In [2]:
```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import mglearn
%matplotlib inline
import seaborn as sns
import platform
from matplotlib import font_manager , rc

if platform.system() == 'Darwin':
  rc('font' , family = 'AppleGothic')
elif platform.system() == 'Windows':
  path = 'C:/Windows/Fonts/malgun.ttf'
  font_name = font_manager.FontProperties(fname = path).get_name()
  rc('font' , family = font_name)
else:
  print('모름')
plt.rcParams['axes.unicode_minus'] = False
import warnings
warnings.filterwarnings('ignore')
```
executed in 4.48s, finished 09:25:57 2023-11-01

In [137]:
```
# 경로 저장
path = 'C:/k_digital/machine/source/titanic'
```
executed in 18ms, finished 09:56:05 2023-11-01

In [344]:
```
titanic = pd.read_csv(path + '/train.csv')
```
executed in 19ms, finished 11:12:10 2023-11-01

In [212]:
```
titanic.columns
```
executed in 15ms, finished 10:06:25 2023-11-01

Out[212]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [139]:
```
titanic.info()
```
executed in 13ms, finished 09:56:06 2023-11-01

```
<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
```

In [214]:
```
# Cabin과 Ticket 제거.
titanic.drop(['Cabin' , 'Ticket'] , axis = 1 , inplace = True)
```
executed in 17ms, finished 10:06:52 2023-11-01

In [215]:
```
# 학습시킬 데이터들과 target 나누기
data = titanic.loc[: , ['Pclass','Sex','Age','Fare','Embarked', 'SibSp','Parch']]
target = titanic.Survived
```
executed in 17ms, finished 10:07:05 2023-11-01

In [216]:
```
data.info()
```
executed in 12ms, finished 10:07:09 2023-11-01

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Pclass    891 non-null    int64  
 1   Sex       891 non-null    object 
 2   Age       714 non-null    float64
 3   Fare      891 non-null    float64
 4   Embarked  889 non-null    object 
 5   SibSp     891 non-null    int64  
 6   Parch     891 non-null    int64  
dtypes: float64(2), int64(3), object(2)
memory usage: 48.9+ KB
```

In [217]:
```
# Embarked에 결측치 존재 , 최빈값으로 채우기
data.Embarked.value_counts()
```
executed in 11ms, finished 10:07:12 2023-11-01

Out[217]:
```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

In [218]:
```
data.Embarked = data.Embarked.fillna('S')
```
executed in 12ms, finished 10:07:14 2023-11-01

In [219]:
```
data.info()
```
executed in 20ms, finished 10:07:14 2023-11-01

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Pclass    891 non-null    int64  
 1   Sex       891 non-null    object 
 2   Age       714 non-null    float64
 3   Fare      891 non-null    float64
 4   Embarked  891 non-null    object 
 5   SibSp     891 non-null    int64  
 6   Parch     891 non-null    int64  
dtypes: float64(2), int64(3), object(2)
memory usage: 48.9+ KB
```

In [220]:
```
data.Age.value_counts()
```
executed in 13ms, finished 10:07:16 2023-11-01

Out[220]:
```
24.00    30
22.00    27
18.00    26
19.00    25
28.00    25
         ..
36.50     1
55.50     1
0.92      1
23.50     1
74.00     1
Name: Age, Length: 88, dtype: int64
```

In [221]:
```
data.Age = round(data.Age , 0)
```
executed in 12ms, finished 10:07:16 2023-11-01

In [222]:
```
age_mean = round(data.Age.mean() , 0)
```
executed in 15ms, finished 10:07:16 2023-11-01

In [223]:
```
#나이의 평균으로 Age 결측치 대체
data.Age = data.Age.fillna(age_mean)
```
executed in 19ms, finished 10:07:17 2023-11-01

In [224]:
```
data.info()
```
executed in 26ms, finished 10:07:17 2023-11-01

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Pclass    891 non-null    int64  
 1   Sex       891 non-null    object 
 2   Age       891 non-null    float64
 3   Fare      891 non-null    float64
 4   Embarked  891 non-null    object 
 5   SibSp     891 non-null    int64  
 6   Parch     891 non-null    int64  
dtypes: float64(2), int64(3), object(2)
memory usage: 48.9+ KB
```

In [225]:
```
data.Age = data.Age.astype('int')
```
executed in 12ms, finished 10:07:18 2023-11-01

