딥러닝을 이용한 타이타닉 생존자 예측

만약...

프로야구 올해 우승팀을 예측하고 싶다면

먼저...

우승에 필요한 여러 특징(feature)을 잡고

데이터를 수집하고

관찰한 수

특징을 이용해서 학습하고

올해의 데이터로 예측을 하게 됩니다.

만약...

고객의 구매 성향을 보고

추천 시스템을 만들고 싶다면

고객의 기존 구매 물품 종류, 금액, 주소지,

성별, 연령, 구매가 이루어지는 시간대

등으로 feature를 잡고 학습할 수 있음

그런 의미에서

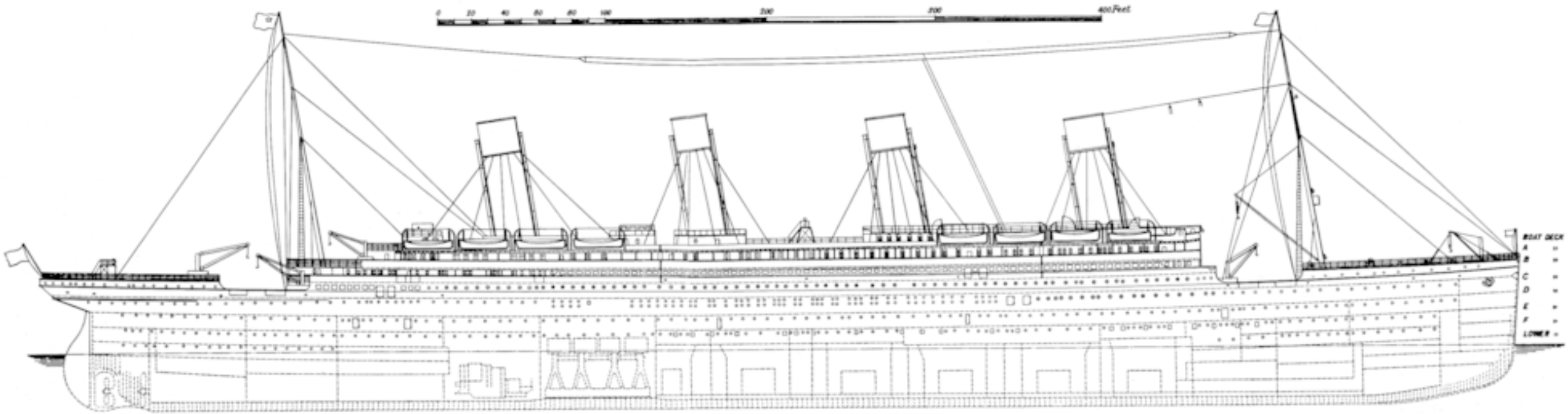
타이타닉 생존자 예측은

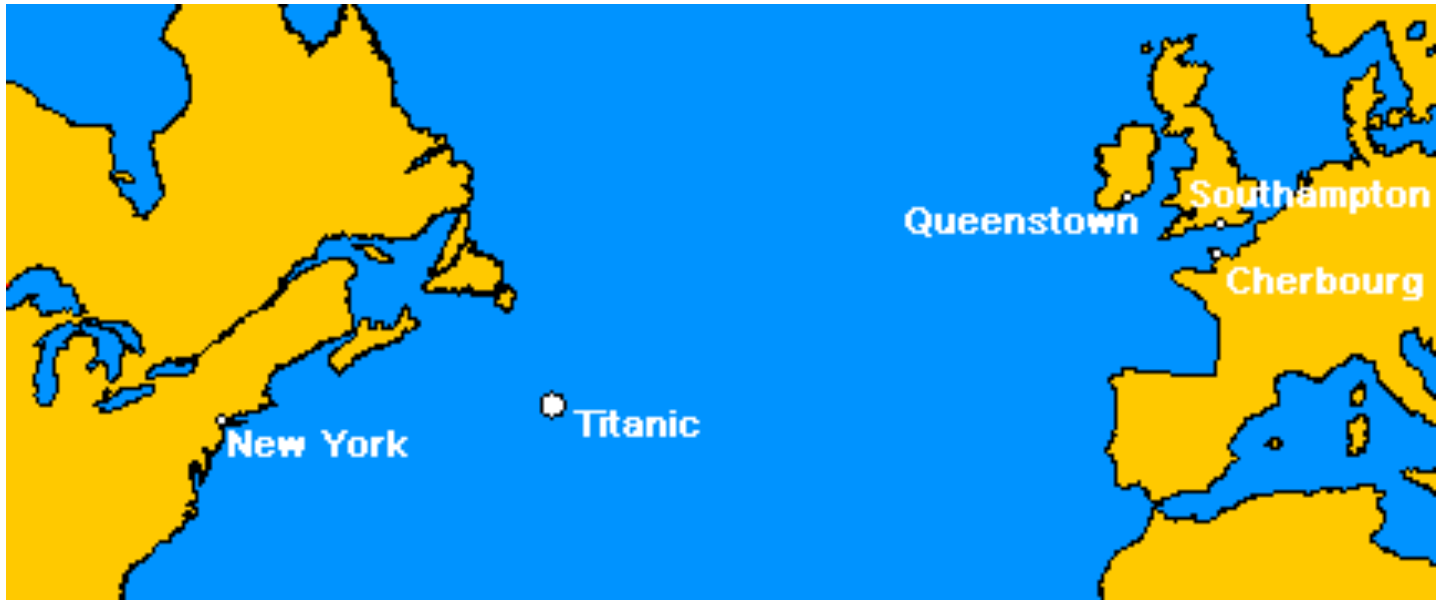
생존한 사람들의 특징을 학습하고

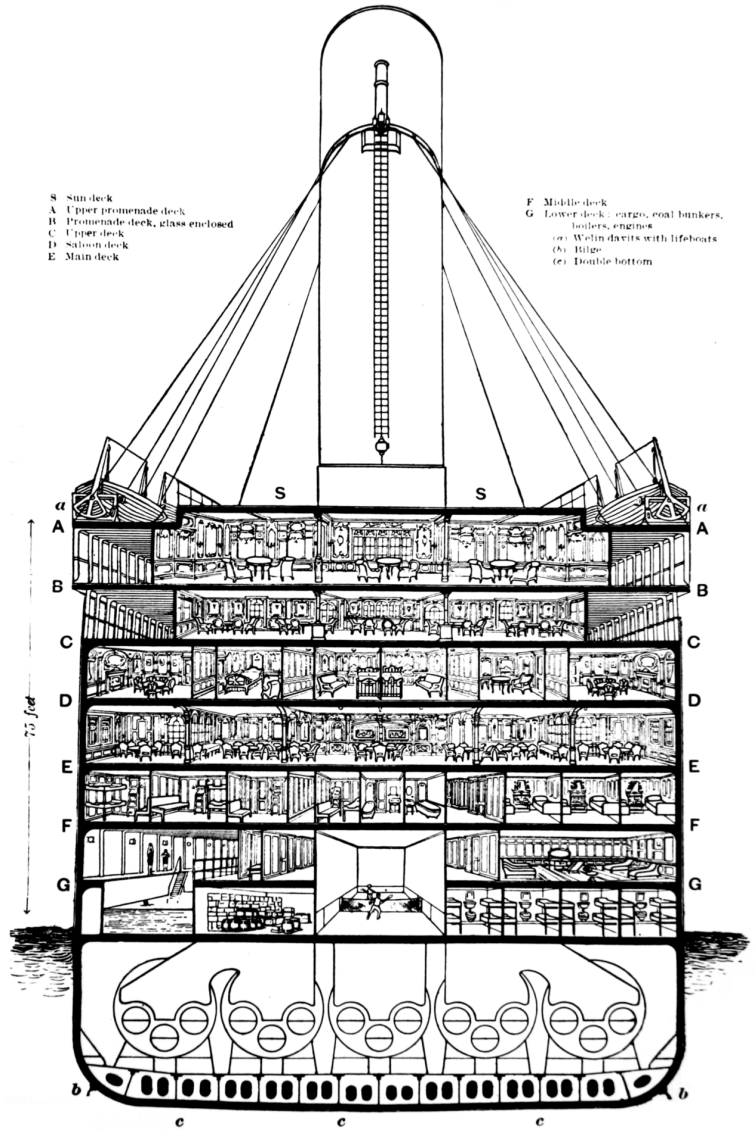
이를 이용해서

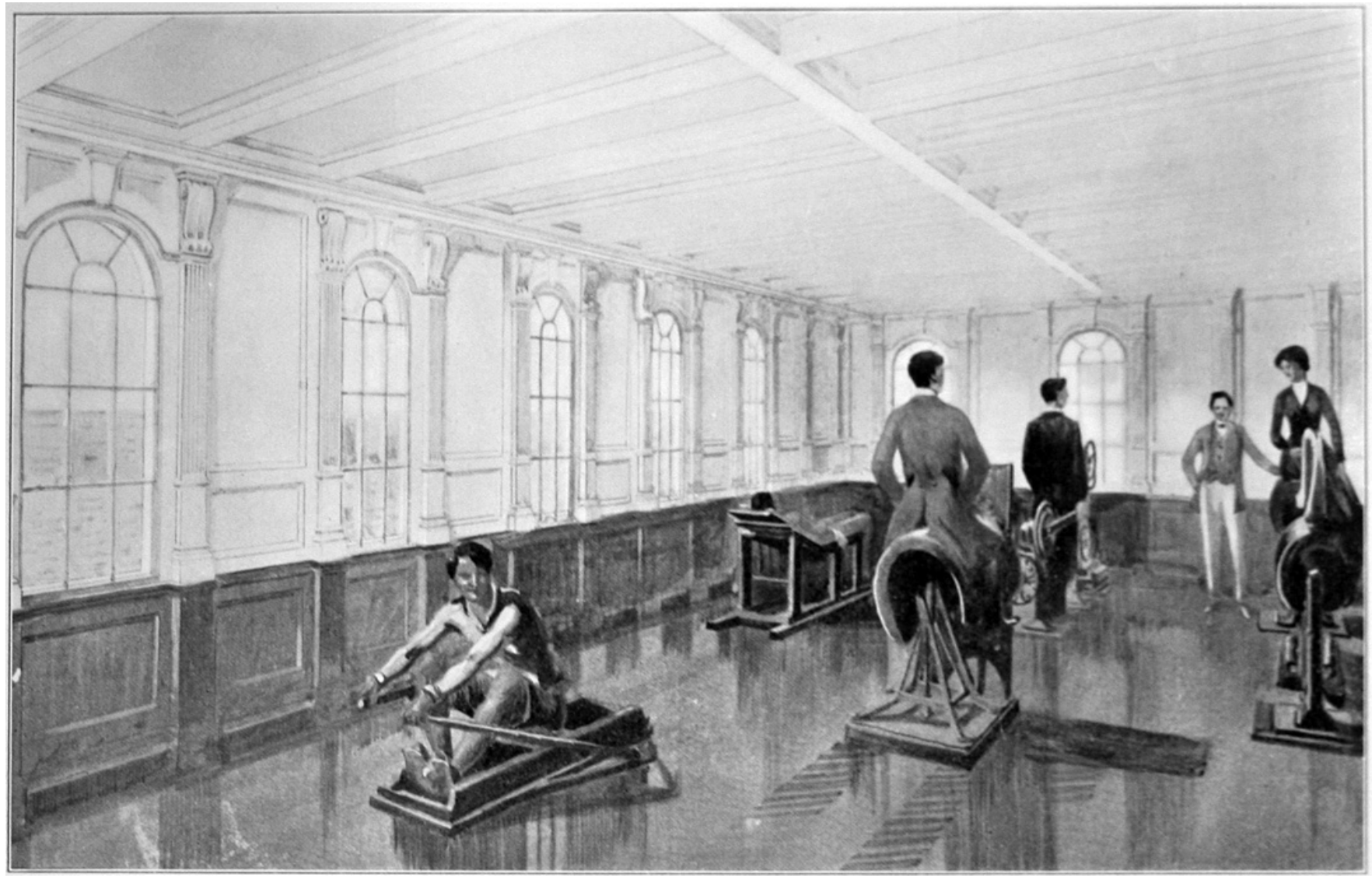
특정한 사람의 데이터를 이용해서

생존 가능성을 확인해 볼 수 있음


0 20 40 60 80 100 200 300 400 Feet
BOAT DECK
A "
B "
C "
D "
E "
F "
LOWER "


Queenstown
Southampton
Cherbourg
Titanic
New York


S Sun deck
A Upper promenade deck
B Promenade deck, glass enclosed
C Upper deck
D Saloon deck
E Main deck
F Middle deck
G Lower deck: cargo, coal bunkers, boilers, engines
(a) Welin davits with lifeboats
(b) Bilge
(c) Double bottom
S
S
a
A
B
C
D
E
F
G
75 feet
b
c



In [1]:

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

import sys
import tensorflow as tf
import keras
import numpy as np
import pandas as pd
```

```
Using TensorFlow backend.
```

In [2]:

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

tf.set_random_seed(777)  # for reproducibility

print('Python version : ', sys.version)
print('TensorFlow version : ', tf.__version__)
print('Keras version : ', keras.__version__)
```

```
Python version :  3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 12:04:33)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)]
TensorFlow version :  1.5.0
Keras version :  2.1.5
```

In [3]:

```
raw_data = pd.read_excel('titanic.xls')
raw_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 14 columns):
pclass       1309 non-null int64
