
Q    123
Name: Embarked, dtype: int64
```

```
merge_df['Embarked'] = merge_df['Embarked'].fillna('S') # null값을 가장 많은 S도시로 치환
```

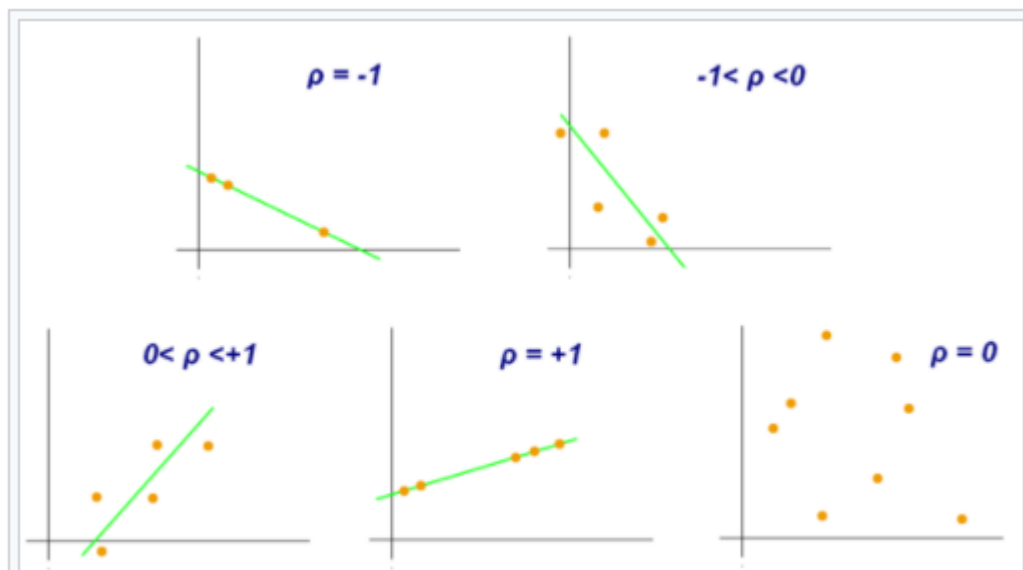


3. 분석

피어슨 상관 계수

위키백과, 우리 모두의 백과사전.

통계학에서, 피어슨 상관 계수(Pearson Correlation Coefficient, PCC)란 두 변수 X 와 Y 간의 선형 상관 관계를 계량화한 수치다. 피어슨 상관 계수는 코시-슈바르츠 부등식에 의해 $+1$ 과 -1 사이의 값을 가지며, $+1$ 은 완벽한 양의 선형 상관 관계, 0 은 선형 상관 관계 없음, -1 은 완벽한 음의 선형 상관 관계를 의미한다. 일반적으로 상관관계는 피어슨 상관관계를 의미하는 상관계수이다.



서로 다른 상관 계수 값 (ρ)을 갖는 산포도 다이어그램의 예

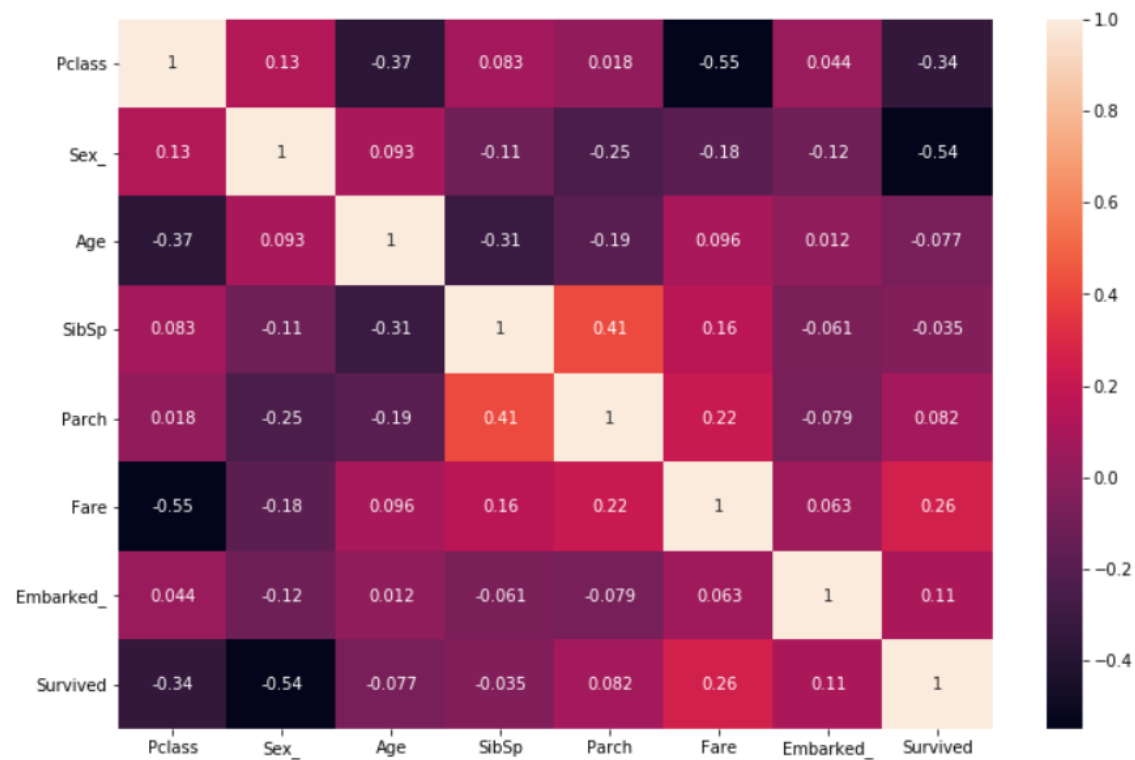
출처:
[피어슨 상관 계수 - 위키백과, 우리 모두의 백과사전 \(wikipedia.org\)](#)

피어슨 상관 계수 - 히트맵

