전공별 AI활용 2분반 – 기말 프로젝트 발표
32204292 모바일시스템공학과 조민혁
32204502 컴퓨터공학과 차지욱

공유 차량 이용자의 데이터 분석 모듈

CONTENTS

목차

목차

Contents

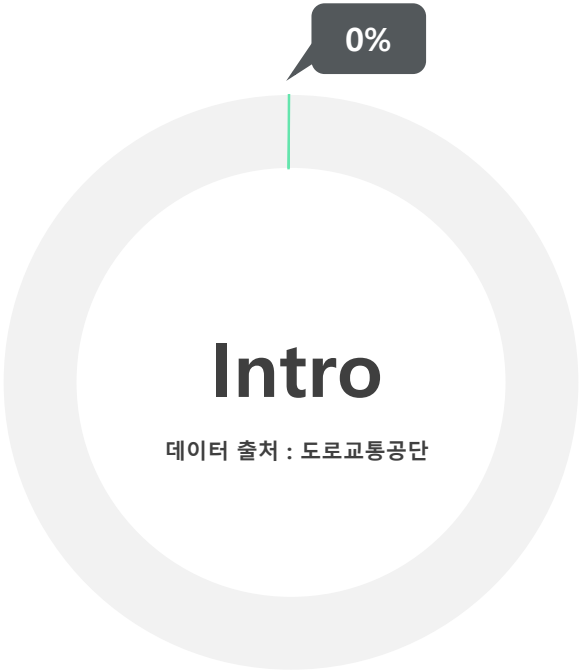
1. ### Intro ###
   - 카셰어링의 개념
   - 데이터 분석으로 알아본 카셰어링 수요

2. ### 문제 제시 및 솔루션 ###
   - 데이터 분석으로 알아본 카셰어링의 문제점
   - 문제를 해결 하기 위한 AI 모델 제시

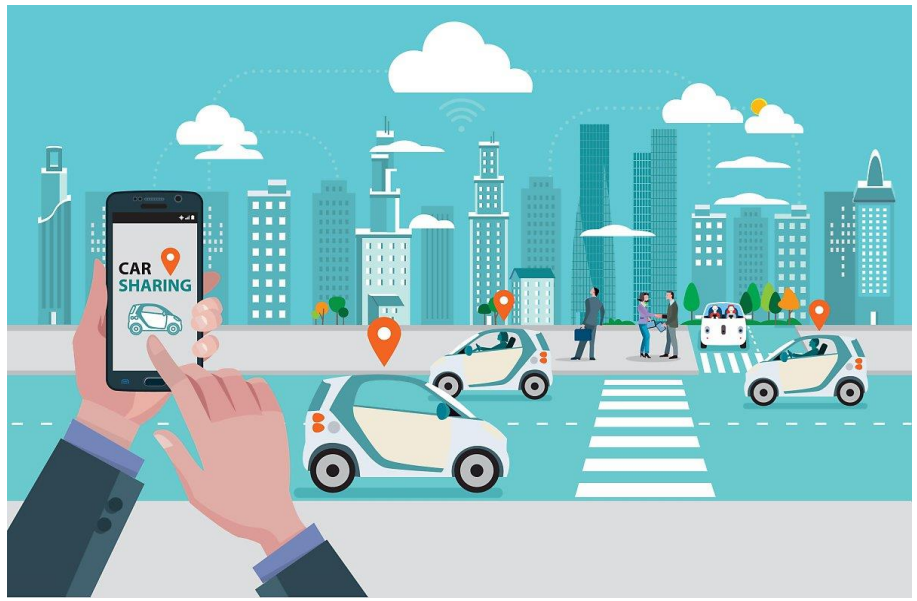
3. ### 프로그램 설명 ###
   - 프로그램의 작동 방식
   - 코드 설명

4. ### Outro ###
   - 활용 방안
   - 마무리

데이터 분석으로 알아본 카셰어링 수요 - 결과



카셰어링의 개념



카셰어링(Car Sharing)

- 자동차를 시간 단위로 여러 사람이 나눠 쓰는 것
- 시내 곳곳에 위치한 무인 거점을 통해 자동차 대여
- 지정된 거점에 반납

* 렌터카와의 차이점
- 거주지 근처에 'Service Zone' 존재
- 인증된 회원이라면 누구나 쉽게 대여 가능

데이터 분석으로 알아본 카셰어링 수요 - 코드

```
# Car_Sharing 시장 현황 제시
# Car_Sharing 수요에 대한 데이터 분석

import pandas as pd
import matplotlib.pyplot as plt
plt.rc('font', family = 'NanumBarunGothic') # 한글 폰트 사용을 위한 코드


df = pd.read_excel("/카셰어링_수요.xlsx")

# 데이터 전처리 - 열 이름 수정

df = df.rename(columns = {'Unnamed: 0': '분류'})
df = df.set_index('분류')


# 이용자 수에 대한 시각화
# 선 그래프로 시간에 따른 이용자 수의 변화를 보여줌
using_car_sr = df.loc['이용자수'] / 10000

plt.style.use('ggplot')

plt.figure(figsize = (14,5))

plt.plot(using_car_sr.index, using_car_sr.values, marker = 'o', markersize = 10)

plt.title('시간에 따른 이용자 수')
plt.xlabel('Time')
plt.ylabel('Users')

plt.legend(labels = ['단위 : 10000'], loc = 'best')
plt.show()

# 이용자 수에 따른 서비스 존과 차량 수의 증가
# 그래프 객체 생성
fig = plt.figure(figsize = (20,10))
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)

#서비스 존에 대한 선 그래프 생성
service_zone_sr = df.loc['서비스존']

ax1.plot(service_zone_sr.index, service_zone_sr.values, marker = 'o', markersize = 10,
         color = 'green', linewidth = 2)

ax1.set_title('시간에 따른 공유차량 서비스 존 수')
ax1.set_xlabel('Time')
ax1.set_ylabel('Service Zone')

# 차량 수에 대한 선 그래프 생성
car_numbers_sr = df.loc['차량수']

ax2.plot(car_numbers_sr.index, car_numbers_sr.values, marker = 'o', markersize = 10,
         color = 'blue', linewidth = 2)

ax2.set_title('시간에 따른 등록된 공유차량 수')
ax2.set_xlabel('Time')
ax2.set_ylabel('Car Numbers')


plt.show()
```

