

# Spring Day 12

## ▼ Spring

### CRUD

#### ▼ CRUD(책 상세 보기)

##### book\_SQL.xml

```
<!-- insert : MyBatis의 데이터 조회 태그 -->
<!-- resultType : 조회한 데이터의 반환타입 -->
<!-- 객체로 데이터를 전송하거나 조회할 수 있다. (ex. parameterType="com.spring.controller.BookDTO" resultType="com.spring.controller.BookDTO") -->
<select id="select_detail" parameterType="hashMap" resultType="hashMap">
  <![CDATA[
    select * from book where book_id = #{bookId}
  ]]>
</select>
```

##### BookRepository.java

```
Map<String, Object> selectDetail(Map<String, Object> map);
```

##### BookRepositoryImpl.java

```
@Override
public Map<String, Object> selectDetail(Map<String, Object> map) {
    //selectOne : 쿼리의 결과 행 수가 0개면 null을 반환, 결과가 여러 개면 예외를 발생시킨다(Mapper.xml의 resultType과 일치시켜야 한다.)
    return this.sqlSessionTemplate.selectOne("book.select_detail", map);
}
```

##### BookService.java

```
Map<String, Object> detail(Map<String, Object> map);
```

##### BookServiceImpl.java

```
@Override
public Map<String, Object> detail(Map<String, Object> map) {
    return this.bookRepository.selectDetail(map);
}
```

##### BookController.java

```
@RequestMapping(value="/detail", method=RequestMethod.GET)
public ModelAndView detail(@RequestParam Map<String, Object> map) {
```

```

    Map<String, Object> detailMap = this.bookService.detail(map);

    ModelAndView mav = new ModelAndView();
    mav.addObject("data", detailMap);
    String bookId = map.get("bookId").toString();
    mav.addObject("bookId", bookId);
    mav.setViewName("/book/detail");
    return mav;
}

```

## detail.jsp

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>책 상세 보기 </title>
</head>
<body>
<h1> 책 상세 보기 </h1>

<p>제목 : ${data.title}</p>
<p>종류 : ${data.category}</p>
<p>가격 : ${data.price}</p>
<p>날짜 : ${data.insert_date}</p>
<br>
<br>
<p><a href="/update?bookId=${bookId}">수정</a></p>

</body>
</html>

```

## ▼ C R U D(책 수정하기)

### book\_SQL.xml

```

<update id="update" parameterType="hashMap" >

    UPDATE book SET title=#{title}, category=#{category}, price=#{price} WHERE book_id = #{bookId}

</update>

```

### BookRepository.java

```

int update(Map<String, Object> map);

```

### BookRepositoryImpl.java

```

@Override
public int update(Map<String, Object> map) {
    return this.sqlSessionTemplate.update("book.update", map);
}

```

### BookService.java

```

boolean update(Map<String, Object> map);

```

## BookServiceImpl.java

```
@Override
public boolean update(Map<String, Object> map) {

    int affectRowCount = this.bookRepository.update(map);

    return affectRowCount == 1;

}
```

## BookController.java

```
@RequestMapping(value="/update", method=RequestMethod.GET)
public ModelAndView update(@RequestParam Map<String, Object> map) {
    Map<String, Object> detailMap = this.bookService.detail(map);

    ModelAndView mav = new ModelAndView();
    mav.addObject("data", detailMap);
    String bookId = map.get("bookId").toString();
    mav.addObject("bookId", bookId);
    mav.setViewName("/book/update");
    return mav;
}

@RequestMapping(value="/update", method=RequestMethod.POST)
public ModelAndView updatePost(@RequestParam Map<String, Object> map) {
    ModelAndView mav = new ModelAndView();

    Boolean isUpdateSuccess = this.bookService.update(map);

    if(isUpdateSuccess) {
        String bookId = map.get("bookId").toString();
        mav.setViewName("redirect:/detail?bookId=" + bookId);
    }else {
        mav = this.update(map);
    }

    return mav;
}
```

## update.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title> 책 수정하기 </title>
</head>
<body>
<h1> 책 수정하기 </h1>

<form action="/update" method="post">
<p>제목 : <input type="text" name="title" value="${data.title }">
<p>종류 : <input type="text" name="category" value="${data.category }">
<p>가격 : <input type="text" name="price" value="${data.price }">
<input type="hidden" name="bookId" value="${bookId }">
<p><input type="submit" value="수정">

</form>

</body>
</html>
```

