

Spring Day 17

▼ Spring

Spring Security & 카카오 로그인

Spring Security와 카카오 로그인을 연동시키려면 여러가지 절차가 필요하다.

진행방식

1. 카카오 로그인 단추 클릭한다.
2. 카카오 API 로 연결한다.
3. 카카오 사이트에서 로그인 진행한다.
4. 카카오 로그인 성공시 데이터(이메일, 성별, 별명 등)를 넘겨받는다.
5. 넘어온 이메일을 DB에서 조회한다(AJAX)
6. DB에 있으면 Spring Security 로그인을 승인한다(Session 생성)
7. DB에 해당 이메일이 존재하지 않으면 회원가입을 진행한다.

login.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<script src="https://code.jquery.com/jquery-3.6.3.min.js" integrity="sha256-pvPw+upLPUjgMXy0G+800xUf+/Im1MZjXxxg0cBQBxU=" crossorigin="anonymous"></script>
<%@ page session="false" %>
<html>
<head>
<title>main</title>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<meta charset="utf-8">
</head>
<body class="text-center">
<!-- <div class="alert alert-dark" role="alert"> -->
<!-- <h1>로그인</h1> -->
<!-- </div> -->

<div class="container" style="width: 21%; padding-top: 200px;">
<main class="form-signin w-100 m-auto">
<form action="/login" method="post">

<h1 class="h3 mb-3 fw-normal">로그인</h1>

<div class="form-floating">
<input type="text" class="form-control" id="username" name="username" placeholder="ID">
<label for="floatingInput">ID</label>
</div>
<div class="form-floating">
<input type="password" class="form-control" id="password" name="password" placeholder="Password">
<label for="floatingPassword">Password</label>
</div>

<div class="checkbox mb-3">
<label>
<input type="checkbox" value="remember-me"> Remember me
</label>
</div>
```

```

</div>
<button class="w-100 btn btn-lg btn-primary" type="submit">로그인</button>
<input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}"/>
<br><br><br>
<a href="javascript:kakaoLogin()">

</a>

<p class="mt-5 mb-3 text-muted">&copy; 2017-2022</p>

</form>
</main>
</div>

</body>
<script src="https://developers.kakao.com/sdk/js/kakao.js"></script>
<script>

// 카카오 로그인
Kakao.init("f94caa3f14bb97c5a461377e09303edb");

function kakaoLogin() {
    window.Kakao.Auth.login({
        //카카오 로그인 성공시 넘겨받을 항목들(카카오 동의항목에서 설정한 값들과 동일해야 한다.)
        scope: 'profile_nickname, account_email, gender',
        success: function (authObj) {window.Kakao.API.request({url:'/v2/user/me',
            success:res => {
                const nickname = res.kakao_account.profile.nickname;
                const email = res.kakao_account.email;
                const gender = res.kakao_account.gender;

                console.log(nickname);
                console.log(email);
                console.log(gender);

                kakaoProcess(nickname, email, gender);
            }
        }}}
    })
}

function kakaoProcess(nickname, email, gender) {
    $.ajax({
        type:"POST",
        url:"/users/kakao",
        data:{nickname: nickname,
            email: email,
            gender: gender
        },
        beforeSend : function(xhr){
            xhr.setRequestHeader("${_csrf.headerName}", "${_csrf.token}");
        },
        success:function(result) {
            alert("카카오 로그인 성공, 메인화면으로 이동합니다.");
            window.location.assign('/');
        },
        error:function(request, status, error) {
            // alert(request.status + " " + request.responseText);
            alert("카카오 로그인 실패, 사이트 회원가입을 하셔야 합니다.");
            window.location.assign('/users/joinkakao?email=' + email);
        }
    })
}
</script>