```
def show_heatmap(self, data_f, a):  
    plt.figure(figsize=(12,8))  
    sns.heatmap(data_f[a].corr(),annot=True)
```

```
visualizer = Visualization()  
  
train_df['Sex_'] = train_df['Sex'].map({'male':1, 'female':0})    # sex_ 속성 추가  
train_df['Embarked_'] = train_df['Embarked'].map({'S':1, 'C':2, 'Q':3})    # Embarked_ 속성 추가  
visualizer.show_heatmap(train_df, ['Pclass', 'Sex_', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked_', 'Survived'])
```

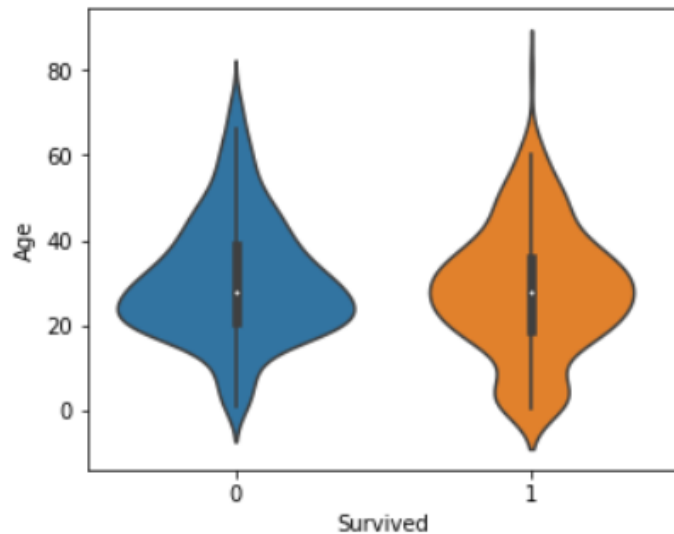

피어슨 상관 계수 - 히트맵



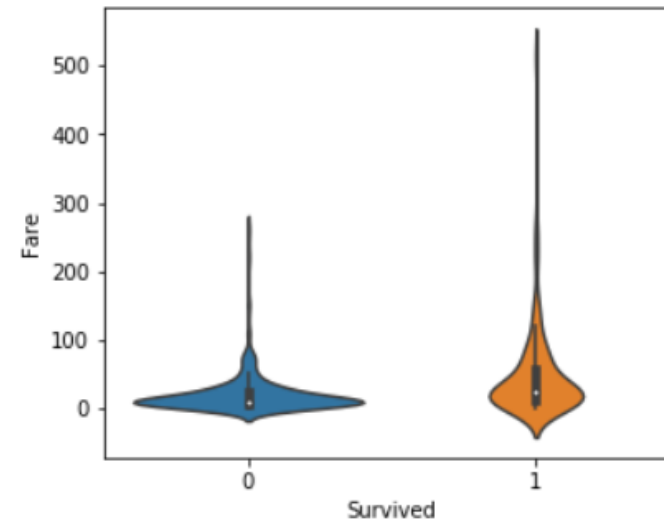
violinplot