In [226]:
```
data.info()
```
executed in 11ms, finished 10:07:18 2023-11-01

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Pclass    891 non-null    int64  
 1   Sex       891 non-null    object 
 2   Age       891 non-null    int32  
 3   Fare      891 non-null    float64
 4   Embarked  891 non-null    object 
 5   SibSp     891 non-null    int64  
 6   Parch     891 non-null    int64  
dtypes: float64(1), int32(1), int64(3), object(2)
memory usage: 45.4+ KB
```

- 학습시킬 열들을 전부 int로 바꾸는 작업

In [227]:
```
data.Sex[data.Sex=='male'] = 1
```
executed in 14ms, finished 10:07:18 2023-11-01

In [228]:
```
data.Sex[data.Sex=='female'] = 0
```
executed in 16ms, finished 10:07:19 2023-11-01

In [229]:
```
data.Embarked[data.Embarked == 'S'] = 0
data.Embarked[data.Embarked == 'C'] = 1
data.Embarked[data.Embarked == 'Q'] = 2
```
executed in 20ms, finished 10:07:19 2023-11-01

In [230]:
```
data[['Sex','Embarked']] = data[['Sex','Embarked']].astype('int')
```
executed in 14ms, finished 10:07:19 2023-11-01

In [231]:
```
titanic['Survived'].value_counts()
```
executed in 11ms, finished 10:07:20 2023-11-01

Out[231]:
```
0    549
1    342
Name: Survived, dtype: int64
```

In [232]:
```
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()
```
executed in 7ms, finished 10:07:20 2023-11-01

In [233]:
```
# train과 test로 나누기
from sklearn.model_selection import train_test_split

train_input , test_input , train_target , test_target = train_test_split(data , t
```
executed in 15ms, finished 10:07:21 2023-11-01

In [285]:
```
#표준화
ss.fit(train_input)
train_scaled = ss.transform(train_input)
test_scaled = ss.transform(test_input)
```
executed in 15ms, finished 10:29:45 2023-11-01

In [235]:
```
#모델 생성
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
```
executed in 6ms, finished 10:07:21 2023-11-01

In [236]:
```
#파라미터 책정
param = {'C' : [1,10,20,30,40] , 'max_iter' : [100,1000,10000,100000]}
```
executed in 6ms, finished 10:07:22 2023-11-01

In [237]:
```
from sklearn.metrics import accuracy_score
```
executed in 12ms, finished 10:07:23 2023-11-01

In [238]:
```
#그리드서치 실행
from sklearn.model_selection import GridSearchCV
gs = GridSearchCV(lr , param , n_jobs = -1)
gs.fit(train_scaled , train_target)

gs.best_params_
```
executed in 2.47s, finished 10:07:26 2023-11-01

Out[238]:
```
{'C': 30, 'max_iter': 100}
```

In [240]:
```
#그리드서치에서 나온 가장 좋은 파라미터 적용
lr = LogisticRegression(C = 30 , max_iter = 100)
lr.fit(train_scaled , train_target)
accuracy_score(test_target , lr.predict(test_scaled))
```

executed in 18ms, finished 10:07:29 2023-11-01

Out[240]: 0.7982062780269058

In [241]:
```
from sklearn.model_selection import cross_validate
scores = cross_validate(lr , train_input , train_target)
scores
```

executed in 87ms, finished 10:07:34 2023-11-01

Out[241]:
```
{'fit_time': array([0.01384616, 0.01050782, 0.01358747, 0.0114994 , 0.0122723
6]),
 'score_time': array([0.0019865 , 0.00099826, 0.00192261, 0.0009973 , 0.0009970
7]),
 'test_score': array([0.78358209, 0.80597015, 0.76865672, 0.81203008, 0.8421052
6])}
```

In [242]:
```
np.mean(scores['test_score'])
```

executed in 11ms, finished 10:07:39 2023-11-01

Out[242]: 0.802468858713949

- 정확도는 0.8쯤..

In [187]:
```
titanic
```
executed in 27ms, finished 09:59:15 2023-11-01

Out[187]:

	PassengerId	Survived	Pclass	Name	Sex	Age	Fare	Embarked	together
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	7.2500	S	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	71.2833	C	1
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	7.9250	S	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	53.1000	S	1
4	5	0	3	Allen, Mr. William Henry	male	35.0	8.0500	S	0
...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	13.0000	S	0
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	30.0000	S	0
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	23.4500	S	3
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	30.0000	C	0
890	891	0	3	Dooley, Mr. Patrick	male	32.0	7.7500	Q	0

891 rows × 9 columns

- test.csv에 있는 데이터들 예측

In [288]:
```
test = pd.read_csv(path + '/test.csv')
```
executed in 15ms, finished 10:30:12 2023-11-01

In [244]:
```
test.info()
```
executed in 11ms, finished 10:07:44 2023-11-01

