

인공지능 아카데미

01강. 머신러닝 이해와 기초 파이썬

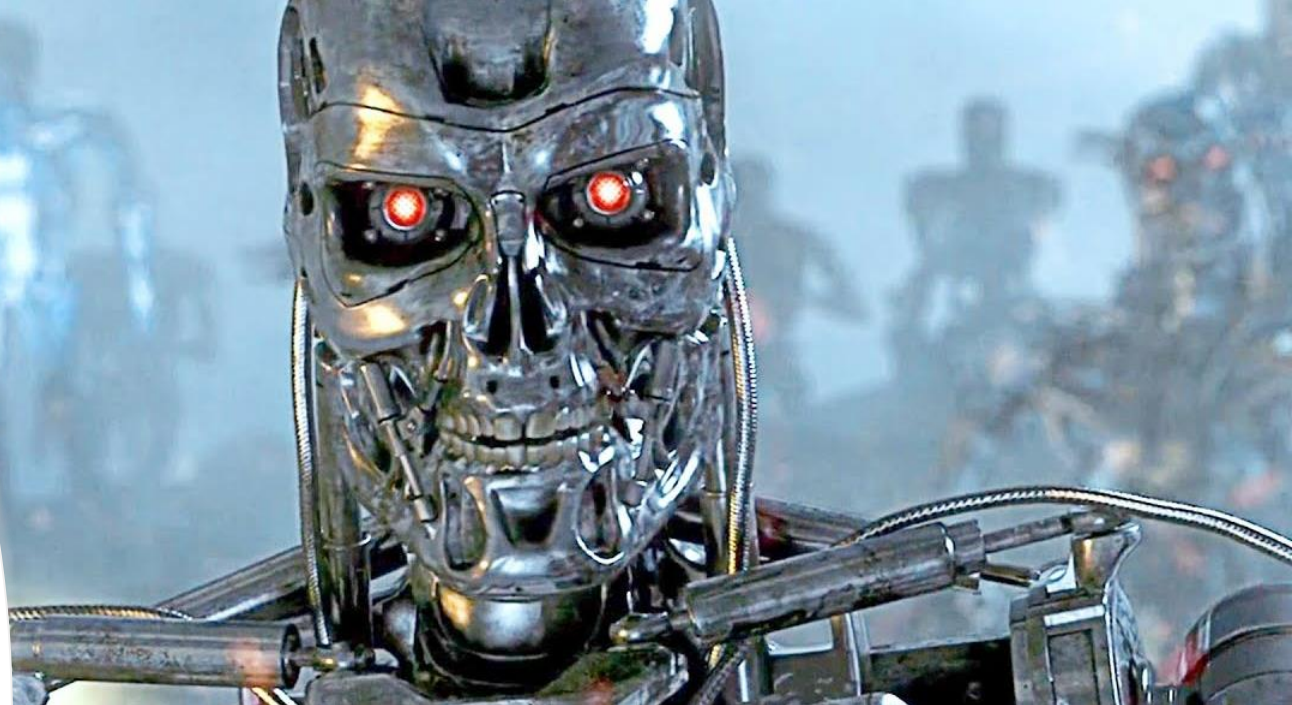


목차

- 인공지능? 머신러닝? 딥러닝?
- 파이썬으로 머신러닝 시작하기
- 타이타닉 데이터를 활용한 실습

인공지능

- 인간의 지능을 기계 등에 인공적으로 구현
- 사람이 해결하지 못하던 문제를 해결
- 사람이 하던 일을 보다 빠르고 정확하게 처리
- 생활이 편리해짐 – ex) 시리, 자율주행 자동차, 추천 시스템
- 인공지능을 어떻게 만들까? => 머신러닝



내가 내일 시험인데 이려고 있는데
충고좀


니가 지금 공부해야 나처럼
병아리 흥내 내는
알바안한다.



머신러닝 (Machine Learning)

사람이 공부를 잘하기 위해 필요한 것

- 사람
- 학습 방법
- 학습 자료
- 학습 시간
- 선생님



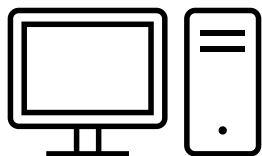
머신러닝 (Machine Learning)

기계가 공부를 잘하기 위해 필요한 것

- 모델(알고리즘)
- 학습 기법
- 데이터
- 학습 시간
- 사람

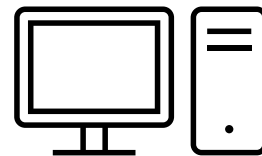
기계가 공부하는 과정

문제	
X	Y
5	25
8	64
3	9
2	4
9	81



○ ⇒ 확인

문제	
X	Y
1	?
4	?
6	?
7	?
10	?



