|  |  |
| --- | --- |
| 학과 | 컴퓨터 소프트웨어 공학부 |
| 학번 | 201995085 |
| 이름 | 정주몽 |

로고, 상징, 엠블럼, 등록 상표이(가) 표시된 사진

자동 생성된 설명

인공지능 기말과제 보고서

목차

1. 인공지능 기말과제 주제 및 개요
2. 주요 코드 설명 및 실행 화면
3. 프로그램 코드

1. 인공지능 기말과제 주제 및 개요

**1) 주제**

CS2(Counter – Strike2)게임의 전장 금지 추천 시스템

**2) 주제 선정 이유**

CS2 게임의 프리미어 매치(랭킹전)는 각 팀별로 원하지 않은 전장을 선택해 금지하고 남은 전장 중에서 경기가 시작된다. 랭킹전인 만큼 승률을 올리기 위해서 AI가 사용자편의 팀의 전장 별 승률을 예측해서 어떤 전장을 금지시키면 좋은 지를 추천해주면 랭킹전의 승률을 올리는데 좋을 것 같다는 생각이 들었다.

**3) 기술 스택**

 프로그래밍 언어 및 라이브러리:

* Python: 데이터 처리 및 머신러닝 모델 구축
* Pandas: 데이터 프레임 처리
* Scikit-learn: 머신러닝 모델 학습 및 예측
* JSON: 데이터 저장 및 교환

 웹 기술:

* HTML5: 웹 페이지 구조
* CSS: 웹 페이지 스타일링
* JavaScript (ES6): 동적 데이터 로딩 및 DOM 조작
* Fetch API: 비동기 데이터 요청

2. 주요 코드 설명 및 실행 화면

1) 구성 파일

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

- Images 폴더: css에 필요한 이미지 파일 저장소

- Ai.py: 데이터의 전처리와 데이터를 학습하는 모델의 파일

- Blitz\_CSGO\_UserDater.xlsx: Blitz.gg사이트에서 얻은 CS2게임의 유저의 맵 승률 데이터 파일

- Main.css: Main.html의 css파일

- Main.html: 웹페이지 작업을 위한 파일

- team\_a\_map\_recommendations.json: ai.py에서 모델이 학습하고 예측한 값을 json으로 저장한 파일

- team\_a\_map\_recommendations.xlsx: ai.py에서 모델이 학습하고 예측한 값을 액샐로 저장한 파일

**2) ai.py**

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

Exel에 저장된 전장의 데이터 값을 불러옴.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

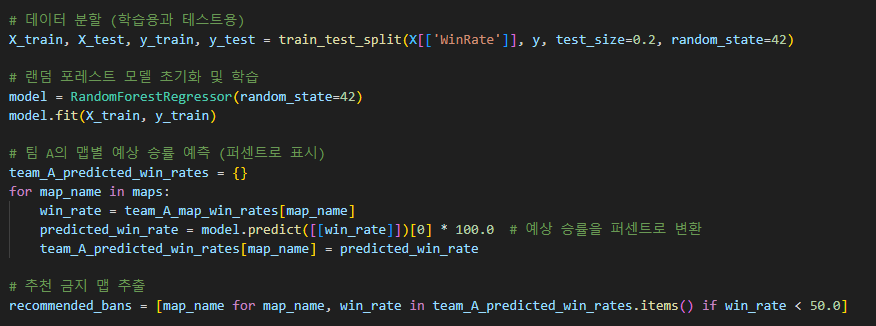
User\_76561199262465443은 사용자를 지정한 것.

랜덤의 9명의 데이터를 선택하고 임의 사용자를 지정한 이유는 원래 목표는 실시간 매치 메이킹에서 정보를 불러와서 하고 싶었지만, 본인의 역량 부족으로 인해 실시간 구동 기능을 구현하지 못하여 랜덤한 9명과 지정된 사용자 한명을 선택해 A와 B 두팀으로 나눠서 랜덤으로 5명씩 배치시켜서 테스트를 진행함.