```
<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: 36.0+ KB
```

In [194]:
```
test['together'] = test.SibSp + test.Parch
```
executed in 17ms, finished 10:01:32 2023-11-01

In [ ]:
```

```

In [289]:
```
test = test.loc[:,['Pclass','Sex','Age','Fare','Embarked', 'SibSp','Parch']]
```
executed in 9ms, finished 10:30:14 2023-11-01

In [258]:
```
test.info()
```
executed in 11ms, finished 10:08:42 2023-11-01

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Pclass    418 non-null    int64  
 1   Sex       418 non-null    object 
 2   Age       332 non-null    float64
 3   Fare      417 non-null    float64
 4   Embarked  418 non-null    object 
 5   SibSp     418 non-null    int64  
 6   Parch     418 non-null    int64  
dtypes: float64(2), int64(3), object(2)
memory usage: 23.0+ KB
```

In [290]:
```
test.Fare = test.Fare.fillna(0)
```
executed in 12ms, finished 10:30:17 2023-11-01

In [291]:
```
test.Age = test.Age.fillna(30)
```
executed in 16ms, finished 10:30:17 2023-11-01

In [292]:
```
test.Sex[test.Sex=='male'] = 1
test.Sex[test.Sex=='female'] = 0
test.Embarked[test.Embarked == 'S'] = 0
test.Embarked[test.Embarked == 'C'] = 1
test.Embarked[test.Embarked == 'Q'] = 2
```
executed in 15ms, finished 10:30:17 2023-11-01

In [293]:
```
test[['Sex','Embarked']] = test[['Sex','Embarked']].astype('int')
```
executed in 18ms, finished 10:30:27 2023-11-01

In [294]:
```
test_predict = ss.transform(test)
```
executed in 11ms, finished 10:30:28 2023-11-01

In [295]:
```
lr.predict(test_predict)
```
executed in 15ms, finished 10:30:38 2023-11-01

Out[295]:
```
array([0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
       1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1,
       1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
       1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0,
       1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0],
      dtype=int64)
```

In [276]:
```
test['pred'] = lr.predict(test_scaled)
```
executed in 12ms, finished 10:22:22 2023-11-01

In [277]:
```
test1 = test.pred
```
executed in 10ms, finished 10:22:29 2023-11-01

In [278]:
```
test1
```
executed in 8ms, finished 10:22:31 2023-11-01

Out[278]:
```
0      0
1      0
2      0
3      0
4      1
      ..
413    0
414    1
415    0
416    0
417    0
Name: pred, Length: 418, dtype: int64
```

In [279]:
```
test1.to_csv('pred.csv')
```
executed in 19ms, finished 10:22:43 2023-11-01

In [280]:
```
titanic
```
executed in 28ms, finished 10:28:12 2023-11-01

Out[280]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Emba
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	53.1000	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	8.0500	
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	13.0000	
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	30.0000	
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	23.4500	
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	30.0000	
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	7.7500	

891 rows × 10 columns

In [314]:
```
prediction = pd.read_csv('pred.csv')
```
executed in 12ms, finished 10:37:32 2023-11-01

In [296]:
```
test_predict
```
executed in 20ms, finished 10:30:47 2023-11-01

Out[296]:
```
array([[ 0.83012938,  0.72705166,  0.3540876 , ...,  2.57764677,
        -0.46037161, -0.47720996],
       [ 0.83012938, -1.37541808,  1.31936068, ..., -0.5530224 ,
         0.40105202, -0.47720996],
       [-0.36497068,  0.72705166,  2.47768838, ...,  2.57764677,
        -0.46037161, -0.47720996],
       ...,
       [ 0.83012938,  0.72705166,  0.66297499, ..., -0.5530224 ,
        -0.46037161, -0.47720996],
       [ 0.83012938,  0.72705166,  0.00658929, ..., -0.5530224 ,
        -0.46037161, -0.47720996],
       [ 0.83012938,  0.72705166,  0.00658929, ...,  1.01231219,
         0.40105202,  0.73949329]])
```

In [302]:
```
from sklearn.neighbors import KNeighborsClassifier
kn = KNeighborsClassifier()
```
executed in 9ms, finished 10:33:13 2023-11-01

In [299]:
```
kn.fit(train_scaled , train_target)
```
executed in 17ms, finished 10:32:01 2023-11-01