</html>

```

joinkakao.jsp

```

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>

<head>
<title>회원가입</title>

```

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
    <meta charset="utf-8">

</head>
<body class="text-center">
<div class="alert alert-dark" role="alert">
<div class="container"><h1>회원가입</h1>

    <form:form modelAttribute="NewUser" action="./joinkakao?${_csrf.parameterName}=${_csrf.token}" class="form-horizontal" enctype="multipart/form-data">
        <fieldset>
            <legend>
                ${addTitle }
            </legend>
            ID : <form:input path="username" class="form-control" value="${user.username}" /><span style="color: red;">카카오 로그인을 통해 회원가입이 가능합니다.</span>
            password : <form:input path="password" type="password" class="form-control" />
            name : <form:input path="uname" class="form-control" />
            email : <form:input path="uemail" class="form-control" />

            <input type="submit" class="btn btn-primary" value="회원가입">

        </fieldset>
    </form:form>

</div>
</div>

</body>
</html>

```

UserRepository.java

```
User existUsername(String email);
```

UserRepositoryImpl.java

```

@Override
public User existUsername(String email) {
    return this.sqlSessionTemplate.selectOne("user.select_email", email);
}

```

UserService.java

```
User existUsername(String email);
```

ServiceImpl.java

```

@Override
public User existUsername(String email) {
    return userRepository.existUsername(email);
}

```

UsersController.java

```
package com.carshop.users;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.context.SecurityContext;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.web.context.HttpSessionSecurityContextRepository;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;

@RequestMapping("users")
@Controller
public class UsersController {

    @Autowired
    UserService userService;

    @Autowired
    BCryptPasswordEncoder bCryptPasswordEncoder;

    @GetMapping("/join")
    public String joinForm(@ModelAttribute("NewUser") User user) {
        return "users/joinform";
    }

    @PostMapping("/join")
    public String submitForm(@ModelAttribute("NewUser") User user) {

        //Spring은 기본적으로 password를 암호화하여 DB에 등록해야만 로그인할 수 있도록 설정되어 있다.

        String encodedPassword = bCryptPasswordEncoder.encode(user.getPassword());
        user.setPassword(encodedPassword);
        userService.setNewUser(user);
        return "login";
    }

    @RequestMapping("/list")
    public String userList(Model model) {
        List<User> list = userService.getAllUserList();
        model.addAttribute("userList", list);
        return "users/list";
    }

    @PostMapping("/list")
    public void updateAuth(@RequestParam Map<String, Object> auth) {
        userService.updateAuth(auth);
    }

    @PostMapping("/updateEnabled")
    public String updateEnabled(@RequestParam Map<String, Object> enabled) {
        userService.updateEnabled(enabled);
        return "users/list";
    }

    @ResponseBody
    @RequestMapping("/removeUser")
    public void ajaxremoveUser(@RequestParam("username") String username) {
        userService.deleteUser(username);
    }

    @RequestMapping("/kakao")
    public String loginCheckKakao(HttpServletRequest req, @RequestParam Map<String, Object> auth) {
        String email = (String)auth.get("email");

        User user = this.userService.existUsername(email);
    }
}
```

```

if(user != null) {
    System.out.println("이미 회원가입한 고객입니다.");
    //DB에 존재하는 회원이므로 로그인 세션처리를 진행한다.
    //JSP 방식의 독자적인 세션처리가 아니라 Spring Security 방식의 세션처리를 해야한다.
    List<GrantedAuthority> list = new ArrayList<GrantedAuthority>();
    list.add(new SimpleGrantedAuthority("ROLE_USER"));

    SecurityContext sc = SecurityContextHolder.getContext();

    // user : 회원정보 객체, null : 비밀번호(회원정보 객체에 이미 암호화된 비밀번호가 존재하기 때문에 null로 처리해도 된다.), list : 권한 설정
    sc.setAuthentication(new UsernamePasswordAuthenticationToken(user, null, list));

    HttpSession session = req.getSession(true);

    session.setAttribute(HttpSessionSecurityContextRepository.SPRING_SECURITY_CONTEXT_KEY, sc);

    //AJAX를 사용하므로 올바른 return 값을 입력해 success 경로로 보내준다.
    return "users/list";
} else {
    System.out.println("회원가입 가능합니다.");
    //AJAX를 사용하므로 존재하지 않는 return 값을 입력해 error 경로로 보내준다.
    return "";
}

}

@GetMapping("/joinkakao")
public String joinkakaoForm(@RequestParam("email") String email, @ModelAttribute("NewUser") User user) {
    String username = email;
    user.setUsername(username);
    return "users/joinkakao";
}

@PostMapping("/joinkakao")
public String submitkakaoForm(@ModelAttribute("NewUser") User user) {

    //Spring은 기본적으로 password를 암호화하여 DB에 등록해야만 로그인을 할 수 있도록 설정되어 있다.

    String encodedPassword = bCryptPasswordEncoder.encode(user.getPassword());
    user.setPassword(encodedPassword);
    userService.setNewUser(user);
    return "login";
}
}