문제	
X	Y
1	1
4	16
6	36
7	49
10	100

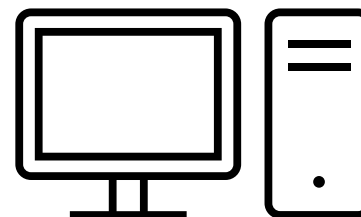
파이썬으로 컴퓨터와 대화하기



1+1=?
2



01100110...
00000010

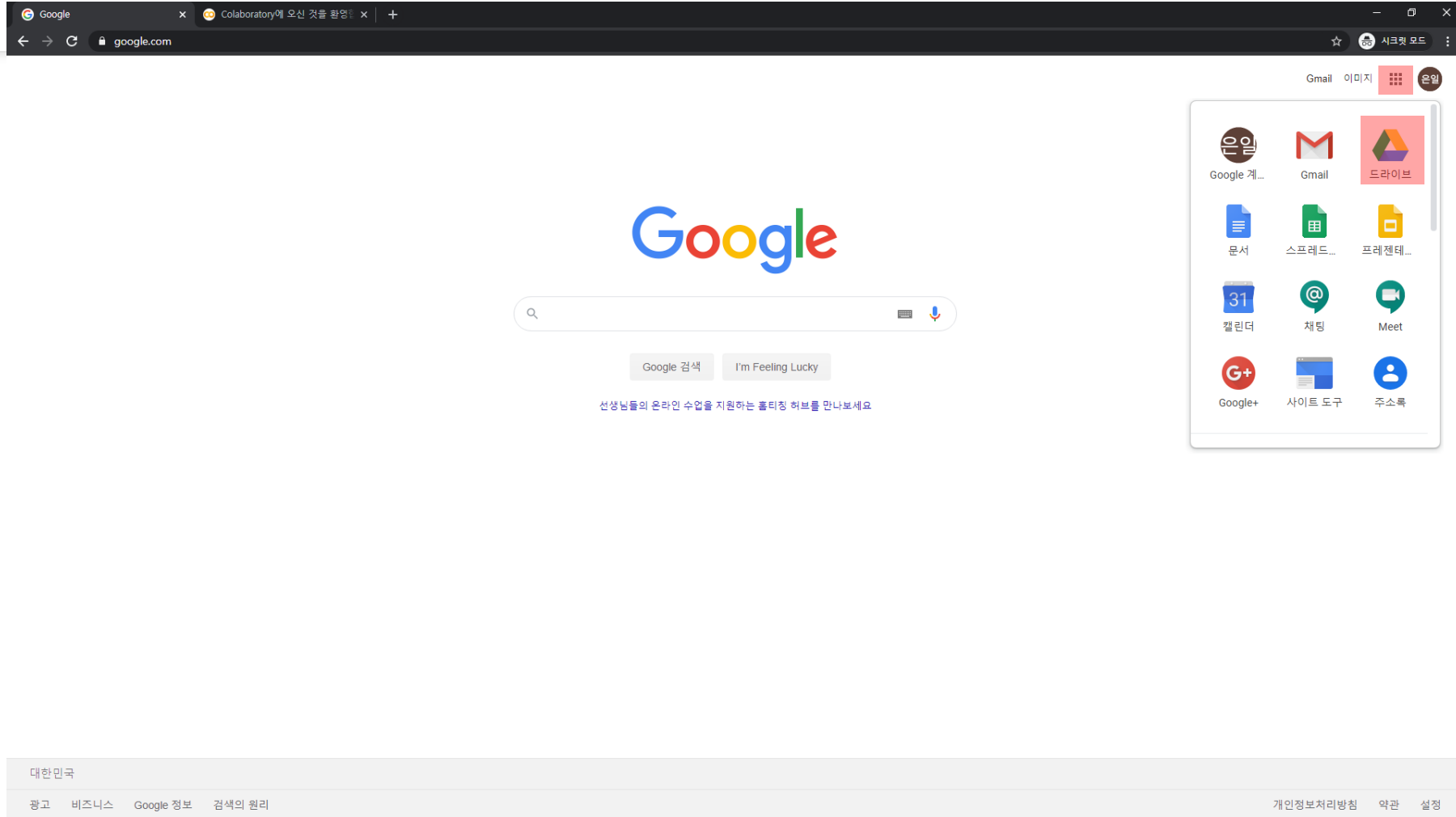




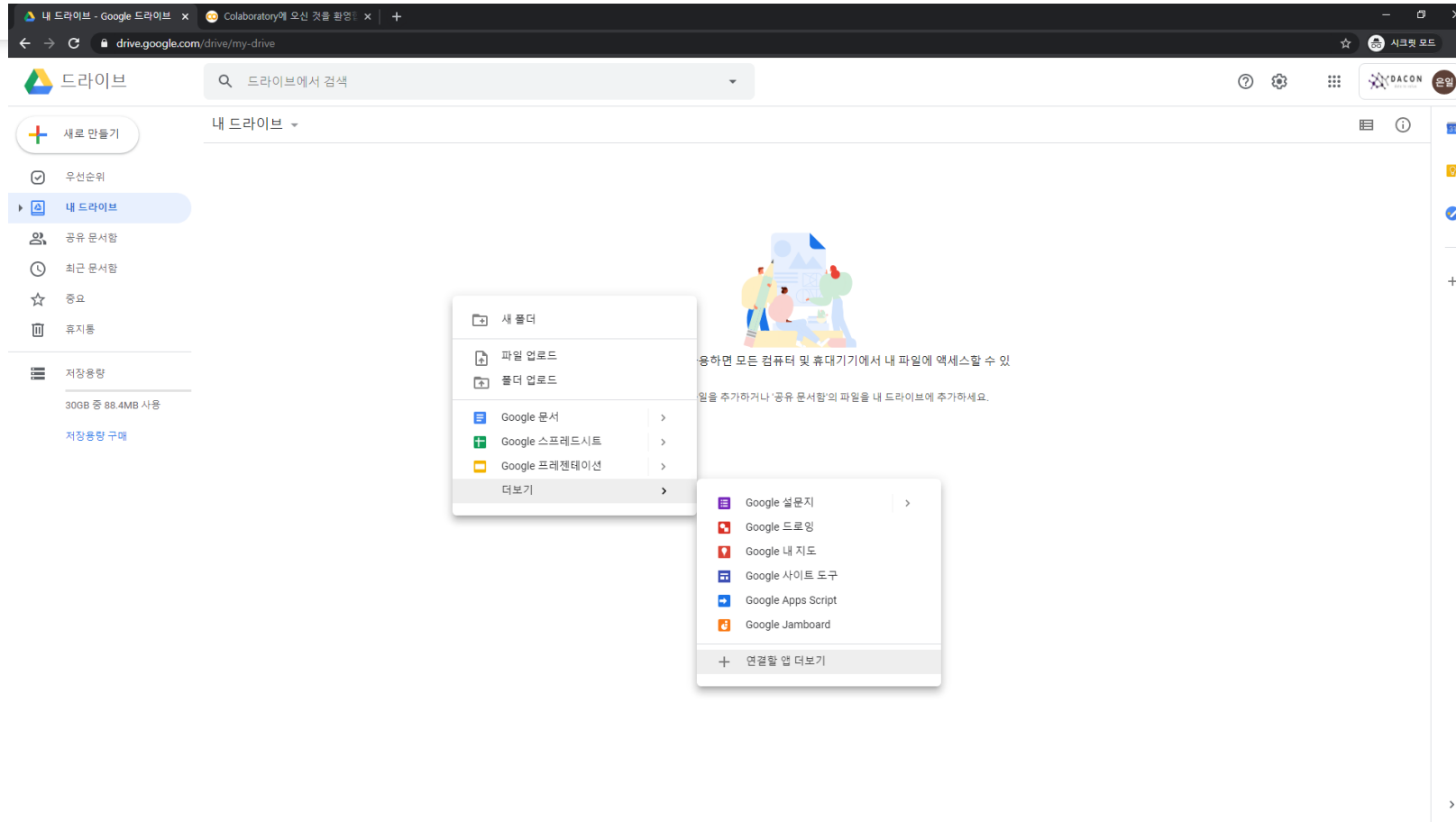
Google Colab으로 파이썬 시작하기



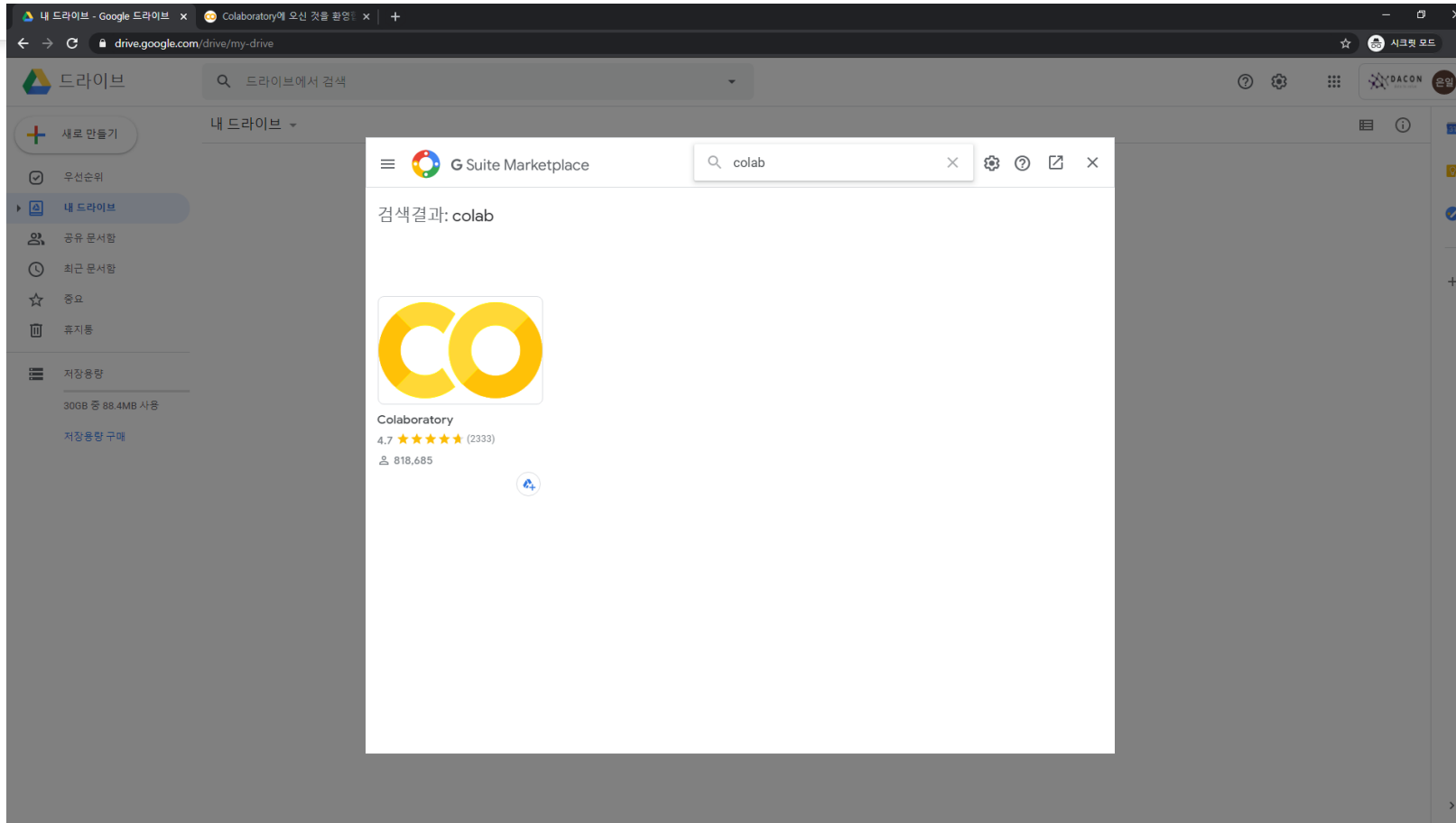
구글 드라이브에 Colab 설치



구글 드라이브에 Colab 설치



구글 드라이브에 Colab 설치



사용할 데이터 - 타이타닉



데이터 다운로드

The screenshot shows a web browser window displaying the DAICON competition page for the Titanic dataset. The page title is "[재난] 타이타닉 : 누가 살아남았을까?". The DAICON logo is visible in the top navigation bar. The page includes a header with navigation links: 교육, 코드공유, 토론, More, and a search bar. Below the header, there is a section for the competition details, including the title, prize (교육), dates (2019.12.11 ~ 2027.01.11 17:59), and a "참여중" button. A horizontal menu below this section contains links: 대회안내, 데이터, 코드 공유, 토론, 리더보드, 팀, and 제출. The "데이터" link is currently selected. The main content area is titled "설명" and contains a list of files: ① train.csv: 타이타닉 탑승자들 중 일부의 인적정보와 생존 여부 데이터, ② test.csv: 타이타닉 탑승자들 중 일부 (train set의 탑승자 제외)의 인적정보 데이터, and ③ sample_submission.csv: submission 파일의 예시. A "다운로드" button is located at the bottom right of the file list. Below the file list, there is a section titled "상세" with three columns: 목록, 컬럼, and 컬럼상세. The "목록" column shows "train.csv(60KB)". The "컬럼" column shows "PassengerId". The "컬럼상세" column shows "탑승객의 고유 아이디".

[재난] 타이타닉 : 누가 살아남았을까?

상금 : 교육

2019.12.11 ~ 2027.01.11 17:59 + Google Calendar

191팀 D-2405

참여중

대회안내 데이터 코드 공유 토론 리더보드 팀 제출

설명

[Files]

- ① train.csv : 타이타닉 탑승자들 중 일부의 인적정보와 생존 여부 데이터
- ② test.csv : 타이타닉 탑승자들 중 일부 (train set의 탑승자 제외)의 인적정보 데이터
- ③ sample_submission.csv : submission 파일의 예시

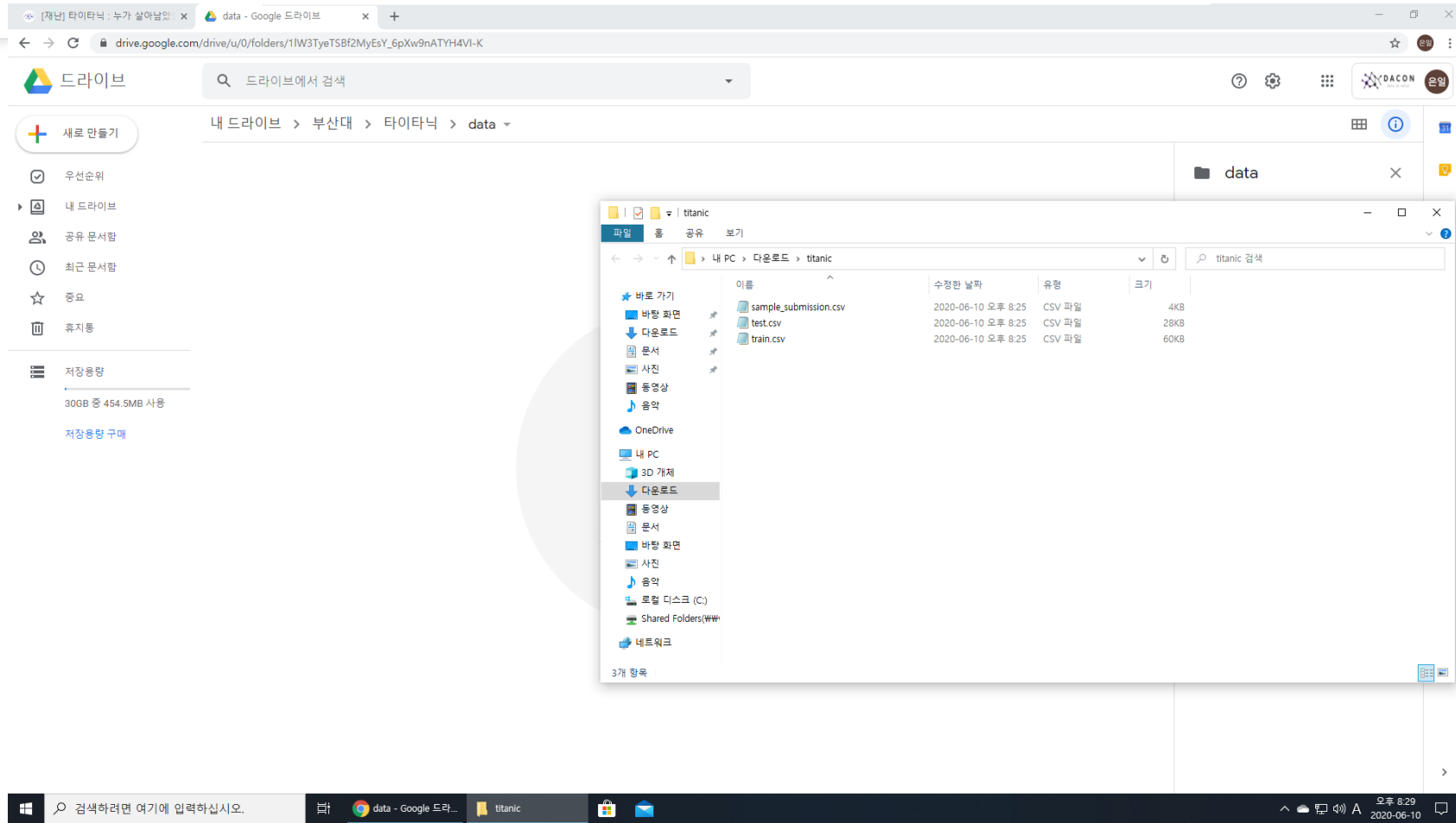
다운로드

상세

목록	컬럼	컬럼상세
train.csv(60KB)	PassengerId	탑승객의 고유 아이디

<https://dacon.io/competitions/open/235539/data/>

데이터 다운로드



Colab 실행

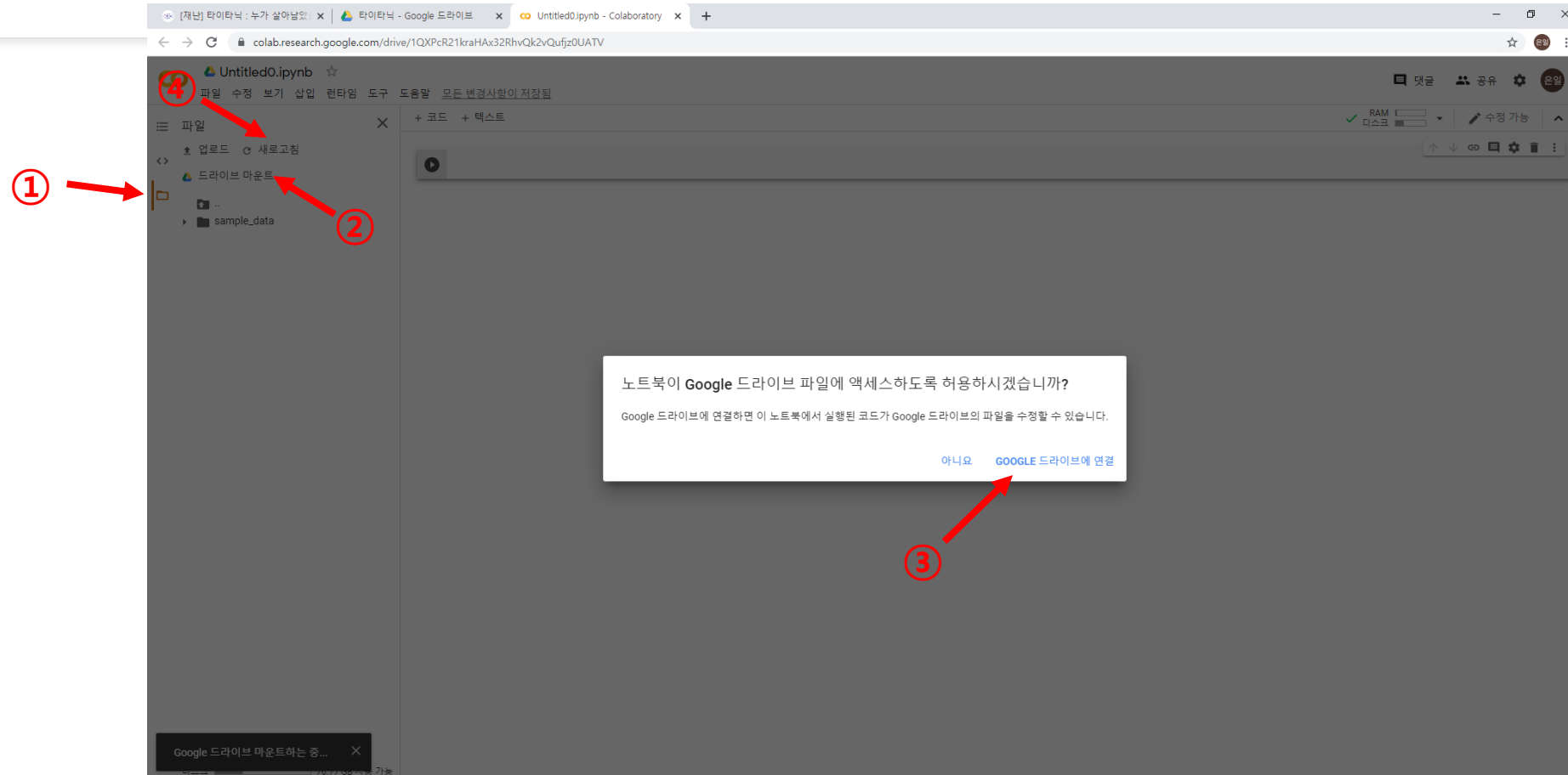
The screenshot displays the Google Drive web interface. The main view shows a folder named '타이타닉' (Titanic) under the '내 드라이브' (My Drive) section. A context menu is open over the 'data' folder, listing options such as '새 폴더' (New Folder), '파일 업로드' (Upload File), '폴더 업로드' (Upload Folder), and various Google apps including Google Colaboratory. The sidebar on the right provides details for the '타이타닉' folder, including its type (Google Drive folder), location (부산대), owner (나), and modification history. A bottom overlay shows a list of uploaded files: train.csv, sample_submission.csv, and test.csv, all marked as '업로드 3개 완료' (3 uploads complete).