Out[299]:
```
KNeighborsClassifier()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [300]:
```
accuracy_score(test_target , kn.predict(test_scaled))
```
executed in 22ms, finished 10:32:21 2023-11-01

Out[300]:
```
0.7982062780269058
```

In [301]:
```
params = {'n_neighbors' : [3,5,7,9,11,13,15,17,19]}
```
executed in 8ms, finished 10:33:07 2023-11-01

In [303]:
```
#그리드서치 실행
from sklearn.model_selection import GridSearchCV
gs = GridSearchCV(kn , params , n_jobs = -1)
gs.fit(train_scaled , train_target)

gs.best_params_
```
executed in 2.56s, finished 10:33:19 2023-11-01

Out[303]:
```
{'n_neighbors': 9}
```

In [304]:
```
kn = KNeighborsClassifier(n_neighbors = 9)
kn.fit(train_scaled , train_target)
accuracy_score(test_target , kn.predict(test_scaled))
```
executed in 32ms, finished 10:33:41 2023-11-01

Out[304]:
```
0.8071748878923767
```

In [306]:
```
from sklearn.ensemble import VotingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression , LinearRegression , Ridge
```

executed in 109ms, finished 10:35:26 2023-11-01

In [311]:
```
lr = LogisticRegression()
knn = KNeighborsClassifier()

#소프트 보팅

vo = VotingClassifier(estimators = [('LR' , lr),('Knn' , knn)] , voting = 'soft'

#학습

vo.fit(train_scaled , train_target)

#예측

pred = vo.predict(test_scaled)

#정확도

accuracy = accuracy_score(test_target , pred)

print('voting 분류기 정확도 : ' , accuracy)

models = [lr,knn]
for i in models:
    i.fit(train_scaled , train_target)
    pred = i.predict(test_scaled)
    model_name = i.__class__.__name__
    score = accuracy_score(test_target , pred)
    print(f'{model_name}의 정확도 : {score}')
```

executed in 46ms, finished 10:37:02 2023-11-01

```
voting 분류기 정확도 :  0.8161434977578476
LogisticRegression의 정확도 : 0.7982062780269058
KNeighborsClassifier의 정확도 : 0.7982062780269058
```

In [312]:
```
vo.predict(test_predict)
```
executed in 22ms, finished 10:37:06 2023-11-01

Out[312]:
```
array([0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
       1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1,
       1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
       1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0],
      dtype=int64)
```

In [316]:
```
prediction.Survived = vo.predict(test_predict)
```
executed in 19ms, finished 10:37:48 2023-11-01

In [318]:
```
prediction.to_csv('pred.csv' , index = False)
```
executed in 20ms, finished 10:38:36 2023-11-01

In [ ]:
```

```

In [319]:
```
from sklearn.ensemble import RandomForestClassifier
```
executed in 10ms, finished 10:40:37 2023-11-01

In [320]:
```
rf = RandomForestClassifier(oob_score = True , random_state = 42 , n_jobs = -1)
rf.fit(train_input , train_target)
```
executed in 186ms, finished 10:40:38 2023-11-01

Out[320]:
```
RandomForestClassifier(n_jobs=-1, oob_score=True, random_state=42)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [321]:
```
params = {
    'max_depth' : [8,16,24],
    'min_samples_leaf' : [1,6,12],
    'min_samples_split' : [2,8,16]
}
```
executed in 12ms, finished 10:40:40 2023-11-01

In [322]:
```
from sklearn.model_selection import GridSearchCV
gs = GridSearchCV(RandomForestClassifier(random_state = 42 , n_jobs = -1) , para
gs.fit(train_input , train_target)
gs.best_params_
```
executed in 7.60s, finished 10:40:58 2023-11-01

Out[322]:
```
{'max_depth': 8, 'min_samples_leaf': 1, 'min_samples_split': 16}
```

In [324]:
```
rf = RandomForestClassifier(max_depth = 8 , min_samples_split = 16 , random_stat
```
executed in 15ms, finished 10:41:30 2023-11-01

In [327]:
```
rf.fit(train_input , train_target)
```
executed in 134ms, finished 10:41:44 2023-11-01