## ▼ C R U D(도서목록)

### book\_SQL.xml

```
<select id="select_list" parameterType="hashMap" resultType="hashMap">
<![CDATA[
    select * from book ORDER BY insert_date desc
]]>
</select>
```

### BookRepository.java

```
List<Map<String, Object>> selectList(Map<String, Object> map);
```

### BookRepositoryImpl.java

```
@Override
public List<Map<String, Object>> selectList(Map<String, Object> map) {
    return this.sqlSessionTemplate.selectList("book.select_list", map);
}
```

### BookService.java

```
List<Map<String, Object>> list(Map<String, Object> map);
```

### BookServiceImpl.java

```
@Override
public List<Map<String, Object>> list(Map<String, Object> map) {
    return this.bookRepository.selectList(map);
}
```

### BookController.java

```
@RequestMapping(value="/list")
public ModelAndView list(@RequestParam Map<String, Object> map) {

    List<Map<String, Object>> list = this.bookService.list(map);

    ModelAndView mav = new ModelAndView();
    mav.addObject("data", list);
    mav.setViewName("/book/list");
    return mav;
}
```

### list.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
```

```

<title>도서목록</title>
</head>
<body>
<h1>도서목록</h1>

<table>
<thead>
<tr>
<th>제목</th>
<th>카테고리</th>
<th>가격</th>
</tr>
</thead>

<tbody>
<c:forEach var="row" items="${data}">
<tr>
<td><a href="/detail?bookId=${row.book_id}">${row.title}</a></td>
<td>${row.category}</td>
<td>${row.price}</td>
</tr>
</c:forEach>
</tbody>
</table>

<br><br>

<a href="/create">도서등록</a>

</body>
</html>

```

## ▼ CRUD(도서삭제)

### book\_SQL.xml

```

<delete id="delete" parameterType="hashMap">

    DELETE FROM book WHERE book_id = #{bookId}

</delete>

```

### BookRepository.java

```

int delete(Map<String, Object> map);

```

### BookRepositoryImpl.java

```

@Override
public int delete(Map<String, Object> map) {
    return this.sqlSessionTemplate.delete("book.delete", map);
}

```

### BookService.java

```

boolean remove(Map<String, Object> map);

```

### BookServiceImpl.java

```

@Override
public boolean remove(Map<String, Object> map) {
    int affectRowCount = this.bookRepository.delete(map);
    return affectRowCount == 1;
}

```

## BookController.java

```

@RequestMapping(value="/delete", method=RequestMethod.POST)
public ModelAndView deletePost(@RequestParam Map<String, Object> map) {
    ModelAndView mav = new ModelAndView();

    Boolean isDeleteSuccess = this.bookService.remove(map);

    if(isDeleteSuccess) {
        mav.setViewName("redirect:/list");
    }else {
        String bookId = map.get("bookId").toString();
        mav.setViewName("redirect:/detail?bookId=" + bookId);
    }

    return mav;
}

```

## detail.jsp

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>책 상세 보기 </title>
</head>
<body>
<h1> 책 상세 보기 </h1>

<p>제목 : ${data.title}</p>
<p>종류 : ${data.category}</p>
<p>가격 : ${data.price}</p>
<p>날짜 : ${data.insert_date}</p>
<br>
<br>
<p><a href="/update?bookId=${bookId}">수정</a> <a href="/list">목록</a></p>
<form action="/delete" method="POST"><input type="hidden" name="bookId" value="${bookId}"><input type="submit" value="삭제"></form>

</body>
</html>

```

## ▼ CRUD(검색 기능)

### book\_SQL.xml(Dynamic Query)

```

<!-- <select id="select_list" parameterType="hashMap" resultType="hashMap"> -->
<!-- <![CDATA[ -->
<!-- select * from book ORDER BY insert_date DESC -->
<!-- ]]> -->
<!-- </select> -->

<select id="select_list" parameterType="hashMap" resultType="hashMap">
<![CDATA[
select * from book WHERE 1=1
]]>
<if test="keyword != null and keyword != ''">
and (title like CONCAT('%',{keyword},%') or category like CONCAT('%',{keyword},%'))
</if>

```

```
ORDER BY insert_date DESC

</select>
```

## Dynamic Query

상황에 따라 분기 처리를 통해 SQL을 동적으로 만드는 쿼리

### WHERE 1=1

검색 조건을 AND로 연결하기 위해 사용한다(관습적인 표현)

```
<if test="keyword != null and keyword != "">
```