이름	소유자	마지막으로 수정한 날짜	파일 크기
data	나	오후 8:29 나	-

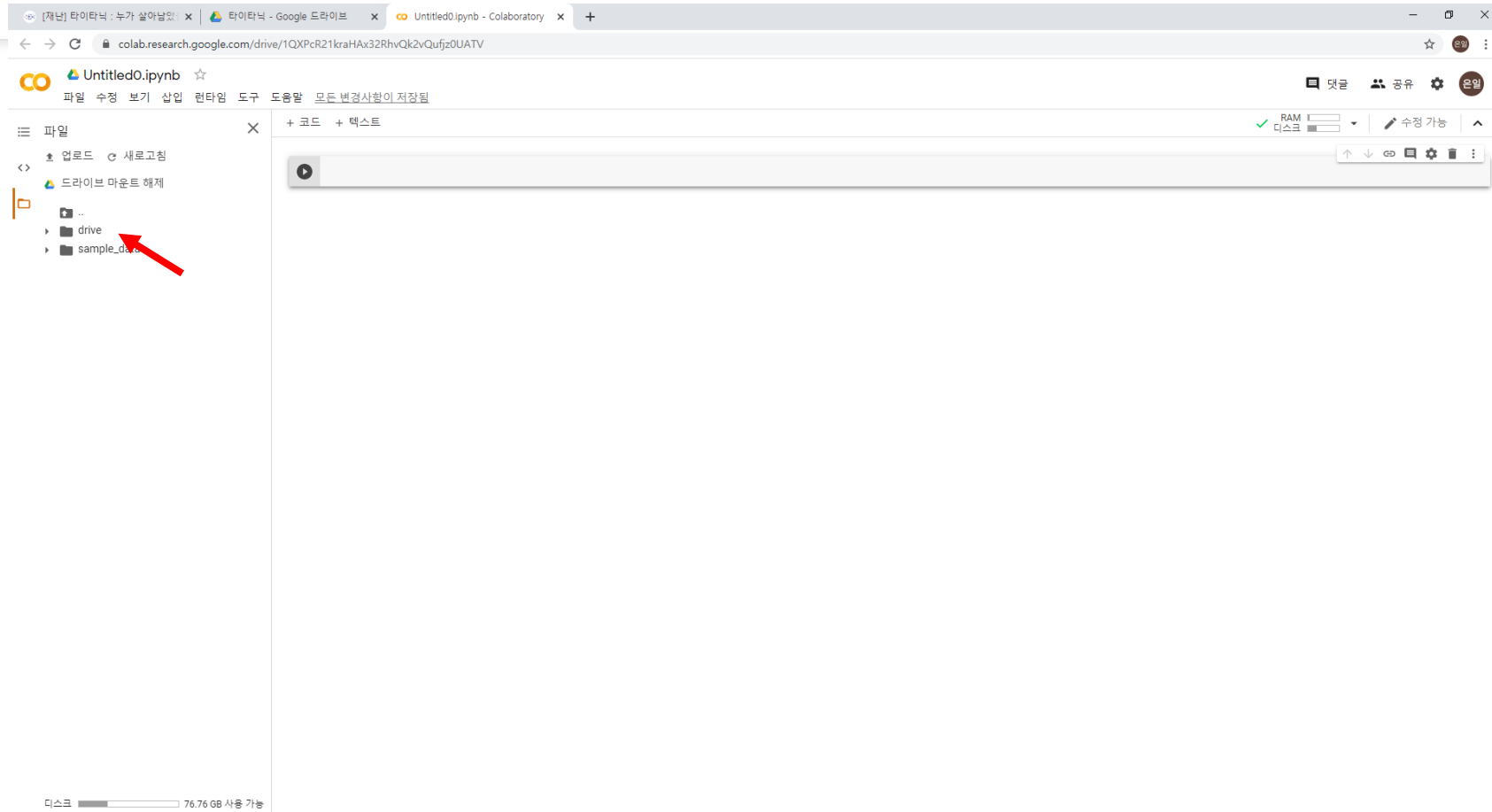
타이타닉	
세부정보	활동
정	
유형	Google 드라이브 폴더
위치	부산대
소유자	나
수정한 날짜	내가 오후 8:26에 수정함
열어본 날짜	내가 오후 8:29에 열어봄
생성한 날짜	Google Drive Web(으)로 오후 8:26에 작성됨

업로드 3개 완료	
train.csv	✓
sample_submission.csv	✓
test.csv	✓

구글 드라이브 마운트



구글 드라이브 마운트



파이썬 기초 1. 연산

[1] 1+1

↳ 2

[2] 32-12

↳ 20

[3] 8*3

↳ 24

[4] 5/2

↳ 2.5

[5] 5//2

↳ 2

[6] 5%2

↳ 1

[7] 5**2

↳ 25

[8] a = 3
b = 2
a+b

↳ 6

파이썬 기초 2.자료형

- 숫자 – 정수(1, 2, 3, 4...), 실수(1.24, 2.234...)
- 문자 – 'Hello World!'

```
[9] txt = 'hello world!'
    txt
```

```
↳ 'hello world!'
```

파이썬 기초 3.자료 구조

- 리스트
 - [0,1,2,3] => 1차원 데이터
 - [[0,1], [2,3]] => 2차원 데이터 [[0,1], [2,3]]
- 튜플 – (0,1,2,3)
 - 리스트와의 차이 : 저장된 요소 수정 불가
- 딕셔너리 – {이름:[철수, 영희], 성별:[여자, 남자]}

Pandas 패키지



- 패키지(라이브러리) : 프로그래밍 언어에서 다양한 기능의 모음
- Pandas : 정형 데이터를 다루는 파이썬 패키지
 - 1차원 데이터 → Series
 - 2차원 데이터 → DataFrame

	이름	출석번호
0	철수	21
1	영희	22

DataFrame

```
df['이름']  
  
0    철수  
1    영희  
Name: 이름, dtype: object
```

Series

Pandas로 데이터 프레임 만들기

1. 데이터프레임으로 만들 딕셔너리 생성

```
di = {'이름': ['철수', '영희'], '출석번호': [21, 22]}  
di  
  
{'이름': ['철수', '영희'], '출석번호': [21, 22]}
```

2. Pandas 패키지 import

```
import pandas as pd
```

3. pd.DataFrame()를 이용하여 데이터 프레임 생성

```
df = pd.DataFrame(di)  
df
```

	이름	출석번호
0	철수	21
1	영희	22

타이타닉 데이터 살펴보기

- train.csv : 타이타닉 탑승자들 중 일부의 인적정보와 생존 여부 데이터
- test.csv : 타이타닉 탑승자들 중 일부(train set의 탑승자 제외)의 인적정보 데이터
- sample_submission.csv : submission 파일의 예시

타이타닉 데이터 살펴보기

생존 정보

	A	B	C	D	E	F	G	H	I	J	K	L
1	Passenger	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	1	0	3	Braund, M	male	22	1	0	A/5 21171	7.25		S
3	2	1	1	Cumings, female	female	38	1	0	PC 17599	71.2833	C85	C
4	3	1	3	Heikkinen female	female	26	0	0	STON/O2	7.925		S
5	4	1	1	Futrelle, M	female	35	1	0	113803	53.1	C123	S
6	5	0	3	Allen, Mr.	male	35	0	0	373450	8.05		S
7	6	0	3	Moran, M	male		0	0	330877	8.4583		C
8	7	0	1	McCarthy, male	male	54	0	0	17463	51.8625	E46	S
9	8	0	3	Palsson, M	male	2	3	1	349909	21.075		S
10	9	1	3	Johnson, M	female	27	0	2	347742	11.1333		S
11	10	1	2	Nasser, M	female	14	1	0	237736	30.0708		C
12	11	1	3	Sandstrom female	female	4	1	1	PP 9549	16.7	G6	S
13	12	1	1	Bonnell, M	female	58	0	0	113783	26.55	C103	S
14	13	0	3	Saunders male	male	20	0	0	A/5. 2151	8.05		S
15	14	0	3	Andersson male	male	39	1	5	347082	31.275		S

탑승자 정보

train.csv 파일을 Excel로 열어본 모습

타이타닉 데이터 살펴보기

	A	B	C	D	E	F	G	H	I	J	K
1	Passenger	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	892	3	Kelly, Mr.	male	34.5	0	0	330911	7.8292	Q	
3	893	3	Wilkes, Mr.	female	47	1	0	363272	7		S
4	894	2	Myles, Mr.	male	62	0	0	240276	9.6875		Q
5	895	3	Wirz, Mr.	male	27	0	0	315154	8.6625		S
6	896	3	Hirvonen, female		22	1	1	3101298	12.2875		S
7	897	3	Svensson, male		14	0	0	7538	9.225		S
8	898	3	Connolly, female		30	0	0	330972	7.6292		Q
9	899	2	Caldwell, male		26	1	1	248738			
10	900	3	Abraham, female		18	0	0	2657			
11	901	3	Davies, Mr.	male	21	2	0	A/4 48871	24.15		S
12	902	3	Ilieff, Mr.	male		0	0	349220	7.8958		S
13	903	1	Jones, Mr.	male	46	0	0	694	26		S
14	904	1	Snyder, M	female	23	1	0	21228	82.2667	B45	S

탑승자 정보

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

생존 예측

test.csv 파일을 Excel로 열어본 모습

Sample_submission.csv 파일을 Excel로 열어본 모습

메모장으로 열어 본 CSV



The screenshot shows a Windows Notepad window titled "train.csv - Windows 메모장". The menu bar includes "파일(F)", "편집(E)", "서식(O)", "보기(V)", and "도움말(H)". The text content is a CSV file with the following header and data rows:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	"Braund, Mr. Owen Harris"	male	22	1	0	A/5 21171	7.25		S
2	1	1	"Cumings, Mrs. John Bradley (Florence Briggs Thayer)"	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	"Heikkinen, Miss. Laina"	female	26	0	0	STON/O2. 3101282	7.925		S
4	1	1	"Futrelle, Mrs. Jacques Heath (Lily May Peel)"	female	35	1	0	113803	53.1	C123	S
5	0	3	"Allen, Mr. William Henry"	male	35	0	0	373450	8.05		S
6	0	3	"Moran, Mr. James"	male		0	0	330877	8.4583		Q
7	0	1	"McCarthy, Mr. Timothy J"	male	54	0	0	17463	51.8625	E46	S
8	0	3	"Palsson, Master. Gosta Leonard"	male	2	3	1	349909	21.075		S
9	1	3	"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)"	female	27	0	2	347742	11.1333		S
10	1	2	"Nasser, Mrs. Nicholas (Adele Achem)"	female	14	1	0	237736	30.0708		C
11	1	3	"Sandstrom, Miss. Marguerite Rut"	female	4	1	1	PP 9549	16.7	G6	S
12	1	1	"Bonnell, Miss. Elizabeth"	female	58	0	0	113783	26.55	C103	S
13	0	3	"Saunderscock, Mr. William Henry"	male	20	0	0	A/5. 2151	8.05		S
14	0	3	"Andersson, Mr. Anders Johan"	male	39	1	5	347082	31.275		S
15	0	3	"Vestrom, Miss. Hulda Amanda Adolfina"	female	14	0	0	350406	7.8542		S
16	1	2	"Hewlett, Mrs. (Mary D Kingcome) "	female	55	0	0	248706	16		S

The status bar at the bottom indicates "Ln 49, Col 61", "100%", "Windows (CRLF)", and "UTF-8".