Out[327]:
```
RandomForestClassifier(max_depth=8, min_samples_split=16, n_jobs=-1,
                       random_state=42)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [329]:
```
rf.predict(test)
```
executed in 35ms, finished 10:42:25 2023-11-01

Out[329]:
```
array([0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
       1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1,
       1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 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, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
       1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0],
      dtype=int64)
```

In [332]:
```
prediction.Survived = rf.predict(test)
```
executed in 36ms, finished 10:43:14 2023-11-01

In [334]:
```
prediction.to_csv('pred.csv' , index = False)
```
executed in 15ms, finished 10:43:29 2023-11-01

In [345]:
```
titanic1
```

executed in 19ms, finished 11:12:15 2023-11-01

Out[345]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7

891 rows × 12 columns

In [343]:
```
data
```
executed in 15ms, finished 11:07:27 2023-11-01

Out[343]:

	Pclass	Sex	Age	Fare	Embarked	SibSp	Parch
0	3	1	22	7.2500	0	1	0
1	1	0	38	71.2833	1	1	0
2	3	0	26	7.9250	0	0	0
3	1	0	35	53.1000	0	1	0
4	3	1	35	8.0500	0	0	0
...	...	...	...	...	...	...	...
886	2	1	27	13.0000	0	0	0
887	1	0	19	30.0000	0	0	0
888	3	0	30	23.4500	0	1	2
889	1	1	26	30.0000	1	0	0
890	3	1	32	7.7500	2	0	0

891 rows × 7 columns

In [346]:
```
titanic['Title'] = titanic.Name.str.extract(' ([A-Za-z]+)W.', expand=False)
```
executed in 17ms, finished 11:19:12 2023-11-01

In [347]:
```
titanic['Title']
```
executed in 7ms, finished 11:19:16 2023-11-01

Out[347]:
```
0        Mr
1       Mrs
2      Miss
3       Mrs
4        Mr
       ...
886     Rev
887    Miss
888    Miss
889      Mr
890      Mr
Name: Title, Length: 891, dtype: object
```

In [348]:
```
titanic.Name
```
executed in 14ms, finished 11:19:28 2023-11-01

Out[348]:
```
0                                Braund, Mr. Owen Harris
1      Cumings, Mrs. John Bradley (Florence Briggs Th...
2                                 Heikkinen, Miss. Laina
3           Futrelle, Mrs. Jacques Heath (Lily May Peel)
4                               Allen, Mr. William Henry
                             ...
886                               Montvila, Rev. Juozas
887                        Graham, Miss. Margaret Edith
888            Johnston, Miss. Catherine Helen "Carrie"
889                               Behr, Mr. Karl Howell
890                                 Dooley, Mr. Patrick
Name: Name, Length: 891, dtype: object
```

In [350]:
```
data
```
executed in 25ms, finished 11:26:55 2023-11-01

Out[350]:

	Pclass	Sex	Age	Fare	Embarked	SibSp	Parch
0	3	1	22	7.2500	0	1	0
1	1	0	38	71.2833	1	1	0
2	3	0	26	7.9250	0	0	0
3	1	0	35	53.1000	0	1	0
4	3	1	35	8.0500	0	0	0
...	...	...	...	...	...	...	...
886	2	1	27	13.0000	0	0	0
887	1	0	19	30.0000	0	0	0
888	3	0	30	23.4500	0	1	2
889	1	1	26	30.0000	1	0	0
890	3	1	32	7.7500	2	0	0

891 rows × 7 columns

In [351]:
```
features = ["Pclass", "Sex", "SibSp", "Parch"]
X = pd.get_dummies(data[features])
```
executed in 21ms, finished 11:26:58 2023-11-01

In [352]:
```
X
```
executed in 26ms, finished 11:27:02 2023-11-01

Out[352]:

	Pclass	Sex	SibSp	Parch
0	3	1	1	0
1	1	0	1	0
2	3	0	0	0
3	1	0	1	0
4	3	1	0	0
...	...	...	...	...
886	2	1	0	0
887	1	0	0	0
888	3	0	1	2
889	1	1	0	0
890	3	1	0	0

891 rows × 4 columns

In [354]:
```
X_test = pd.get_dummies(test[features])
```
executed in 5ms, finished 11:27:57 2023-11-01

In [355]:
```
model = RandomForestClassifier(n_estimators=100, max_depth=3, random_state=2)
model.fit(X, target)
predictions = model.predict(X_test)
```
executed in 119ms, finished 11:28:00 2023-11-01

In [356]:
```
predictions
```
executed in 8ms, finished 11:28:04 2023-11-01

Out[356]:
```
array([0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0,
       1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1,
       1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
       1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0,
       1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0],
      dtype=int64)
```

In [359]:
```
prediction.Survived = predictions
prediction.to_csv('pred.csv' , index = False)
```
executed in 12ms, finished 11:28:47 2023-11-01