survived     1309 non-null int64
name         1309 non-null object
sex          1309 non-null object
age          1046 non-null float64
sibsp        1309 non-null int64
parch        1309 non-null int64
ticket       1309 non-null object
fare         1308 non-null float64
cabin        295 non-null object
embarked     1307 non-null object
boat         486 non-null object
body         121 non-null float64
home.dest    745 non-null object
dtypes: float64(3), int64(4), object(7)
memory usage: 143.2+ KB
```

In [4]:

```
raw_data.describe()
```

	pclass	survived	age	sibsp	parch	fare	body
count	1309.000000	1309.000000	1046.000000	1309.000000	1309.000000	1308.000000	121.000000
mean	2.294882	0.381971	29.881135	0.498854	0.385027	33.295479	160.809917
std	0.837836	0.486055	14.413500	1.041658	0.865560	51.758668	97.696922
min	1.000000	0.000000	0.166700	0.000000	0.000000	0.000000	1.000000
25%	2.000000	0.000000	21.000000	0.000000	0.000000	7.895800	72.000000
50%	3.000000	0.000000	28.000000	0.000000	0.000000	14.454200	155.000000
75%	3.000000	1.000000	39.000000	1.000000	0.000000	31.275000	256.000000
max	3.000000	1.000000	80.000000	8.000000	9.000000	512.329200	328.000000

In [5]:

```
raw_data.head()
```

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	2	NaN	St Louis, MO
1	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
2	1	0	Allison, Miss. Helen Loraine	female	2.0000	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON
3	1	0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1	2	113781	151.5500	C22 C26	S	NaN	135.0	Montreal, PQ / Chesterville, ON
4	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON

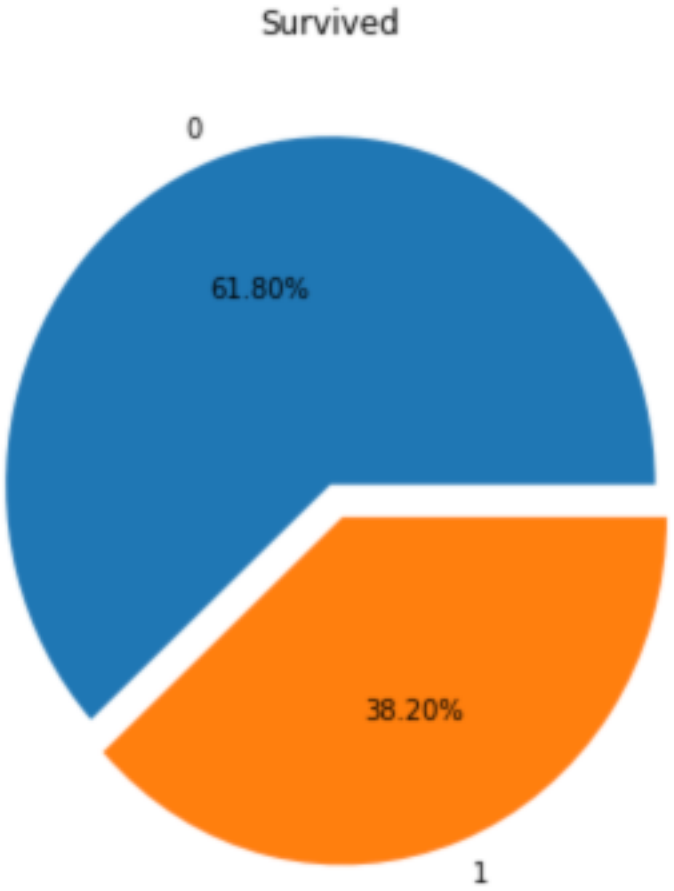
전체적인 생존률과 생존자의 수는?

