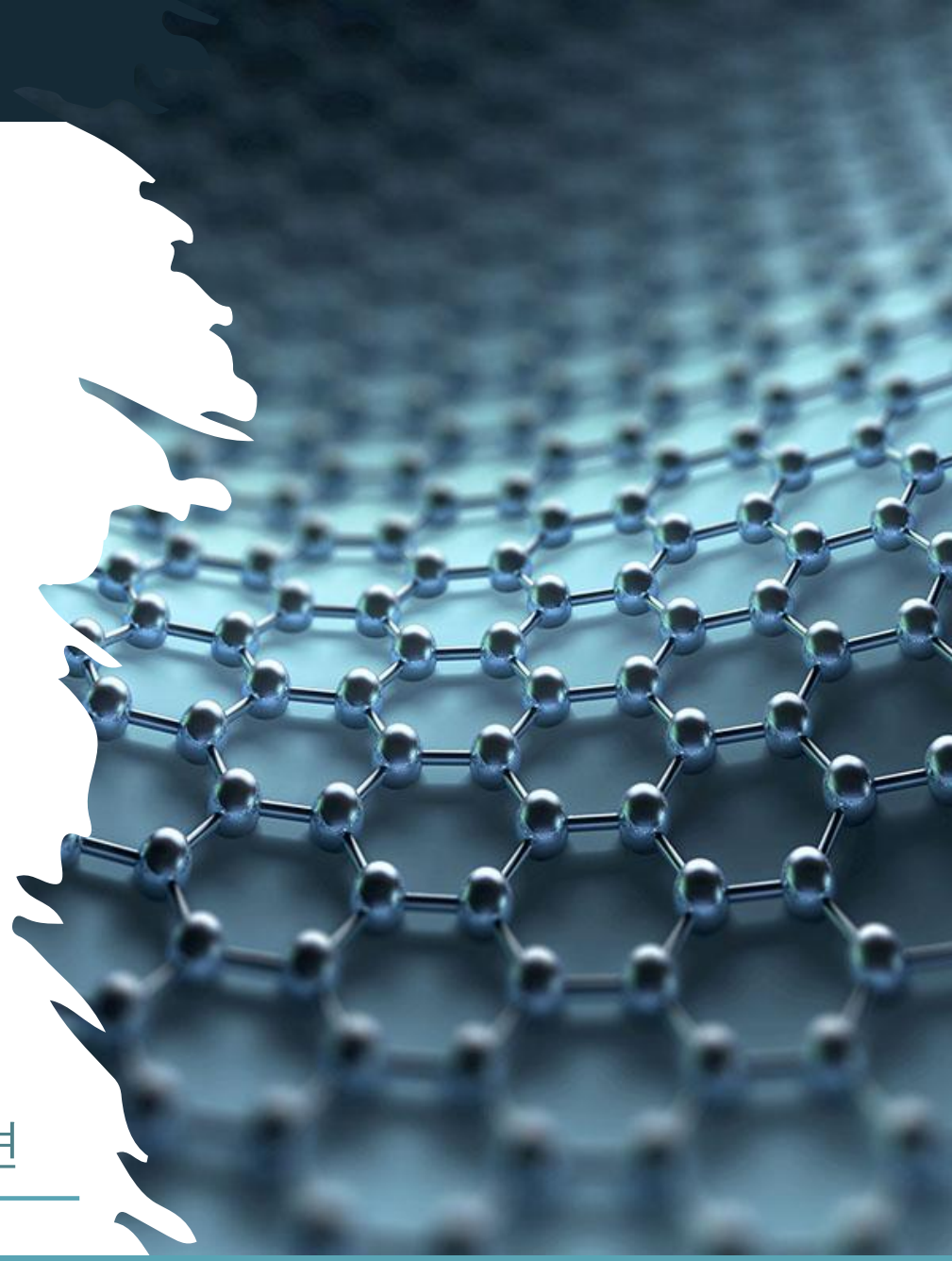
Stackoverflow 설문을 통한 개발자 연봉 분석



팀 명 : 이루지명

발표자 : 강진영

참여자 : 백지명, 강진영, 고예성, 박현식, 안영준, 조세연

CONTENTS

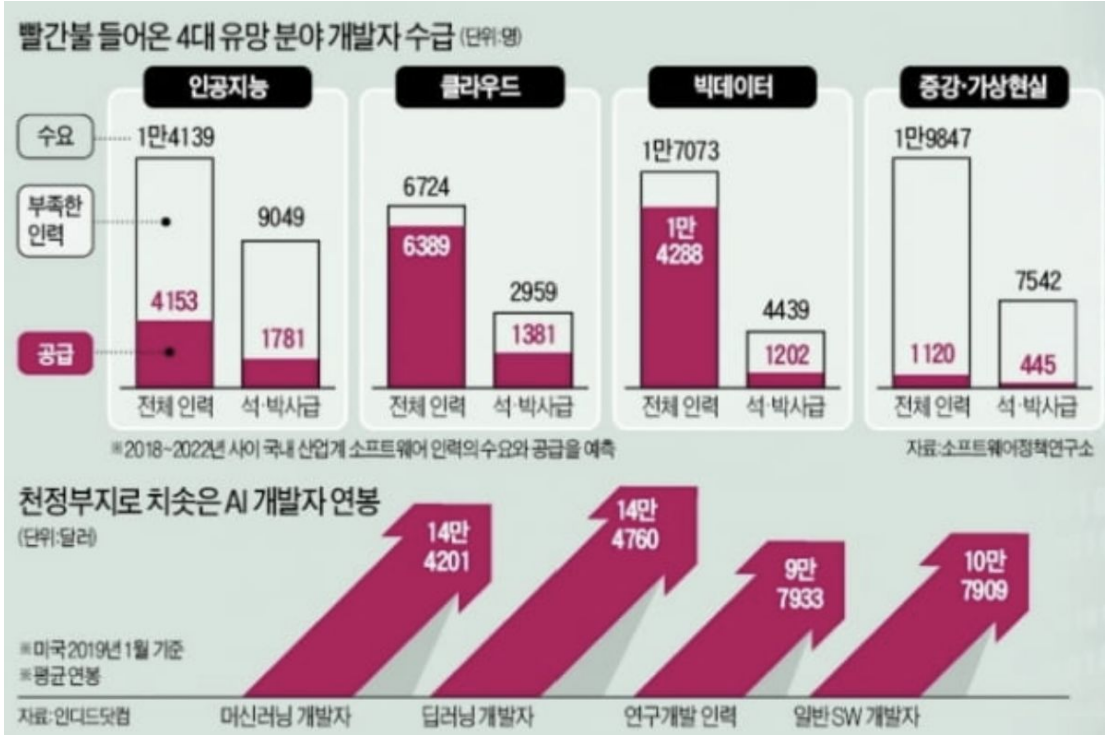
1. 분석 개요

1. 분석 개요

“제발 저희 회사로 와주세요”...몸값 더 뛰었다는 ‘경력’ 개발자

"IT 개발자 연봉, 조건만 넣으면 바로 계산"

| 스택오버플로, 개발자 연봉 계산기 공개



stack overflow

Stack Overflow Developer Survey 2011-2022

Dataset of the Stack Overflow Developer Survey from 2011 to 2022

2. 데이터 전처리

2. 데이터 전처리

연봉의 결측값 처리

```python
def calculate_average_salary(df, career):
    salaries = df.loc[(df['YearsCodePro'] == career) & df['ConvertedCompYearly'].notnull(), 'ConvertedCompYearly']
    average_salary = np.mean(salaries)
    return average_salary

unique_careers = df['YearsCodePro'].unique()
average_salaries = {}

for career in unique_careers:
    if career is not np.nan:
        average_salaries[career] = calculate_average_salary(df, career)

sorted_salaries = sorted(average_salaries.items(), key=lambda x: x[1], reverse=True)

salary_input = {career: average_salary for career, average_salary in sorted_salaries}

for career, average_salary in sorted_salaries:
    print(f"경력 {career}: 평균 연봉 {average_salary}")
# 결측값 대체
df['ConvertedCompYearly'] = df.apply(lambda row: salary_input.get(row['YearsCodePro'], row['ConvertedCompYearly']) if np.isnan(row['ConvertedCompYearly']) else row['ConvertedCompYearly'], axis=1)
```

연봉에 결측값이 존재하는 경우, 경력에 따라

해당 연봉의 **평균값**을 구하여 결측치에 채워넣음

2. 데이터 전처리

경력

```
df['YearsCodePro'].dtypes
```

```
dtype('O')
```

해당 데이터 확인 결과,
문자열인 Object 타입으로 반환

```
df['YearsCodePro'] = df['YearsCodePro'].replace({