머신러닝 과정

- 데이터 가져오기
- 데이터 전처리
- 데이터 시각화
- 특징(Feature) 선택
- 모델 설계 & 학습
- 결과 예측

Padas로 csv파일 열기 - pd.read_csv()

```
[1] import pandas as pd
```

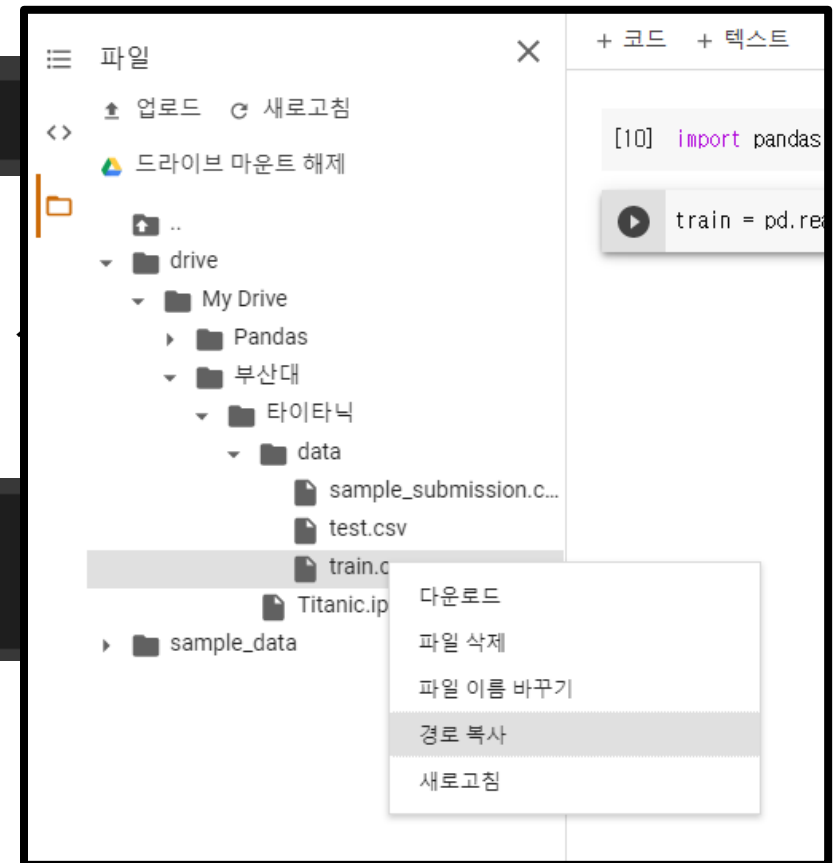
Pandas 패키지 가져오기

Pandas 패키지 이름을 pd로 줄여서

```
[2] train = pd.read_csv('/content/drive/My Drive/부산대/타이타닉/data/train.csv')  
test = pd.read_csv('/content/drive/My Drive/부산대/타이타닉/data/test.csv')  
submission = pd.read_csv('/content/drive/My Drive/부산대/타이타닉/data/sample_submission.csv')
```

Pandas로 CSV 파일 읽기

CSV 파일 위치 및 파일 이름



Train – pd.DataFrame.head()

위에서 10줄만 출력, 숫자 입력 안하면 5줄 출력

[4] train.head(10)

	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			26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacc		35.0	1	0	113803	53.1000	C123	S
4	5	0	3			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
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C

2차원 데이터
DataFrame

DataFrame 정보 - pd.DataFrame.info()

[4] train.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
```

[5] test.info()

```
<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
```

결측치 - pd.DataFrame.isna()

```
[8] train.isna()
```



	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	False	False	False	False	False	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	True	False
...
886	False	False	False	False	False	False	False	False	False	False	True	False
887	False	False	False	False	False	False	False	False	False	False	False	False
888	False	False	False	False	False	True	False	False	False	False	True	False
889	False	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	True	False

891 rows × 12 columns

결측치 pd.DataFrame.sum()

```
train.shape
```

```
(891, 12)
```

```
[6] train.isna().sum()
```

```
train.isna().sum()/train.shape[0]
```

PassengerId	0.000000
Survived	0.000000
Pclass	0.000000
Name	0.000000
Sex	0.000000
Age	0.198653
SibSp	0.000000
Parch	0.000000
Ticket	0.000000
Fare	0.000000
Cabin	0.771044
Embarked	0.002245

dtype: float64

```
test.isna().sum()
```

```
test.isna().sum()/test.shape[0]
```

PassengerId	0.000000
Pclass	0.000000
Name	0.000000
Sex	0.000000
Age	0.205742
SibSp	0.000000
Parch	0.000000
Ticket	0.000000
Fare	0.002392
Cabin	0.782297
Embarked	0.000000

dtype: float64

특정 컬럼 선택 - pd.DataFrame['column']

[4] train.head(10)

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	male	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	male	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	male	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	male	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	male	14.0	1	0	237736	30.0708	NaN	C

[9] train['Age']

0	22.0
1	38.0
2	26.0
3	35.0
4	35.0
...	...
886	27.0
887	19.0
888	NaN
889	26.0
890	32.0

Name: Age, Length: 891, dtype: float64

1차원 데이터 Series

여러가지 통계 정보

mean(), min(), max(), median(), describe()

[4] train.head(10)

[9] train['Age']

022.0

138.0

226.0

335.0

435.0

...

88627.0

88719.0

888NaN

88926.0

89032.0

Name: Age, Length: 891, dtype: float64

[11] train['Age'].mean()

29.69911764705882

[12] train['Age'].min()

0.42

[13] train['Age'].max()

80.0

[14] train['Age'].median()

28.0

[15] train['Age'].describe()

count714.000000

mean29.699118

std14.526497

min0.420000

25%20.125000

50%28.000000

75%38.000000

max80.000000

Name: Age, dtype: float64

91012Nasser, Mrs. Nicholas (Adele Achem)female14.01023773630.0708NaNCS

결측치 처리 - pd.Series.fillna()

```
[4] train.head(10)
```

	PassengerId	Survived	Pclass	
0	1	0	3	
1	2	1	1	Cumings,
2	3	1	3	
3	4	1	1	Fut
4	5	0	3	
5	6	0	3	
6	7	0	1	
7	8	0	3	
8	9	1	3	Johnson, M
9	10	1	2	

```
[16] train['Age'] = train['Age'].fillna(28)  
test['Age'] = test['Age'].fillna(28)
```

```
[17] train.isna().sum()
```

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

	Parch	Ticket	Fare	Cabin	Embarked
0		A/5 21171	7.2500	NaN	S
0		PC 17599	71.2833	C85	C
0		STON/O2. 3101282	7.9250	NaN	S
0		113803	53.1000	C123	S
0		373450	8.0500	NaN	S
0		330877	8.4583	NaN	Q
0		17463	51.8625	E46	S
1		349909	21.0750	NaN	S
2		347742	11.1333	NaN	S
0		237736	30.0708	NaN	C

카운트 – pd.Series.value_counts()

[4] train.head(10)

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3					0	A/5 21171	7.2500	NaN	S
1	2	1	1					0	PC 17599	71.2833	C85	C
2	3	1	3					0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1					0	113803	53.1000	C123	S
4	5	0	3					0	373450	8.0500	NaN	S
5	6	0	3					0	330877	8.4583	NaN	Q
6	7	0	1					0	17463	51.8625	E46	S
7	8	0	3					1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C

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

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

```
[ ] train['Embarked'] = train['Embarked'].fillna('S')  
    test['Embarked'] = test['Embarked'].fillna('S')
```

생존자, 사망자

[4] train.head(10)

	PassengerId	Survived	Pclass
0	1	0	3
1	2	1	1
2	3	1	3
3	4	1	1
4	5	0	3
5	6	0	3
6	7	0	1
7	8	0	3
8	9	1	3
9	10	1	2

[19] train['Survived']

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

[18] train['Survived'].value_counts()

0	549
1	342

Name: Survived, dtype: int64

SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	A/5 21171	7.2500	NaN	S
1	0				C
0					S
1					S
0					S
0					Q
0	0	17463	51.8625	E46	S
3	1	349909	21.0750	NaN	S
0	2	347742	11.1333	NaN	S
1	0	237736	30.0708	NaN	C

생존, 사망 데이터 분리

```
[4] train.head(10)
```



	PassengerId	Survived	Pclass
0	1	0	
1	2	1	
2	3	1	
3	4	1	
4	5	0	
5	6	0	
6	7	0	
7	8	0	
8	9	1	
9	10	1	

```
[22] train['Survived']==1
```



```
0    False
1     True
2     True
3     True
4    False
...
886   False
887    True
888   False
889    True
890   False
Name: Survived, Length: 891, dtype: bool
```

```
[ ] survived = train[train['Survived']==1]
    dead = train[train['Survived']==0]
```

```
[ ] survived
```



	PassengerId	Survived	Pclass
1	2	1	1 Cumings, M
2	3	1	3
3	4	1	1 Futre
8	9	1	3 Johnson, M
9	10	1	2

생존 데이터

```
[23] survived = train[train['Survived']==1]
     dead = train[train['Survived']==0]
```

```
[24] survived
```

↗

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
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
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C
...
875	876	1	3	Najib, Miss. Adele Kiamie "Jane"	female	15.0	0	0	2667	7.2250	NaN	C
879	880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0	1	11767	83.1583	C50	C
880	881	1	2	Shelley, Mrs. William (Imanita Parrish Hall)	female	25.0	0	1	230433	26.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C

342 rows x 12 columns

Pclass에 따른 생존 여부

```
[4] train.head(10)
```

	PassengerId	Survived	Pclass	Name
0	1	0	3	Braund, Mr. Owen Ha
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs)
2	3	1	3	Heikkinen, Miss. L
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May P
4	5	0	3	Allen, Mr. William He
5	6	0	3	Moran, Mr. Jas
6	7	0	1	McCarthy, Mr. Timot
7	8	0	3	Palsson, Master. Gosta Leon
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina B
9	10	1	2	Nasser, Mrs. Nicholas (Adele Ach

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

```
[ ] 3    491  
    1    216  
    2    184  
    Name: Pclass, dtype: int64
```

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

```
[ ] 1    136  
    3    119  
    2     87  
    Name: Pclass, dtype: int64
```

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

```
[ ] 3    372  
    2     97  
    1     80  
    Name: Pclass, dtype: int64
```

ticket	Fare	Cabin	Embarked
171	7.2500	NaN	S
7599	71.2833	C85	C
282	7.9250	NaN	S
3803	53.1000	C123	S
3450	8.0500	NaN	S
0877	8.4583	NaN	Q
7463	51.8625	E46	S
9909	21.0750	NaN	S
7742	11.1333	NaN	S
7736	30.0708	NaN	C

Series로 DataFrame 생성 - pd.DataFrame()

```
[ ] survived_cnt = survived['Pclass'].value_counts()  
    dead_cnt = dead['Pclass'].value_counts()
```

```
[ ] df = pd.DataFrame([survived_cnt, dead_cnt])
```

```
[ ] df
```

```
↳
```

	1	2	3
Pclass	136	87	119
Pclass	80	97	372

```
[ ] df.index = ['survived', 'dead']
```

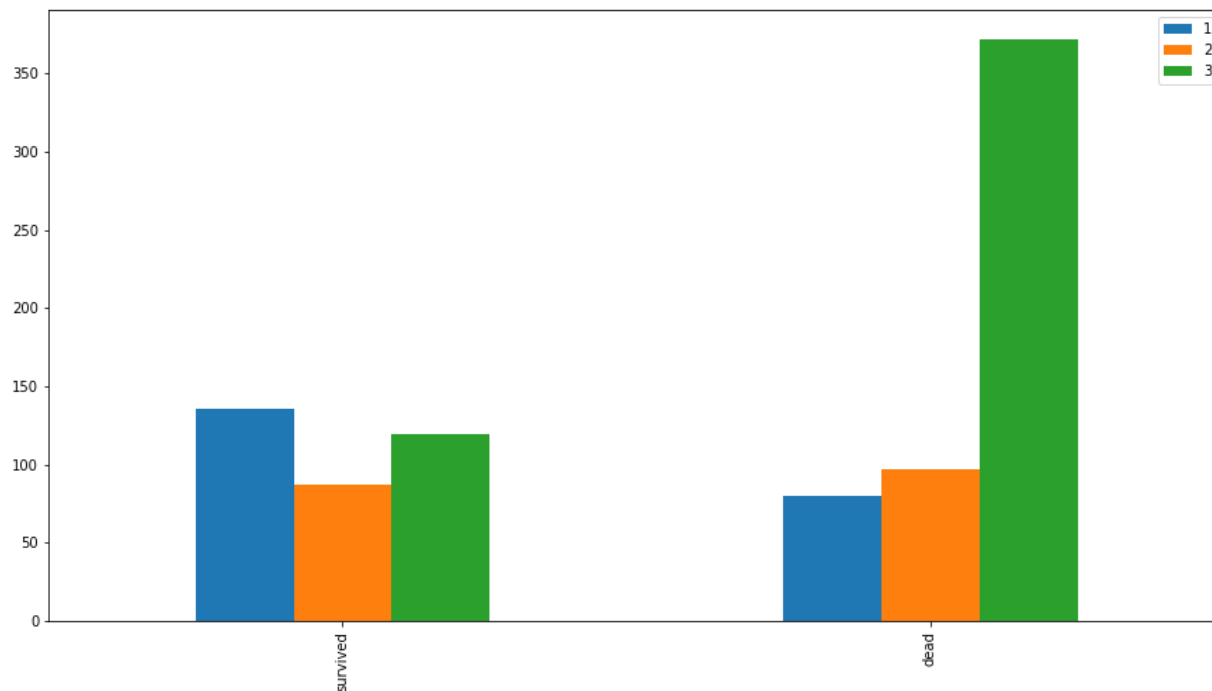
```
[ ] df
```

```
↳
```

	1	2	3
survived	136	87	119
dead	80	97	372

그래프 그리기 – pd.DataFrame.plot()

```
[ ] df.plot(kind='bar', figsize=(15,8))
```



생존률 계산

행과열 자리 바꾸기 - `pd.DataFrame.T`

```
[38] df.T
```

	survived	dead
1	136	80
2	87	97
3	119	372

새로운 컬럼 추가 - `pd.DataFrame[new_column] = pd.Series`

```
[43] df = df.T
      df['suival_rate'] = 100*df['survived']/(df['survived']+df['dead'])
      df
```

	survived	dead	suival_rate
1	136.0	80.0	62.962963
2	87.0	97.0	47.282609
3	119.0	372.0	24.236253

생존률 = $100 * \text{생존자} / (\text{생존자} + \text{사망자})$

```
[41] df = df.T
      100*df['survived']/(df['survived']+df['dead'])
```

1	62.962963
2	47.282609
3	24.236253

dtype: float64

그래프~생존률 한번에