```

user_SQL.xml

```


<select id="select_email" parameterType="String" resultType="com.carshop.users.User">
<![CDATA[
    SELECT * FROM users WHERE username = #{email}
]]>
</select>

```

Spring Security 복호화가 불가능한 이유[단방향 암호화 & Bcrypt]

OKKY - Spring Security 복호화 질문드립니다.

안녕하세요 신입 개발자입니다 현재 토이 프로젝트 진행하면서 기존 세션을 통한 로그인하던걸 시큐리티로 변환하면서 시큐리티를 처음 사용하는데 암호화 방식을 써서 DB에 데이터를 넣고 사용자의 정보 변경기능에서 데이터베이스의 값을 가져오려면 matches를 사용하여 true일때 데이터 베이스 데이터를 넘겨받아서 출력하려고하는데 기존 비밀번호가 1234 였다면 모델로 넘

 <https://okky.kr/questions/1144216>

Spring Security[로그인 세션 생성하기]

[Spring Security] 로그인 폼 없이 수동으로 로그인 세션 생성

0. 들어가기 전에 일단적으로는 로그인을 위해서 아이디와 비밀번호를 입력하는 폼 형태가 있으며 로...

<https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=platinasnow&logNo=221398649213>



▼ Python

Machine Learning

보스턴 주택 가격 예측

```
# 보스턴 주택 가격 예측

import pandas as pd
import numpy as np
from sklearn import datasets
import matplotlib.pyplot as plt
import seaborn as sns

# 기존 내장 데이터 셋에 인종차별 요소가 포함되어 있어 [load_boston()]이 삭제되었다.
# housing = datasets.load_boston()

# 페이지에서 가져오기(보스턴 주택 가격 예측)
data_url = "http://lib.stat.cmu.edu/datasets/boston"
raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
target = raw_df.values[1::2, 2]

data = pd.DataFrame(data)

target = pd.DataFrame(target)

df = pd.concat([data, target], axis=1)

df.columns = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'LSTAT', 'B', 'Target']

df.corr()

plt.figure(figsize=(12,12))
sns.heatmap(df.corr(), annot=True)

df.corr().loc['LSTAT', 'Target'].abs().sort_values(ascending=False)

# 집 가격(Target)과 상관 관계가 높은 4개만 추출

plot_cols = ['Target', 'RM', 'PTRATIO', 'INDUS', 'TAX']

plot_df = df.loc[:, plot_cols]
plot_df
```

```
# 선형회귀선

plt.figure(figsize=(12, 12))
for idx, col in enumerate(plot_cols[1:]):
    ax1 = plt.subplot(2,2, idx+1)
    sns.regplot(x=col, y=plot_cols[0], data=plot_df, ax=ax1)

sns.displot(x='Target', data=df)
# 집값이 20만 달러인 데이터가 가장 많고 좌우 대칭 구조를 볼 수 있다.
# 집값이 50만 달러인 데이터도 많은 것을 확인할 수 있다.
```

피쳐 스케일링

```
# 피쳐 스케일링

# 피쳐(열)의 데이터들의 단위에 따른 상대적 영향력의 차이를 제거할 필요가 있다.
# 즉 피쳐(열)들의 단위를 비슷한 수준으로 맞춰주어야 한다.

# ex. 만약에 가격의 단위를 50,000,000원이라고 가정해 보면
# 계산을 할 때 가격의 단위 때문에 계산의 결과값을 왜곡시킬 수 있다.

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

df_scaled = df.iloc[:, :-1] # Target(집값)은 제외한다.
scaler.fit(df_scaled)
df_scaled = scaler.transform(df_scaled)

df.iloc[:, :-1] = df_scaled[:, :]
df

df.iloc[:, :-1]

# 학습용 데이터(80%)와 검증용 데이터(20%)로 분할하기

from sklearn.model_selection import train_test_split

X_data = df.loc[:, 'RM':'LSTAT']
y_data = df.loc[:, 'Target']

X_train, X_test, y_train, y_test = train_test_split(X_data, y_data,
                                                    test_size=0.2,
                                                    shuffle=True,
                                                    random_state=12)
```