    'Less than a year': '1',
    '1 to 2 years': '2',
    '2 to 3 years': '3',
    '3 to 4 years': '4',
    '4 to 5 years': '5',
    '5 to 6 years': '6',
    '6 to 7 years': '7',
    '7 to 8 years': '8',
    '8 to 9 years': '9',
    '9 to 10 years': '10',
    '10 to 11 years': '11',
    '11 to 12 years': '12',
    '12 to 13 years': '13',
    '13 to 14 years': '14',
    '14 to 15 years': '15',
    '15 to 16 years': '16',
    '16 to 17 years': '17',
    '17 to 18 years': '18',
    '18 to 19 years': '19',
    '19 to 20 years': '20',
    '20 or more years': '25'
})
```

해당 데이터를 숫자형으로 변환하기 위해,
YearsCodePro의 값을 임의로 수정
20년 이상인 경우 **25년**으로 치환

```
df['YearsCodePro'] = pd.to_numeric(df['YearsCodePro'], errors='coerce').astype('Int64')
```

해당 데이터를 **숫자형**으로 변환

2. 데이터 전처리

개발자 유형

```
df['DeveloperType'].value_counts()
```

```
Web developer
10683
Web developer; Desktop applications developer
1849
Mobile developer
1556
Web developer; Mobile developer
1503
Desktop applications developer
1433

...
Web developer; Data scientist; Systems administrator; DevOps specialist; Quality assurance engineer
1
Web developer; Embedded applications/devices developer; Developer with a statistics or mathematics b
1
Web developer; Mobile developer; Embedded applications/devices developer; Graphics programming; Mach
administrator        1
Web developer; Mobile developer; Embedded applications/devices developer; Graphics programming; Deve
1
Web developer; Mobile developer; Developer with a statistics or mathematics background; Data scienti
1
Name: DeveloperType, Length: 1823, dtype: int64
```