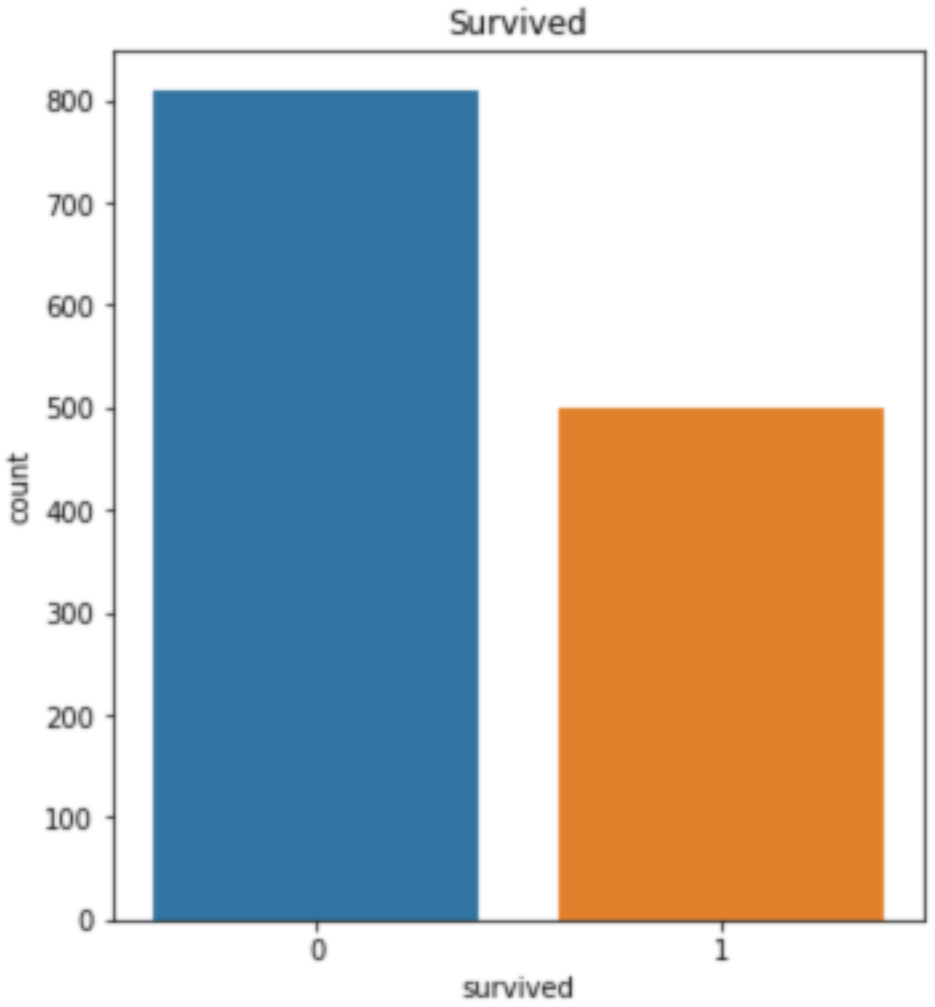
In [6]:

```
f,ax=plt.subplots(1,2,figsize=(12,6))

raw_data['survived'].value_counts().plot.pie(explode=[0,0.1],
                                             autopct='%1.2f%%',ax=ax[0])
ax[0].set_title('Survived')
ax[0].set_ylabel('')

sns.countplot('survived', data=raw_data,ax=ax[1])
ax[1].set_title('Survived')
plt.show()
```

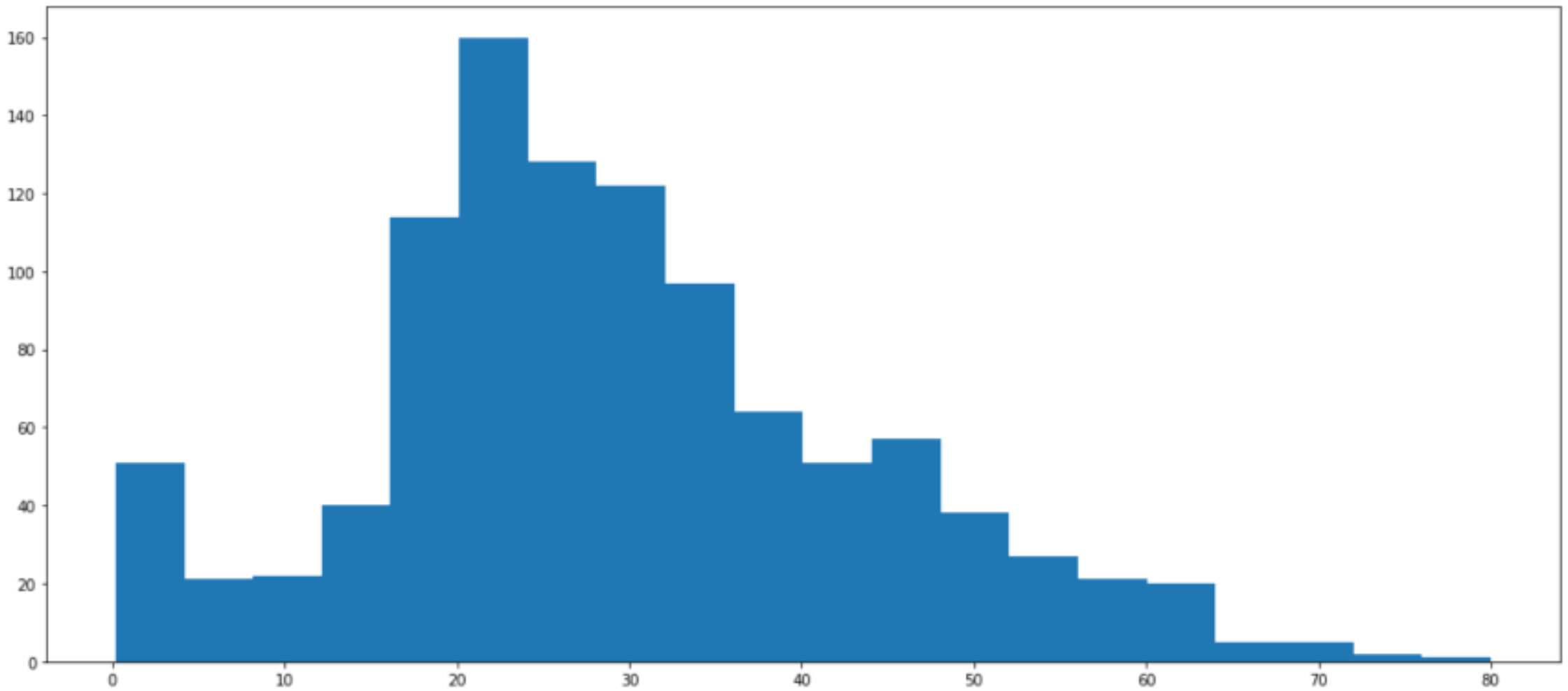

Survived
0
61.80%
38.20%
1


Survived
800
700
600
500
400
300
200
100
0
count
0
1
survived

탑승자의 연령대는?

In [7]:

```
raw_data['age'].hist(bins=20,figsize=(18,8),grid=False);
```



선실 등급별 현황은?

In [8]:

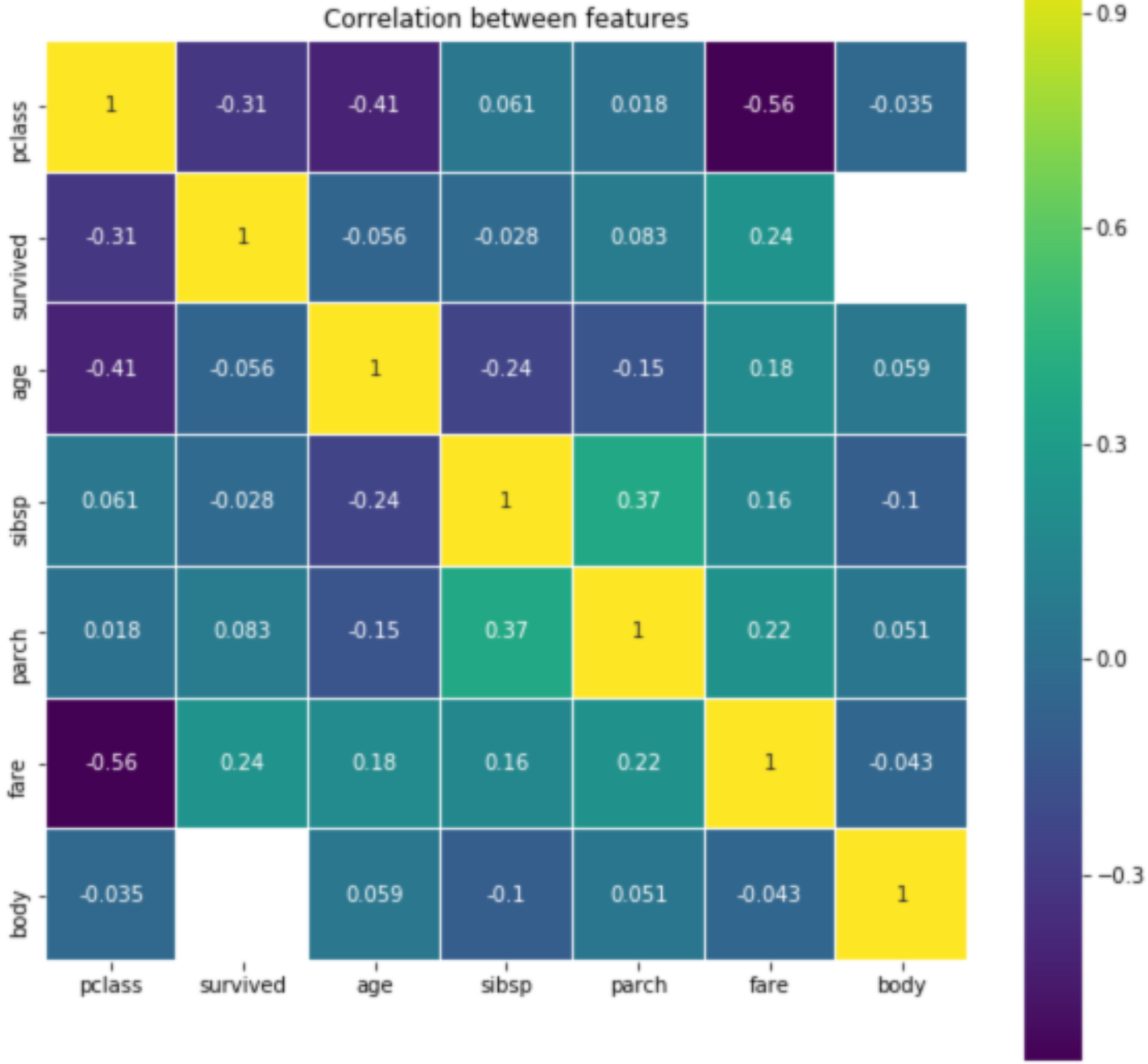
```
raw_data.groupby('pclass').mean()
```

pclass	survived	age	sibsp	parch	fare	body
1	0.619195	39.159918	0.436533	0.365325	87.508992	162.828571
2	0.429603	29.506705	0.393502	0.368231	21.179196	167.387097
3	0.255289	24.816367	0.568406	0.400564	13.302889	155.818182

생존 여부와 다른 정보와의 상관관계는?

In [10]:

```
plt.figure(figsize=(10, 10))
sns.heatmap(raw_data.corr(), linewidths=0.01, square=True,
            annot=True, cmap=plt.cm.viridis, linecolor="white")
plt.title('Correlation between features')
plt.show()
```


Correlation between features
pclass
survived
age
sibsp
parch
fare
body
1 -0.31 -0.41 0.061 0.018 -0.56 -0.035
-0.31 1 -0.056 -0.028 0.083 0.24
-0.41 -0.056 1 -0.24 -0.15 0.18 0.059
0.061 -0.028 -0.24 1 0.37 0.16 -0.1
0.018 0.083 -0.15 0.37 1 0.22 0.051
-0.56 0.24 0.18 0.16 0.22 1 -0.043
-0.035 0.059 -0.1 0.051 -0.043 1
0.9
0.6
0.3
0.0
−0.3

연령, 성별, 선실 등급별 생존율은?

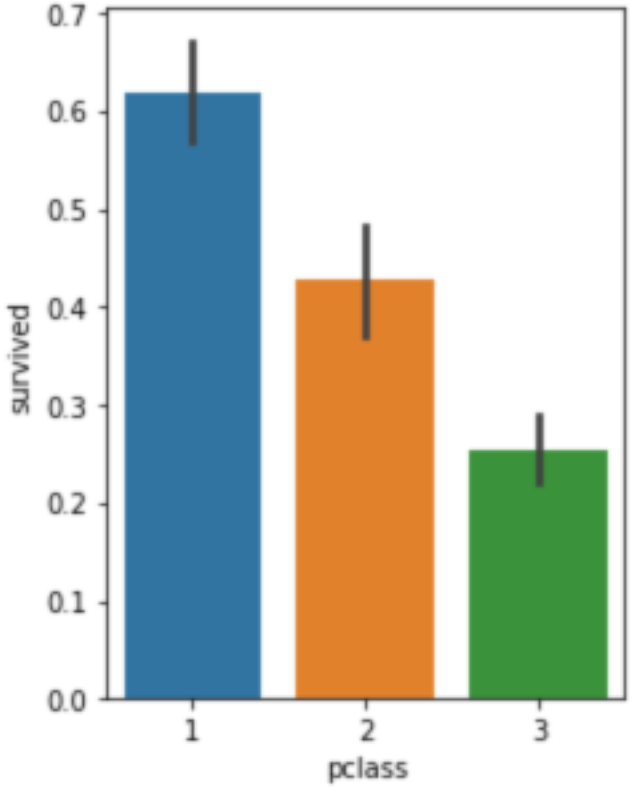
In [11]:

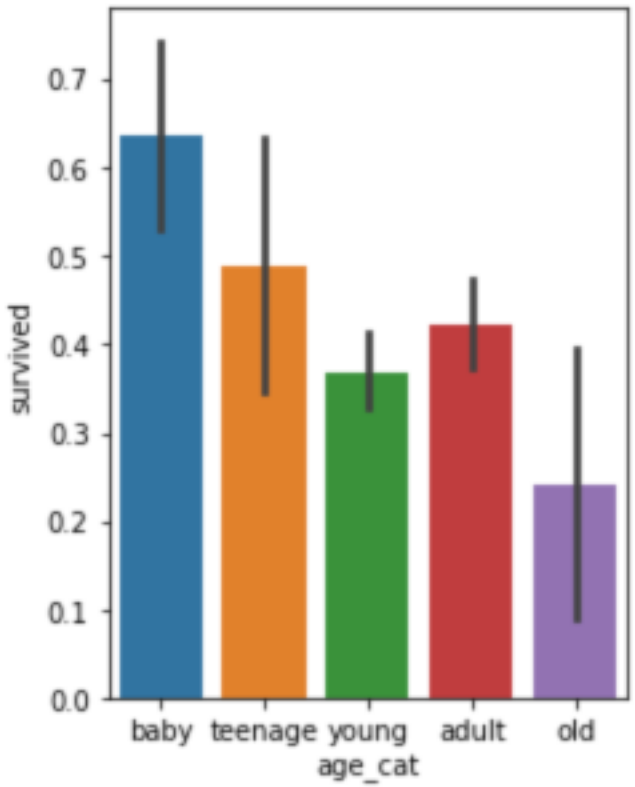
```
raw_data['age_cat'] = pd.cut(raw_data['age'], bins=[0, 7, 15, 30, 60, 100],
                             include_lowest=True,
                             labels=['baby', 'teenage', 'young', 'adult', 'old'])
raw_data.head()
```

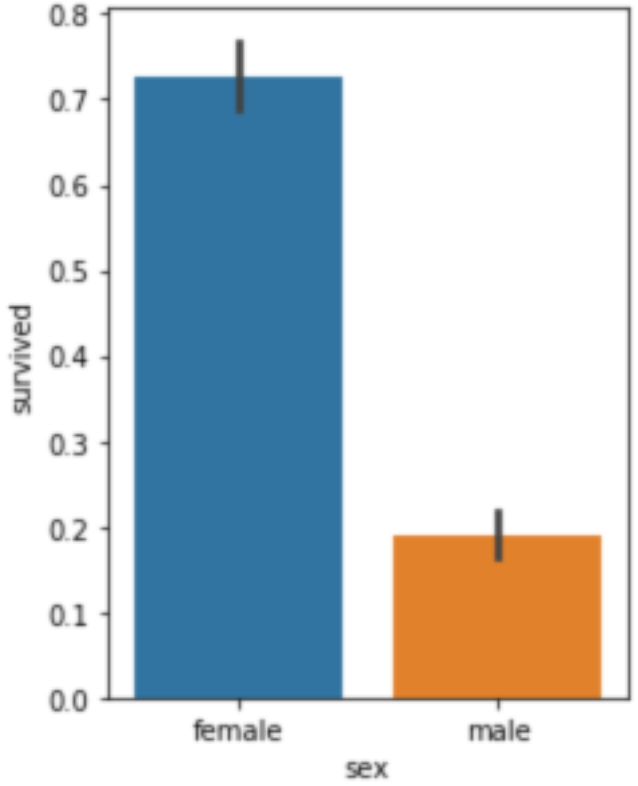
	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest	age_cat
0	1	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	2	NaN	St Louis, MO	young
1	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON	baby
2	1	0	Allison, Miss. Helen Loraine	female	2.0000	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON	baby
3	1	0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1	2	113781	151.5500	C22 C26	S	NaN	135.0	Montreal, PQ / Chesterville, ON	young
4	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON	young