MyBatis의 if문이다(test : 조건식)

```
like CONCAT('%',{keyword},'%')
```

키워드를 포함하는 결과 찾기

ex) like '김%' : 김하나, 김돌, 김셋

ex) like '%김' : 맛김, 조미김, 양반김

ex) like '%김%' : 참치김밥, 치즈김밥, 소고기김밥

## BookController.java

```
// @RequestMapping(value="/list")
// public ModelAndView list(@RequestParam Map<String, Object> map) {
//
//     List<Map<String, Object>> list = this.bookService.list(map);
//
//     ModelAndView mav = new ModelAndView();
//     mav.addObject("data", list);
//     mav.setViewName("/book/list");
//     return mav;
// }

@RequestMapping(value="/list")
public ModelAndView list(@RequestParam Map<String, Object> map) {

    List<Map<String, Object>> list = this.bookService.list(map);

    ModelAndView mav = new ModelAndView();
    mav.addObject("data", list);

    if(map.containsKey("keyword")) {
        mav.addObject("keyword", map.get("keyword"));
    }

    mav.setViewName("/book/list");
    return mav;
}
```

## list.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
```

```

<html>
<head>
<meta charset="UTF-8">
<title>도서목록</title>
</head>
<body>
<h1>도서목록</h1>

<form>
<p>
<input type="text" placeholder="검색" name="keyword" value="${keyword}">
<input type="submit" value="검색">
</p>
</form>

<table>
<thead>
<tr>
<th>제목</th>
<th>카테고리</th>
<th>가격</th>
</tr>
</thead>

<tbody>
<c:forEach var="row" items="${data}">
<tr>
<td><a href="/detail?bookId=${row.book_id}">${row.title}</a></td>
<td>${row.category}</td>
<td>${row.price}</td>
</tr>
</c:forEach>
</tbody>
</table>

<br><br>

<a href="/create">도서등록</a>

</body>
</html>

```

## <![CDATA[ ]]>

### Mybatis 에서 CDATA 사용하기

마이바티스에서는 CDATA 구문을 많이 씁니다 이유는 쿼리문에 문자열 비교연산자나 부등호를 처리할 때가 있습니다 그러면 < 와 같은 기호를 괄호인지 아니면 비교연산자 인지 확인이 되지않아요 이외에도 특수문자 사용하는데 제한이있습니다 그런이유에서 하나의 규칙같이 부등호가 없는 쿼리문에도 전부CDATA를 쓰는곳도 많습니다

👉 <https://epthffh.tistory.com/entry/Mybatis-에서-CDATA-사용하기>



## WHERE 1=1을 사용하는 이유

### [MYSQL] WHERE 1=1 사용하는 이유?? 주의사항으로는??

안녕하세요, 오늘은 MYSQL 질의문에서 WHERE 1=1을 사용하는 이유에 대해서 알아보도록 하겠습니다. 다른 개발자가 개발한 프로젝트를 유지 보수하거나, 처음 개발자로 입문하여 선임들이 작성한 쿼리문을 보면 WHERE 1=1 조건문을 볼 수 있는데요, 어떤 이유로 사용하는지 한번 알아보시다. 1. WHERE 1=1 참(true)을 의미 말그대

👉 <https://ssd0908.tistory.com/entry/MYSQL-WHERE-11-사용하는-이유-주의사항으로는>

**WHERE 1=1  
사용하는 이유**

## ▼ Python



# Titanic

## 타이타닉 데이터 분석

```
# 타이타닉

# 1. 전처리
# 2. 데이터분석
# 3. 인공지능(머신러닝)
# 4. 예측시스템 (ex. 디카프리오, 21, 남성, 3등석 >> 생존률)
# 5. 연예인 닮은 꼴 찾기 API

# 가족과 함께 탑승한 1등실 21세 남성의 생존 확률은 얼마일까?
# 홀로 탑승한 3등실 20세 여성의 생존 확률은 얼마일까?

import pandas as pd          # 계산 관련 모듈
import numpy as np           # 분석 관련 모듈
import matplotlib.pyplot as plt # 기본 그림 관련 모듈
import seaborn as sns        # 고급 그림 관련 모듈

# !pip install xlrd

raw_data = pd.read_excel('titanic.xlsx') # 판다스로 엑셀 파일 읽어오기
raw_data.info()                         # 데이터프레임의 대략적인 정보 확인하기