개발자 유형의 종류가 중복된 형태

```
developer = []

# 'DeveloperType' 컬럼 값 순회
for dev in df['DeveloperType']:
    # NaN 값이 아닌 경우에만 처리
    if isinstance(dev, str):
        # ';'로 구분된 역할을 개별 역할로 분리하여 developer 리스트에 추가
        dev_list = dev.split(';')
        for role in dev_list:
            developer.append(role.strip())  # 공백 제거 후 추가
0
# 중복된 개발자 역할 제거
columns = list(set(developer))


# 개발자 역할을 열로 추가하고 초기화
for column in columns:
    df[column] = 0

# 개발자 역할에 해당하는 열에 1로 할당
for i, dev in enumerate(df['DeveloperType']):
    if isinstance(dev, str):
        dev_list = dev.split(';')
        for role in dev_list:
            role = role.strip()
            if role in columns:
                df.at[i, role] = 1
```

개발자의 유형의 중복값을 제거

개발자의 유형을 모두 파생변수로 만들고

해당 개발자 유형인 경우 1, 아니면 0으로 채움

2. 데이터 전처리

대륙

```
df['Country'].value_counts()
```

```
United States of America                                  8389
Germany                                                   2748
United Kingdom of Great Britain and Northern Ireland      2525
India                                                     2034
Canada                                                    1418
                                                          ...
Haiti                                                        1
Fiji                                                         1
Suriname                                                     1
Somalia                                                      1
Seychelles                                                   1
Name: Country, Length: 159, dtype: int64
```

국가의 데이터 종류가 다양하여

대륙으로 분류하기로 판단

```
continent_mapping = {
    'Slovakia': 'Europe',
    'Austria': 'Europe',
    'United Kingdom of Great Britain and Northern Ireland': 'Europe',
    'India': 'Asia',
    'Sweden': 'Europe',
    'Spain': 'Europe',
    'Germany': 'Europe',
    'Turkey': 'Asia',
    'Canada': 'North America',
    'France': 'Europe',
    'Switzerland': 'Europe',
    'Russian Federation': 'Europe',
    'Ukraine': 'Europe',
    'United States of America': 'North America',
    'Portugal': 'Europe',
    'Brazil': 'South America',
    'Bulgaria': 'Europe',
```

국가별 대륙 딕셔너리를 생성하고

Continent 파생변수를 생성,

Country 컬럼은 제거

2. 데이터 전처리

인종

```
df['Ethnicity']
```

```
0                              White or of European descent
1                              White or of European descent
2                              White or of European descent
3                              White or of European descent
5                              White or of European descent
                                 ...
51386                          White or of European descent
51387              East Asian; White or of European descent
51388    Black or of African descent; Hispanic or Latin...
51390                          White or of European descent
51391                          White or of European descent
Name: Ethnicity, Length: 33033, dtype: object
```

인종의 데이터 종류 확인

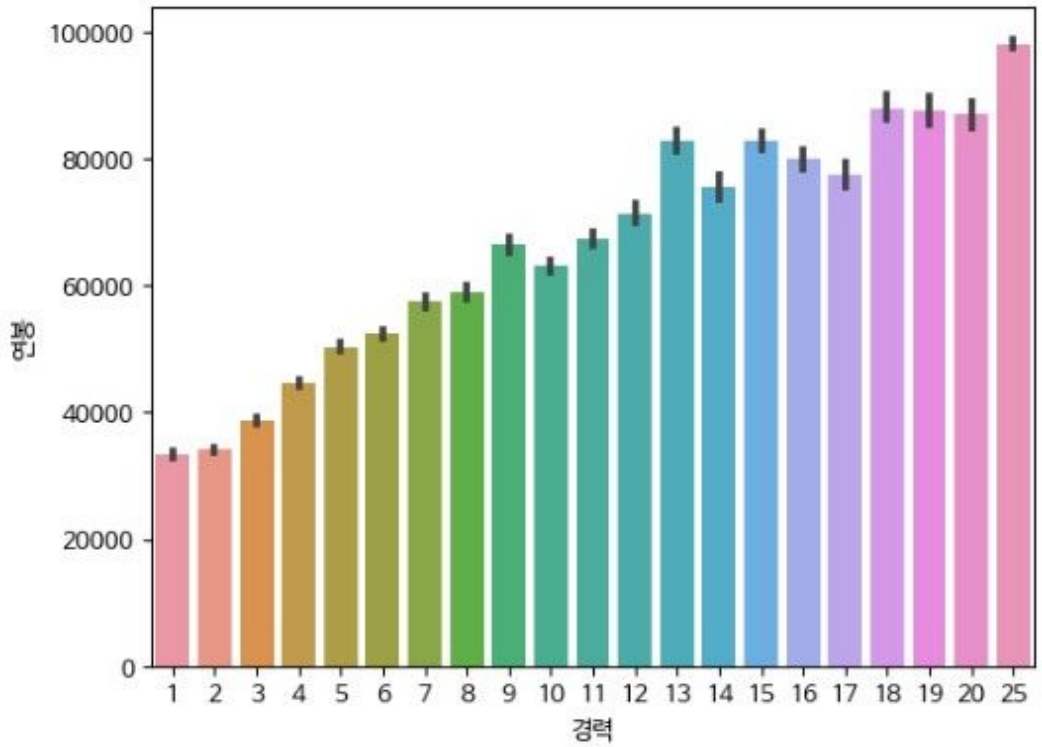
```
df = df[df['Ethnicity'].map(df['Ethnicity'].value_counts()) >= 500]
df_encoded = pd.get_dummies(df, columns='Ethnicity')
```

