

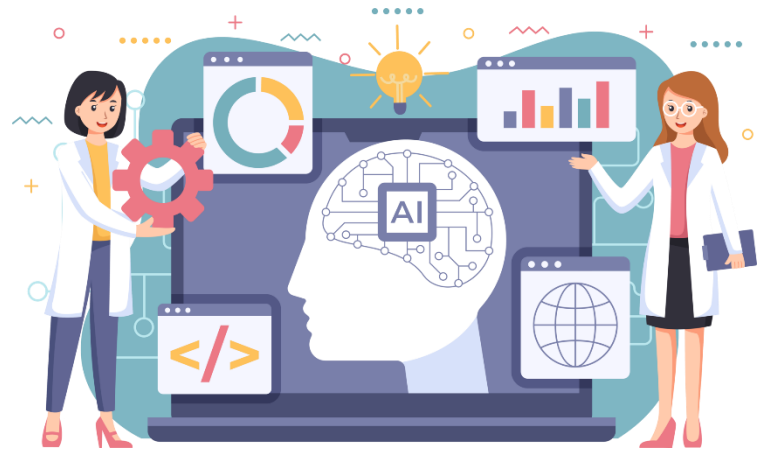


상관 분석

인공지능 RPA – 9주차

학습 내용

1. 상관분석 실습
2. Fast API - 파일 업로드



인공지능 실습

_데이터의 처리 과정



1. 데이터 정제 (Data Cleaning) : 불필요하거나 일치하지 않는 데이터를 제거
2. 데이터 통합 (Data Integration) : 다수의 데이터 소스들을 결합
3. 데이터 선택 (Data Selection) : 필요한 데이터들을 데이터 저장소로부터 검색
4. 데이터 변환 (Data Transformation) : 데이터 마이닝/모델링을 하기에 적합한 형태로 데이터 가공
5. 데이터 마이닝 / 모델링 (Data Mining, Modeling) : 지능적 방법들을 적용하여 지식(데이터 패턴, 관계 등) 추출
6. 데이터 검증 (Data Evaluation) : 찾아낸 지식(데이터 패턴, 관계 등)를 검증
7. 데이터 시각화 (Data Presentation) : 발견한 지식을 사용자에게 효과적으로 보여주기 위해 시각화



_상관분석

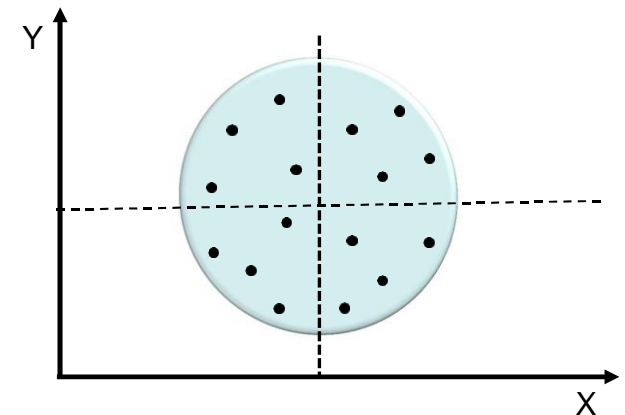
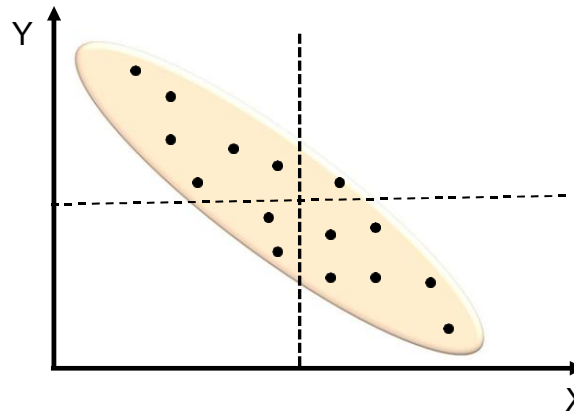
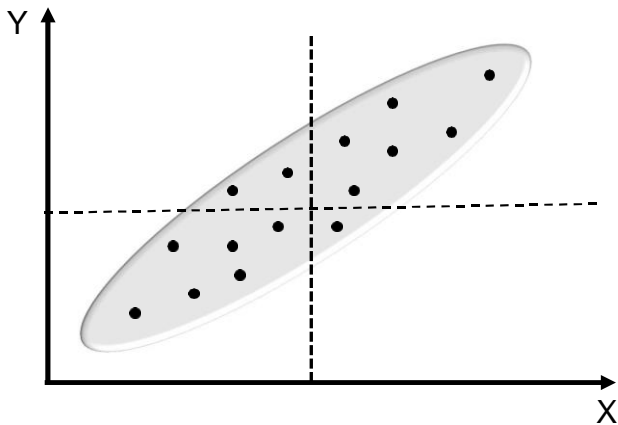
상관분석(correlation analysis) : 두 변수 간의 선형관계가 존재하는지 또는 존재하지 않는지를 분석