텍스트, 스크린샷, 폰트, 디스플레이이(가) 표시된 사진

자동 생성된 설명

팀 A와 팀 B 간의 각 맵별 승률 데이터를 포함하는 pandas DataFrame인 X와 이에 대응하는 타겟 변수 y를 생성하는 과정



데이터를 학습용과 테스트용으로 분할 후, 랜덤 포레스트 모델을 초기화하고 학습시키며, 팀 A의 각 맵별 예상 승률을 예측하고 추천 금지 맵을 추출하는 과정.



train\_test\_split 함수를 사용하여 X[['WinRate']]와 y를 학습 데이터 (X\_train, y\_train)와 테스트 데이터 (X\_test, y\_test)로 분할.

test\_size=0.2는 테스트 데이터의 비율을 20%로 설정.

random\_state=42는 재현 가능한 난수 시드를 설정하여 분할 과정을 동일하게 함.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

RandomForestRegressor를 사용하여 랜덤 포레스트 회귀 모델을 초기화.

random\_state=42는 모델을 초기화할 때 사용되는 난수 시드로 설정.

fit 메서드를 사용하여 X\_train과 y\_train 데이터로 모델을 학습시킴.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

각 맵별로 지정 사용자가 있는 팀 A의 예상 승률을 계산.

team\_A\_map\_win\_rates에서 각 맵별 평균 승률을 가져와 model.predict를 사용하여 예상 승률을 계산.

계산된 예상 승률은 퍼센트로 변환하여 team\_A\_predicted\_win\_rates 사전에 저장.



team\_A\_predicted\_win\_rates에서 예상 승률이 50% 미만인 맵을 추천 금지 맵으로 선택.

리스트 컴프리헨션을 사용하여 조건에 맞는 맵 이름을 추출하여 recommended\_bans 리스트에 저장.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

예측한 결과를 json파일로 저장함.

**3) team\_a\_map\_recommendations.json**

**텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

Ai.py파일을 실행하면 저장완료 메시지와 함께 json파일이 디렉토리에 저장이 됨

스크린샷, 텍스트, 폰트이(가) 표시된 사진

자동 생성된 설명

예측한 결과를 json파일로 저장된 파일 내용

**4) Main.html**

**텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명**

document.addEventListener('DOMContentLoaded', () => { ... });: HTML 문서가 완전히 로드된 후 실행되는 함수입니다. 여기서 JSON 데이터를 가져오고 테이블에 데이터를 추가합니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

맵 이름과 이미지 링크 객체 설정

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

팀 A의 맵별 예상 승률을 표에 추가.

각 맵의 이미지와 이름을 함께 표시하고, 예상 승률을 퍼센트로 표시

for...of 반복문을 사용하여 팀 A 맵별 예상 승률과 추천 금지 맵을 표에 동적으로 추가. 각 맵에는 이미지와 이름이 함께 표시.

최종화면

스크린샷, 텍스트, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

3. 프로그램 코드

1) ai.py

import pandas as pd

from sklearn.ensemble import RandomForestRegressor

from sklearn.model\_selection import train\_test\_split

# 엑셀 파일에서 데이터 불러오기

file\_path = 'Blitz\_CSGO\_UserDater.xlsx'

df = pd.read\_excel(file\_path)

# 맵 목록

maps = [

    'de\_dust2',

    'de\_inferno',

    'de\_mirage',

    'de\_nuke',

    'de\_overpass',

    'de\_vertigo'

]

# SteamID가 76561199262465443인 유저 데이터 선택

user\_76561199262465443 = df[df['SteamID'] == '76561199262465443']

# 랜덤으로 9명의 데이터 선택 (SteamID가 76561199262465443인 유저를 제외)

selected\_users = df[df['SteamID'] != '76561199262465443'].sample(n=9, random\_state=1)  # random\_state 설정 추가

# 팀 A에 SteamID가 76561199262465443인 유저와 나머지 4명 배치

team\_A = pd.concat([user\_76561199262465443, selected\_users.sample(n=4, random\_state=2)])

# 팀 B에 남은 5명 배치

team\_B = selected\_users.drop(team\_A.index)

# 각 맵별 팀 A와 팀 B의 평균 승률 계산 (퍼센트로 표시)

team\_A\_map\_win\_rates = {}

team\_B\_map\_win\_rates = {}

for map\_name in maps:

    team\_A\_map\_win\_rates[map\_name] = team\_A[map\_name].mean()  # 퍼센트로 변환하지 않고 계산

    team\_B\_map\_win\_rates[map\_name] = team\_B[map\_name].mean()  # 퍼센트로 변환하지 않고 계산

# AI 모델 학습을 위한 데이터 준비

X = pd.DataFrame({

    'Team': ['A', 'B'] \* len(maps),

    'Map': maps \* 2,

    'WinRate': [team\_A\_map\_win\_rates[map\_name] for map\_name in maps] +

               [team\_B\_map\_win\_rates[map\_name] for map\_name in maps]