# 각 데이터열의 의미 확인
# pclass 선실의 등급 1등실이 가장 비싸고 3등실이 가장 저렴
# survived 생존여부 1이 생존 0이 사망
# age 나이
# sibsp 함께 탑승한 형제자매나 배우자의 수
# parch 함께 탑승한 부모 또는 자녀의 수
# fare 탑승 요금

# 데이터 살펴보기

raw_data

raw_data.describe() # 데이터프레임의 대략적인 통계적 정보 확인하기(숫자로 이루어진 열만 확인가능)
# 나이의 경우 80세가 최고령이고 가장 어린나이는 0.16살. 즉, 갓난아이라도 탑승했다.

f,ax = plt.subplots(1,2,figsize=(12,6))
# matplotlib 로 한번에 2개의 그래프를 표현.
# 1, 2는 1행 2열 의미하고 figsize=(12,6) 가로 * 세로의 크기를 의미한다.
# f와 ax 전체 기본 프레임과 그 위에 올라가는 그림을 나타낸다.

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

# raw_data 에 있는 survived(생존) 정보를 파이 그래프로 그린다.
# value_counts() : 1(생존 수)와 0(사망 수)으로 표시
# explode=[0,0.1] : 값 사이의 여백 설정(파이가 떨어지게)
# autopct='%1.2f%%' : 1과 0의 퍼센트 값을 계산하여 2자리의 소수점 아래까지 % 를 붙여서 표현
# ax=ax[0] 2개의 자리중 1번째(왼쪽)에 위치

ax[0].set_title('Survived') # 제목 설정
ax[0].set_ylabel('')        # 라벨 제거

# 여기서부터는 오른쪽 2번째 그림
sns.countplot(x='survived', data=raw_data, ax=ax[1]) # sns seaborn 으로 그림을 표현
ax[1].set_title('Survived') # 제목
plt.show()

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

# 전체 탑승객들의 나이 정보로 만든 히스토그램.
# 가장 많은 인원수는 20대 초반으로 보인다.

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

# groupby 를 사용하여 선실의 등급의 평균을 살펴 본다.
# 가장 비싼 1등실의 생존률이 61%로 상당히 높은 것을 볼 수 있다.

raw_data.corr()

# corr() 함수는 상관계수를 계산해 주는 함수이다. 각 열들의 상호 상관계수를 확인할 수 있다.

plt.figure(figsize=(10, 10))
# 기본 틀

sns.heatmap(raw_data.corr(), linewidths=0.01, square=True,
            annot=True, cmap=plt.cm.viridis, linecolor="white")

# seaborn 의 heatmap.
```

```

# 데이터는 raw_data.corr()
# "annot=True" : 숫자 표시
# cmap : 컬러 색상

plt.title('Correlation between features')
# 그림의 제목 설정

plt.show()

# 상관계수는 두 데이터의 상관관계만을 계산한 것이다(인과관계와 다르다.)
# 선실 등급이 낮을수록 생존율이 높다는 것(~31%)과
# 요금이 높을수록 생존율이 높다는 것을 쉽게 그림으로 확인할 수 있다.

raw_data['age_cat'] = pd.cut(raw_data['age'],
                             bins=[0, 3, 7, 15, 30, 60, 100],
                             include_lowest=True,
                             labels=['baby', 'children', 'teenage',
                                      'young', 'adult', 'old'])

raw_data.head()

# 나이등급(열)을 하나 새롭게 만들어서 관찰해 보자.
# Pandas 에 있는 cut 함수를 사용하여 나이 구분 기준을 만들어 각각 라벨을 붙였다.
# 0 ~ 3살 : baby
# 3 ~ 7살 : children
# "include_lowest=True" : 제일 작은 값이 '초과'의 개념에서 빠지지 않게 포함 시키는 옵션("include_lowest=False"라면 1 ~ 3살이 baby 가 된다.)

raw_data.groupby('age_cat').mean()

# baby 등급의 경우 pclass 평균이 2.58이다. 즉, 3등실에 많이 있었음을 보여준다.
# 따라서 최종 결과를 보면 baby 일수록 생존율이 높다는 것을 알 수 있다.
# adult ~ old 로 갈수록 1등실에 많이 탑승하였지만 생존율이 낮아짐을 볼 수 있다.