```
def violin_plot(self, data_f, a, b):  
    plt.figure(figsize=(5,4))  
    plt.subplot(1,1,1)  
    sns.violinplot(x=a,y=b,data=data_f)
```

```
visualizer.violin_plot(train_df, 'Survived', 'Age')
```

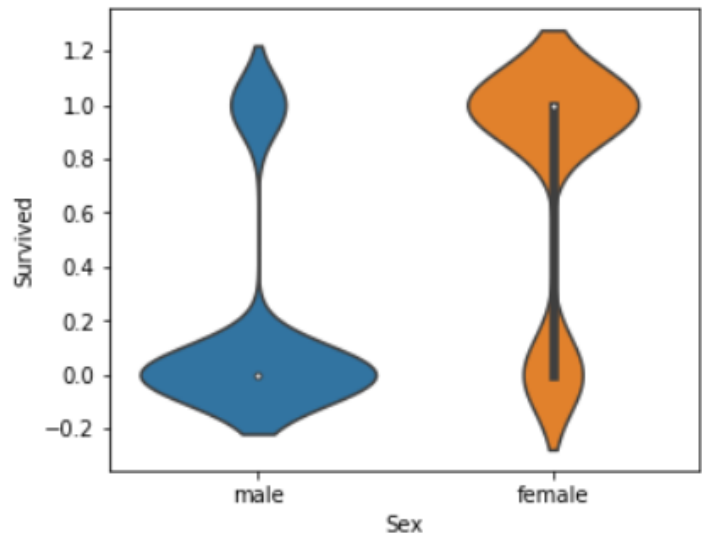


```
visualizer.violin_plot(train_df, 'Survived', 'Fare')
```



violinplot

```
visualizer.violin_plot(train_df, 'Sex', 'Survived')
```



```
women = train_df.loc[train_df.Sex == 'female']["Survived"]  
rate_women = sum(women)/len(women)
```

```
print("% of women who survived:", rate_women)
```

% of women who survived: 0.7420382165605095