In [12]:

```
plt.figure(figsize=[12,4])
plt.subplot(131)
sns.barplot('pclass', 'survived', data=raw_data)
plt.subplot(132)
sns.barplot('age_cat', 'survived', data=raw_data)
plt.subplot(133)
sns.barplot('sex', 'survived', data=raw_data)
plt.subplots_adjust(top=1, bottom=0.1, left=0.10, right=1, hspace=0.5, wspace=0.5)
plt.show()
```


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
survived
1
2
3
pclass

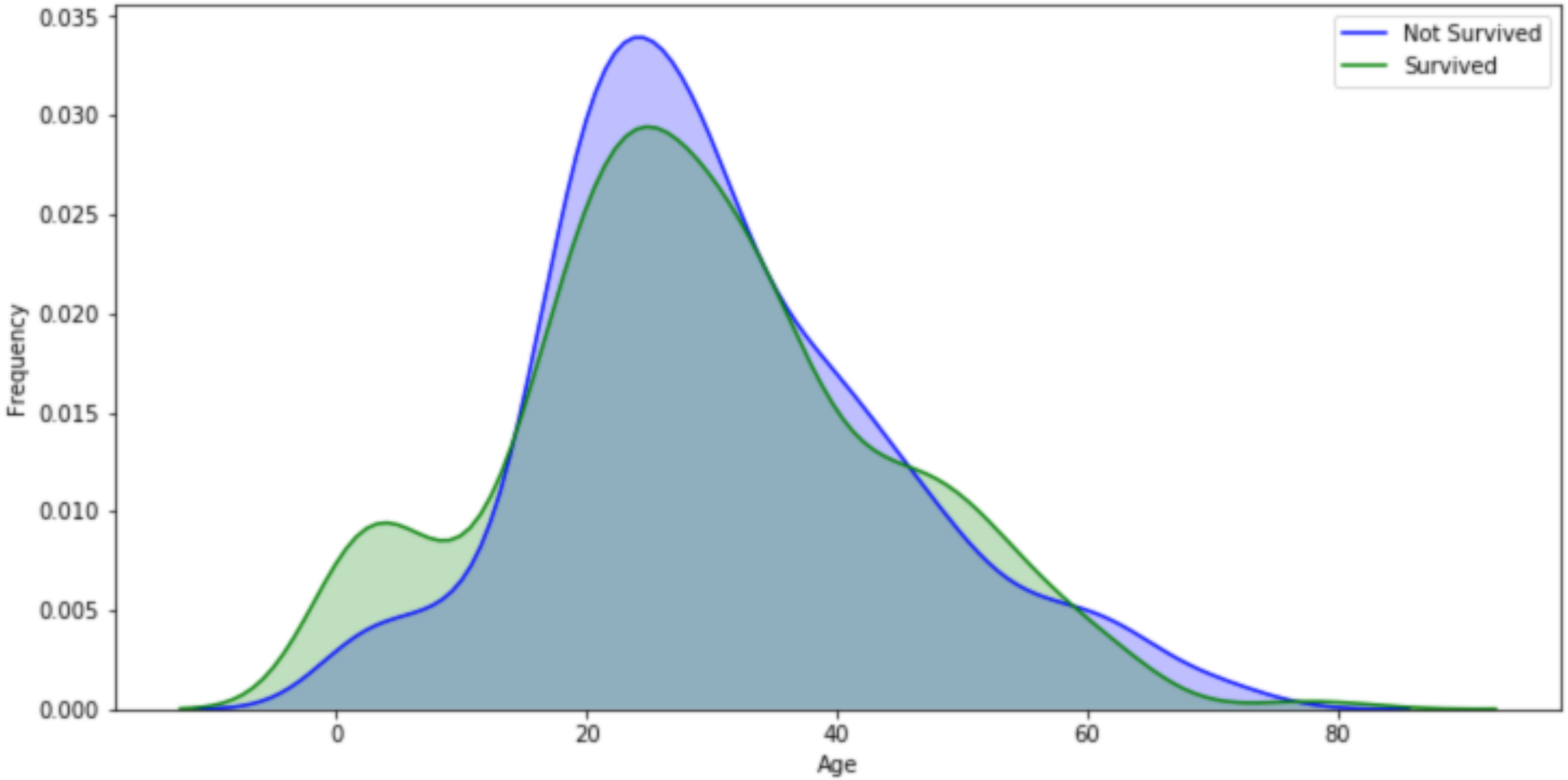

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
survived
baby
teenage
young
adult
old
age_cat


0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
survived
female
male
sex

연별별 생존여부의 분포곡선은?

In [13]:

```
f,ax = plt.subplots(figsize=(12,6))
g = sns.kdeplot(raw_data["age"][(raw_data["survived"] == 0) & \
                                (raw_data["age"].notnull())],
                ax = ax, color="Blue", shade = True)
g = sns.kdeplot(raw_data["age"][(raw_data["survived"] == 1) & \
                                (raw_data["age"].notnull())],
                ax =g, color="Green", shade= True)
g.set_xlabel("Age")
g.set_ylabel("Frequency")
g = g.legend(["Not Survived","Survived"])
```

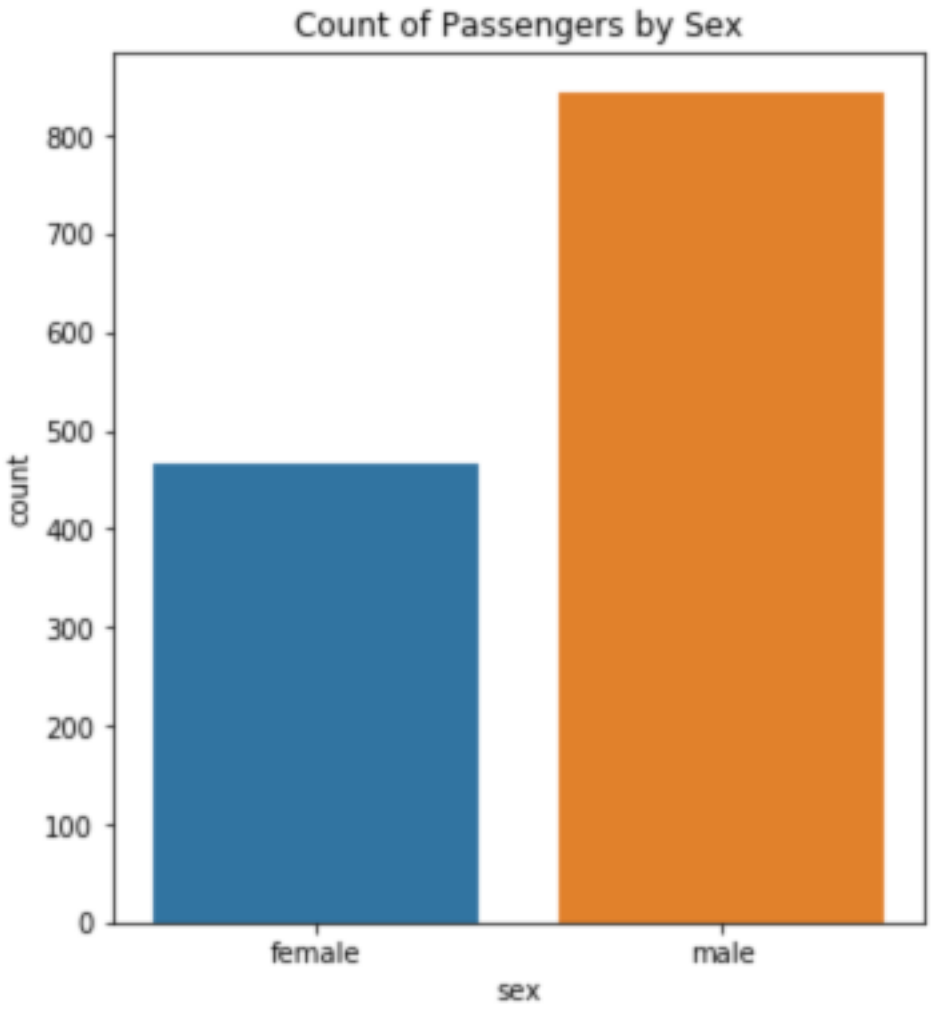

Not Survived
Survived
Frequency
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
Age