코드 설명

Pandas와 matplotlib.plt를 불러와 데이터 가공 및 시각화 시작

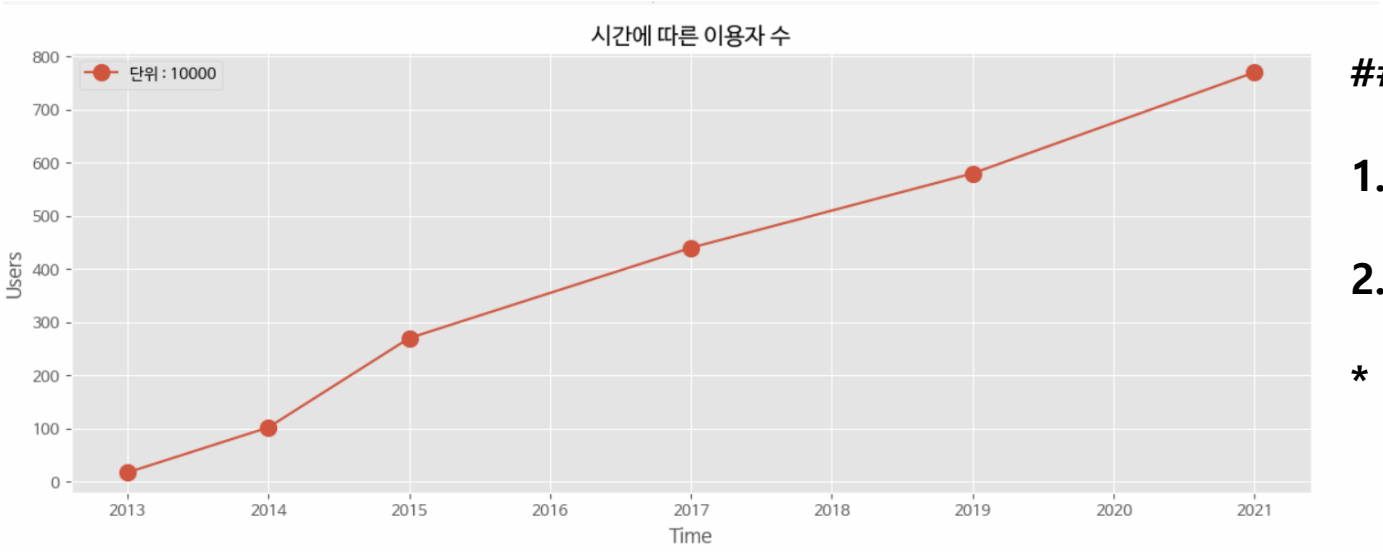
열 이름을 변경하는 데이터 전처리 진행

이용자 수, 차량 수, 서비스 존 수에 대한 시각화 진행

3개의 객체를 하나의 캔버스에 담기 위해 2x2 서브 플롯 생성

선 그래프를 통한 데이터 시각화 진행

데이터 분석으로 알아본 카셰어링 수요 - 결과

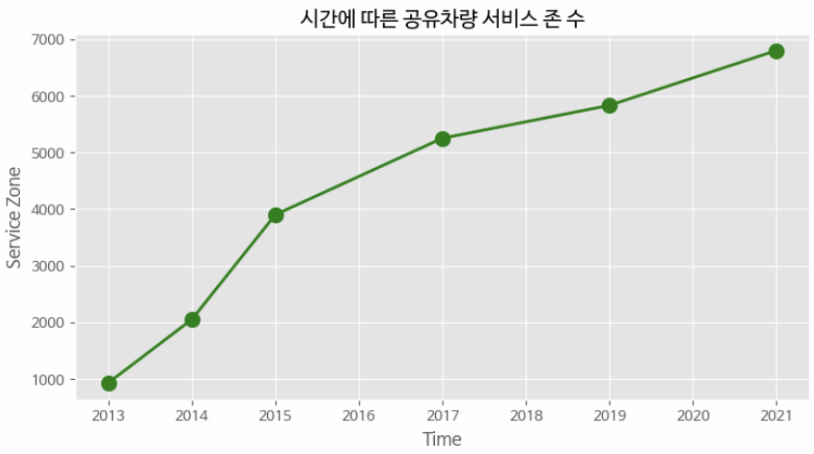


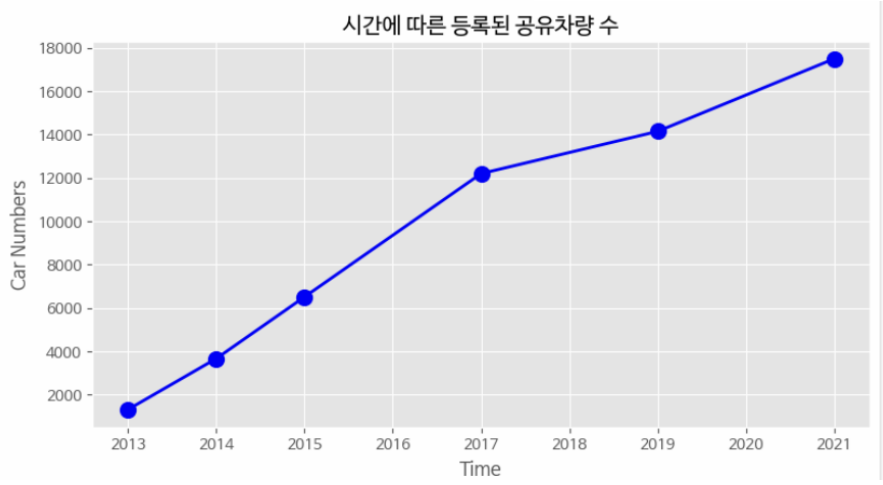
시각화 결과 분석

1. 시간에 따른 이용자 수, 서비스 존, 공유차량 수가 증가함

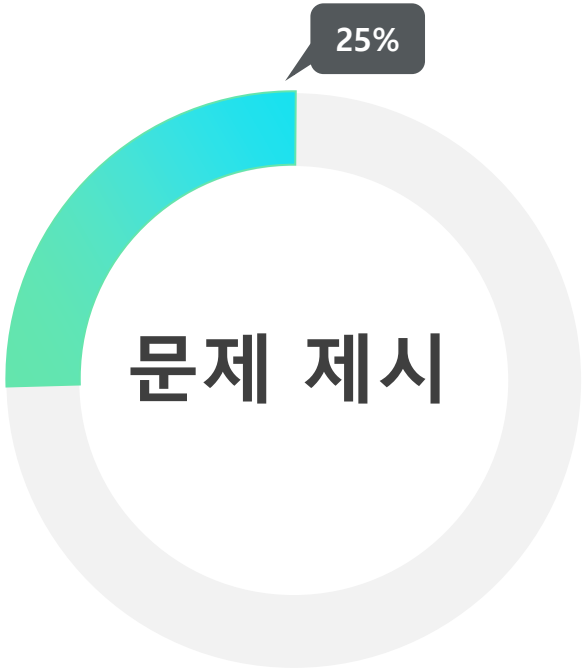
2. 카셰어링에 대한 수요와 공급이 증가했다는 것을 알 수 있음

* 선 그래프 선택 이유 : 시간에 따른 변화를 가장 잘 보여주겠다고 판단





카셰어링 문제 제시



문제 제시

카셰어링 사고 통계 분석 - 코드

```
# 문제 제시 - 카셰어링 사고 통계 (카셰어링 사고 통계, 운전자 연령별 구성비, 사고 운전자 연령)
# 카셰어링 사고 통계 -> 막대 그래프로 (사고건수, 부상자수, 사망자수) 를 보여줌

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

plt.rc('font', family = 'NanumBarunGothic')

df = pd.read_excel("/카셰어링_사고통계.xlsx")

# 데이터 가공 전 데이터 세팅
df.rename(columns = {'Unnamed: 0' : '분류'}, inplace = True)
df.set_index('분류', inplace = True)

df_acci_inju = df.loc[['사고건수', '부상자수']]
df_dead = df.loc['사망자수']

# 막대 그래프 bar() 으로 사고건수, 사망자수, 부상자수 를 한번에 보여줌

fig, axes = plt.subplots(1, 2, figsize=(20, 7))

# 첫 번째 서브플롯 (사고 건 수, 부상자 수)
df_acci_inju.T.plot(kind='bar', color=['red', 'green'], ax=axes[0])
axes[0].set_title('2017, 2019, 2021 - 사고 건 수, 부상자 수', size=20)
axes[0].set_ylabel('수')

axes[0].set_xticklabels(axes[0].get_xticklabels(), rotation = 0)

# 두 번째 서브플롯 (사망자 수)
df_dead.plot(kind='bar', color='yellow', ax=axes[1])
axes[1].set_title('2017, 2019, 2021 - 사망자 수', size=20)
axes[1].set_ylabel('수')
axes[1].legend(labels = ['사망자 수'], loc = 'best', fontsize = 15)

axes[1].set_xticklabels(axes[1].get_xticklabels(), rotation = 0)

# 전체 그래프에 대한 타이틀 추가
plt.suptitle('카셰어링 사고 통계', size=30)

# 서브플롯 간 간격 조정
plt.tight_layout()

plt.show()
```