데이터의 개수가 **500개 이상**인
데이터만 재저장

3. 상관관계 분석

3. 독립변수와 종속변수의 상관관계 분석

경력

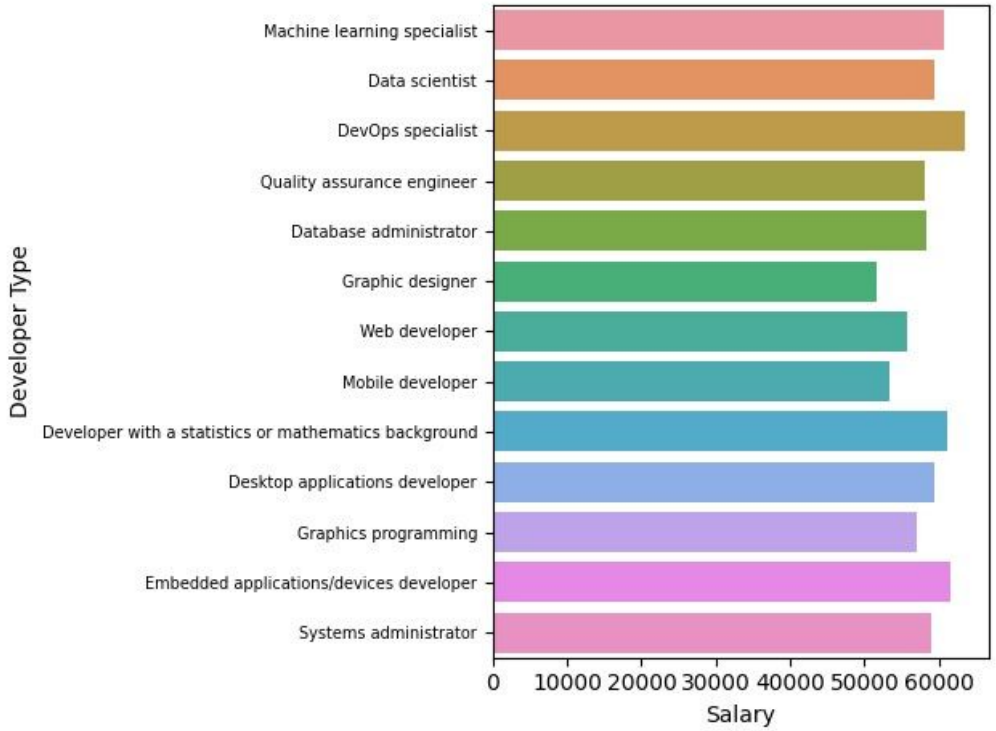


■ 경력

최고 값 : 25년 이상

최소 값 : 1년 이하

개발자 유형



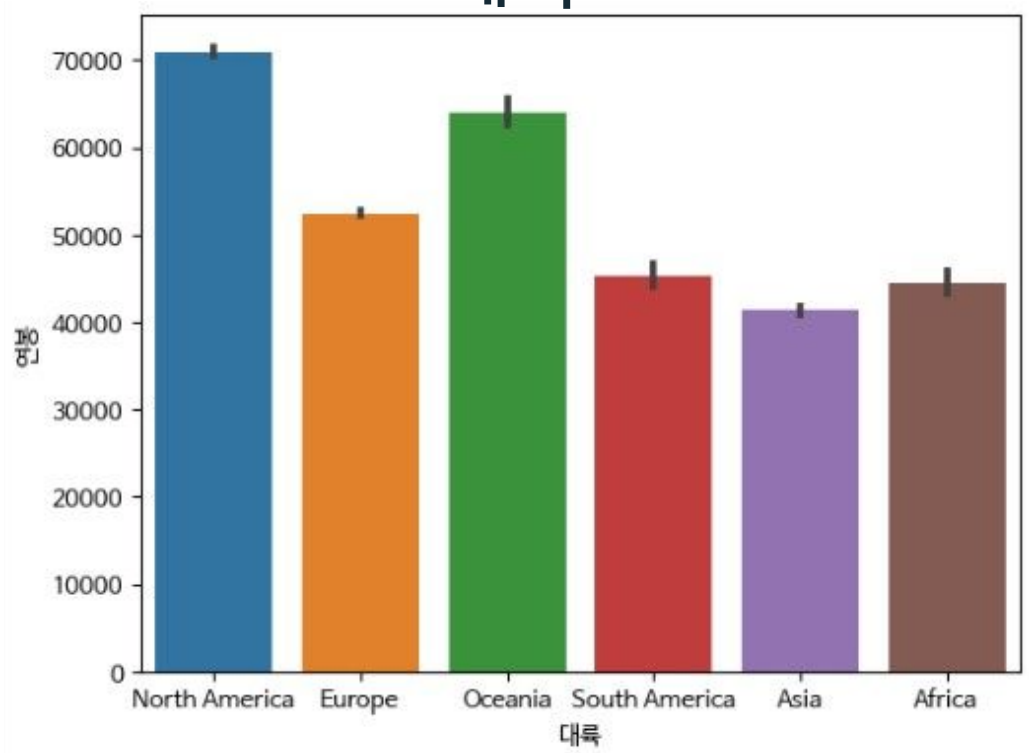
■ 개발자 유형

최고 값 : DevOps specialist

최소 값 : Graphic designer