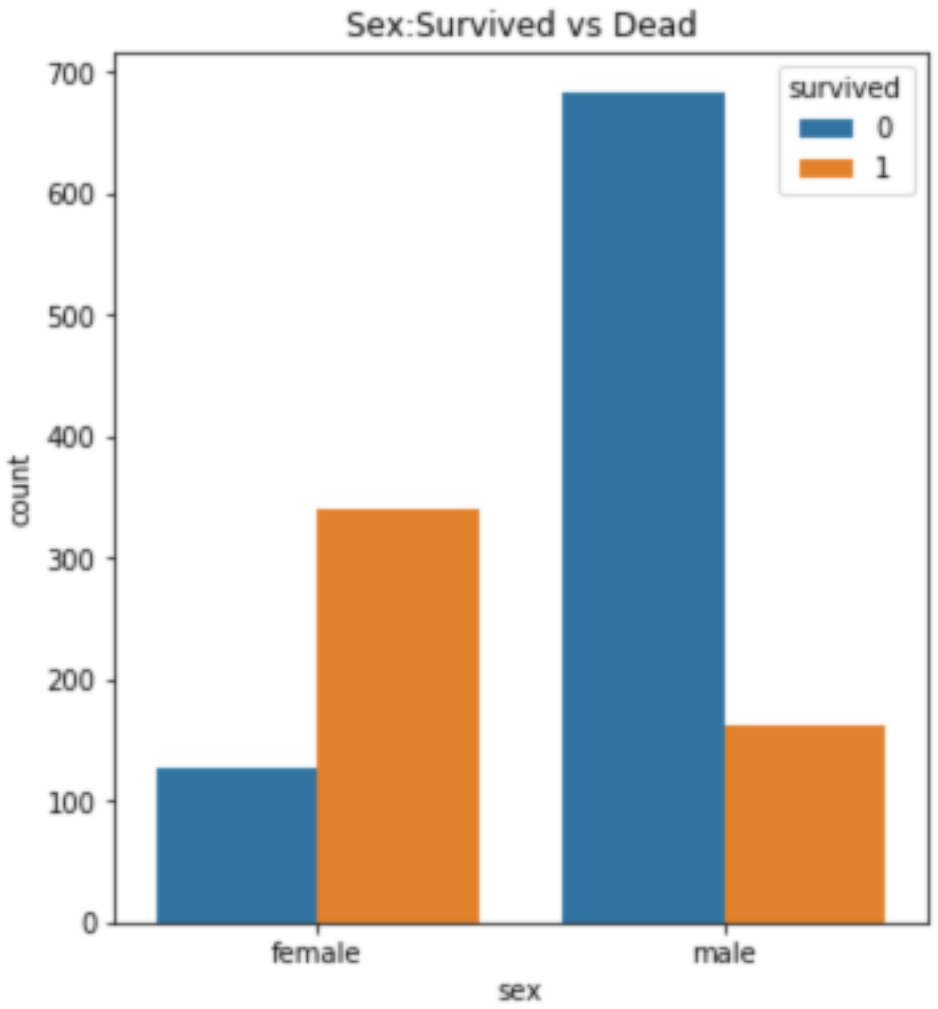
성별 생존율과 생존자 수는?

In [14]:

```
f,ax=plt.subplots(1,2,figsize=(12,6))
sns.countplot('sex',data=raw_data, ax=ax[0])
ax[0].set_title('Count of Passengers by Sex')

sns.countplot('sex',hue='survived',data=raw_data, ax=ax[1])
ax[1].set_title('Sex:Survived vs Dead')
plt.show()
```


Count of Passengers by Sex
count
sex
female
male
0
100
200
300
400
500
600
700
800


Sex:Survived vs Dead
survived
0
1
count
sex
female
male
0
100
200
300
400
500
600
700

보트에 탑승한 사람의 생존율은?

In [15]:

```
boat_survivors = raw_data[raw_data['boat'].notnull()]
boat_survivors.head()
```

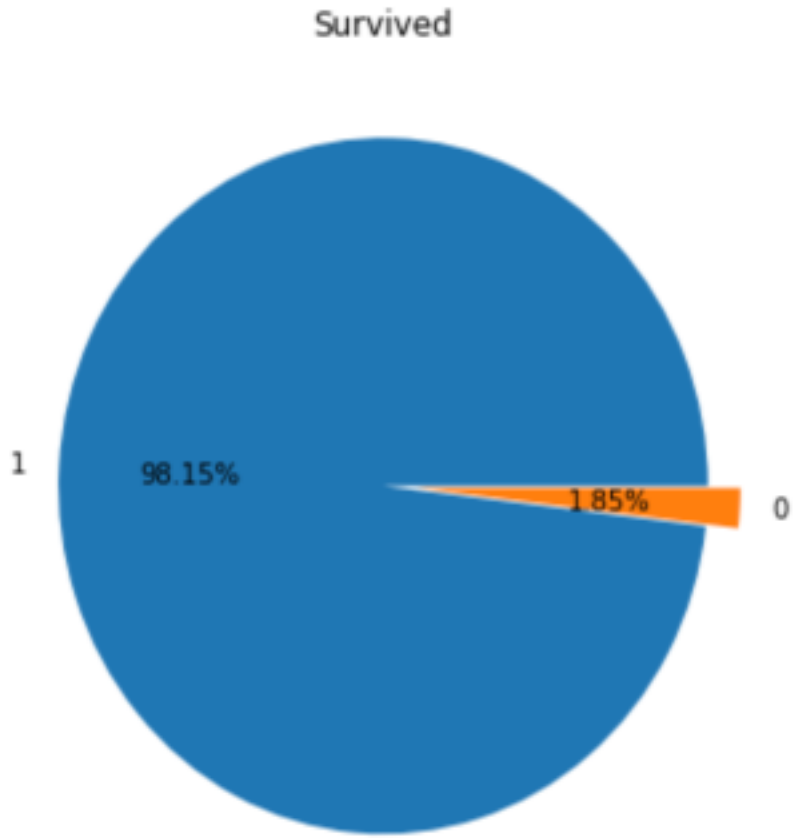
	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest	age_cat
0	1	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	2	NaN	St Louis, MO	young
1	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON	baby
5	1	1	Anderson, Mr. Harry	male	48.0000	0	0	19952	26.5500	E12	S	3	NaN	New York, NY	adult
6	1	1	Andrews, Miss. Kornelia Theodosia	female	63.0000	1	0	13502	77.9583	D7	S	10	NaN	Hudson, NY	old
8	1	1	Appleton, Mrs. Edward Dale (Charlotte Lamson)	female	53.0000	2	0	11769	51.4792	C101	S	D	NaN	Bayside, Queens, NY	adult