})

y = [0, 1] \* len(maps)  # A를 0, B를 1로 인코딩

# 데이터 분할 (학습용과 테스트용)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X[['WinRate']], y, test\_size=0.2, random\_state=42)

# 랜덤 포레스트 모델 초기화 및 학습

model = RandomForestRegressor(random\_state=42)

model.fit(X\_train, y\_train)

# 팀 A의 맵별 예상 승률 예측 (퍼센트로 표시)

team\_A\_predicted\_win\_rates = {}

for map\_name in maps:

    win\_rate = team\_A\_map\_win\_rates[map\_name]

    predicted\_win\_rate = model.predict([[win\_rate]])[0] \* 100.0  # 예상 승률을 퍼센트로 변환

    team\_A\_predicted\_win\_rates[map\_name] = predicted\_win\_rate

# 추천 금지 맵 추출

recommended\_bans = [map\_name for map\_name, win\_rate in team\_A\_predicted\_win\_rates.items() if win\_rate < 50.0]

# JSON 파일로 저장

import json

output\_json\_path = 'team\_a\_map\_recommendations.json'

output\_data = {

    'team\_A\_map\_predictions': team\_A\_predicted\_win\_rates,

    'recommended\_bans': recommended\_bans

}

with open(output\_json\_path, 'w') as json\_file:

    json.dump(output\_data, json\_file)

print(f"결과가 {output\_json\_path}에 저장되었습니다.")

**2) Main.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>CS:GO 팀 A 맵별 예상 승률 및 추천 금지 맵</title>

    <link rel="stylesheet" href="Main.css">

</head>

<body>

    <h2>CS:GO 팀 A 맵별 예상 승률</h2>

    <table>

        <thead>

            <tr>

                <th>맵</th>

                <th>예상 승률</th>

            </tr>

        </thead>

        <tbody id="teamA\_predictions">

            <!-- 여기에 JavaScript로 데이터를 추가할 예정 -->

        </tbody>

    </table>

    <h2>추천 금지 맵</h2>

    <table id="recommended\_bans\_table">

        <thead>

            <tr>

                <th>맵</th>

            </tr>

        </thead>

        <tbody id="recommended\_bans">

            <!-- 여기에 JavaScript로 데이터를 추가할 예정 -->

        </tbody>

    </table>

    <script>

        document.addEventListener('DOMContentLoaded', () => {

            fetch('team\_a\_map\_recommendations.json')

                .then(response => response.json())

                .then(data => {

                    const teamAPredictions = document.getElementById('teamA\_predictions');

                    const recommendedBans = document.getElementById('recommended\_bans');

                    // 맵 이름과 이미지 링크 객체

                    const mapsData = {

                        'de\_dust2': {

                            name: 'Dust 2',

                            image: 'images/de\_dust2.jpg'

                        },

                        'de\_inferno': {

                            name: 'Inferno',

                            image: 'images/de\_inferno.jpg'

                        },

                        'de\_mirage': {

                            name: 'Mirage',

                            image: 'images/de\_mirage.jpg'

                        },

                        'de\_nuke': {

                            name: 'Nuke',

                            image: 'images/de\_nuke.jpg'

                        },

                        'de\_overpass': {

                            name: 'Overpass',

                            image: 'images/de\_overpass.jpg'

                        },

                        'de\_vertigo': {

                            name: 'Vertigo',

                            image: 'images/de\_vertigo.jpg'

                        }

                    };

                    // 팀 A 맵별 예상 승률 추가

                    const teamAData = data.team\_A\_map\_predictions;

                    for (const [map, winRate] of Object.entries(teamAData)) {

                        const tr = document.createElement('tr');

                        const tdMap = document.createElement('td');

                        const tdWinRate = document.createElement('td');

                        const img = document.createElement('img');

                        // 이미지와 맵 이름 추가

                        img.src = mapsData[map].image;

                        img.alt = mapsData[map].name;

                        img.style.width = '100px'; // 이미지 크기 조절

                        tdMap.appendChild(img);

                        tdMap.innerHTML += `<br>${mapsData[map].name}`; // 줄 바꿈과 맵 이름 추가

                        // 예상 승률 추가

                        tdWinRate.textContent = `${winRate.toFixed(2)}%`;

                        tr.appendChild(tdMap);

                        tr.appendChild(tdWinRate);

                        teamAPredictions.appendChild(tr);

                    }

                    // 추천 금지 맵 표 생성

                    const recommendedBansTable = document.getElementById('recommended\_bans');

                    const bansData = data.recommended\_bans;

                    bansData.forEach(map => {

                        const tr = document.createElement('tr');