Linear Regression(선형 회귀)

```
# Linear Regression(선형 회귀)

from sklearn.linear_model import LinearRegression

lr = LinearRegression()

lr.fit(X_train, y_train)

y_test_pred = lr.predict(X_test)

plt.scatter(X_test['LSTAT'], y_test, label='y_test')
plt.scatter(X_test['LSTAT'], y_test_pred, c='r', label='y_pred')
plt.legend()
```

모델 성능 평가

```
# 모델 성능 평가

# 예측값이 실제 데이터를 정확하게 설명한다고 보기 어렵다. 실제값과 예측값 사이의 오차가 발생하기 때문이다.
# 모델 성능 평가에서 이 실제값과 예측값(정답)의 차이. 즉, 잔차(residuals)를 이용할 수 있다.

# 잔차는 오차를 나타내는 지표이므로 잔차값이 작을수록 모델의 성능이 좋다.

# MSE (Mean Squared Error) : 잔차(실제값과 예측값의 차이)의 제곱을 평균한 값
# 잔차는 상황에 따라 양수 & 음수가 될 수 있다.
# 그렇기 때문에 부호 간 계산으로 잔차의 합이 상쇄되는 것을 방지하기 위하여 잔차를 제곱하여 합을 구한다.

# MAE (Mean Absolute Error): 잔차(실제값과 예측값의 차이)의 절대값을 평균한 값

from sklearn.metrics import mean_squared_error

y_train_pred = lr.predict(X_train)

train_mse = mean_squared_error(y_train, y_train_pred)
print("훈련용 데이터셋 MSE :", train_mse)
test_mse = mean_squared_error(y_test, y_test_pred)
print("검증용 데이터셋 MSE :", test_mse)
```

```
# 훈련용 데이터셋 MSE : 31.26948078799356
# 검증용 데이터셋 MSE : 25.5356642865603
```

```
# K-Fold

from sklearn.model_selection import cross_val_score

lr = LinearRegression()
mse_scores = -1 * cross_val_score(lr, X_train, y_train, cv=5,
                                  scoring='neg_mean_squared_error')

print('개별 폴드 MSE : ', mse_scores)
print('전체 평균 MSE : ', np.mean(mse_scores))
```

과대적합(Overfitting) & 과소적합(Underfitting)

과대적합 & 과소적합

과대적합(Overfitting) 모델은 학습에 사용한 데이터와 비슷한 데이터는 잘 예측하지만 새로운 데이터는 예측 능력이 낮아진다.
(훈련 데이터에 지나치게 적응해버리는 현상)

과소적합(Underfitting) 모델은 훈련 데이터의 특성을 파악하기 어려울 정도의 작은 수의 데이터로 학습을 시킬때 발생한다.

단항식이 아닌 다항식으로 선형 회귀식을 만들면 구조가 복잡해지게 되어 모델의 예측 능력을 높일 수 있다.

```
from sklearn.preprocessing import PolynomialFeatures
```

```
# 데이터 셋을 2차 함수식으로 변환한다.
pf = PolynomialFeatures(degree = 2)
X_train_poly = pf.fit_transform(X_train)
X_train_poly
```

2차식으로 변환된 데이터 셋을 선형회귀 모델에 넣어 학습시킨다.

```
lr = LinearRegression()
lr.fit(X_train_poly, y_train)

y_train_pred = lr.predict(X_train_poly)
train_mse = mean_squared_error(y_train, y_train_pred)
print("Train MSE:", train_mse)

X_test_poly = pf.fit_transform(X_test)
y_test_pred = lr.predict(X_test_poly)
test_mse = mean_squared_error(y_test, y_test_pred)
print("Test MSE:", test_mse)

# 훈련용 데이터셋 MSE : 23.37217126318056
# 검증용 데이터셋 MSE : 15.122177340747417
```