3. 독립변수와 종속변수의 상관관계 분석

대륙

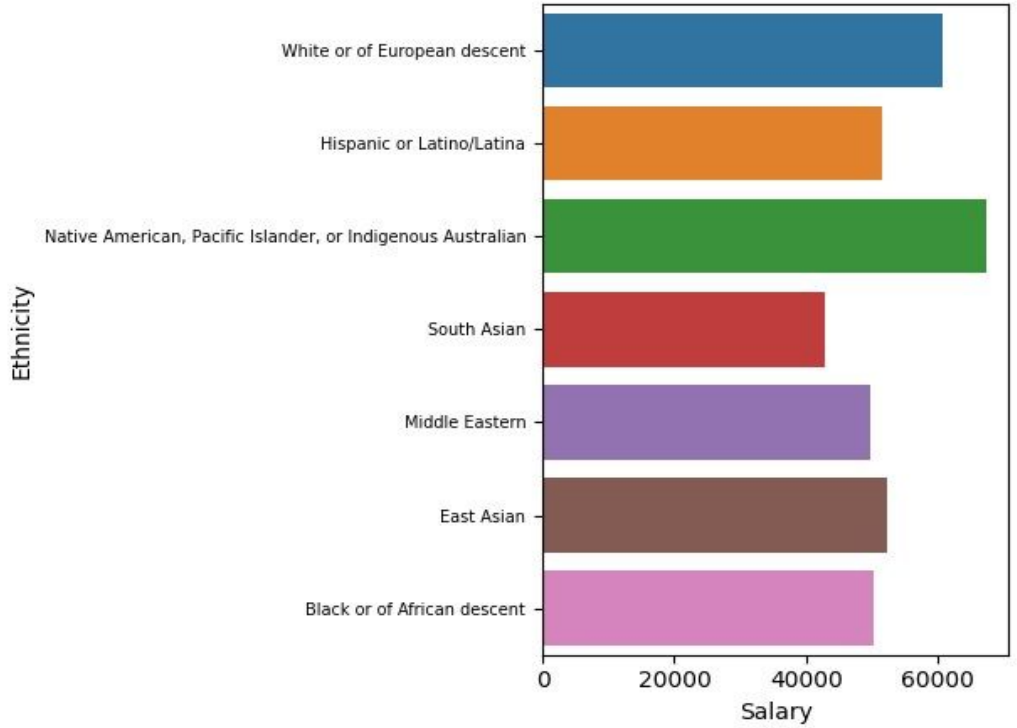


■ 대륙

최고 값 : North America

최소 값 : Asia

인종



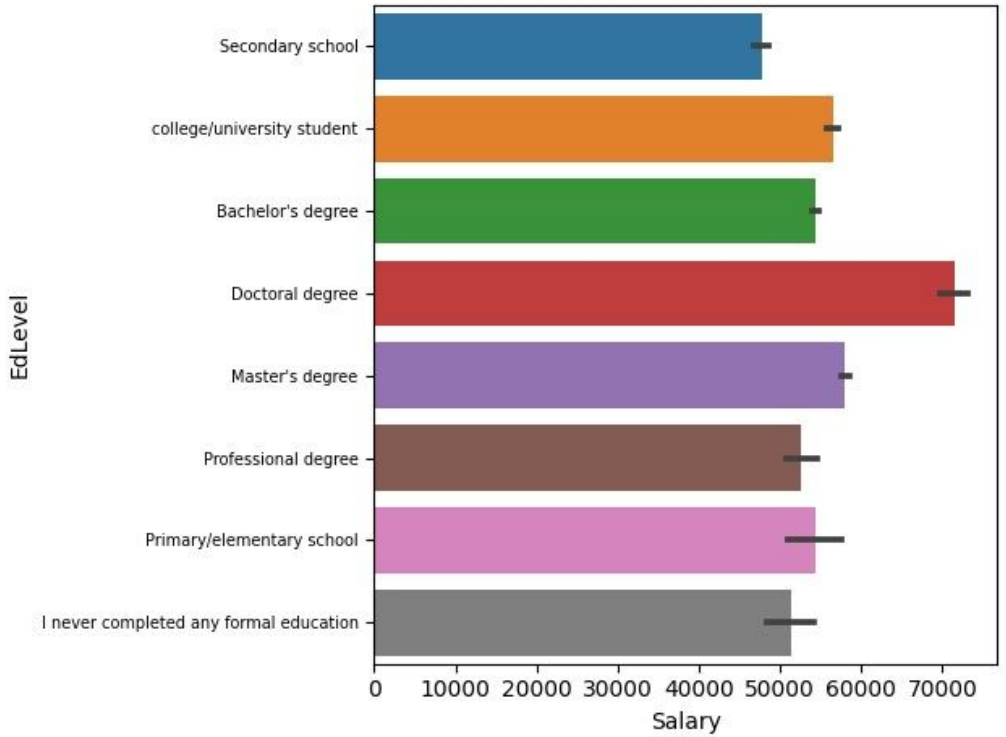
■ 인종

최고 값 : Native American, Pacific Islander, or Indigenous Australian

최소 값 : South Asian

3. 독립변수와 종속변수의 상관관계 분석

학위