plt.figure(figsize=[14,4])
plt.subplot(131)
sns.barplot(x='pclass', y='survived', data=raw_data)
plt.subplot(132)
sns.barplot(x='age_cat', y='survived', data=raw_data)
plt.subplot(133)
sns.barplot(x='sex', y='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()

# 나이가 어릴수록 생존율이 높은 것을 볼 수 있다.
# 특히 여성의 생존율이 월등히 높은 것을 볼 수 있다.

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

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

# 여성 탑승객 수는 약 450명, 남성 탑승객 수는 약 850명 정도이다.
# 여성이 남성의 절반 정도이다.
# 그런데 생존자 수를 살펴보면 여성의 월등히 많이 생존하였음을 볼 수 있다.

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

# 보트 번호가 있으면 구명 보트에 탑승한 사람이다.
# 결측치는 제외(NaN이 아닌 항목만 선택)한 데이터이다.

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

# 구명 보트에 탑승한 사람은 생존율이 98.15% 이다.
# 오히려 보트 함몰은 인공지능을 학습 시킬때 포함하면 안된다(분별력이 떨어지기 때문이다.)

# 사회적 지위에 따른 생존율
raw_data['name']

raw_data['name'][0]
# name 열에는 이름뿐 아니라 사회적 지위도 포함되어 있다.

raw_data['name'][0].split(',')[1]
# 항상 두번째 단어가 사회적 지위이기때문에

raw_data['name'][0].split(',')[1].split('.')[0]

```

```

# 심표 뒤부터 마침표 앞까지가 사회적 지위를 나타낸다.

raw_data['name'][0].split(',')[1].split('.')[0].strip()
# 빈칸까지 제거하여 한사람의 사회적 지위를 추출하였다.

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

# unique() : 데이터 안의 고유값들을 알고 싶을 때 사용하는 함수.
titles = raw_data['title'].unique()
titles

# 이름에서 분리한 사회적 지위만으로 새로운 열(title)을 만들었다.

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

# 각 지위별 남자와 여자 수.
# 판다스의 크로스탭(crosstab)은 지정된 컬럼을 인수로 하여 두번째 지정된 컬럼에 대한 개수를 알려준다.

raw_data['title'] = raw_data['title'].replace('Mlle', 'Miss')
# Mlle 를 Miss로 변경하고

raw_data['title'] = raw_data['title'].replace('Ms', 'Miss')
# Ms 를 Miss로 변경하고

raw_data['title'] = raw_data['title'].replace('Mme', 'Mrs')
# Mme 를 Mrs로 변경했다.

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()

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

raw_data['sex']

# 데이터 형이 숫자가 아니면 모델링을 할 수 없다. 따라서 성을 숫자로 변경해야 한다.

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

# 여성은 0으로
# 남성은 1로
# 그 외는 NaN으로 바꾼다.

raw_data['sex'] = tmp
raw_data['sex']

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()

raw_data.info()

# 머신러닝 알고리즘 적용시에는 반드시 결측치가 없어야 한다.
# 결측치를 제거하는 방법은 삭제, 대체 등이 있다.

# 결측치 제거 방법중 삭제를 적용.
# 1309개 -> 1045개

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()

```

## 예측 모델 만들기

```

# 예측 모델 만들기

train_pre = raw_data[['pclass', 'sex', 'age', 'sibsp', 'parch', 'fare']]
train_pre.head()

# 예측 모델을 만들 때는 모든 열값을 사용할 필요는 없다.
# 따라서 [문자로 이루어진 열] 또는 [숫자라도 변별력에 문제가 있는 열]은 제외한다.

# 전체 데이터를 학습용(training)과 검증용(test)으로 나누어야 한다.
# ex) 전체 데이터 1000개, 학습용 900개, 검증용 100개 [9:1]

# !pip install scikit-learn

from sklearn.model_selection import train_test_split
# 전체 데이터를 나누어주는 모듈

X_train, X_test, y_train, y_test = train_test_split(train_pre,
                                                    raw_data[['survived']],
                                                    test_size=0.1,
                                                    random_state=13)

# "random_state=13" : 재현가능(for reproducibility)하도록 난수의 초기값을 설정해준다(아무 숫자나 넣어주어도 된다.)

# 90%
# X_train 훈련용 문제지 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare'
# y_train 훈련용 답안지 survived

# 10%
# X_test 검증용 문제지 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare'
# y_test 검증용 답안지 survived

X_train.info()

y_train.info()

X_test.info()

y_test.info()

y_train.head()