```
men = train_df.loc[train_df.Sex == 'male']["Survived"]  
rate_men = sum(men)/len(men)
```

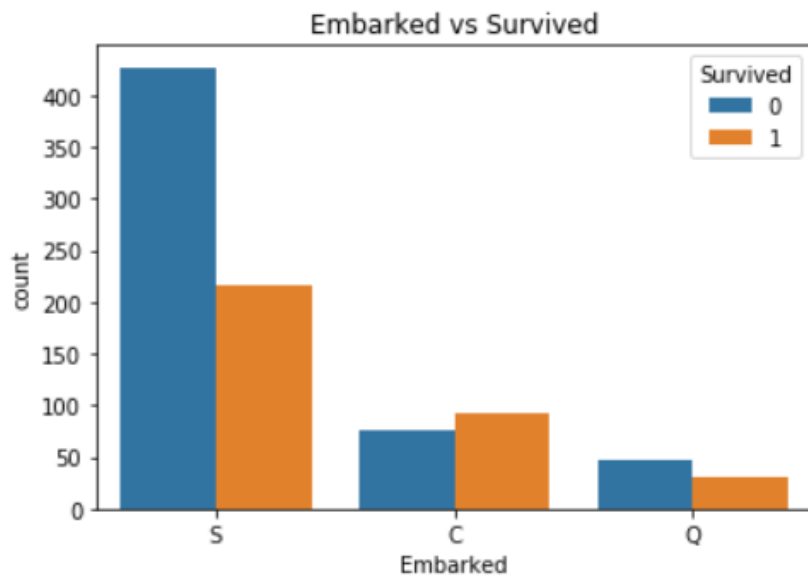
```
print("% of men who survived:", rate_men)
```

% of men who survived: 0.18890814558058924