- 변수들 간의 선형성의 강도에 대한 통계적 분석이라 할 수 있음
ex) 가계소득과 저축간 관계, 흡연량과 폐암발병률 관계
- 인과성은 다른 이야기다

관계정도를 계산하는 방법 : 공분산을 이용

- 공분산은 두 변수 X, Y가 서로 어떤 패턴(pattern)을 보여주는 가를 나타냄.
- $Cov(X, Y) > 0$ 이면 X가 증가(감소)할 때 Y도 증가(감소)
- $Cov(X, Y) < 0$ 이면 X가 증가(감소)할 때 Y는 감소(증가)
- $Cov(X, Y) = 0$ 이면 두 변수는 아무런 상관이 없음

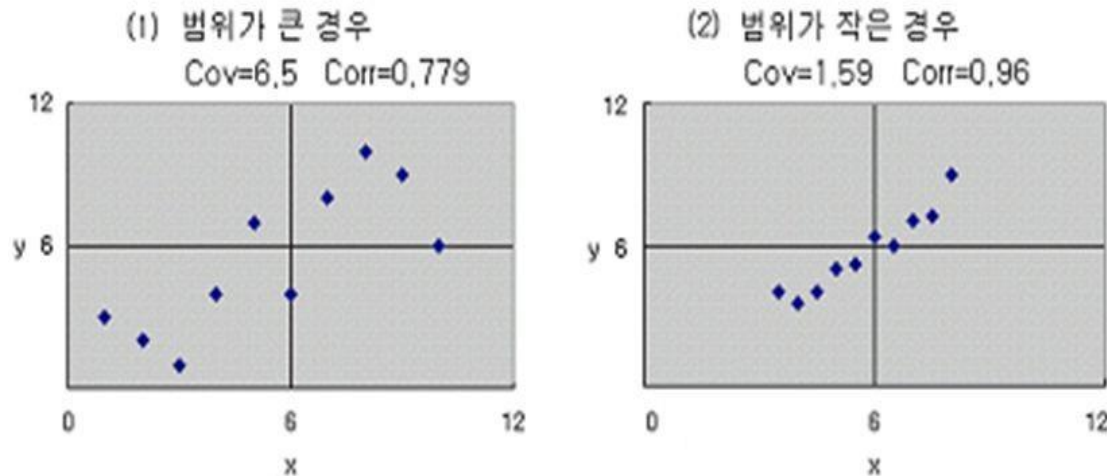
$$Cov(X, Y) = \sigma_{XY} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{n}$$



_상관 분석

공분산으로 변수간 상관성을 판단할 수 있을까?

- 공분산이 커도 두 변수간 상관성이 적은 경우가 있음 → 상관계수 필요



관계의 정도는 상관계수 (correlation coefficient) 를 이용

- 상관계수는 공분산을 각각의 표준편차로 나누어 표준화한 값을 나타내며 'r'로 표기함.

$$\rho = \text{Corr}(X, Y) = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}(X) \text{Var}(Y)}} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y}$$

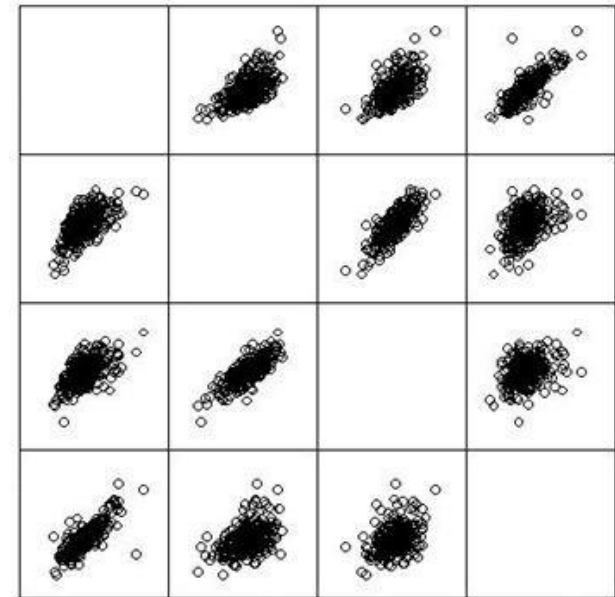
$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