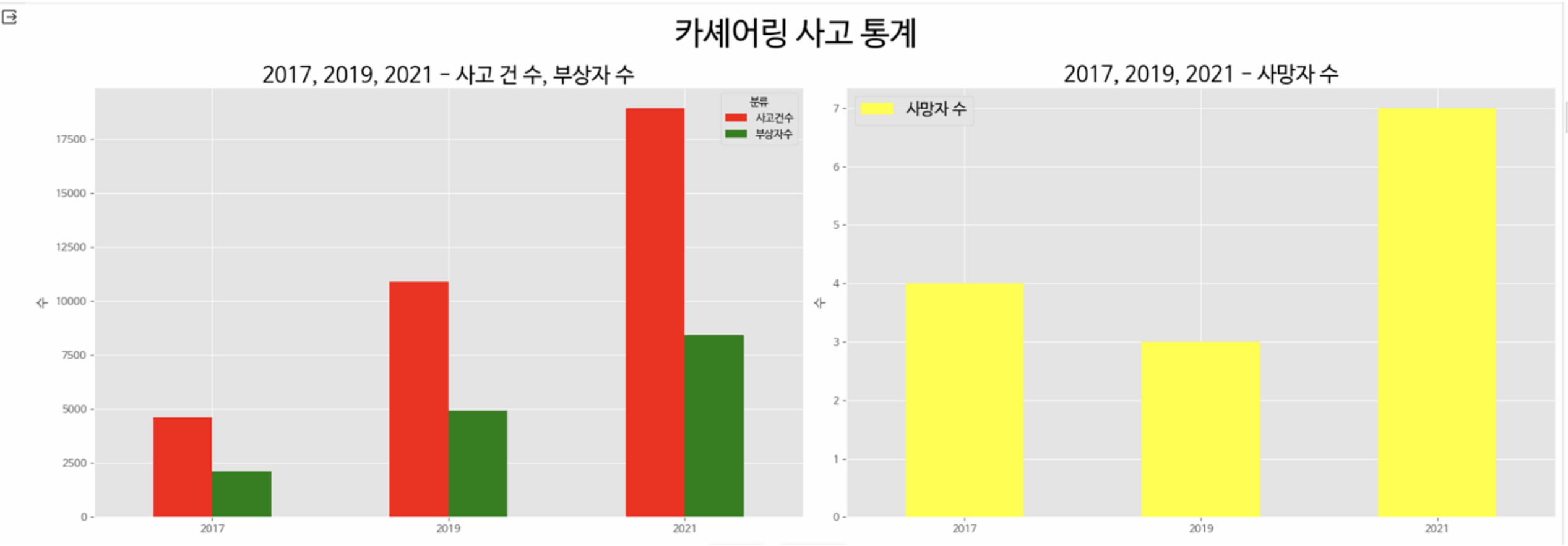
코드 분석

사고건수, 부상자 수, 사망자 수 열들을 선택하여 각각의 데이터 프레임 제작

수요 데이터 분석과는 다르게 bar() 을 이용한 시각화 이용

문제 제시

카셰어링 사고 통계 분석 - 시각화



시각화 결과 분석

2개년 단위로 사고 건 수와 부상자 수는 증가하였음

사망자 수는 2019년에 감소하였지만, 2021년 2배 이상 증가

* Bar 그래프 선정 이유

: 시간에 따른 변화를 보여주기보다 해당 년도에 발생한 사건의 추세를 보여주기에 Bar 그래프가 직관적으로 잘 들어온다고 판단

문제 제시

카셰어링 연령별 구성비 – 코드

```
# 카셰어링 연령별 구성비 현황
# 연령별 구성비에 따른 데이터 시각화 -> 막대 그래프 이용
import pandas as pd
import matplotlib.pyplot as plt

plt.rc('font', family = 'NanumBarunGothic')

# 데이터 로드
df = pd.read_excel('/카셰어링_연령별구성비.xlsx')

# 데이터 가공 전 데이터 세팅
df.rename(columns={'Unnamed: 0': '분류'}, inplace=True)
df.set_index('분류', inplace=True)

# 연령대별 데이터 선택
df_degree_age = df.loc[['20대', '30대', '40대', '50대']] /1000
df_20_age = df.loc[['20대초반', '20대후반']] /1000

# 막대 그래프 그리기
fig, axes = plt.subplots(2, 1, figsize=(10, 10))

# 20대, 30대, 40대, 50대의 막대 그래프
df_degree_age.T.plot(kind='bar', ax=axes[0], cmap='viridis')
axes[0].set_title('20대, 30대, 40대, 50대 연령별 구성비', size=16)
axes[0].set_xlabel('연도', size=14)
axes[0].set_ylabel('인원수', size=14)
axes[0].legend(title='연령대', bbox_to_anchor=(1.05, 1), loc='upper left')
axes[0].set_xticklabels(axes[0].get_xticklabels(), rotation = 0)

for ax in axes:
    legend_text = ax.legend(title='단위 : 천(1000)', bbox_to_anchor=(1.05, 1), loc='upper left').get_texts()
    plt.setp(legend_text)
```

코드 분석

시각화하고자 하는 연령대별 열들을 선택함

사고 통계 시각화와 동일하게 막대 그래프로 시각화 진행

'20대 ~ 50대' 데이터와 '20대 초반 ~ 20대 후반' 두 데이터를 시각화하기 위해 서브 플롯을 나눔

문제 제시

카셰어링 연령별 구성비 – 코드

```
# 20대 초반, 20대후반의 막대 그래프
df_20_age.T.plot(kind='bar', ax=axes[1], cmap='viridis')
axes[1].set_title('20대 초반과 20대 후반의 연령별 구성비', size=16)
axes[1].set_xlabel('연도', size=14)
axes[1].set_ylabel('인원수', size=14)
axes[1].legend(title='연령대', bbox_to_anchor=(1.05, 1), loc='upper left')
axes[1].set_xticklabels(axes[1].get_xticklabels(), rotation = 0)

for ax in axes:
    legend_text = ax.legend(title='단위 : 천(1000)', bbox_to_anchor=(1.05, 1), loc='upper left').get_texts()
    plt.setp(legend_text)

# 전체 그래프에 대한 타이틀 추가
plt.suptitle('카셰어링 이용자들의 연령별 구성비', size=20)

# 서브플롯 간 간격 조정
plt.tight_layout()

plt.show()
```