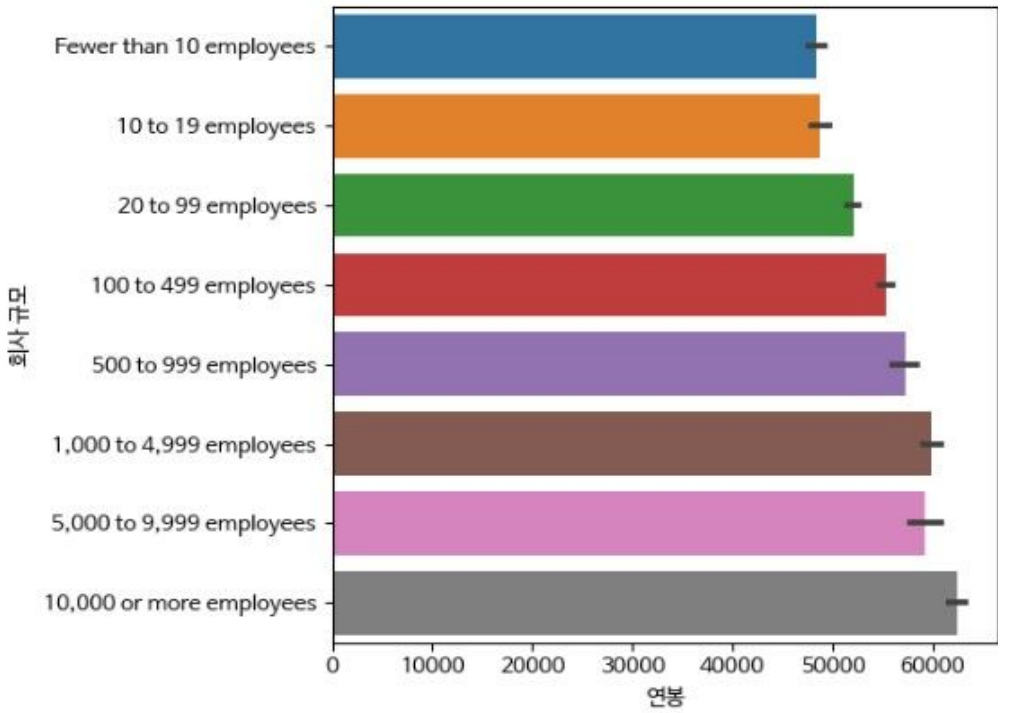


■ 학위

최고 값 : Doctoral degree

최소 값 : Secondary school

회사 규모



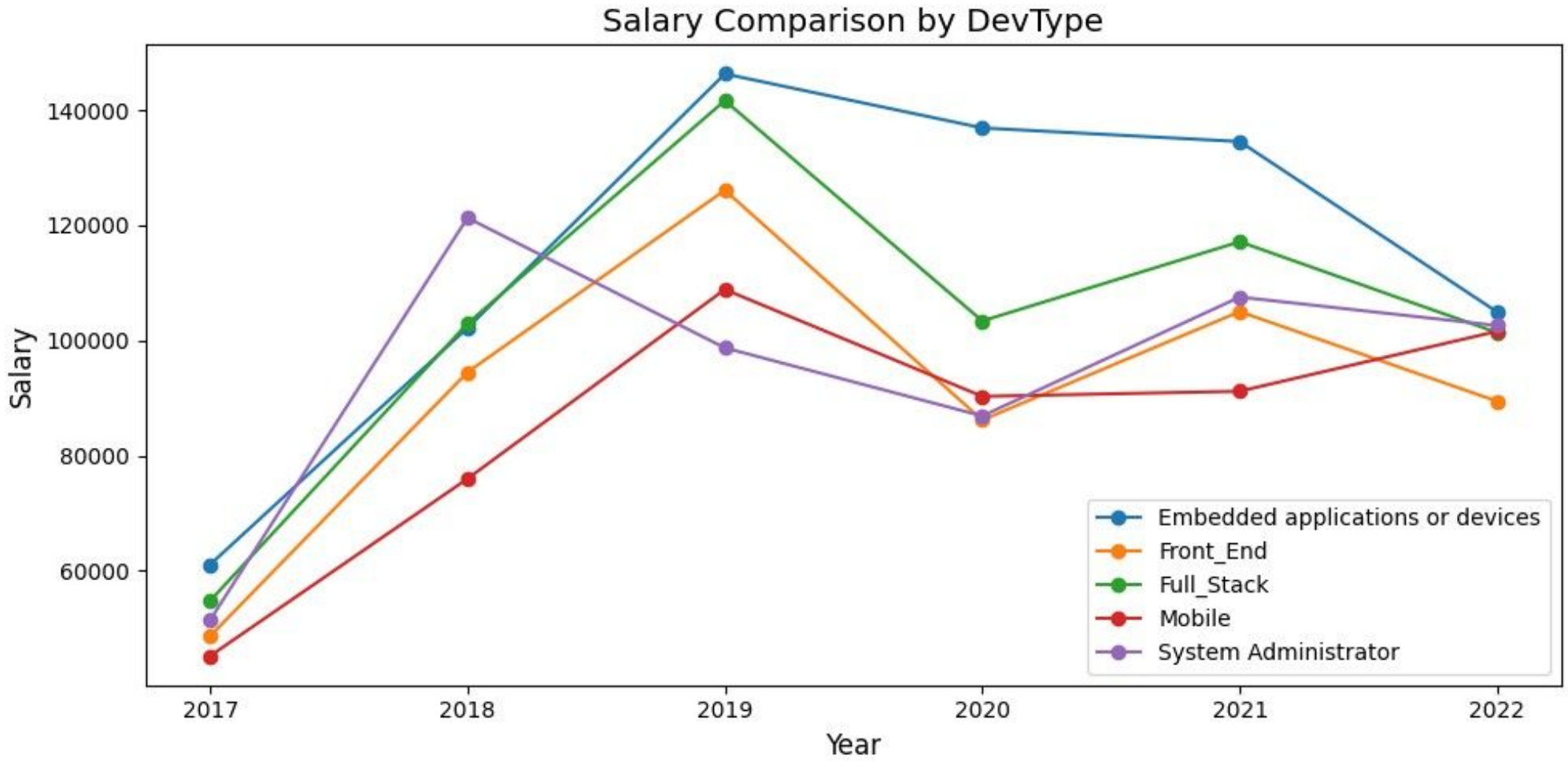
■ 회사 규모

최고 값 : 10,000 or more employees

최소 값 : Fewer than 10 employees

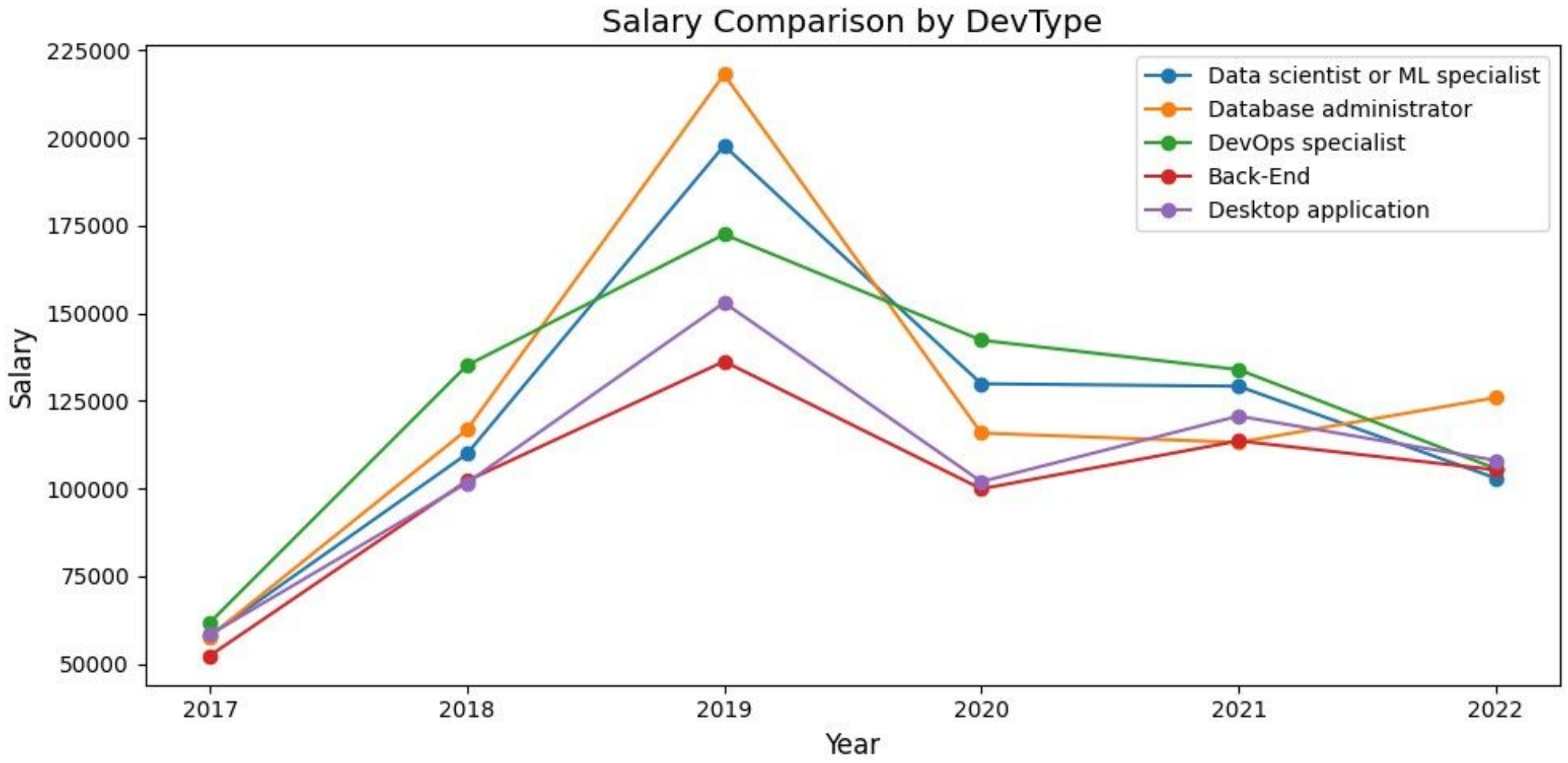
4. 개발자 연봉의 변화 추이

4. 개발자 유형별 연봉 변화 추이



- 시스템 관리자는 2018년 기준 연봉이 가장 높았지만, 2019년에 최하위를 기록함
- 임베디드 시스템은 2019년에 최고치를 찍고 하락세를 보이고 있음
- 전반적으로 19년도에 비해 20년도에 연봉 하락율이 큼

4. 개발자 유형별 연봉 변화 추이



- 데이터베이스 관리자는 제일 높은 연봉을 기록함(2019), 그러나 2020년에 절반에 가까운 하락을 보임
- 마찬가지로 19년도까지는 올랐지만, 20년도는 전반적으로 내려감