_상관 분석의 해석

상관계수의 값에 따라 판단

- $-1 < r < 1$
- 상관계수의 절대값이 클수록 산점도의 띠 폭은 좁아짐

상관계수 절대값	해 석
0.2 이하	상관관계 거의 없음
0.2 ~ 0.4	낮은 상관관계
0.4 ~ 0.6	보통 관계
0.6 ~ 0.8	높은 상관관계
0.8 이상	매우 높은 상관관계



_상관분석 실습

1. 라이브러리 불러오기

- `import pandas as pd`
- `import matplotlib.pyplot as plt`
- `import seaborn as sns`

2. 데이터 불러오기 및 데이터 선택

- `w = pd.read_csv('ch5-1.csv')`
- `w_n = w.iloc[:,1:5]`

3. 상관 분석 수행

- `w_cor = w_n.corr(method = 'pearson')`

4. 결과 보기 및 시각화

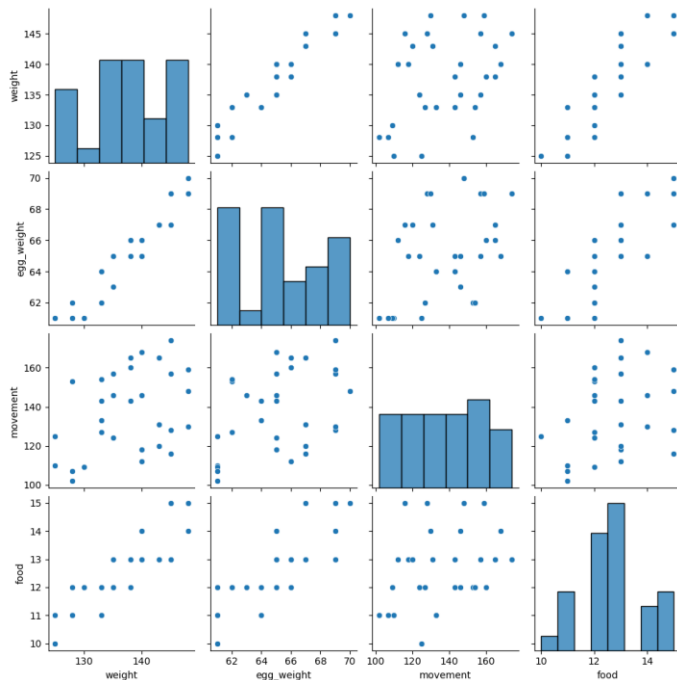
- `print(w_cor, end='\n\n')`
- `sns.pairplot(w_n)`
- `plt.figure(figsize = (10,7))`
- `sns.heatmap(w_cor, annot = True, cmap = 'Blues')`
- `plt.show()`



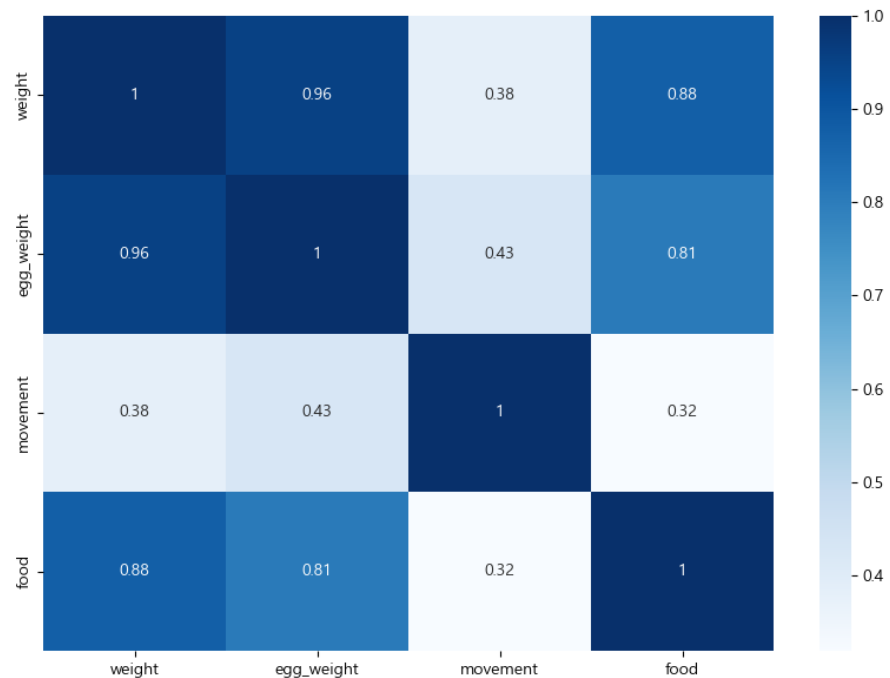
_상관분석 실습

	weight	egg_weight	movement	food
weight	1.000000	0.957169	0.380719	0.877574
egg_weight	0.957169	1.000000	0.428246	0.808147
movement	0.380719	0.428246	1.000000	0.319011
food	0.877574	0.808147	0.319011	1.000000