# random 으로 섞여있기 때문에 인덱스가 엉켜 있다.

# 인덱스 재조정하기

X_train = X_train.reset_index()
X_train = X_train.drop(['index'], axis=1)

X_test = X_test.reset_index()
X_test = X_test.drop(['index'], axis=1)

y_train = y_train.reset_index()
y_train = y_train.drop(['index'], axis=1)

y_test = y_test.reset_index()
y_test = y_test.drop(['index'], axis=1)

from sklearn.tree import DecisionTreeClassifier
# sklearn 의 DT(결정나무 알고리즘)로 모델을 생성한다.

tree_clf = DecisionTreeClassifier(max_depth=3, random_state=13)
# 객체 생성시 옵션으로 깊이를 3으로 하고 난수의 초기값을 설정해준다.

tree_clf.fit(X_train, y_train) # 지도 학습
# 훈련용 데이터와 훈련용 답안지를 넣어 학습을 시킨다.

print('Score: {}'.format(tree_clf.score(X_train, y_train)))

# 윗줄에서 만든 모델에 훈련용 데이터를 넣어 결과를 예측한다.

# 훈련용 데이터로 만든 모델에 훈련용 데이터를 넣었을 때의 정확도는 81.1% 이다.

from sklearn.metrics import accuracy_score
# 정확도 계산 모듈

y_pred = tree_clf.predict(X_test)
# 위에서 생성한 모델에 10%의 검증용 문제지를 넣은 결과

print("Test Accuracy is ", accuracy_score(y_test, y_pred)*100)
# 그 결과와 10%의 검증용 답안지를 비교했을 때의 정확도

# !pip install graphviz

from sklearn.tree import export_graphviz

```

```

export_graphviz(
    tree_clf,
    out_file="titanic.dot",
    feature_names=['pclass', 'sex', 'age', 'sibsp', 'parch', 'fare'],
    class_names=['Unsurvived', 'Survived'],
    rounded=True,
    filled=True
)

import graphviz
with open("titanic.dot") as f:
    dot_graph = f.read()
dot = graphviz.Source(dot_graph)
dot.format = 'png'
dot.render(filename='titanic_tree', directory='images/decision_trees', cleanup=True)
dot

# pclass, sex, age, sibsp, parch, fare
dicaprio = [3., 1., 19., 0., 0., 5.]

tree_clf.predict_proba([dicaprio])

# [0.875, 0.125] 비생존확률, 생존확률

# pclass, sex, age, sibsp, parch, fare
dicaprio = [1., 1., 19., 0., 0., 5.]

tree_clf.predict_proba([dicaprio])

# [0.65467626, 0.34532374]

# pclass, sex, age, sibsp, parch, fare
winslet = [3., 0., 17., 1., 1., 100.]
tree_clf.predict_proba([winslet])

# [0.9047619, 0.0952381] 비생존확률, 생존확률

# pclass, sex, age, sibsp, parch, fare
winslet = [1., 0., 17., 1., 1., 100.]
tree_clf.predict_proba([winslet])


# [0.02419355, 0.97580645]

```

## Python 람다 표현식

파이썬 코딩 도장: 32.1 람다 표현식으로 함수 만들기

코딩 도장: 따라하기, 연습하기, 심사하기로 배우는 프로그래밍 철저입문

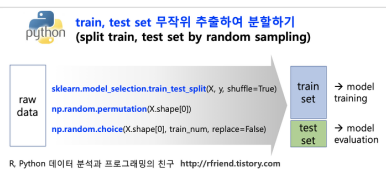
 <https://dojang.io/mod/page/view.php?id=2359>

## 데이터셋 분할하기

[Python numpy] Train, Test 데이터셋 분할하기 (split train and test set)

기계학습에서 모델을 학습하는데 사용하는 train set, 적합한 모델의 성능을 평가하는데 사용하는 test set 으로 나누어놓고 시작합니다. 이번 포스팅에서는 2차원 행렬 형태의 데이터셋을 무작위로 샘플링하여 Train set, Test set 으로 분할하는 방법을 소개하겠습니다. (1) scikit-learn 라이브러리 model\_selection 클래스의 train\_test\_split

🔗 <https://rfrriend.tistory.com/519>



R, Python 데이터 분석과 프로그래밍의 친구 <http://rfrriend.tistory.com>

## Graphviz

Download

Graph Visualization Software

 <https://graphviz.org/download/>