5. 결론

5. 결론

- 개발자의 연봉은 **경력과 학력 및 회사규모**와 양의 상관관계임

- 개발자의 연봉은 지속적으로 상승할 것으로 예측하였으나, 데이터로 보았을때 2019년 가장 높았고 2020년부터 하락한 것으로 나타남

- 개발자의 연봉은 코로나 팬데믹 영향이 비교적 적은 직업중 하나였으나, 팬데믹 영향이 있었음으로 추측

- 설문조사 데이터의 특성에 따라 주관적이기 때문에 분석의 오류 가능성이 큼

6. 트러블 슈팅

6. 트러블 슈팅

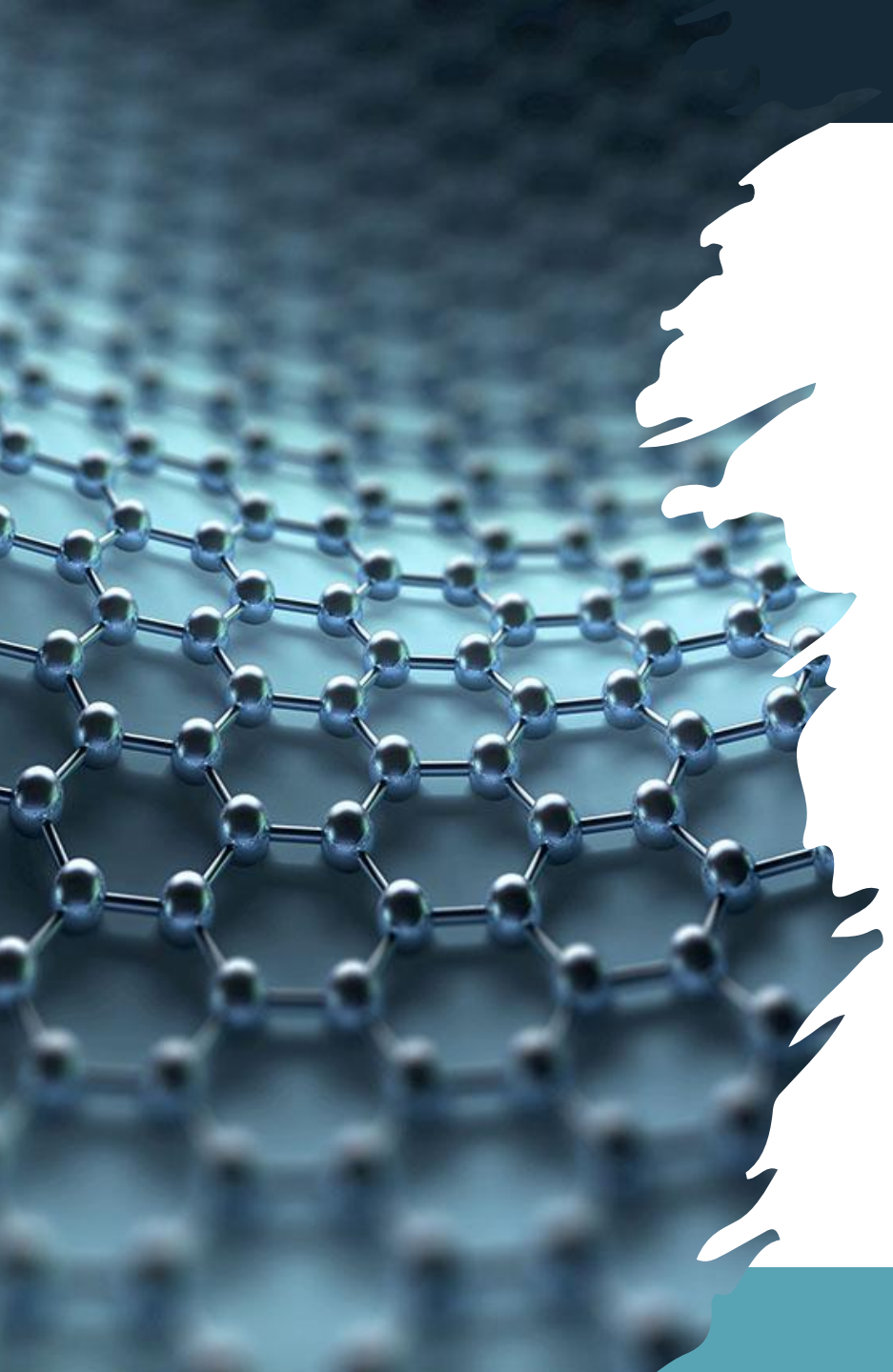
- 설문조사로 이루어진 데이터셋이기 때문에 신뢰도가 떨어지는 점
- 연도별 데이터 컬럼 개수와 종류의 일관성이 떨어짐 ➡ 연도별 같은 피쳐들만 남기고 명칭 통일화
- 연도별 데이터에서 컬럼의 데이터 타입이 서로 다르게 구성 ➡ 수치형 데이터를 범주화
- 다중 선택형 질문으로 인한 데이터 중복 ➡ 중복값 제거 및 개수가 더 많은 데이터로 통합
- 결측값 및 이상치 데이터 제거 후의 데이터 수가 처음 데이터 수보다 적어짐 ➡ 머신러닝 모델 학습에 필요한 데이터 수는 충족
- 초기 가설인 ‘해가 거듭할수록 개발자의 연봉이 상승할 것이다.’와는 다른 결과 도출

7. 참고문헌

Page1. 기사1 https://zdnet.co.kr/view/?no=20170921081136

Page1. 기사2 https://www.hankyung.com/it/article/202301314996i

Page1. 기사3 https://www.mk.co.kr/news/it/10747151 /



감사합니다