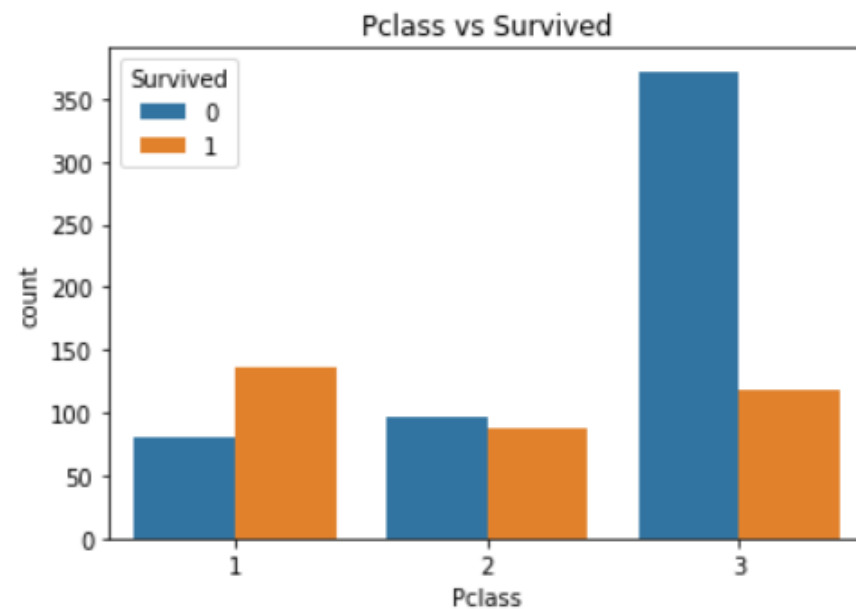
countplot

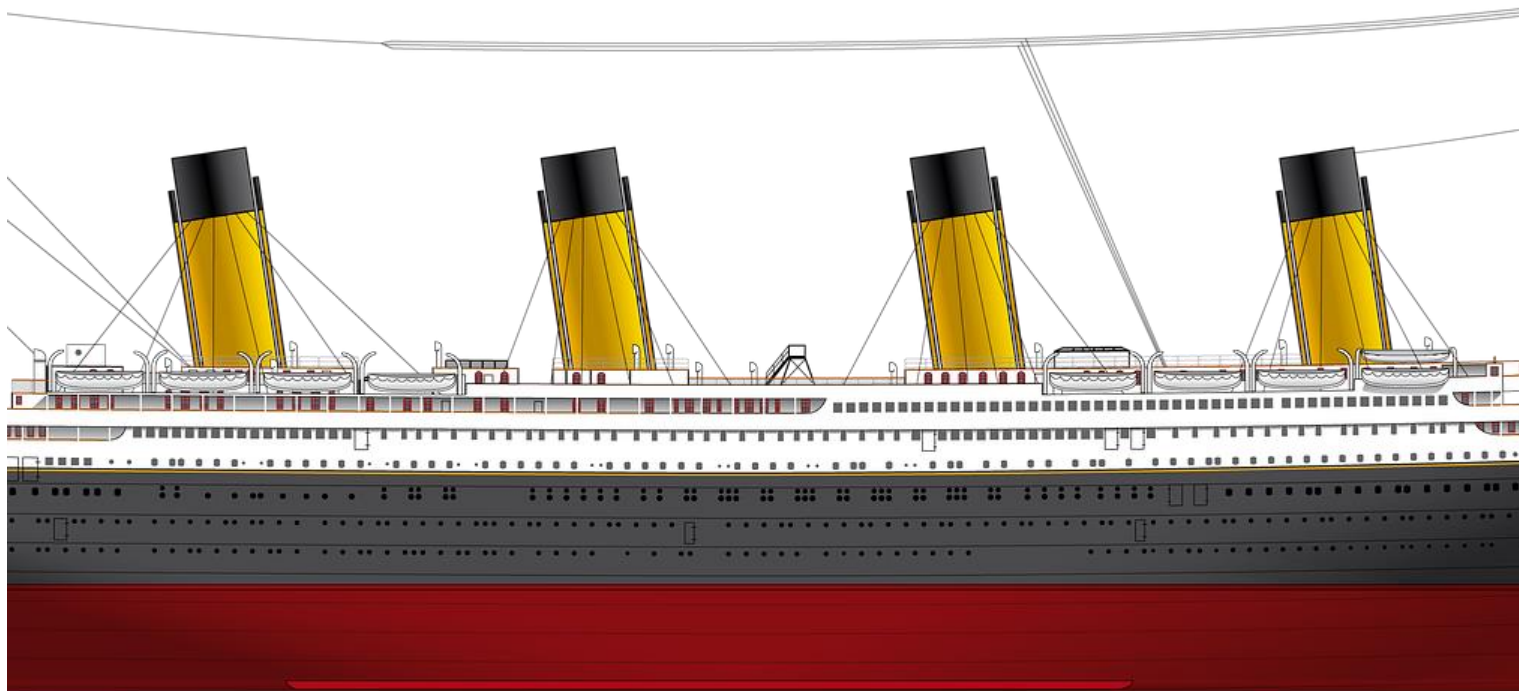
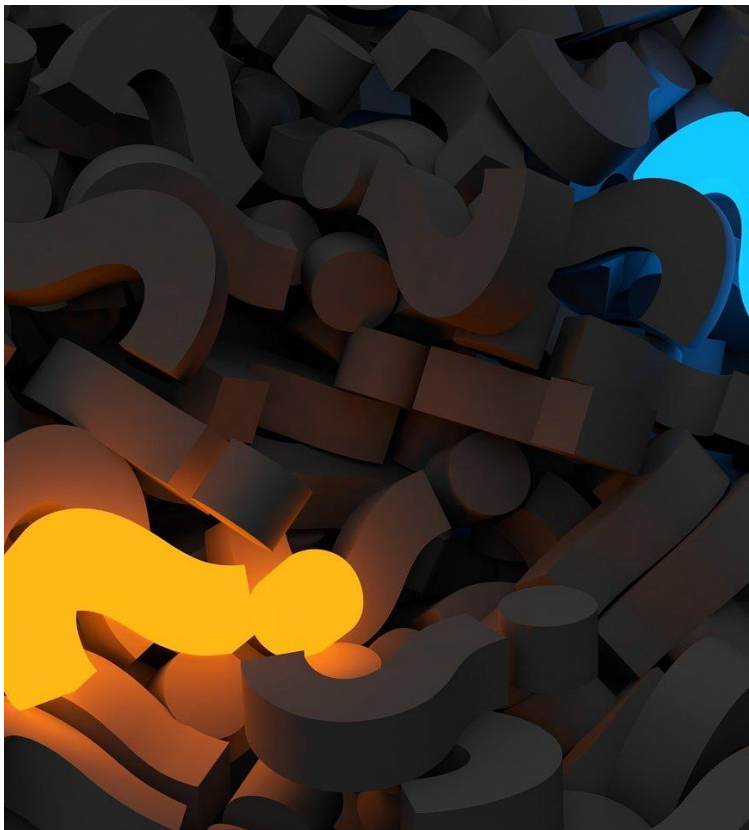
```
def chart(self, x, y, data_f):  
    # seaborn 시각화  
    sns.countplot(x, hue=y, data=data_f)  
  