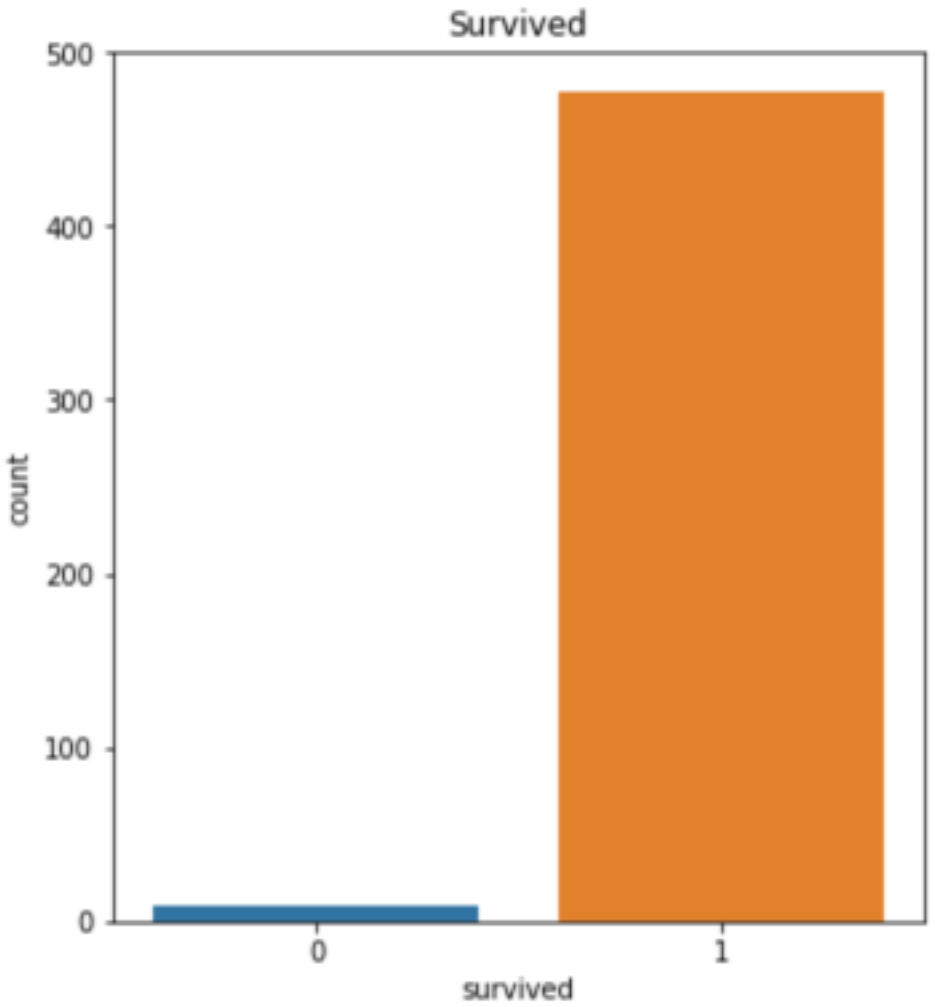
In [16]:

```
f,ax=plt.subplots(1,2,figsize=(12,6))

boat_survivors['survived'].value_counts().plot.pie(explode=[0,0.1],
                                                   autopct='%1.2f%%',ax=ax[0])
ax[0].set_title('Survived')
ax[0].set_ylabel('')

sns.countplot('survived',data=boat_survivors,ax=ax[1])
ax[1].set_title('Survived')
plt.show()
```


Survived
1
98.15%
1.85%
0


Survived
500
400
300
200
100
0
count
0
1
survived

사회적 신분과 성별로 본 생존 상황은?

In [17]:

```
raw_data['title'] = raw_data['name'].map(lambda x:
                                         x.split(',')[1].split('.')[0].strip())
titles = raw_data['title'].unique()
titles
```

```
array(['Miss', 'Master', 'Mr', 'Mrs', 'Col', 'Mme', 'Dr', 'Major', 'Capt',
       'Lady', 'Sir', 'Mlle', 'Dona', 'Jonkheer', 'the Countess', 'Don',
       'Rev', 'Ms'], dtype=object)
```

In [18]:

```
raw_data.head()
```

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest	age_cat	titl
0	1	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	2	NaN	St Louis, MO	young	Miss
1	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON	baby	Mast
2	1	0	Allison, Miss. Helen Loraine	female	2.0000	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON	baby	Miss
3	1	0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1	2	113781	151.5500	C22 C26	S	NaN	135.0	Montreal, PQ / Chesterville, ON	young	Mr

In [19]:

```
pd.crosstab(raw_data['title'], raw_data['sex'])
```

sex title	female	male
Capt	0	1
Col	0	4
Don	0	1
Dona	1	0
Dr	1	7
Jonkheer	0	1
Lady	1	0
Major	0	2
Master	0	61
Miss	260	0
Mlle	2	0
Mme	1	0
Mr	0	757
Mrs	197	0
Ms	2	0
Rev	0	8
Sir	0	1
the Countess	1	0

In [20]:

```
raw_data['title'] = raw_data['title'].replace('Mlle', 'Miss')
raw_data['title'] = raw_data['title'].replace('Ms', 'Miss')
raw_data['title'] = raw_data['title'].replace('Mme', 'Mrs')

Rare = ['Lady','the Countess','Countess','Capt', 'Master',
        'Col','Don','Dr','Major','Rev','Sir','Jonkheer', 'Dona']

for each in Rare:
    raw_data['title'] = raw_data['title'].replace(each, 'Rare')

raw_data['title'].unique()
```

```
array(['Miss', 'Rare', 'Mr', 'Mrs'], dtype=object)
```

In [21]:

```
print (raw_data[['title', 'survived']].groupby(['title'], as_index=False).mean())
```

```
   title  survived
0   Miss  0.678030
1     Mr  0.162483
2    Mrs  0.787879
3   Rare  0.466667
```

데이터 중에서 수치적 데이터를 정리하고

생존여부를 학습할

Feature를 정리함

In [22]:

```
raw_data.head()
```

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest	age_cat	title
0	1	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	2	NaN	St Louis, MO	young	Miss
1	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON	baby	Rare
2	1	0	Allison, Miss. Helen Loraine	female	2.0000	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON	baby	Miss

In [23]:

```
tmp = []
for each in raw_data['sex']:
    if each == 'female':
        tmp.append(1)
    elif each == 'male':
        tmp.append(0)
    else:
        tmp.append(np.nan)
```

In [24]:

```
raw_data['sex'] = tmp
raw_data.head()
```

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest	age_cat	title
0	1	1	Allen, Miss. Elisabeth Walton	1	29.0000	0	0	24160	211.3375	B5	S	2	NaN	St Louis, MO	young	Miss

In [25]:

```
raw_data['survived'] = raw_data['survived'].astype('float')
raw_data['pclass'] = raw_data['pclass'].astype('float')
raw_data['sex'] = raw_data['sex'].astype('float')
raw_data['sibsp'] = raw_data['sibsp'].astype('float')
raw_data['parch'] = raw_data['parch'].astype('float')
raw_data['fare'] = raw_data['fare'].astype('float')
raw_data.head()
```

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest	age_cat	title
0	1.0	1.0	Allen, Miss. Elisabeth Walton	1.0	29.0000	0.0	0.0	24160	211.3375	B5	S	2	NaN	St Louis, MO	young	Miss
1	1.0	1.0	Allison, Master. Hudson Trevor	0.0	0.9167	1.0	2.0	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON	baby	Rare

In [26]:

```
raw_data = raw_data[raw_data['age'].notnull()]
raw_data = raw_data[raw_data['sibsp'].notnull()]
raw_data = raw_data[raw_data['parch'].notnull()]
raw_data = raw_data[raw_data['fare'].notnull()]
raw_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1045 entries, 0 to 1308
Data columns (total 16 columns):
pclass       1045 non-null float64
survived     1045 non-null float64
```

In [27]:
```
x_data = raw_data.values[:, [0,3,4,5,6,8]]
y_data = raw_data.values[:, [1]]
```

In [28]:
```
x_data.shape, y_data.shape
```

```
((1045, 6), (1045, 1))
```

정리된 데이터를 훈련용 데이터와 시험용으로 나눔

In [29]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_data, y_data,
                                                    test_size=0.1, random_state=7)
```

In [30]:
```
len(X_train), len(X_test), len(y_train), len(y_test)
```

```
(940, 105, 940, 105)
```

In [31]:

```
import tensorflow as tf
import keras
from keras.models import Sequential
from keras.layers.core import Dense
np.random.seed(7)

print('tensorflow version : ', tf.__version__)
print('keras version : ', keras.__version__)
```