```
column = 'Pclass'
```

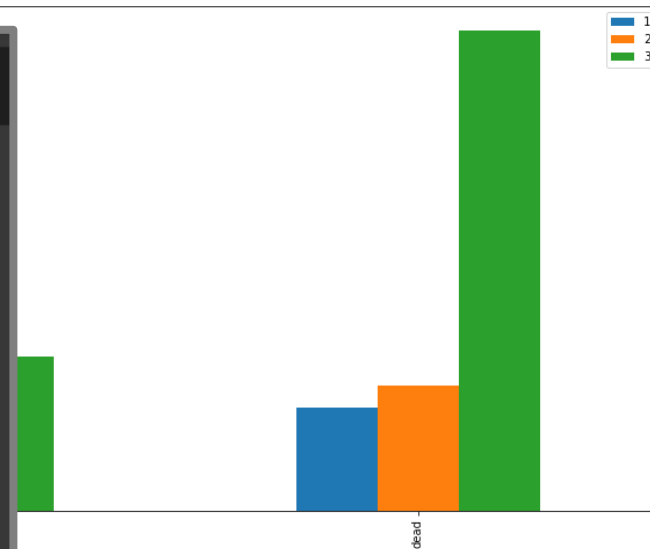
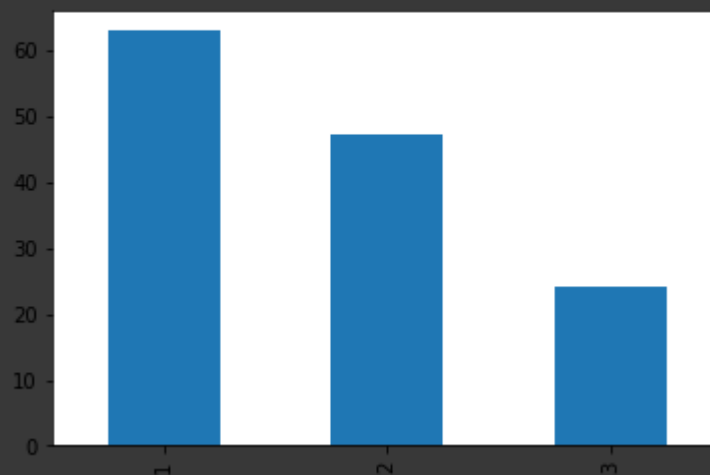
```
survived = train[train['Survived']==1]  
dead = train[train['Survived']==0]  
survived_cnt = survived[column].value_counts()  
dead_cnt = dead[column].value_counts()  
df = pd.DataFrame([survived_cnt, dead_cnt])  
df.index = ['survived', 'dead']  
df.plot(kind='bar', figsize=(15,8))
```

```
df = df.T  
df['survival_rate'] = 100*df['survived']/df['dead']
```

	survived	dead	survival_rate
1	136	80	62.962963
2	87	97	47.282609
3	119	372	24.236253

```
[48] df['survival_rate'].plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6f62421860>
```



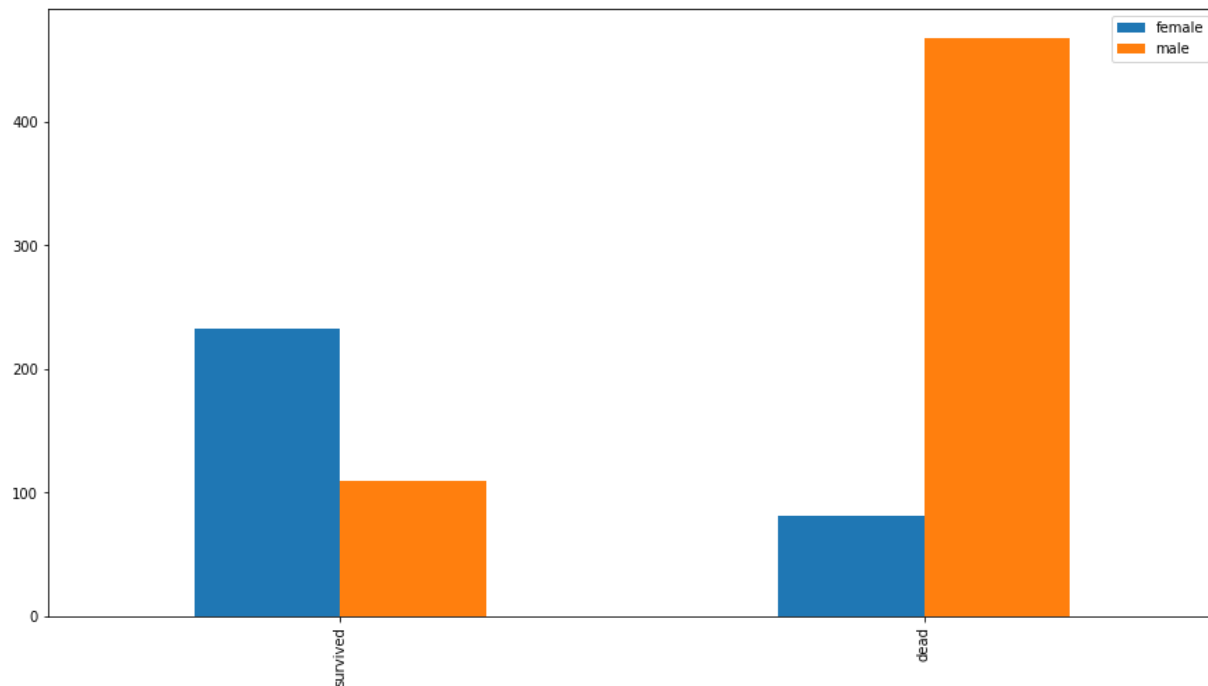
다른 컬럼 시각화

```
[60] column = 'Sex'
```

```
survived = train[train['Survived']==1]
dead = train[train['Survived']==0]
survived_cnt = survived[column].value_counts()
dead_cnt = dead[column].value_counts()
df = pd.DataFrame([survived_cnt, dead_cnt])
df.index = ['survived', 'dead']
df.plot(kind='bar', figsize=(15,8))

df = df.T
df['survival_rate'] = 100*df['survived']/(df['survived']+df['dead'])
df
```

	survived	dead	survival_rate
female	233	81	74.203822
male	109	468	18.890815



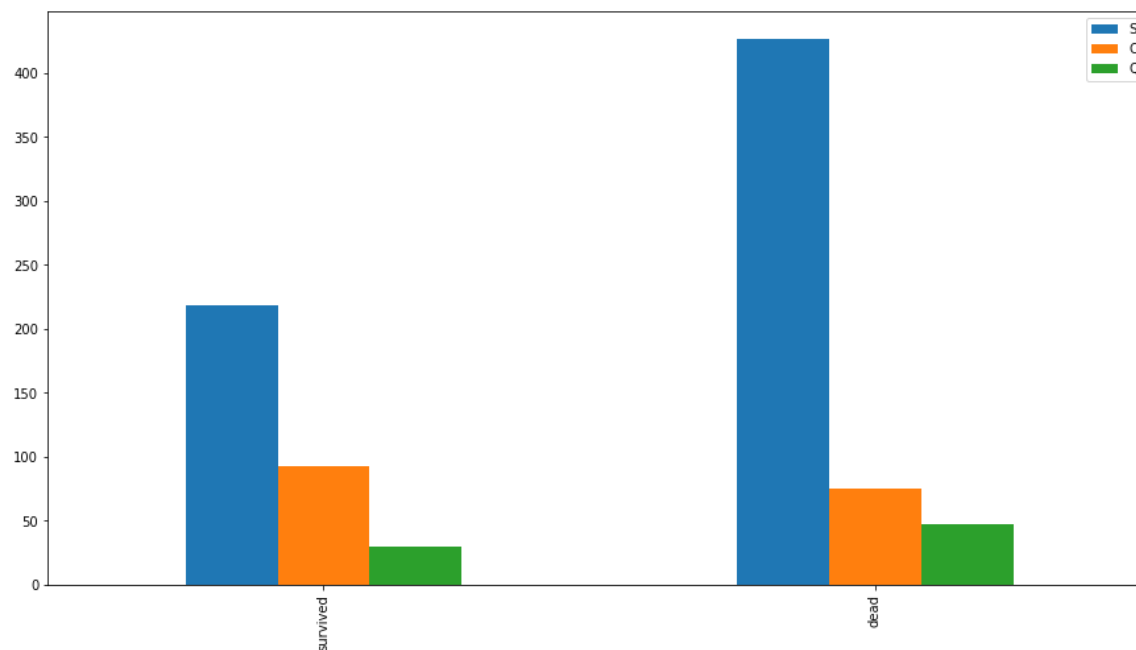
다른 컬럼 시각화

```
[61] column = 'Embarked'
```

```
survived = train[train['Survived']==1]
dead = train[train['Survived']==0]
survived_cnt = survived[column].value_counts()
dead_cnt = dead[column].value_counts()
df = pd.DataFrame([survived_cnt, dead_cnt])
df.index = ['survived', 'dead']
df.plot(kind='bar', figsize=(15,8))

df = df.T
df['survival_rate'] = 100*df['survived']/(df['survived']+df['dead'])
df
```

	survived	dead	survival_rate
S	219	427	33.900929
C	93	75	55.357143
Q	30	47	38.961039



두 개의 컬럼을 이용해 새로운 컬럼 생성

```
train['family'] = train['SibSp'] + train['Parch']
```

```
[ ] train['SibSp']
```

0	1
1	1
2	0
3	1
4	0
...	...
886	0
887	0
888	1
889	0
890	0

Name: SibSp, Length: 891

```
[ ] train['Parch']
```

0	0
1	0
2	0
3	0
4	0
...	...
886	0
887	0
888	2
889	0
890	0

Name: Parch, Length: 891

Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	family
male	22.0	1	0	A/5 21171	7.2500	NaN	S	1
male	38.0	1	0	PC 17599	71.2833	C85	C	1
male	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	0
male	35.0	1	0	113803	53.1000	C123	S	1
male	35.0	0	0	373450	8.0500	NaN	S	0

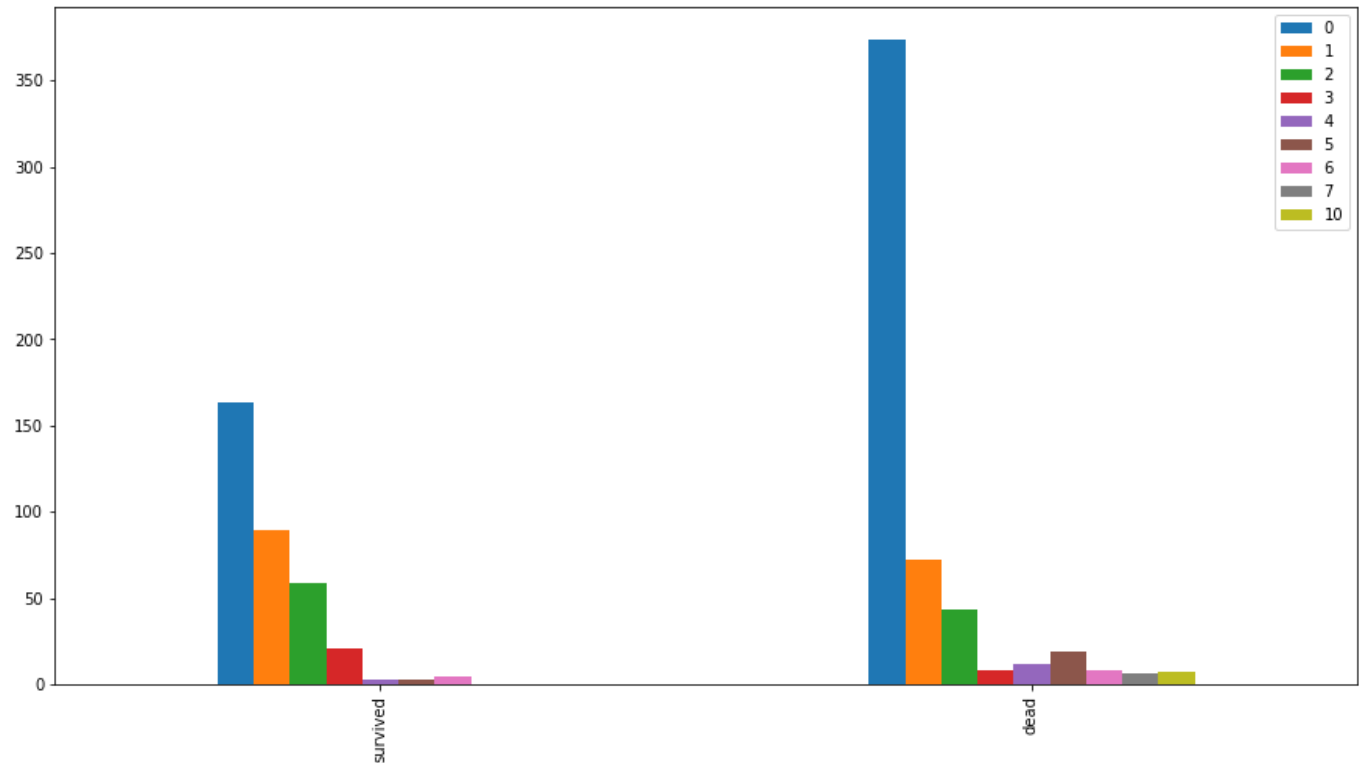
family 컬럼 시각화