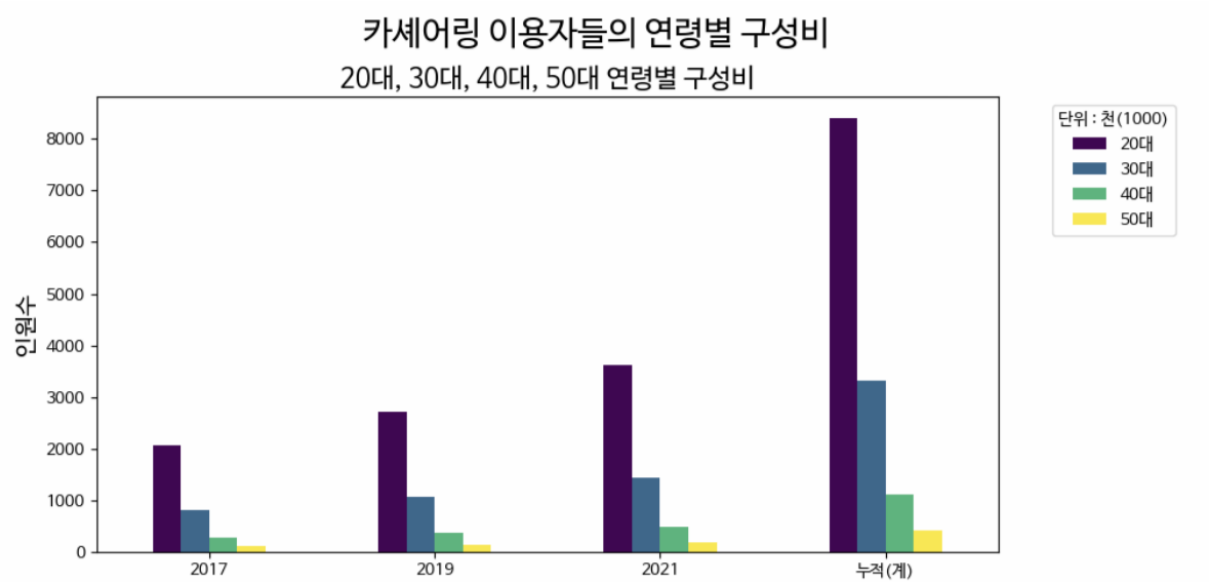
코드 분석

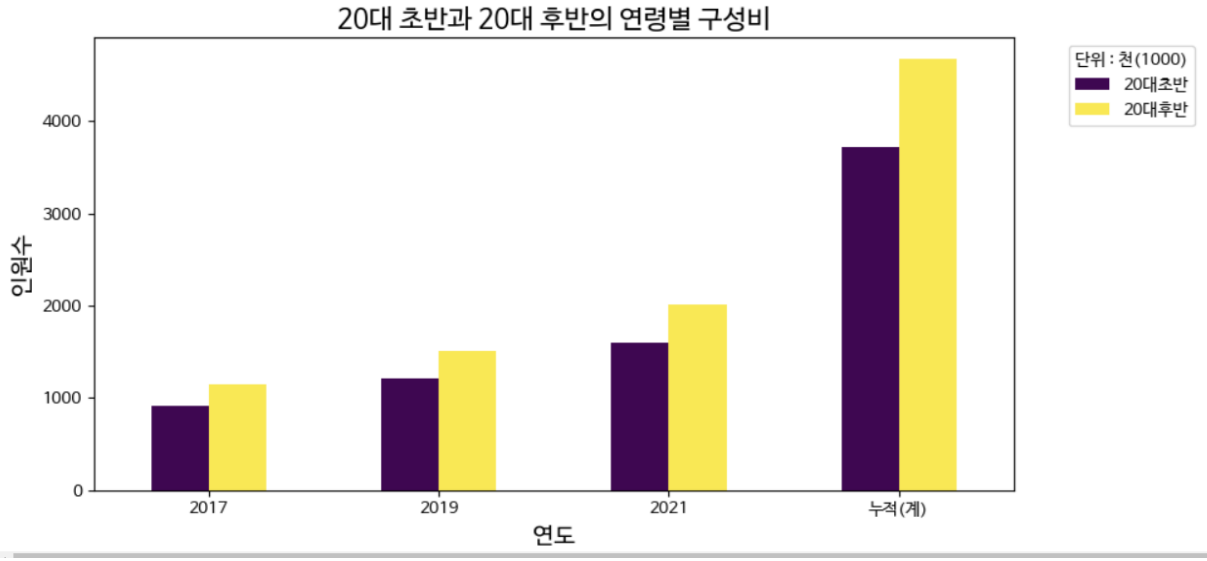
20대 초반, 20대 후반 데이터도 이전 데이터와 동일하게 진행

subtitle을 통한 전체 그래프에 대한 제목 추가

문제 제시

카셰어링 연령별 구성비 – 시각화





시각화 결과 분석

시각화 결과 20대 이용자가 압도적으로 많았음

이에 따른 20대 초반과 후반의 이용자 수를 시각화한 결과
20대 초반보다 20대 후반 이용자 수가 더 많았음

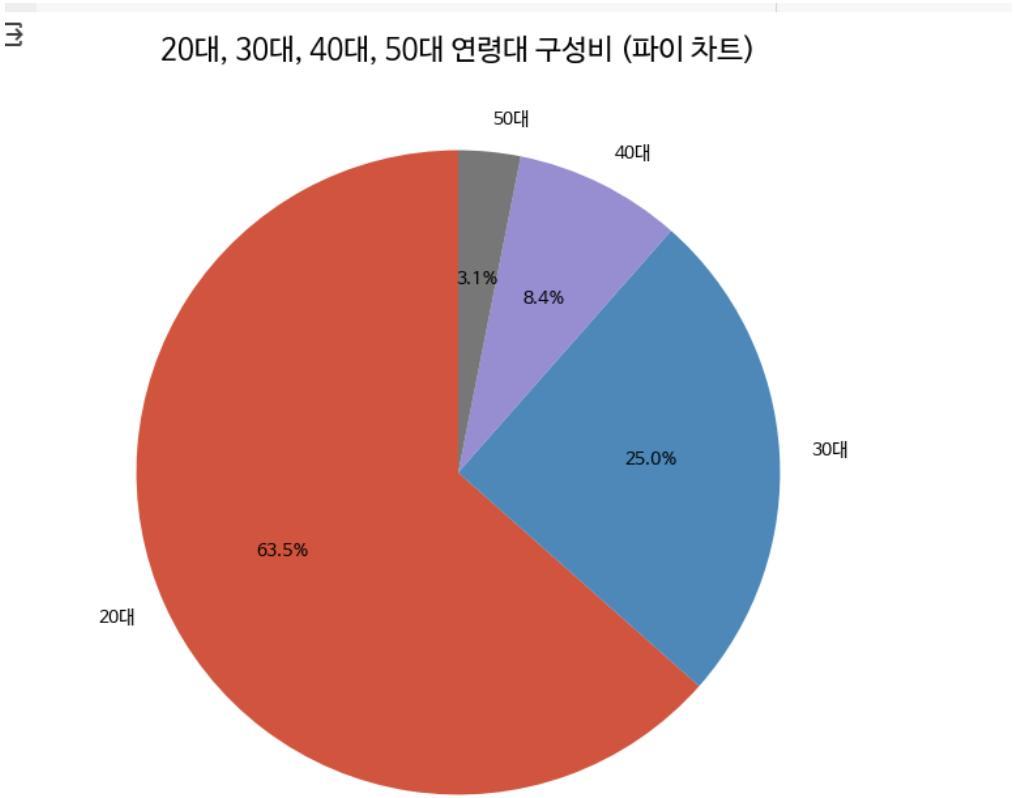
* 막대 그래프 선택 이유

: 이전 시각화 결과와 동일하게 연도별 각 연령대를 파악하기에
막대 그래프가 가장 직관적이라고 판단

문제 제시

카셰어링 연령별 구성비 – 코드 및 시각화

```python
import pandas as pd
import matplotlib.pyplot as plt

# 데이터 로드
df = pd.read_excel('/카셰어링_연령별구성비.xlsx')

# 데이터 가공 전 데이터 세팅
df.rename(columns={'Unnamed: 0': '분류'}, inplace=True)
df.set_index('분류', inplace=True)

# 연령대별 데이터 선택
df_degree_age = df.loc[['20대', '30대', '40대', '50대']]

# 파이 차트 그리기
fig, ax = plt.subplots(figsize=(8, 8))

# 20대, 30대, 40대, 50대의 파이 차트
df_degree_age.T.sum().plot(kind='pie', autopct='%1.1f%%', startangle=90, ax=ax)
ax.set_title('20대, 30대, 40대, 50대 연령대 구성비 (파이 차트)', size=16)

plt.show()
```



코드 분석 및 시각화 결과 분석

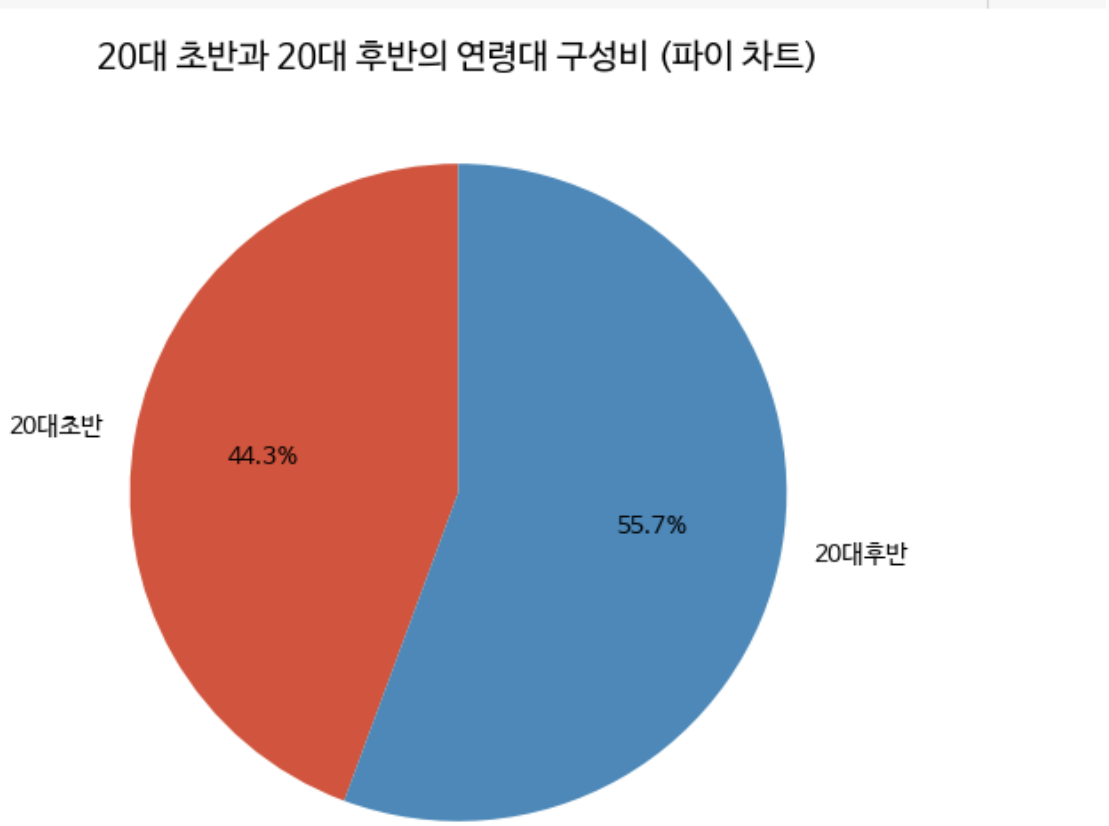
비율로 표현해서 이용자 수를 파악하기 위해 파이 차트 이용

비율로 파악한 결과 20대가 63.5 % 의 점유율을 차지하고 있었음

문제 제시

카셰어링 연령별 구성비 – 코드 및 시각화

```
[ ] # 20대 초반, 20대 후반에 대한 파이 차트로 비율 확인
    import pandas as pd
    import matplotlib.pyplot as plt

    # 데이터 로드
    df = pd.read_excel('/카셰어링_연령별구성비.xlsx')

    # 데이터 가공 전 데이터 세팅
    df.rename(columns={'Unnamed: 0': '분류'}, inplace=True)
    df.set_index('분류', inplace=True)

    # 연령대별 데이터 선택
    df_20_age = df.loc[['20대초반', '20대후반']]

    # 파이 차트 그리기
    fig, axes = plt.subplots(1, 1, figsize=(12, 6))

    # 20대 초반, 20대 후반의 파이 차트
    df_20_age.T.sum().plot(kind='pie', autopct='%1.1f%%', startangle=90, ax=axes)
    axes.set_title('20대 초반과 20대 후반의 연령대 구성비 (파이 차트)', size=14)

    plt.show()
```