과대적합(Overfitting)의 예시

```
from sklearn.preprocessing import PolynomialFeatures

pf = PolynomialFeatures(degree = 10)
X_train_poly = pf.fit_transform(X_train)

lr = LinearRegression()
lr.fit(X_train_poly, y_train)

y_train_pred = lr.predict(X_train_poly)
train_mse = mean_squared_error(y_train, y_train_pred)
print("Train MSE:", train_mse)

X_test_poly = pf.fit_transform(X_test)
y_test_pred = lr.predict(X_test_poly)
test_mse = mean_squared_error(y_test, y_test_pred)
print("Test MSE:", test_mse)

# 10차 다항식(degree = 10)으로 전환한 결과, MSE가 급격히 증가하였다.
# 즉, 과대적합 상태로 신규 데이터에 대한 예측 능력이 현저히 낮아졌음을 의미한다.
```


KNN(K Nearest Neighbor)

```
# KNN(K Nearest Neighbor)

# KNN에서 적용 가능한 변수는 오직 K의 개수 뿐이다.
# K의 개수에 따라 모델의 예측 능력이 달라지므로
# 예측 능력이 가장 높은 최적의 K 값을 찾는 것이 가장 중요하다.
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
import pandas as pd
import pickle
import matplotlib.pyplot as plt
import seaborn as sns

with open('basketball_test.pkl', 'rb') as test_data:
    test = pickle.load(test_data)

with open('basketball_train.pkl', 'rb') as train_data:
    train = pickle.load(train_data)

train

train.shape[0]
```

```
# 최적의 K값 찾기

#최적의 K값을 찾기 위해 교차 검증을 수행할 K의 범위를 3부터 학습 데이터 절반까지 지정하기.

max_k_range = train.shape[0] // 2 # [80 // 2] : 80을 2로 나누었을 때, 몫의 값.

k_list = []

for i in range(3, max_k_range, 2): # 홀수 요소
    k_list.append(i)

# [3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39]

X_train = train[['3P', 'BLK', 'TRB']]
y_train = train[['Pos']]

scores = []

#교차 검증을 각 K값을 대상으로 수행해 검증 결과를 저장
for k in k_list:
    knn = KNeighborsClassifier(n_neighbors=k)
    score = cross_val_score(knn, X_train, y_train.values.ravel(),
                            cv = 10, scoring="accuracy") # [cv=10] : 폴드(fold)의 수
    scores.append(score.mean())
scores

plt.plot(k_list, scores)

# K값이 [7, 9] 일때의 정확도가 가장 높다(0.9375)
```

```
# 피처의 수가 2개일 때, 정확도 구하기

max_k_range = train.shape[0] // 2

k_list = []

for i in range(3, max_k_range, 2):
    k_list.append(i)

X_train = train[['3P', 'BLK']]
y_train = train[['Pos']]

scores = []

for k in k_list:
    knn = KNeighborsClassifier(n_neighbors=k)
    score = cross_val_score(knn, X_train, y_train.values.ravel(),
                            cv = 10, scoring="accuracy")
    scores.append(score.mean())
scores

plt.plot(k_list, scores)
```

```
# K값이 [3, 5, 7] 일때의 정확도가 가장 높다(0.975)
```

```
# 피처의 수가 1개일 때, 정확도 구하기['3P']

max_k_range = train.shape[0] // 2

k_list = []

for i in range(3, max_k_range, 2):
    k_list.append(i)

X_train = train[['3P']]
y_train = train[['Pos']]

scores = []

for k in k_list:
    knn = KNeighborsClassifier(n_neighbors=k)
    score = cross_val_score(knn, X_train, y_train.values.ravel(),
                            cv = 10, scoring="accuracy")
    scores.append(score.mean())
scores

plt.plot(k_list, scores)

# K값이 [13, 15, 17] 일때의 정확도가 가장 높다(0.9)
```

```
# 피처의 수가 1개일 때, 정확도 구하기['BLK']

max_k_range = train.shape[0] // 2

k_list = []

for i in range(3, max_k_range, 2):
    k_list.append(i)

X_train = train[['BLK']]
y_train = train[['Pos']]

scores = []

for k in k_list:
    knn = KNeighborsClassifier(n_neighbors=k)
    score = cross_val_score(knn, X_train, y_train.values.ravel(),
                            cv = 10, scoring="accuracy")
    scores.append(score.mean())
scores

plt.plot(k_list, scores)

# 전반적으로 정확도가 높다(0.975)
```

```
# [7, 9]
# [3, 5, 7]
# [13, 15, 17]

# 지금까지 다양한 파라미터 적용으로 찾은 최적의 K값은 7이다.

knn = KNeighborsClassifier(n_neighbors=7)
X_train = train[['3P', 'BLK']]
y_train = train[['Pos']]

score = cross_val_score(knn, X_train, y_train.values.ravel(),
                        cv = 10, scoring="accuracy")

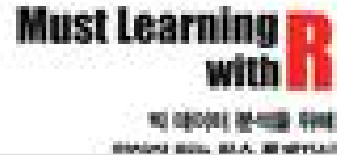
score
```