산점도

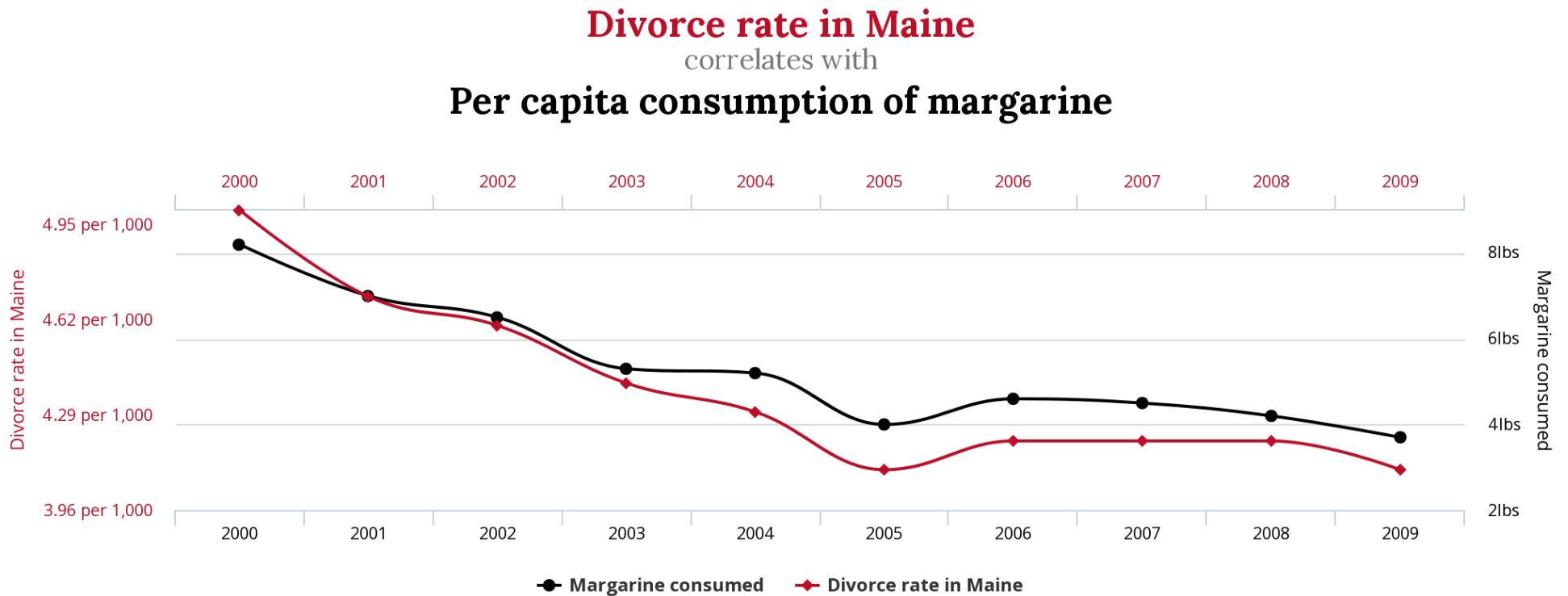


상관행렬도



_상관 분석

이혼률과 마가린 소비량

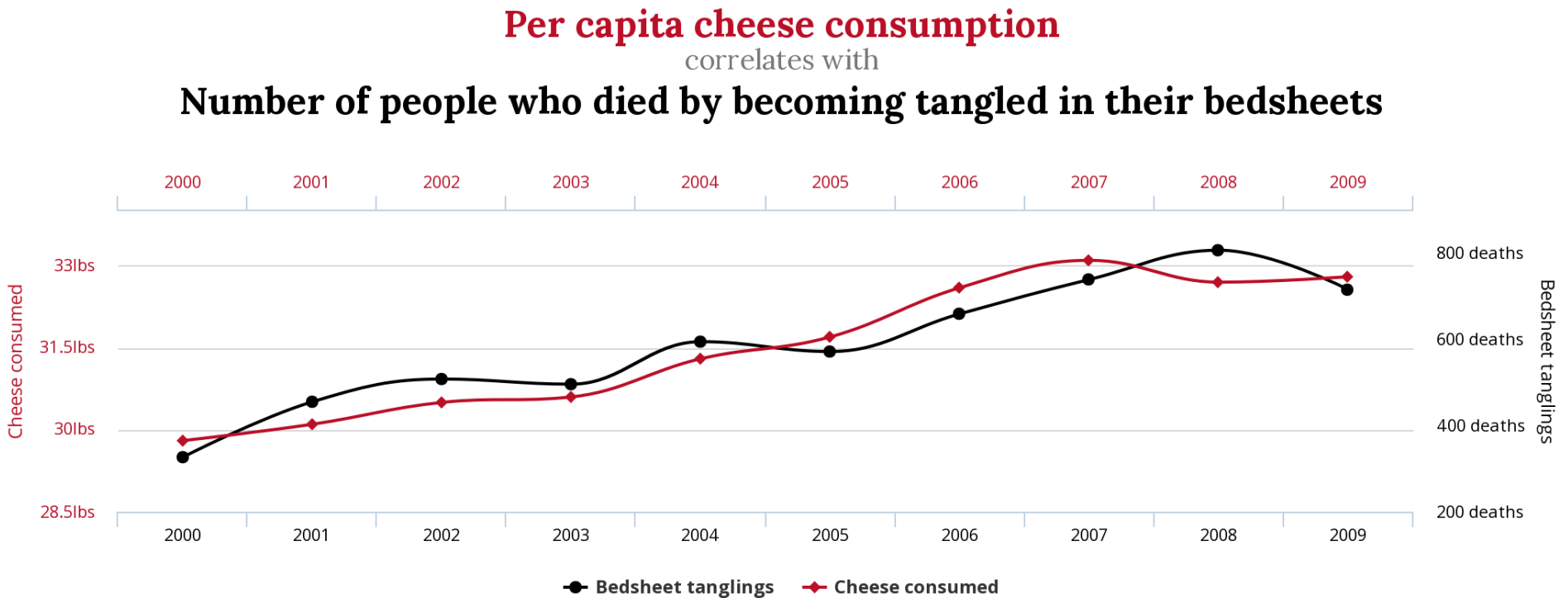


tylervigen.com



_상관 분석

1인당 치즈 소비량과 침대 시트에 얽혀 죽은 사망자 수



tylervigen.com



RPA 실습

_FastAPI

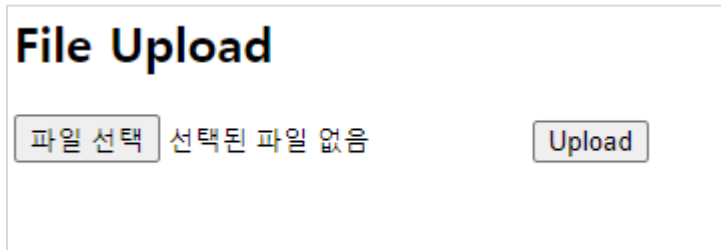
Fast API 파일 업로드 코딩

- 클라이언트 코딩(Front)
- 서버 코딩 (Back)

1. 클라이언트 코딩

: static 폴더안에 upload.html 파일 생성

```
<h2>File Upload</h2>
<input type="file" id="file">
<button onclick="uploadFile()">Upload</button>
```



```
<script>
  async function uploadFile() {
    const fileInput = document.getElementById('file');
    const formData = new FormData();
    formData.append('file', fileInput.files[0]);

    const response = await fetch('/uploadfile/', {
      method: 'POST',
      body: formData,
    });
    const data = await response.json();
    alert(`File uploaded to: ${data.location}`);
  }
</script>
```



_FastAPI

2. 서버 코딩 : Form 요청에 대응하는 API

```
from fastapi import File, UploadFile
import shutil
from pathlib import Path
```

```
@app.post("/uploadfile/")
```

```
async def create_upload_file(file: UploadFile = File(...)):
```

```
    save_path = Path("static/uploads") / file.filename
```

```
    save_path.parent.mkdir(parents=True, exist_ok=True)
```

```
    with save_path.open("wb") as buffer:
```

```
        shutil.copyfileobj(file.file, buffer)
```

```
    return {"filename": file.filename, "location": str(save_path)}
```



_FastAPI

3. 업로드 파일 확인

<http://127.0.0.1:8000/uploads/업로드한 파일명>