문제 제시

카셰어링 사고 운전자 연령별 구성비 – 코드

```
# 카셰어링 사고 운전자 연령별 구성비 현황
# 위와 동일하게 막대그래프 및 파이 차트로 확인

import pandas as pd
import matplotlib.pyplot as plt

# 데이터 로드
df = pd.read_excel("/카셰어링_사고운전자_연령별구성비.xlsx")

# 데이터 가공 전 데이터 세팅
df.rename(columns={'Unnamed: 0': '분류'}, inplace=True)
df.set_index('분류', inplace=True)

# 연령대별 데이터 선택
df_degree_age = df.loc[['20대', '30대이상']]
df_20s_age = df.loc[['20대초반', '20대후반']]

# 막대 그래프 그리기
fig, axes = plt.subplots(1, 2, figsize=(12, 6))

# 20대, 30대이상의 막대 그래프
df_degree_age.T.plot(kind='bar', ax=axes[0])
axes[0].set_title('20대와 30대이상의 운전자 연령별 구성비', size=14)
axes[0].set_xlabel('연도', size=12)
axes[0].set_ylabel('운전자 수', size=12)
axes[0].legend(title='연령대', bbox_to_anchor=(1.05, 1), loc='upper left')
axes[0].set_xticklabels(axes[0].get_xticklabels(), rotation = 0)

# 20대초반, 20대후반의 막대 그래프
df_20s_age.T.plot(kind='bar', ax=axes[1])
axes[1].set_title('20대 초반과 20대 후반의 운전자 연령별 구성비', size=14)
axes[1].set_xlabel('연도', size=12)
axes[1].set_ylabel('운전자 수', size=12)
axes[1].legend(title='연령대', bbox_to_anchor=(1.05, 1), loc='upper left')
axes[1].set_xticklabels(axes[1].get_xticklabels(), rotation = 0)

# 전체 그래프에 대한 타이틀 추가
plt.suptitle('카셰어링 사고 운전자 연령별 구성비', size=16)

# 서브플롯 간 간격 조정
plt.tight_layout()
```

```
# 파이 차트 그리기
fig, axes = plt.subplots(1, 2, figsize=(12, 6))

# 20대, 30대이상의 파이 차트
df_degree_age.T.sum().plot(kind='pie', autopct='%1.1f%%', startangle=90, ax=axes[0])
axes[0].set_title('20대와 30대이상의 운전자 연령별 구성비', size=14)

# 20대초반, 20대후반의 파이 차트
df_20s_age.T.sum().plot(kind='pie', autopct='%1.1f%%', startangle=90, ax=axes[1])
axes[1].set_title('20대 초반과 20대 후반의 운전자 연령별 구성비', size=14)

# 전체 그래프에 대한 타이틀 추가
plt.suptitle('카셰어링 사고 운전자 연령별 구성비', size=16)

# 서브플롯 간 간격 조정
plt.tight_layout()

plt.show()
```

코드 분석

이용자 수를 바탕으로 사고 운전자를 분석하기 위한 코드를 작성

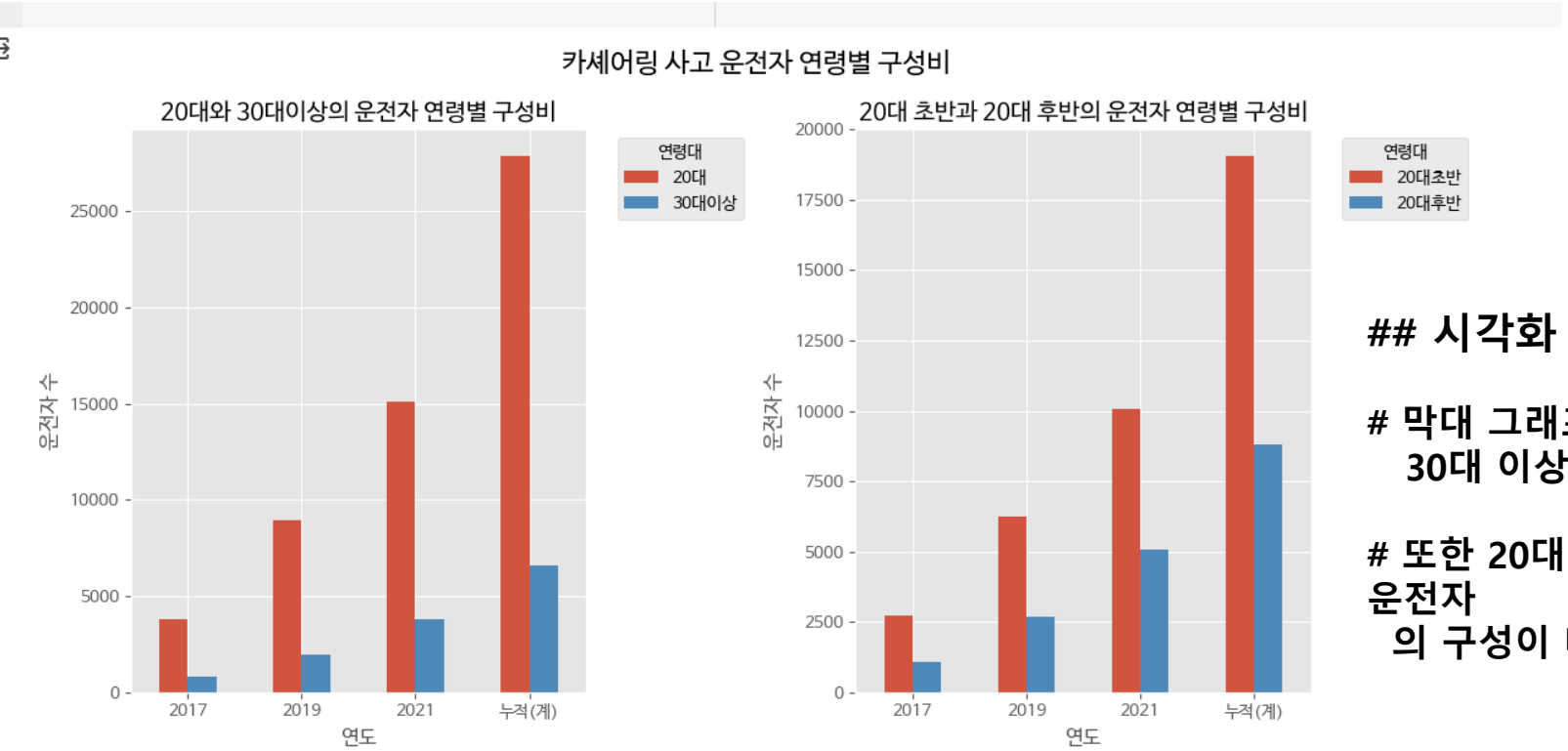
20대가 가장 많았으므로 20대에 대한 데이터와, 30대 이상에 대한 데이터
를 바탕으로 데이터 가공 진행