```
[52] column = 'family'
```

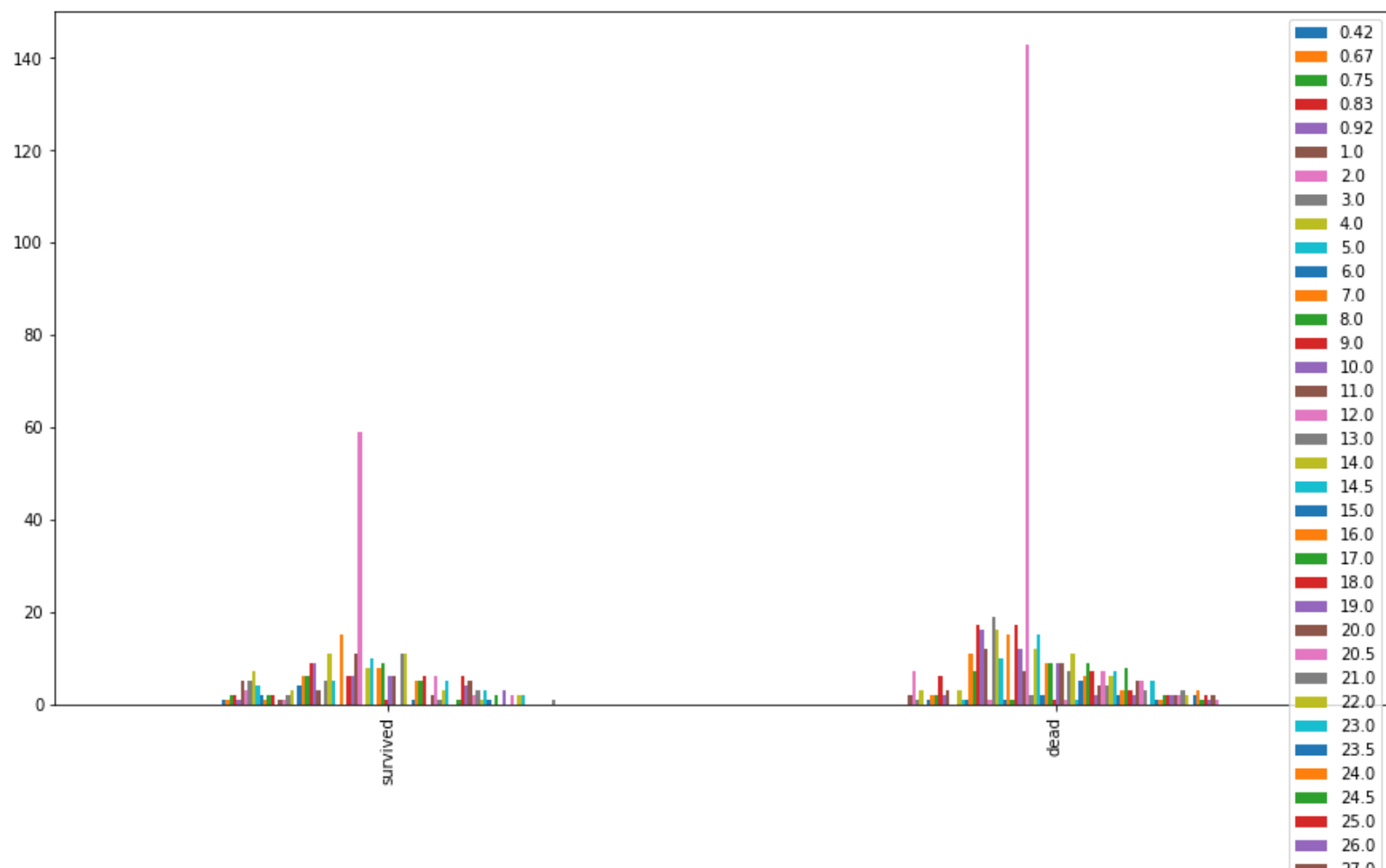
```
survived = train[train['Survived']==1]
dead = train[train['Survived']==0]
survived_cnt = survived[column].value_counts()
dead_cnt = dead[column].value_counts()
df = pd.DataFrame([survived_cnt, dead_cnt])
df.index = ['survived', 'dead']
df.plot(kind='bar', figsize=(15,8))

df = df.T
df['survival_rate'] = 100*df['survived']/(df['survived']+df['dead'])
df
```

	survived	dead	survival_rate
0	163.0	374.0	30.353818
1	89.0	72.0	55.279503
2	59.0	43.0	57.843137
3	21.0	8.0	72.413793
4	3.0	12.0	20.000000
5	3.0	19.0	13.636364
6	4.0	8.0	33.333333
7	NaN	6.0	NaN
10	NaN	7.0	NaN



Age 컬럼 시각화



Age 컬럼 binning

- 10살 미만 => 0
- 10살 이상 20살 미만 => 1
- 20살 이상 30살 미만 => 2
- 30살 이상 40살 미만 => 3
- 40살 이상 50살 미만 => 4
- 50살 이상 60살 미만 => 5
- 60살 이상 70살 미만 => 6
- 70살 이상 => 7

pd.DataFrame.loc[조건, 컬럼] = 값

```
[ ] train.loc[train['Age'] < 10, 'Age_bin'] = 0
    train.loc[(train['Age'] >= 10) & (train['Age'] < 20), 'Age_bin'] = 1
    train.loc[(train['Age'] >= 20) & (train['Age'] < 30), 'Age_bin'] = 2
    train.loc[(train['Age'] >= 30) & (train['Age'] < 40), 'Age_bin'] = 3
    train.loc[(train['Age'] >= 40) & (train['Age'] < 50), 'Age_bin'] = 4
    train.loc[(train['Age'] >= 50) & (train['Age'] < 60), 'Age_bin'] = 5
    train.loc[(train['Age'] >= 60) & (train['Age'] < 70), 'Age_bin'] = 6
    train.loc[train['Age'] >= 70, 'Age_bin'] = 7

    test.loc[test['Age'] < 10, 'Age_bin'] = 0
    test.loc[(test['Age'] >= 10) & (test['Age'] < 20), 'Age_bin'] = 1
    test.loc[(test['Age'] >= 20) & (test['Age'] < 30), 'Age_bin'] = 2
    test.loc[(test['Age'] >= 30) & (test['Age'] < 40), 'Age_bin'] = 3
    test.loc[(test['Age'] >= 40) & (test['Age'] < 50), 'Age_bin'] = 4
    test.loc[(test['Age'] >= 50) & (test['Age'] < 60), 'Age_bin'] = 5
    test.loc[(test['Age'] >= 60) & (test['Age'] < 70), 'Age_bin'] = 6
    test.loc[test['Age'] >= 70, 'Age_bin'] = 7
```

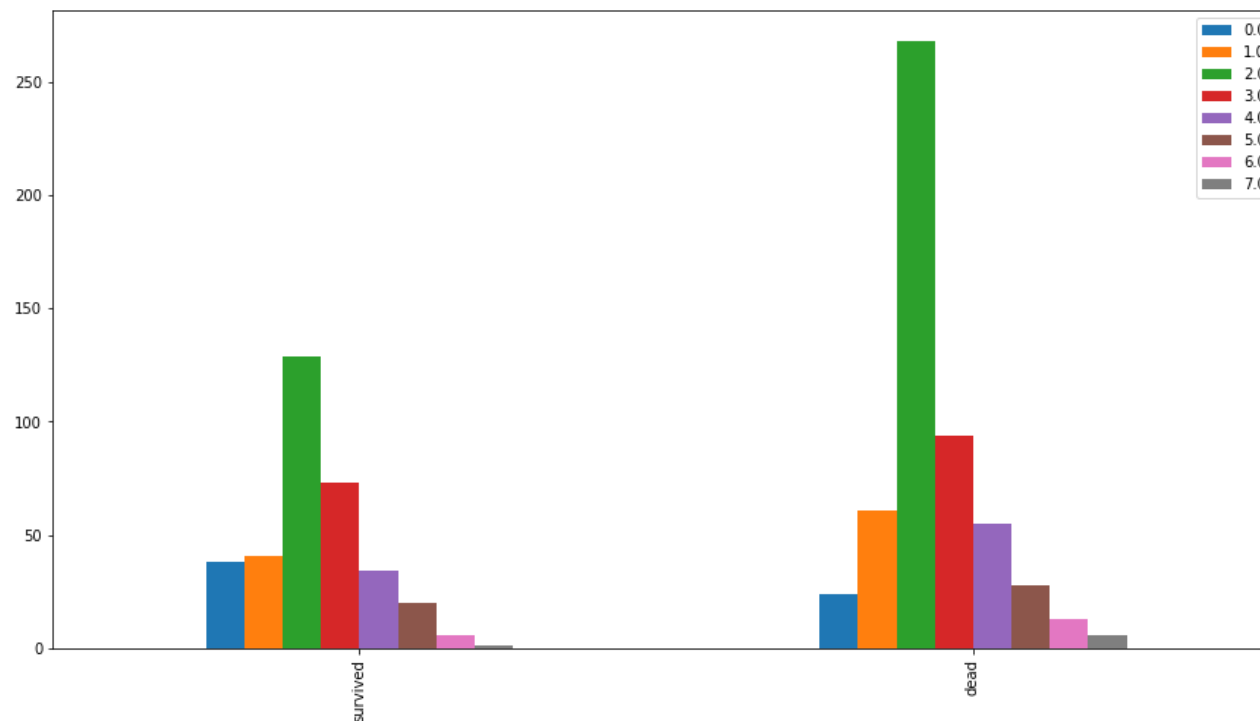
Age_bin 컬럼 시각화

```
[65] column = 'Age_bin'
```

```
survived = train[train['Survived']==1]
dead = train[train['Survived']==0]
survived_cnt = survived[column].value_counts()
dead_cnt = dead[column].value_counts()
df = pd.DataFrame([survived_cnt, dead_cnt])
df.index = ['survived', 'dead']
df.plot(kind='bar', figsize=(15,8))
```

```
df = df.T
df['survival_rate'] = 100*df['survived']/(df['survived']+df['dead'])
df
```

	survived	dead	survival_rate
0.0	38	24	61.290323
1.0	41	61	40.196078
2.0	129	268	32.493703
3.0	73	94	43.712575
4.0	34	55	38.202247
5.0	20	28	41.666667
6.0	6	13	31.578947
7.0	1	6	14.285714



Feature 선택

```
[71] train.head(10)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0
1	2	1	1	Cotton, Mr. Albert J. (Ellen)	male	38.0	1	0
2	3	1	3	Heikkinen, Mr. Teodor	male	26.0	0	0
3	4	1	1	Hart, Mr. William	male	35.0	0	0
4	5	0	3	Black, Mr. John	male	34.0	1	0
5	6	1	1	McCarthy, Mr. Timothy J	male	54.0	0	0
6	7	0	1	Palsson, Master. Gosta Leonard	male	2.0	3	1
7	8	0	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0

```
[72] x_train = train[['Pclass', 'Sex', 'Embarked', 'Age_bin', 'family' ]]  
test_ = test[['Pclass', 'Sex', 'Embarked', 'Age_bin', 'family' ]]  
y_train = train['Survived']
```

```
[75] y_train
```

		Age_bin
0	0	0.0
1	1	2.0
2	1	0.0
3	1	2.0
4	0	0.0
...
886	0	0.0
887	1	2.0
888	0	1.0
889	1	1.0
890	0	2.0