    plt.title('{} vs {}'.format(x,y))  
    plt.show()
```

```
visualizer.chart('Embarked', 'Survived', train_df)
```



```
visualizer.chart('Pclass', 'Survived', train_df)
```





4. 예측

train.csv로 머신러닝 모델 학습

```
loader = fileload()
loader.show_files()
train_df = loader.load_csv('../input/titanic/train.csv')
train_df.drop(['Name'], axis = 1, inplace = True)
train_df.drop(['Ticket'], axis = 1, inplace = True)
train_df.drop(['Cabin'], axis = 1, inplace = True)
train_df = train_df.dropna(axis=0) # null값 제거
train_df.isnull().sum() # 각 열마다 null값이 몇개 있는지 세기
```

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

```
PassengerId    0
Survived        0
Pclass         0
Sex            0
Age           0
SibSp          0
Parch          0
Fare           0
Embarked       0
dtype: int64
```

```
train_df['Sex_'] = train_df['Sex'].map({'male':1, 'female':0}) # se
x_ 속성 추가
train_df['Embarked_'] = train_df['Embarked'].map({'S':1, 'C':2, 'Q':3})
# Embarked_ 속성 추가
```

```
gildong = MachineLearning()
gildong.split_4_parts(train_df, 0.8, ['Pclass', 'Sex_', 'Age', 'SibS
p', 'Parch', 'Fare', 'Embarked_'], 'Survived') # 학습용(문제, 정답),
테스트용(문제, 정답)으로 데이터 나누기
```


train.csv로 머신러닝 모델 학습

```
gildong.learn_DTR()    # DecisionTreeRegressor  
gildong.predict_score()
```

Score: -0.127