20대 내에서도 초반&후반에 대한 사고 데이터 분석을 하기 위해 코드작성

막대 그래프와 파이 차트로 그래프 그리기 진행

문제 제시

카셰어링 사고 운전자 연령별 구성비 – 시각화



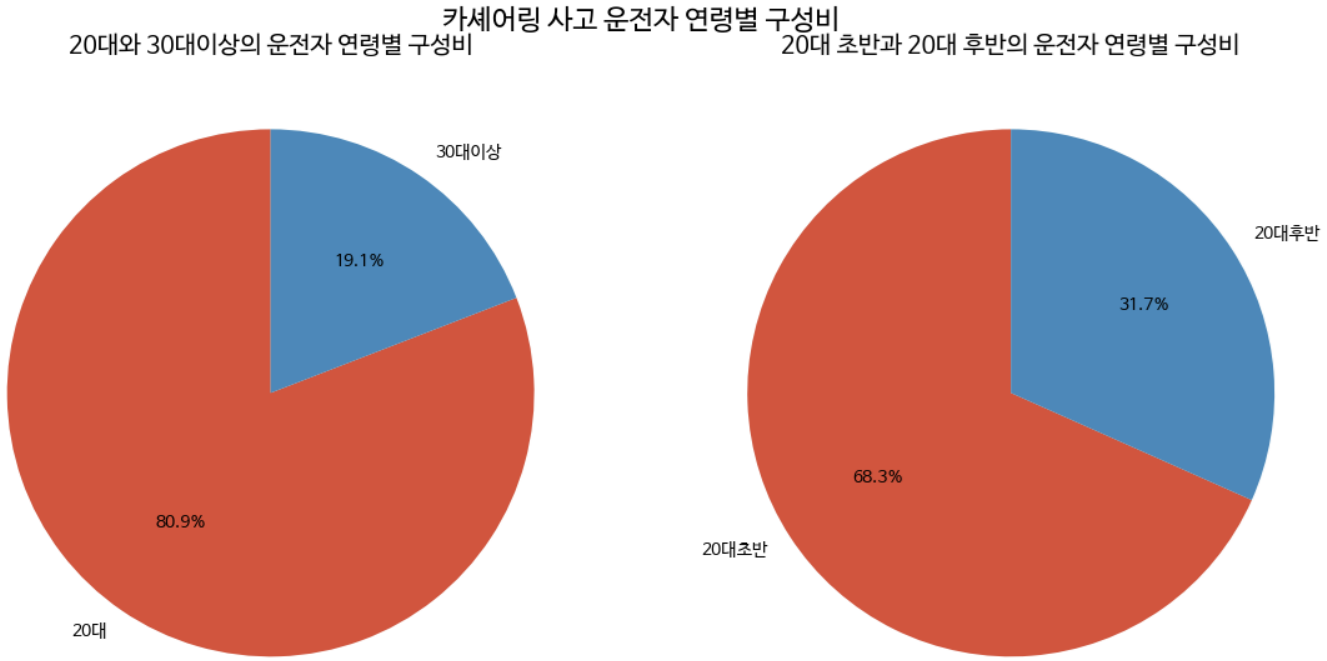
시각화 결과 분석

막대 그래프로 시각화 결과 사고 운전자에 대한 연령별 구성이 30대 이상 운전자들보다 20대가 많은 걸 알 수 있음

또한 20대 초반과 20대 후반 내애서는 20대 초반이 사고 운전자
의 구성이 더 많았음

문제 제시

카셰어링 사고 운전자 연령별 구성비 – 시각화



시각화 결과 분석

파이 차트로 비율을 시각화한 결과

20대가 전체 중 80.9%의 사고 운전자 연령 비율을 보여줌

20대 초반과 후반에서는 20대 초반이 68.3%의 비율을 가지
있음

문제 제시

사고 통계별 면허 취득 기간 분석 - 코드

```
# 문제 제시 - 사고 통계별 면허 취득 기간
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

plt.rc('font', family = 'NanumBarunGothic')

df = pd.read_excel("/면허취득경과년수_사고데이터.xls")

# 공백 제거
df.rename(columns = {'가해운전자 면허취득경과년수 ': '사고운전자 면허취득경과수'}, inplace = True)

df['사고운전자 면허취득경과수'] = df['사고운전자 면허취득경과수'].astype(str)

# 시각화에 필요한 데이터로 설정
df_2 = df.iloc[1:9, [0, 1, 3, 5]]
df_2[['2017', '2019', '2021']] = df_2[['2017', '2019', '2021']].astype(int)

# 시각화 코드
fig, axes = plt.subplots(1, 3, figsize=(20, 5), sharex=True)

sns.barplot(x='사고운전자 면허취득경과수', y='2017', data=df_2, ax=axes[0], dodge = 10)
axes[0].set_title('2017')

sns.barplot(x='사고운전자 면허취득경과수', y='2019', data=df_2, ax=axes[1], dodge = 10)
axes[1].set_title('2019')

sns.barplot(x='사고운전자 면허취득경과수', y='2021', data=df_2, ax=axes[2], dodge = 10)
axes[2].set_title('2021')

plt.show()
```

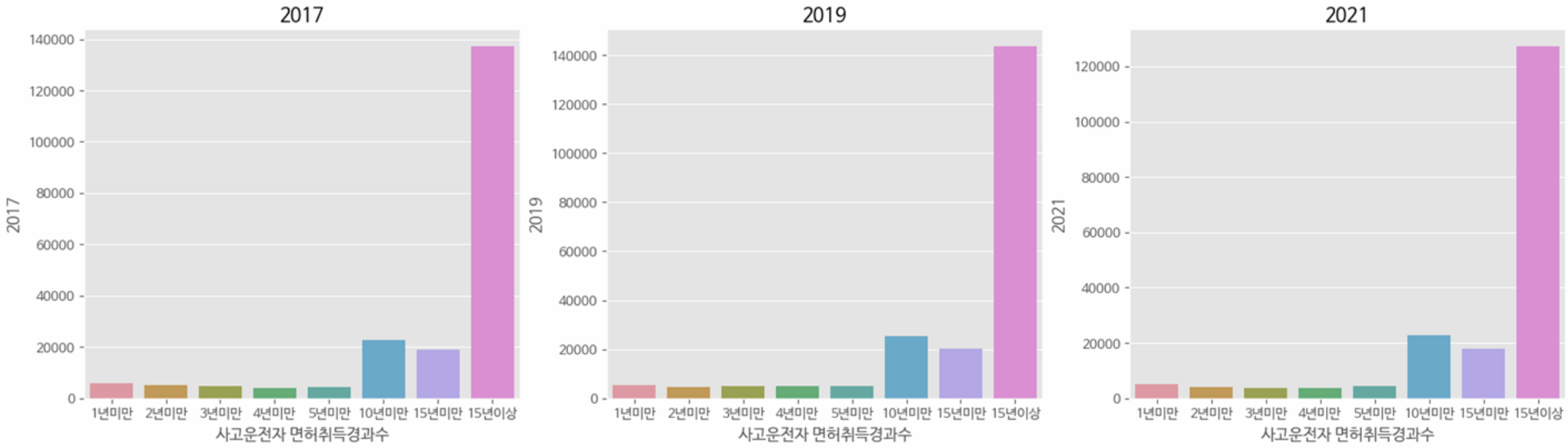
코드 분석

사고 대표 유형 중 하나인 초보 운전과의 관계를 파악하기 위해
  면허 취득 경과 수에 따른 데이터를 분석

barplot()의 시각화를 통해 진행

문제 제시

사고 통계별 면허 취득 기간 분석 - 시각화



시각화 결과 분석

예상과 다르게 15년 이상 운전자들이 압도적으로 사고 통계가 많았음

기사를 찾아보니 위와 같은 내용이 있음

경력이 오래된 운전자들은 차량을 부주의하게 조작하거나 차선을 갑자기 변경하고, 교통 상황이 좋지 않거나 속도제한이 있는 구간에서도 지나친 속도를 내는 경향 탓에 사고 위험이 더 크다는 연구 결과도 있다.

출처 : 세계일보

[1년 미만, 5년 미만] 운전자 들은 면허 취득 기간이 지날 수록 사고 발생 건이 감소하는 추세를 보임

솔루션

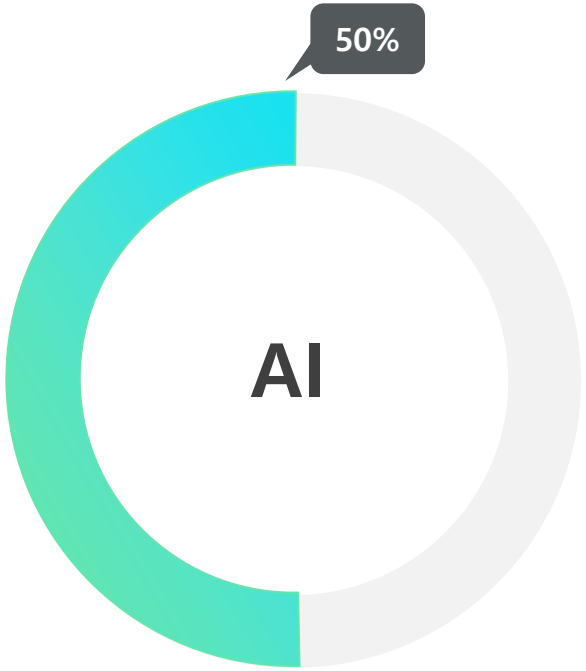
데이터 분석을 통한 정리

- **카셰어링 이용자 수, 서비스 존, 차량 수 증가 -> 수요의 증가**
- **그에 따른 사고 발생도 증가**
- **연령별 구성비 분석 -> 20대 이용자가 압도적으로 많았음**
- **사고 운전자 분석 -> 20대 이용자가 많았음**
- **면허 취득 기간에 따른 분석 -> 취득 기간 5년 미만 까지는 감소하는 추세를 보임**
- **15년 이상 운전자 사고 비율 압도적으로 많음 -> 부주의로 인한 사고 발생**