Name: Survived, Length: 891, dtype: int64

x_train=>문제, y_train=>답, test_=>시험 문제

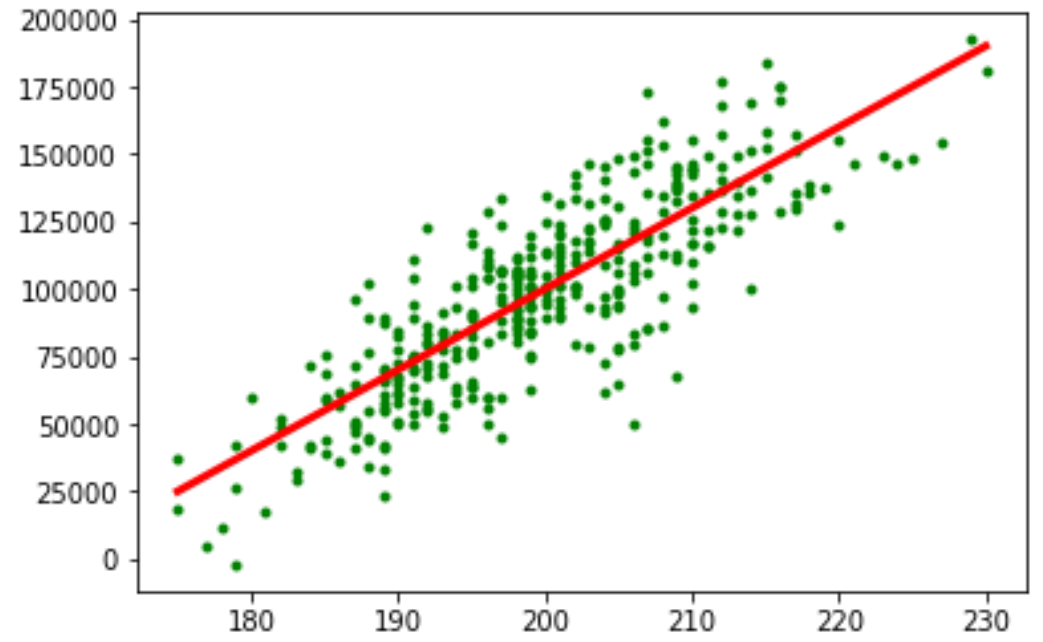
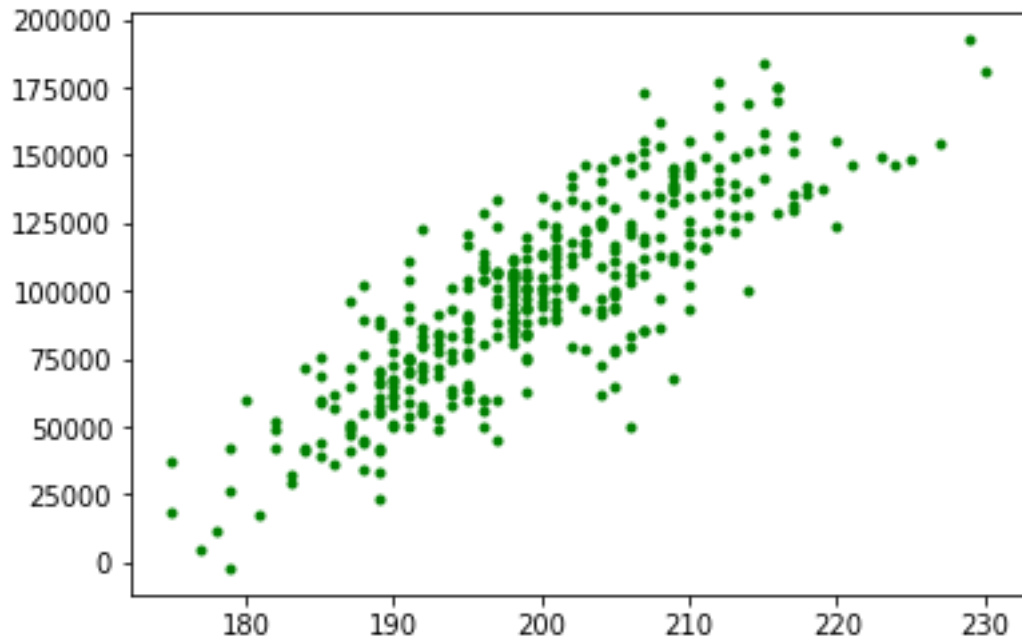
mapping - pd.DataFrame.map()

```
[74] x_train['Sex'] = x_train['Sex'].map({'male':0, 'female':1})  
test_['Sex'] = test_['Sex'].map({'male':0, 'female':1})  
  
x_train['Embarked'] = x_train['Embarked'].map({'S':0, 'C':1, 'Q':2})  
test_['Embarked'] = test_['Embarked'].map({'S':0, 'C':1, 'Q':2})
```

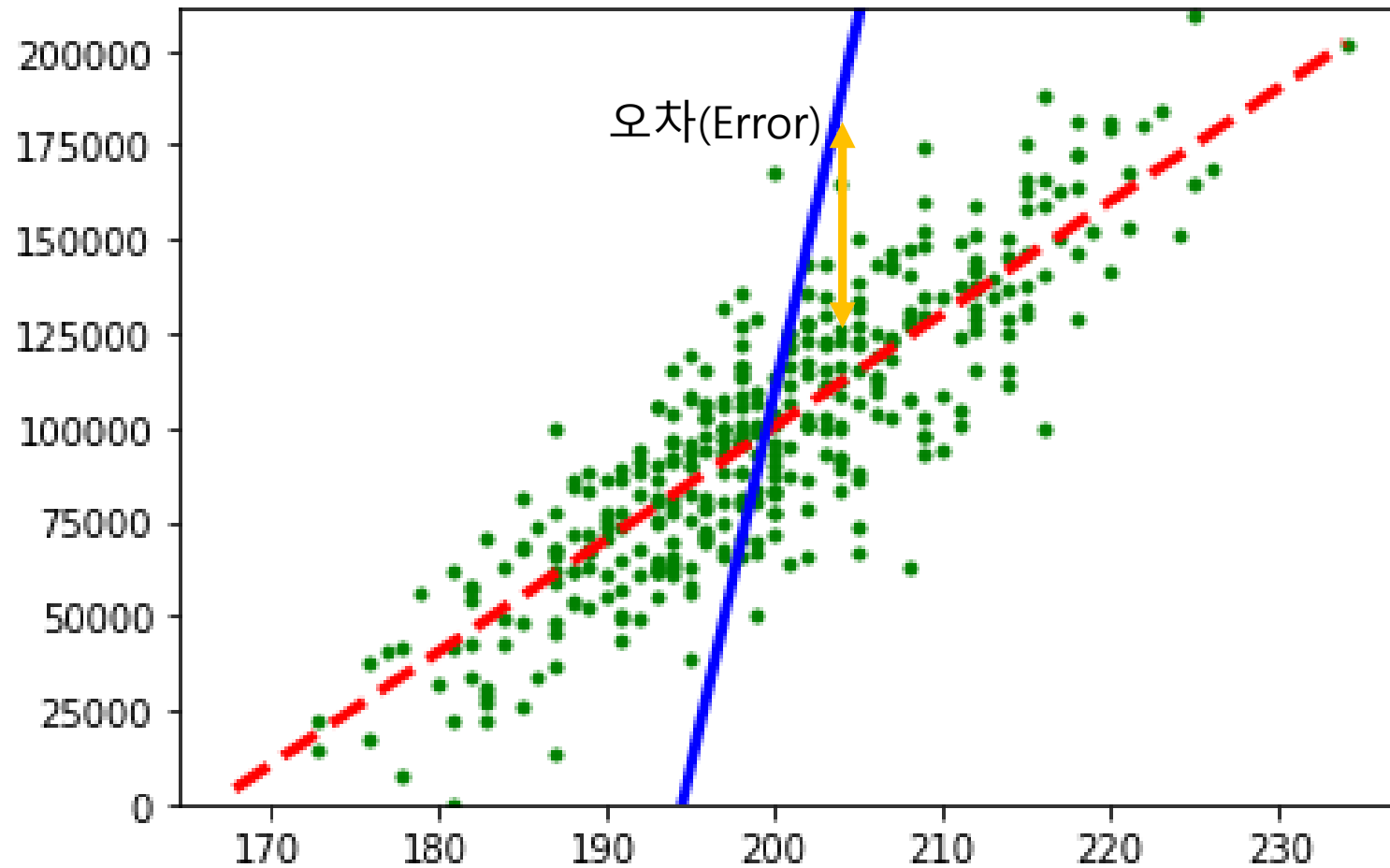
```
[76] x_train.head()
```

	Pclass	Sex	Embarked	Age_bin	Fare_bin
0	3	0	0	2.0	0.0
1	1	1	1	3.0	2.0
2	3	1	0	2.0	0.0
3	1	1	0	3.0	2.0
4	3	0	0	3.0	0.0

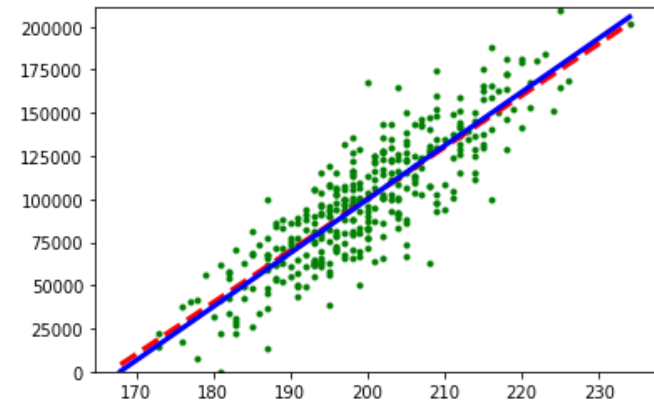
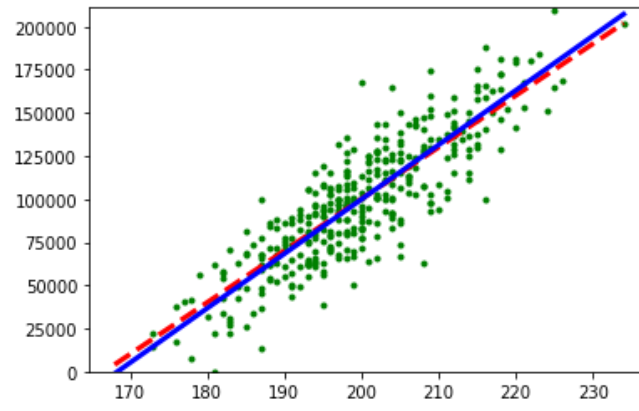
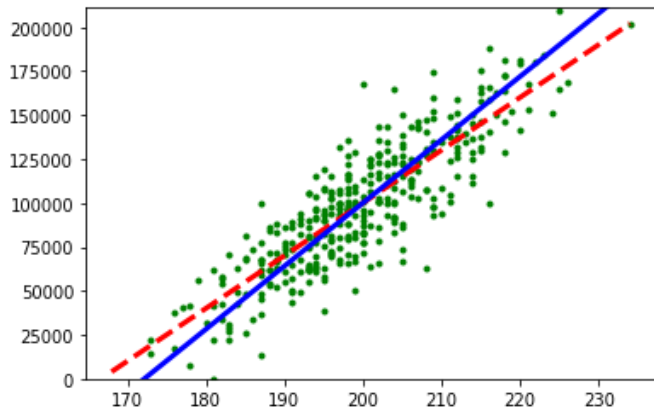
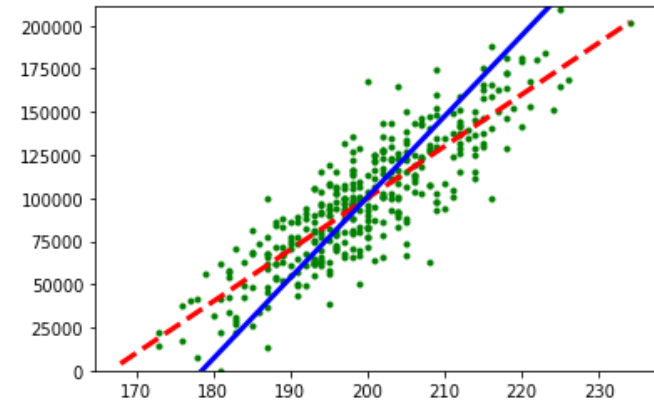
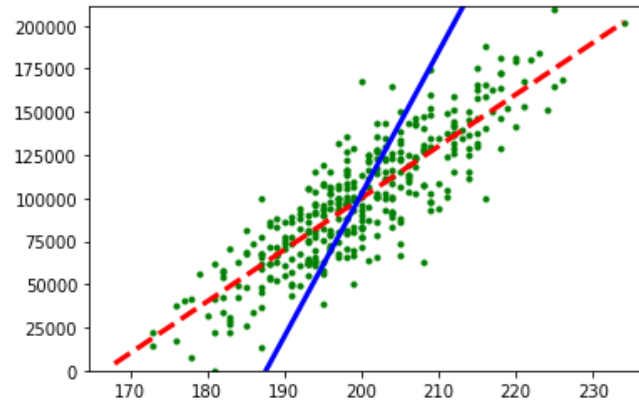
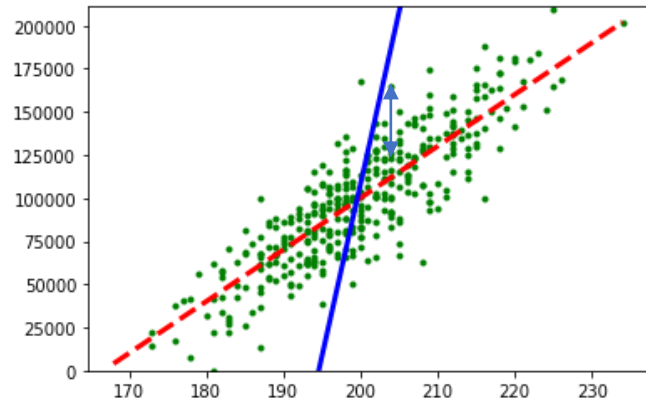
회귀 분석(Regression Analysis)