```
gildong.learn_KNR()    # KNeighborsRegressor  
gildong.predict_score()
```

Score: -0.048

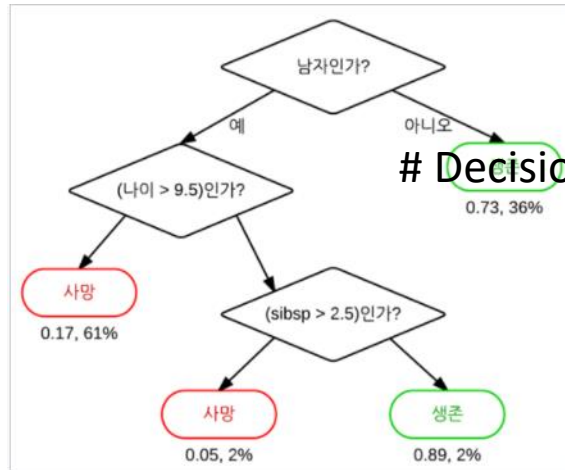
```
gildong.learn_LR()     # LinearRegression  
gildong.predict_score()
```

Score: 0.354

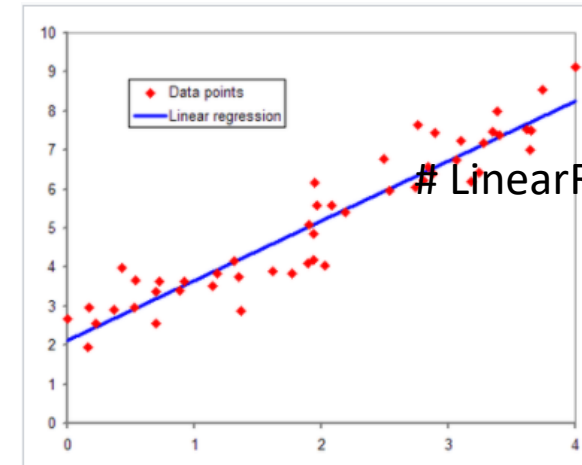
```
gildong.learn_RFR()    # RandomForestRegressor  
gildong.predict_score()
```

Score: 0.304

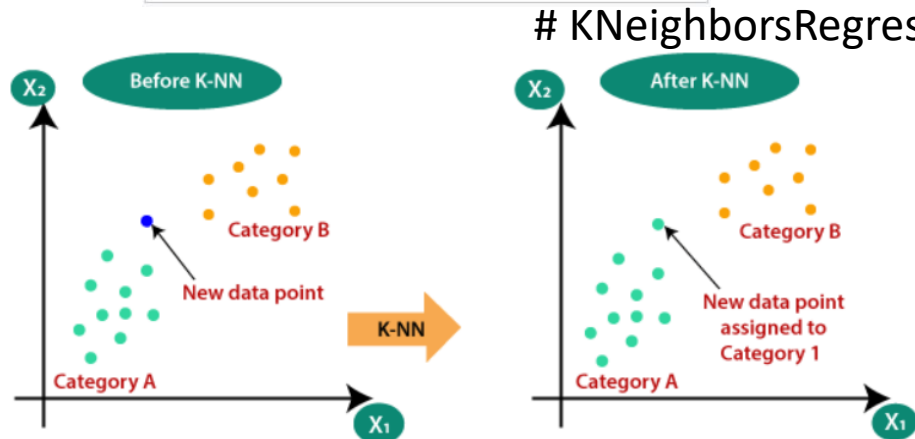
머신러닝 모델



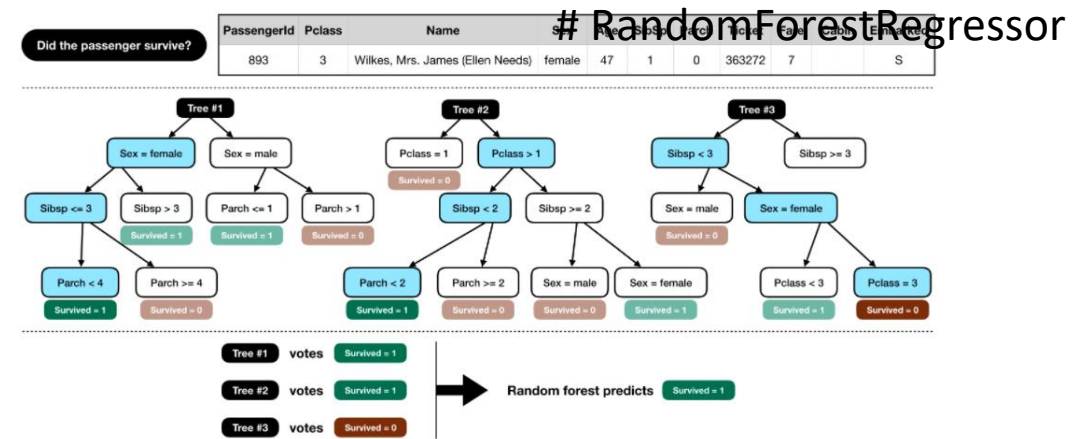
DecisionTreeRegressor



LinearRegression



KNeighborsRegressor



LinearRegression으로 예측

```
test_df['Sex_'] = test_df['Sex'].map({'male':1, 'female':0})    # sex_ 속성 추가
test_df['Embarked_'] = test_df['Embarked'].map({'S':1, 'C':2, 'Q':3})    # Embarked_ 속성 추가
test_var = test_df[['Pclass', 'Sex_', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked_']]
```