문제를 해결하기 위한 AI 제시

" 사용자의 주행 패턴을
통해 점수를 산출해주는
AI가 어느정도 해결해주지
않을까? "

AI 모델에 대한 설명

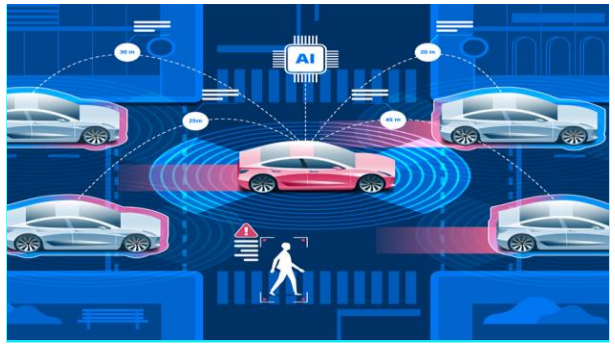


프로그램 작동 방식

“ 주행하기 전 사용자 정보 입력 ”

“ 주행하는 동안 주행 상태에 대한 데이터를 추출”

“ 주행 후 최종 데이터를 바탕으로 점수 산출 ”



주행 전

- 사용자로부터 사전 데이터 입력 받음
- 입력받은 데이터로 부터 점수 산출
- 점수 산출은 사전에 정한 기준을 바탕으로 점수 계산

주행 중

- 주행하는 동안 실시간으로 주행 정보 데이터를 추출
- 깜박이 횟수와 속도 변화를 카운트

주행 후

- 주행 전 데이터와 추출된 주행 중 데이터 엑셀로 보유
- 엑셀 데이터 값을 바탕으로 최종 점수 산출

프로그램 코드 설명 – 주행 전

```
# *** 주행 전 사용자 입력 부분 ***
# - 사용자 입력 데이터 -> 연령대, 사고 전적, 면허 취득 기간
# 사용자로 부터 입력 받는 데이터의 점수는 총점의 40%를 차지
# 사고 전적 -> 50%, 면허 취득 기간 -> 25%, 연령대 -> 25%

# 사고 전적에 대한 구체화 - 배점 100점
# 사고 있으면 -> 70점, 사고 없으면 -> 100점

# 연령대에 대한 구체화 - 배점 100점
# 20대 초반 -> 60점, 20대 후반 -> 70점
# 30대 -> 80점, 40대 -> 90점, 50대 이상 -> 100점

# 면허 취득 기간에 따른 점수 배점
# 1년 미만 -> 30점, 2년 미만 -> 40점, 3년 미만 -> 50점, 4년 미만 -> 60점
# 5년 미만 -> 70점, 10년 미만 -> 60점, 15년 미만 -> 50점, 15년 이상 -> 40점
```

주석 설명

 # 사용자 입력은
- 연령대
- 사고 전적
- 면허 취득 기간

: 데이터 분석을 바탕으로 얻어낸 결론에서 사고 전적을 추가하여
입력 받도록 함

: 사고 전적이 있으면 낮은 점수를 얻음 (50%)

: 20대 초반에 가까운 이용자일수록 낮은 점수를 얻음 (25%)

: 15년 이상 운전자는 부주의 위험성때문에 40점 배점하고,
운전 면허 기간이 짧을 수록 초보 운전에 가깝다 판단하여
낮은 점수 부여 (25%)

***** 사용자로 부터 입력 받는 주행 전 데이터는 총점의 40% 차지하도록 함 *****

주행 전 – main 함수

```
# main 함수
def main():
    print("=======================================")
    print("=======                         =======")
    print("====== 카셰어링 점수 산출 시스템 ======")
    print("=======                         =======")
    print("=======================================")
    print("₩n 주행 전 고객님에 대한 사전 정보를 입력받습니다.")

    # Question 객체 생성
    q = Question()

    # 잘못된 입력은 다시 입력받도록 함
    while True:
        start_question = input("₩n 시작하시겠습니까 ? (네/아니요)₩n")
        if start_question == "네":
            q.ask_first_question() # 첫 번째 질문 진행
            q.ask_second_question() # 두 번째 질문 진행
            q.ask_third_question() # 세 번째 질문 진행
            break
        elif start_question == "아니요":
            print("프로그램을 종료합니다")
            break
        else:
            print("잘못된 입력입니다. 다시 입력바랍니다.")
```

코드 분석

 # 질문 객체를 생성하여 첫 번째, 두 번째, 세 번째 질문과 답변을 진행

 # 잘못된 입력은 다시 입력 받도록 진행

주행 전 – main 함수

```
    # 질문이 모두 끝나면 점수를 반환하여 출력해줌
    if q.get_count() == 3:
        score = q.cal_score()
        print("₩n========================================")
        print("₩n고객님의 점수는", score, "입니다.₩n")
        print("========================================")
        print("점수를 참고하셔서 안전 운전 부탁드립니다.")
        print("주행을 시작합니다")

        # 입력받은 데이터를 바탕으로 데이터프레임 생성
        data = {
            "과거 사고 전적": [q.first_answer],
            "연령대": [q.second_answer],
            "면허 취득 기간": [q.third_answer],
            "사고 전적 점수": [q.first_question_score],
            "연령대 점수": [q.second_question_score],
            "면허 취득 기간 점수": [q.third_question_score],
            "총점": [score]
        }

        df = pd.DataFrame(data)

        # 엑셀 파일로 저장
        df.to_excel("client_score.xlsx", index=False)
        print("데이터를 엑셀 파일로 저장했습니다: client_score.xlsx")


if __name__ == "__main__":
    main()
```

코드 분석

질문과 답변이 끝나면 점수를 반환받아 출력하도록 함

그 후 사용자 정보를 데이터프레임으로 만들어 엑셀 파일로 저장

프로그램 설명

프로그램 코드 설명_ 주행 전 첫 번째 함수

```
# 데이터프레임을 사용하기 위한 pandas import
import pandas as pd

# 필요한 변수들 초기화 되도록 함
class Question:
    def __init__(self):
        self.first_question = ""
        self.first_answer = ""
        self.first_question_score = 0
        self.second_question_score = 0
        self.third_question_score = 0
        self.question_count = 0
        self.second_answer = 0
        self.third_answer = 0
        self.your_score = 0.0

    # 첫 번째 질문
    def ask_first_question(self):
        print("========== 질문 1 ==========")
        while True:
            self.first_answer = input("과거 사고 전적이 있으십니까? (네/아니요)₩n")
            if self.first_answer == "네":
                self.first_question_score = 70
                self.question_count += 1
                break
            elif self.first_answer == "아니요":
                self.first_question_score = 100
                self.question_count += 1
                break
            else:
                print("잘못된 입력입니다. 다시 입력 바랍니다.")
```

코드 분석

필요한 변수들 초기화 진행

첫 번째 질문 진행 및 첫 번째 질문에 대한 점수 부여

프로그램 설명

프로그램 코드 설명– 주행 전 두 번째& 세 번째 함수

```python
# 두 번째 질문
def ask_second_question(self):
    print("========== 질문 2 ==========")
    while True:
        self.second_answer = int(input("현재 본인의 나이를 입력해주세요: "))
        if 20 <= self.second_answer < 25:
            self.second_question_score = 60
            self.question_count += 1
            break
        elif 25 <= self.second_answer < 30:
            self.second_question_score = 70
            self.question_count += 1
            break
        elif 30 <= self.second_answer < 40:
            self.second_question_score = 80
            self.question_count += 1
            break
        elif 40 <= self.second_answer < 50:
            self.second_question_score = 90
            self.question_count += 1
            break
        elif self.second_answer >= 50:
            self.second_question_score = 100
            self.question_count += 1
            break
        else:
            print("잘못된 입력입니다. 다시 입력 바랍니다.")
```

```python
# 세 번째 질문
def ask_third_question(self):
    print("========== 질문 3 ==========")
    while True:
        self.third_answer = int(input("면허 취득 경과 기간을 입력해주세요 (입력 단위 : 년(year), 1년이 경과하지 않았다면 0으로 입력해주세요.)₩n"))
        if self.third_answer < 1:
            self.third_question_score = 30
            self.question_count += 1
            break
        elif self.third_answer < 2:
            self.third_question_score = 40
            self.question_count += 1
            break
        elif self.third_answer < 3:
            self.third_question_score = 50
            self.question_count += 1
            break
        elif self.third_answer < 4:
            self.third_question_score = 60
            self.question_count += 1
            break
        elif self.third_answer < 5:
            self.third_question_score = 70
            self.question_count += 1
            break
        elif self.third_answer < 10:
            self.third_question_score = 60
            self.question_count += 1
            break
        elif self.third_answer < 15:
            self.third_question_score = 50
            self.question_count += 1
            break
        elif self.third_answer >= 15:
            self.third_question_score = 40
            self.question_count += 1
            break
        else:
            print("잘못된 입력입니다. 다시 입력 바랍니다.")
```