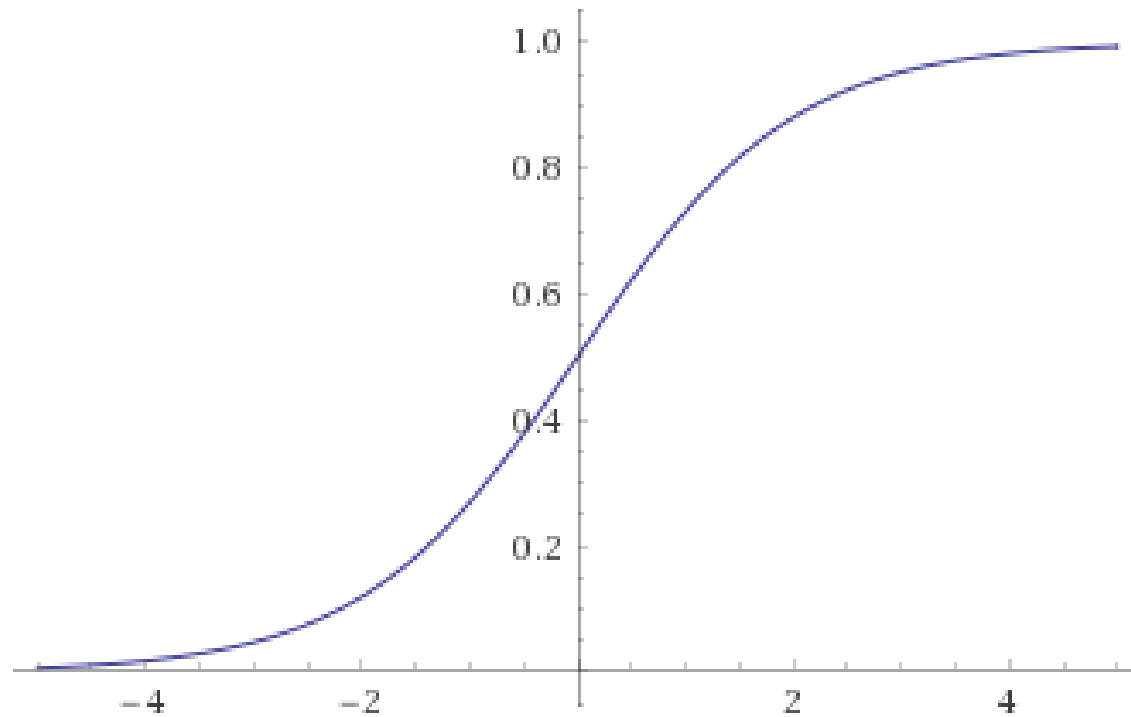
오차(Error)



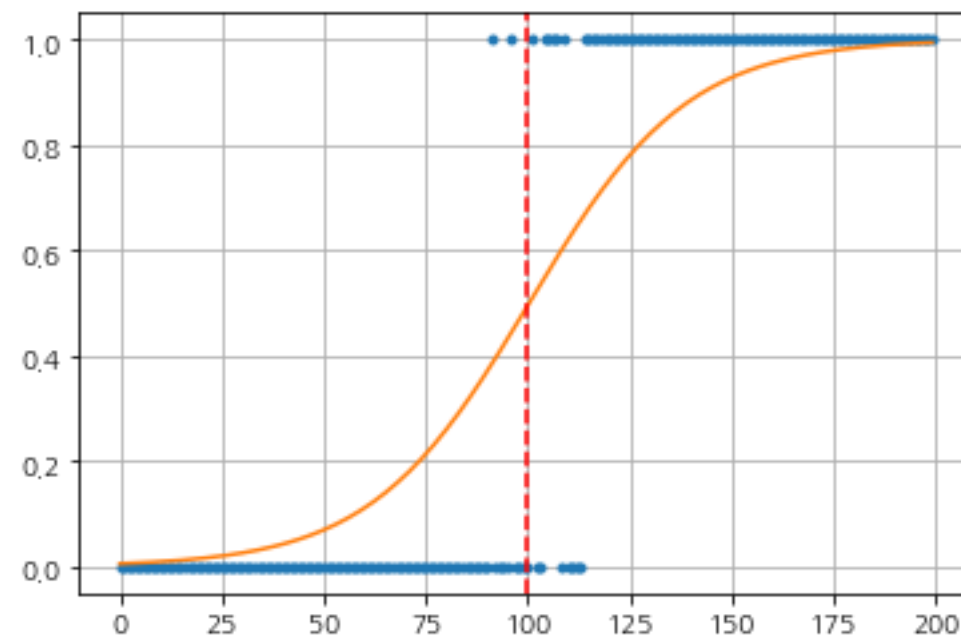
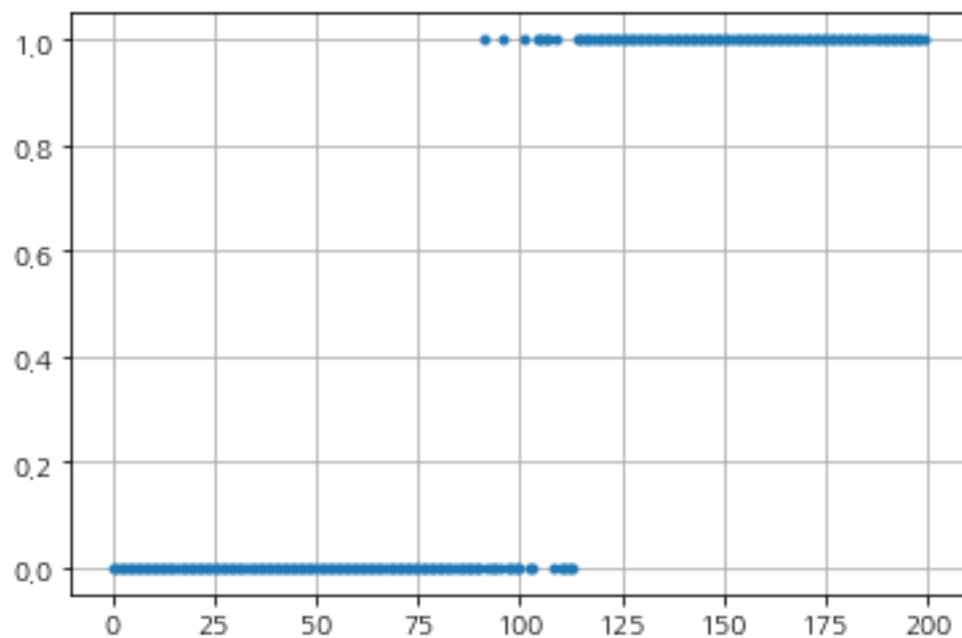
회귀 분석(Regression Analysis)



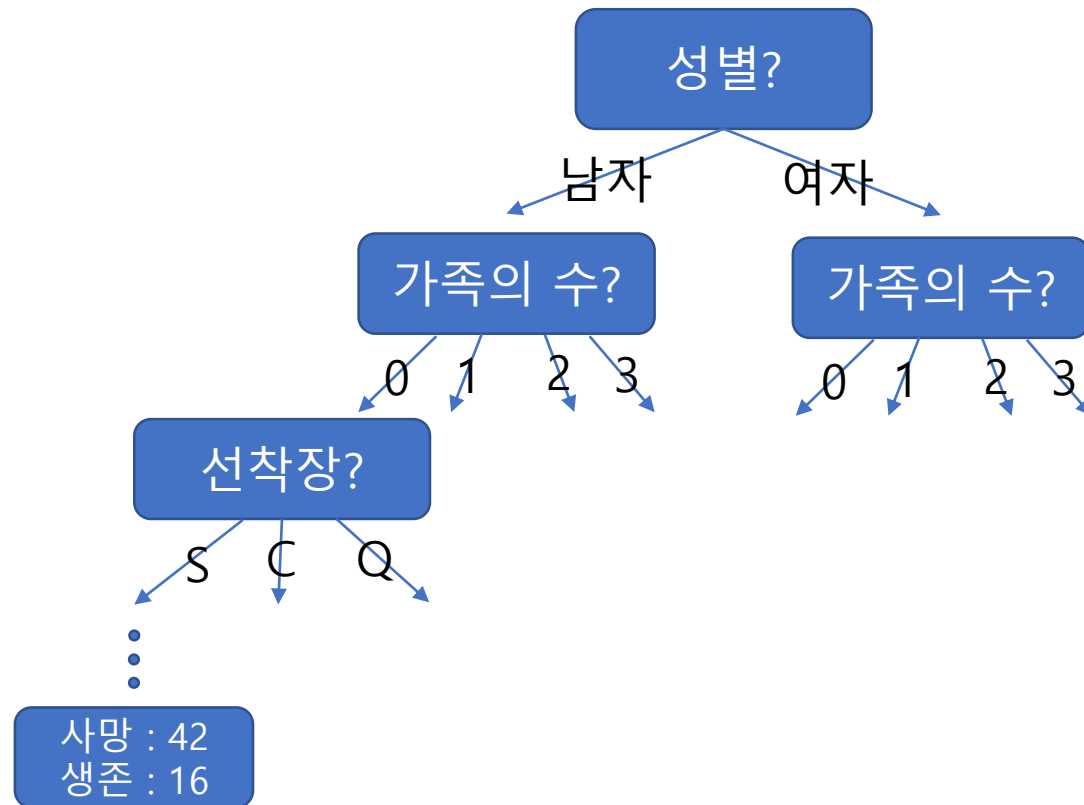
로지스틱 회귀(LogisticRegression)



로지스틱 회귀(LogisticRegression)



의사결정나무(DecisionTree)



sklearn 패키지 – 머신러닝 패키지

```
[77] from sklearn.linear_model import LogisticRegression  
      from sklearn.tree import DecisionTreeClassifier
```

```
[82] lr_model = LogisticRegression()  
      lr_model.fit(x_train, y_train)  
  
      lr_model.score(x_train, y_train)
```

```
0.7901234567901234
```

```
[83] dt_model = DecisionTreeClassifier()  
      dt_model.fit(x_train, y_train)  
  
      dt_model.score(x_train, y_train)
```

```
0.8585858585858586
```

테스트셋 예측 – model.predict()

```
[84] dt_model.predict(test_)
```

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

```
[85] submission['Survived'] = dt_model.predict(test_)
      submission
```

```
PassengerId  Survived
```

```
0            892        0
```

```
1            893        0
```

```
2            894        0
```

```
3            895        0
```

```
4            896        0
```

```
...          ...        ...
```

```
413          1305        0
```

```
414          1306        1
```

```
415          1307        0
```

```
416          1308        0
```

```
417          1309        0
```

```
418 rows × 2 columns
```

결과 저장 – pd.DataFrame.to_csv()

```
[84] dt_model.predict(test_)
```

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

```
[ ] submission.to_csv('/content/drive/My Drive/부산대/타이탄尼克/submission.csv', index=False)
```

```
0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1,
1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0,
1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1,
0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0,
1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0,
0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0,
1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,
0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0])
```

```
[85] submission['Survived'] = dt_model.predict(test_)
      submission
```

```
PassengerId  Survived
```

```
0           892        0
```

```
1           893        0
```

```
2           894        0
```

```
3           895        0
```

```
4           896        0
```

```
...         ...        ...
```

```
413         1305        0
```

```
414         1306        1
```

```
415         1307        0
```

```
416         1308        0
```

```
417         1309        0
```

```
418 rows x 2 columns
```

생존 확률 model.predict_proba()

```
pred_proba = dt_model.predict_proba(test_)  
pd.DataFrame(pred_proba)
```

	0	1
0	1.000000	0.000000
1	1.000000	0.000000
2	1.000000	0.000000
3	0.896552	0.103448
4	0.500000	0.500000

...
413	0.896552	0.103448
414	0.000000	1.000000
415	0.783784	0.216216
416	0.896552	0.103448
417	0.250000	0.750000

418 rows × 2 columns

```
pd.DataFrame(pred_proba)[1]
```

0	0.000000
1	0.000000
2	0.000000
3	0.103448
4	0.500000

...

416 0.103448

417 0.750000

Name: 1, Length: 418, dtype: float64

```
[65] submission.to_csv('/content/drive/My Drive/부산대/타이타닉/submission_proba.csv', index=False)
```

```
submission['Survived'] = pred_proba[:,1]  
submission
```

	PassengerId	Survived
--	-------------	----------

0	892	0.000000
1	893	0.000000

4	896	0.500000
---	-----	----------

...

413	1305	0.103448
-----	------	----------

414	1306	1.000000
-----	------	----------

415	1307	0.216216
-----	------	----------

416	1308	0.103448
-----	------	----------

417	1309	0.750000
-----	------	----------

418 rows × 2 columns

Confusion Matrix

- True(생존)을 True(생존)으로 예측(TP)
- True(생존)을 False(사망)으로 예측(FN)
- False(사망)을 True(생존)으로 예측(FP)
- False(사망)을 False(사망)으로 예측(TN)

	생존 예측	사망 예측
생존 (30)	20 TP	10 FN
사망 (70)	50 FP	20 TN

Confusion Matrix를 이용한 평가 지표

- 정확도(Accuracy) – True를 True로 False를 False로 얼마나 맞추는가
- 정밀도(Precision) – 모델이 True로 분류한 결과를 믿을 수 있는가
- 재현도(Recall) – 모델이 True를 얼마나 잘 찾는가

Confusion Matrix를 이용한 평가 지표

		Predicted class	
		<i>P</i>	<i>N</i>
Actual Class	<i>P</i>	True Positives (TP)	False Negatives (FN)
	<i>N</i>	False Positives (FP)	True Negatives (TN)

- Accuracy(정확도) =
$$\frac{TP + TN}{TP + TN + FP + FN}$$

- Precision(정밀도) =
$$\frac{TP}{TP + FP}$$

- Recall(재현도) =
$$\frac{TP}{TP + FN}$$

양성률과 음성률

		Predicted class	
		<i>P</i>	<i>N</i>
Actual Class	<i>P</i>	True Positives (TP)	False Negatives (FN)
	<i>N</i>	False Positives (FP)	True Negatives (TN)

True Positive Rate = $\frac{TP}{TP + FN}$
(1을 맞춘 비율, TPR)

False Positive Rate = $\frac{FP}{FP + TN}$
(0을 틀린 비율, FPR)

ROC곡선과 AUC

- True 판정 기준(확률 임계값) 변화에 따른 양성률과 음성률의 변화

$$Y(p) = \begin{cases} 0 & (p < 0.5) \\ 1 & (p \geq 0.5) \end{cases}$$

- 평가에 유리한 것은?
 1. 0, 1로 제출
 2. 확률로 제출