MSE(Mean Squared Error)

1. MSE(Mean Squared Error)

1. MSE(Mean Squared Error) **MSE**를 이해하...

<https://wikidocs.net/34063>



cross_val_score()

교차 검증을 위한 Cross_val_score() 함수 사용법

사이킷 런(Scikit-Learn) 에서 보다 쉬운 교차검증 API - cross_val_score() - 사이킷 런에서는 교차검증(K-Fold or StratifiedKFold) 를 더 쉽게 할 수 있는 API를 제공하는 데 그것은 cross_val_score 이다. cross_val_score 에는 여러가지 변수 값들이 있지만 중요한 몇가지만 알아보자. estimator = > 분류 알고리즘(Classifier) 또는 회귀

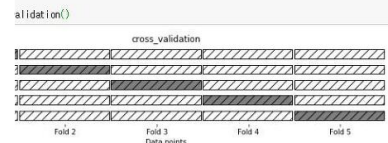
<https://guru.tistory.com/36>



교차검증 (cross-validation)

scikit-learn의 train_test_split() 함수를 사용하여 데이터를 훈련 세트와 테스트 세트로 한 번 나누는 것보다 더 성능이 좋은 평가방법은 교차검증."Cross Validation" 이다. K-겹 교차검증에서 K에는 5 or 10과 같은 특정 숫자가 들어가며 데이터를 비슷한 크기의 집합 'K개'로 나눈다. 이를 폴드(fold)라고 한다. (1) 단순 교차검증

<https://homeproject.tistory.com/6>



pickle - 'wb' & 'rb'

강의노트 04. 파이썬 pickle 모듈 - 초보몽키의 개발공부로그

<https://wayhome25.github.io/cs/2017/04/04/cs-04/>

KNN(K Nearest Neighbor)

[실습] k-최근접 이웃(k-Nearest Neighbor, kNN)

[https://github.com/wikibook/machine-learning]에서 다운로드 받은 농구선수에 대한 데이터를 사용하여 kNN 알고리즘을 적용해보는 실습을 하였다. 목표는 임의의 농구선수의 포지션을 예측하는 것이다. 데이터 획득 import pandas as pd df = pd.read_csv('./data/basketball_stat.csv') df.head() colab에서 실행하였고 파일을 드라이브

<https://kky042386.tistory.com/103>

	Player	POS	3P	2P	TRB	AST	STL	BLK
0	Alex Abrines	SG	1.4	0.6	1.3	0.6	0.5	0.1
1	Steven Adams	C	0.0	4.7	7.7	1.1	1.1	1.0
2	Alexis Ajinca	C	0.0	2.3	4.5	0.3	0.5	0.6
3	Chris Andersen	C	0.0	0.8	2.6	0.4	0.4	0.6
4	Will Barton	SG	1.5	3.5	4.3	3.4	0.8	0.5