```
tensorflow version :  1.5.0
keras version :  2.1.5
```

학습 모델 구축

In [32]:

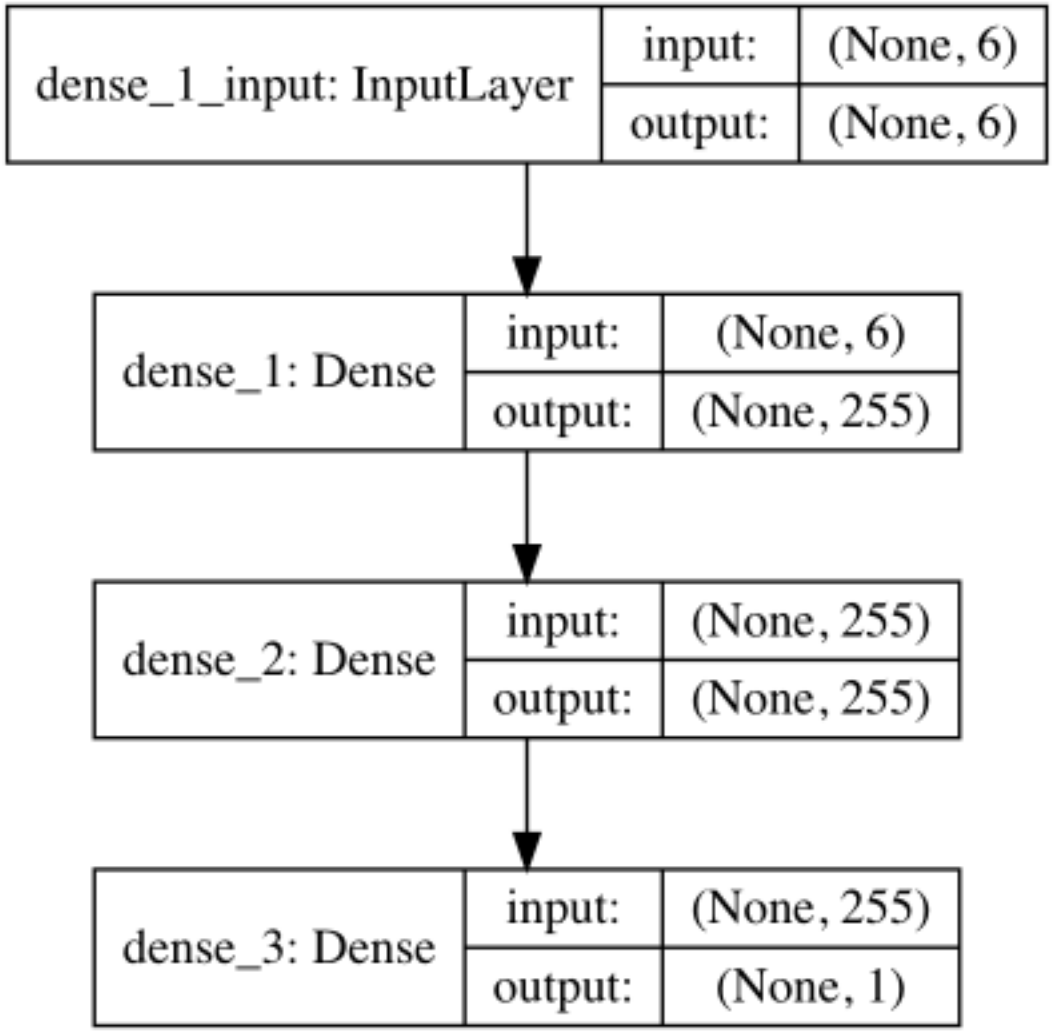
```
model = Sequential()
model.add(Dense(255, input_shape=(6,), activation='relu'))
model.add(Dense(255, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='mse', optimizer='Adam', metrics=['accuracy'])
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 255)               1785
_________________________________________________________________
dense_2 (Dense)              (None, 255)               65280
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 256
=================================================================
Total params: 67,321
Trainable params: 67,321
Non-trainable params: 0
_________________________________________________________________
```

In [33]:

```
from IPython.display import SVG
from keras.utils.vis_utils import model_to_dot

SVG(model_to_dot(model, show_shapes=True).create(prog='dot', format='svg'))
```


dense_1_input: InputLayer
input: (None, 6)
output: (None, 6)
dense_1: Dense
input: (None, 6)
output: (None, 255)
dense_2: Dense
input: (None, 255)
output: (None, 255)
dense_3: Dense
input: (None, 255)
output: (None, 1)

In [34]:
```
hist = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=500)
```

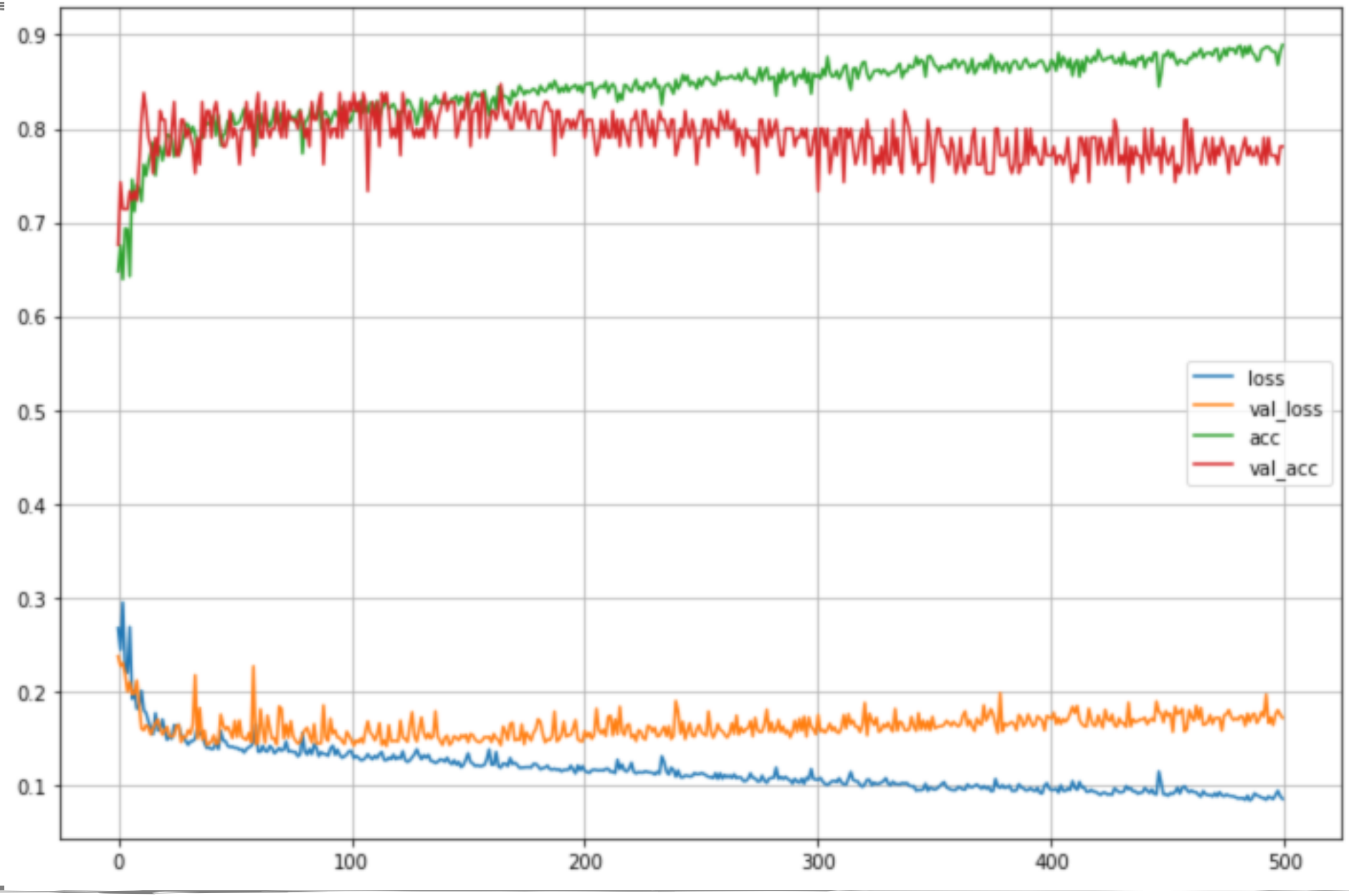
```
Epoch 23/500
940/940 [==============================] - 0s 58us/step - loss: 0.1527 - acc: 0.7926 -
val_loss: 0.1523 - val_acc: 0.7714
Epoch 24/500
940/940 [==============================] - 0s 59us/step - loss: 0.1496 - acc: 0.7894 -
val_loss: 0.1521 - val_acc: 0.8000
Epoch 25/500
940/940 [==============================] - 0s 59us/step - loss: 0.1644 - acc: 0.7702 -
val_loss: 0.1531 - val_acc: 0.8286
Epoch 26/500
940/940 [==============================] - 0s 57us/step - loss: 0.1634 - acc: 0.7777 -
val_loss: 0.1632 - val_acc: 0.7714
```

In [35]:
```
hist.history.keys()
```

```
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```

In [42]:

```
plt.figure(figsize=(12,8))
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.plot(hist.history['acc'])
plt.plot(hist.history['val_acc'])
plt.legend(['loss','val_loss', 'acc','val_acc'])
plt.grid()
plt.show()
```


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
100
200
300
400
500
loss
val_loss
acc
val_acc

예측하고 싶은 데이터 정의

- 영화에서 디카프리오와 윈슬렛의 데이터를 표현

In [37]:
```
dicaprio = np.array([3., 0., 19., 0., 0., 5.]).reshape(1,6)
winslet = np.array([1., 1., 17., 1., 2., 100.]).reshape(1,6)
```

```
In [38]: model.predict(dicaprio)
```

```
array([[0.1133607]], dtype=float32)
```

```
In [39]: model.predict(winslet)
```

```
array([[0.9999969]], dtype=float32)
```

- softmax의 결과를 확률로 볼 수는 없지만,
- 가능성 정도로 볼 수 있다면
- 디카프리오의 생존 가능성은 정말 낮음