```
gildong.learn_LR()
gildong.predict(test_var)
```

```
[ 0.08399135  0.46814656  0.08502059  0.13220996  0.60238579  0.2128899
 0.62506854  0.2404597   0.69999882  0.06845473  0.12576973  0.36159314
 0.9825365   0.03265487  0.82710021  0.78722876  0.2530701   0.16814378
 0.59228882  0.53280946  0.26295752  0.21961628  0.98816801  0.50481533
 0.81843118 -0.06128032  1.01209457  0.15885556  0.39400996  0.02355627
 0.1131528   0.22296856  0.51467841  0.54656943  0.41722732  0.18362551
 0.63909686  0.68264009  0.14486525  0.14149544  0.03641322  0.47322977
 0.04525699  0.79902627  0.83669504  0.14435562  0.36822858  0.12421469
 0.80369947  0.547549   0.45586218  0.30422947  0.73598864  0.8595862
 0.29821396  0.00847543  0.08242469  0.14426664  0.07546466  1.01252281
 0.19388336  0.27352652  0.18764403  0.67463672  0.49119475  0.81399916
 0.69945521  0.36070761  0.45454249  0.7228062   0.66226052  0.16911477]
```

LinearRegression로 예측

```
pre_list = gildong.predict01(test_var)
answer_list = answer_df['Survived'].values.tolist()
```

```
from sklearn.metrics import accuracy_score

accuracy_score(answer_list, pre_list)
```

0.9736842105263158

```
def predict01(self, data):
    # 테스트 문제 전부 주고 테스트하기, test
    predicted = self.gildong.predict(data)
    pre_list = []
    for i in predicted:
        if i >= 0.5:
            pre_list.append(1)
        if i < 0.5:
            pre_list.append(0)
    return pre_list
```



5. 결과


Competition 제출


```
sub_dict = {'PassengerId': range(892,1310), 'Survived': pre_list}
submit_df = pd.DataFrame(sub_dict)
submit_df = submit_df.set_index('PassengerId')
submit_df
```

```
submit_df.to_csv("Titanic_prediction.csv")
```

	Survived
PassengerId	
892	0
893	0
894	0
895	0
896	1
...	...
1305	0
1306	1
1307	0
1308	0
1309	0

418 rows × 1 columns

9773 kohkah  0.76794 1 1m

Your First Entry 
Welcome to the leaderboard!
Your score represents your submission's accuracy. For example, a score of 0.7 in this competition indicates you predicted Titanic survival correctly for 70% of people.
What next? You've got a few options:

- 🔥 Learn skills that can improve your score in [our Intro to Machine Learning course by Dan Becker](#).
- 🔍 Check out [the discussion forum](#) to find lots of tutorials and insights from other competitors.
- 🏆 Find a new challenge by entering one of our [open, active competitions](#) or searching our [public datasets](#).

결론 요약

가설1. 노약자를 우대하지 않았을까?

- 나이에 따른 생존 여부에서, 상대적으로 어린이와 노인은 생존율이 더 컸지만, 전체적으로 나이에 따른 생존율은 크게 두드러지지 않았다
- 성별에 따른 생존 여부에서, 여자의 생존율이 압도적으로 더 많았다.

가설2. 선실 위치에 따라 다르지 않을까?

- Cabin(선실)에 따른 생존 여부는 Cabin의 결측값이 많아 측정할 수 없었다.
- Embarked(탑승한 곳)에 따른 생존 여부는 크게 유의미하지 않다.

3. 지불 요금, PCLASS에 따른 생존율

- 상대적으로 지불 요금이 많으면 생존율이 더 컸다.
- Pclass(사회경제적 지위와 같다)에 따른 생존 여부에서, 생존자가 같은 수일 때 낮은 등급의 사람들이 더 많이 사망했다.
- 즉 경제적 여건이 되면, 살 확률이 높아진다는 것이다.



감사합니다.

발표자 이름 : 고가현

웹 사이트 :

<https://www.kaggle.com/kohkah/titanic-code>