                        const tdMap = document.createElement('td');

                        const img = document.createElement('img');

                        // 이미지와 맵 이름 추가

                        img.src = mapsData[map].image;

                        img.alt = mapsData[map].name;

                        img.style.width = '100px'; // 이미지 크기 조절

                        tdMap.appendChild(img);

                        tdMap.innerHTML += `<br>${mapsData[map].name}`; // 줄 바꿈과 맵 이름 추가

                        tr.appendChild(tdMap);

                        recommendedBansTable.appendChild(tr);

                    });

                })

                .catch(error => console.error('Error loading the JSON file:', error));

        });

    </script>

</body>

</html>

**3)Main.css**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>CS:GO 팀 A 맵별 예상 승률 및 추천 금지 맵</title>

    <link rel="stylesheet" href="Main.css">

</head>

<body>

    <h2>CS:GO 팀 A 맵별 예상 승률</h2>

    <table>

        <thead>

            <tr>

                <th>맵</th>

                <th>예상 승률</th>

            </tr>

        </thead>

        <tbody id="teamA\_predictions">

            <!-- 여기에 JavaScript로 데이터를 추가할 예정 -->

        </tbody>

    </table>

    <h2>추천 금지 맵</h2>

    <table id="recommended\_bans\_table">

        <thead>

            <tr>

                <th>맵</th>

            </tr>

        </thead>

        <tbody id="recommended\_bans">

            <!-- 여기에 JavaScript로 데이터를 추가할 예정 -->

        </tbody>

    </table>

    <script>

        document.addEventListener('DOMContentLoaded', () => {

            fetch('team\_a\_map\_recommendations.json')

                .then(response => response.json())

                .then(data => {

                    const teamAPredictions = document.getElementById('teamA\_predictions');

                    const recommendedBans = document.getElementById('recommended\_bans');

                    // 맵 이름과 이미지 링크 객체

                    const mapsData = {

                        'de\_dust2': {

                            name: 'Dust 2',

                            image: 'images/de\_dust2.jpg'

                        },

                        'de\_inferno': {

                            name: 'Inferno',

                            image: 'images/de\_inferno.jpg'

                        },

                        'de\_mirage': {

                            name: 'Mirage',

                            image: 'images/de\_mirage.jpg'

                        },

                        'de\_nuke': {

                            name: 'Nuke',

                            image: 'images/de\_nuke.jpg'

                        },

                        'de\_overpass': {

                            name: 'Overpass',

                            image: 'images/de\_overpass.jpg'

                        },

                        'de\_vertigo': {

                            name: 'Vertigo',

                            image: 'images/de\_vertigo.jpg'

                        }

                    };

                    // 팀 A 맵별 예상 승률 추가

                    const teamAData = data.team\_A\_map\_predictions;

                    for (const [map, winRate] of Object.entries(teamAData)) {

                        const tr = document.createElement('tr');

                        const tdMap = document.createElement('td');

                        const tdWinRate = document.createElement('td');

                        const img = document.createElement('img');

                        // 이미지와 맵 이름 추가

                        img.src = mapsData[map].image;

                        img.alt = mapsData[map].name;

                        img.style.width = '100px'; // 이미지 크기 조절

                        tdMap.appendChild(img);

                        tdMap.innerHTML += `<br>${mapsData[map].name}`; // 줄 바꿈과 맵 이름 추가

                        // 예상 승률 추가

                        tdWinRate.textContent = `${winRate.toFixed(2)}%`;

                        tr.appendChild(tdMap);

                        tr.appendChild(tdWinRate);

                        teamAPredictions.appendChild(tr);

                    }

                    // 추천 금지 맵 표 생성

                    const recommendedBansTable = document.getElementById('recommended\_bans');

                    const bansData = data.recommended\_bans;

                    bansData.forEach(map => {

                        const tr = document.createElement('tr');

                        const tdMap = document.createElement('td');

                        const img = document.createElement('img');

                        // 이미지와 맵 이름 추가

                        img.src = mapsData[map].image;

                        img.alt = mapsData[map].name;

                        img.style.width = '100px'; // 이미지 크기 조절

                        tdMap.appendChild(img);

                        tdMap.innerHTML += `<br>${mapsData[map].name}`; // 줄 바꿈과 맵 이름 추가

                        tr.appendChild(tdMap);

                        recommendedBansTable.appendChild(tr);

                    });

                })

                .catch(error => console.error('Error loading the JSON file:', error));

        });

    </script>

</body>

</html>