코드 분석

두 번째 질문과 세 번째 질문도 첫 번째 질문과 동일하게 진행

프로그램 설명

실행 화면

```
=======================================
=======                         =======
====== 카셰어링 점수 산출 시스템 ======
=======                         =======
=======================================

 주행 전 고객님에 대한 사전 정보를 입력받습니다.

 시작하시겠습니까 ? (네/아니요)
네
========== 질문 1 ==========
과거 사고 전적이 있으십니까? (네/아니요)
아니요
========== 질문 2 ==========
현재 본인의 나이를 입력해주세요: 24
========== 질문 3 ==========
면허 취득 경과 기간을 입력해주세요 (입력 단위 : 년(year), 1년이 경과하지 않았다면 0으로 입력해주세요.)
4
점수 산출을 시작합니다.

=======================================

고객님의 점수는 82.5 입니다.

=======================================
점수를 참고하셔서 안전 운전 부탁드립니다.
주행을 시작합니다
데이터를 엑셀 파일로 저장했습니다: client_score.xlsx
```

A1 | 과거 사고 전적

	A	B	C	D	E	F	G	H
1	거 사고 전	연령대	허 취득 기	고 전적 점	연령대 점수	취득 기간	총점	
2	아니요	24	4	100	60	70	82.5	

실행 화면 설명

** 코드 실행 화면 및 엑셀에 저장된 결과*

 # 프로그램과 질문 & 답변 식으로 진행

 # 결과는 엑셀에 저장하여 최종 점수 산출에 데이터가 되도록 함

주행 데이터 기준 선정

[운전자 위험운전 행동 11대 유형]

과속		급가속		급감속		급차로 변경 (초당회전각)		급회전 (누적회전각)	
과속	징기 과속	급가속	급출발	급감속	급정지	급진로변경 (15~30°)	급앞지르기 (30~60°)	급좌우회전 (60~120°)	급U턴 (160~180°)

프로그램 코드 설명 – 주행 데이터

```python
import pandas as pd
import numpy as np

# 랜덤 시드 설정
np.random.seed(42)

# 데이터 프레임 생성을 위한 빈 리스트
data = []

# 주행 시간 설정 (예: 10분 동안 1초 간격)
drive_time = 600  # 초 단위
time_interval = 1  # 초 단위로 변경

# 초기 상태 설정
current_speed = np.random.randint(20, 80)  # 초기 속도
blink_status = 0  # 초기 깜빡이 상태 (0: 꺼짐)
blink_count = 0  # 초기 깜빡이 조작 횟수
rapid_acceleration_count = 0  # 초기 급가속 횟수
rapid_deceleration_count = 0  # 초기 급감속 횟수
prev_blink_status = 0  # 이전 깜빡이 상태

# 최대 속도 및 최소 속도 설정
max_speed = 120
min_speed = -6
```

코드 분석

데이터프레임 활용을 위한 pandas import
난수 활용을 위한 numpy import

추적을 위해 랜덤 시드 설정

데이터프레임을 위한 빈 리스트 설정

초기 주행 환경 설정 (시간, 추적 간격)

데이터 초기화

주행 환경 추가 설정 (실생활과 가장 밀접하게 조성)

프로그램 설명

프로그램 코드 설명 – 주행 데이터

```python
# 주행 데이터 생성
for timestamp in range(0, drive_time, time_interval):
    # 현재 속도가 10 미만인 경우
    if current_speed < 10:
        # 랜덤 속도 증감율 생성 (-5에서 15 사이)
        speed_change = np.random.randint(-5, 16)
    # 현재 속도가 0인 경우
    elif current_speed == 0:
        # 랜덤 속도 증감율 생성 (0에서 15 사이)
        speed_change = np.random.randint(0, 16)
    else:
        # 그 외의 경우 랜덤 속도 증감율 생성 (-30에서 15 사이)
        speed_change = np.random.randint(-30, 16)

    # 특이 속도 증감율의 절대값이 11 이상인 경우 급가속 횟수 증가
    if speed_change > 10:
        rapid_acceleration_count += 1
    # 감속 횟수 증가
    elif speed_change < -7.5:
        rapid_deceleration_count += 1

    current_speed += speed_change

    # 최대 및 최소 속도 제한
    current_speed = max(min(current_speed, max_speed), min_speed)

    # 랜덤 깜빡이 상태 업데이트
    blink_status = np.random.choice([0, 1])

    # 깜빡이 상태가 0에서 1로 변하면서 blink_count 증가
    if blink_status == 1 and prev_blink_status == 0:
        blink_count += 1
```

코드 분석

주어진 초기 주행 환경 설정을 토대로 주행 데이터 기록

난수로 구성하되 국내 도심 주행 환경에 맞게 난수 값 보정

급가속/급감속 횟수 기록

난수 값 추가 보정

깜빡이 등 점멸 횟수 기록

프로그램 코드 설명 – 주행 데이터

```
    # 데이터를 리스트에 추가
    data.append([timestamp, current_speed, speed_change, blink_status, blink_count, rapid_acceleration_count, rapid_deceleration_count])

    # 이전 깜빡이 상태 업데이트
    prev_blink_status = blink_status

# 데이터 프레임 생성
columns = ['Time', 'Current Speed', 'Speed Change', 'Blink Status', 'Blink Count', 'Rapid Acceleration Count', 'Rapid Deceleration Count']
df = pd.DataFrame(data, columns=columns)

# 운행 시간 포맷 변경 (초를 분:초로 변환)
df['Time'] = pd.to_datetime(df['Time'], unit='s').dt.strftime('%M:%S')

# "Time" 열 이름으로 변경
df.rename_axis("Time", inplace=True)

# 전치한 데이터프레임을 엑셀 파일로 내보내기
excel_filename_transposed = "AIProjectCarDriveData.xlsx"
df.to_excel(excel_filename_transposed, index=True, header=True)

print(f"차량 주행 데이터가 성공적으로 내보내졌습니다.")
```

코드 분석

데이터프레임에 맞는 형식으로 변환

데이터프레임으로 변환 후 엑셀 파일로 내보내기

최종 메시지 출력

프로그램 코드 설명– 주행 데이터 실행 결과(엑셀)

Time	Time	Current Speed	Speed Change	Blink Status	Blink Count	Rapid Acceleration Count	Rapid Deceleration Count
0	00:00	56	-2	0	0	0	0
1	00:01	68	12	1	1	1	0
2	00:02	58	-10	0	1	1	1
3	00:03	46	-12	0	1	1	2
4	00:04	26	-20	0	1	1	3
5	00:05	19	-7	0	1	1	3
6	00:06	24	5	1	2	1	3
7	00:07	17	-7	0	2	1	3
8	00:08	8	-9	0	2	1	4
9	00:09	4	-4	1	3	1	4
10	00:10	10	6	1	3	1	4
11	00:11	17	7	1	3	1	4
12	00:12	7	-10	0	3	1	5
13	00:13	13	6	1	4	1	5
14	00:14	4	-9	0	4	1	6
15	00:15	10	6	0	4	1	6
16	00:16	6	-4	0	4	1	6
17	00:17	10	4	1	5	1	6
18	00:18	7	-3	1	5	1	6
19	00:19	17	10	0	5	1	6

프로그램 설명

프로그램 코드 설명– 최종 분석 코드

```python
import pandas as pd

# 데이터프레임 A 읽어오기
df_A = pd.read_excel('AIProjectCarDriveData.xlsx', index_col='Time')

# 주행 데이터에 대한 점수 계산
initial_score = 100
acceleration_penalty = 10 * (df_A['Rapid Acceleration Count'] >= 5).iloc[-1]
deceleration_penalty = 10 * (df_A['Rapid Deceleration Count'] >= 10).iloc[-1]
blink_penalty = 15 * (df_A['Blink Count'] <= 15).iloc[-1]
speed_penalty = 50 * (df_A['Current Speed'] > 130).iloc[-1]

# 주행데이터에 대한 총 점수 계산
df_A['Driving Score'] = initial_score - acceleration_penalty - deceleration_penalty - blink_penalty - speed_penalty

# 데이터프레임 B 읽어오기
df_B = pd.read_excel('client_score.xlsx')

# 사용자 점수와 주행데이터 점수의 평균 계산
df_B['Final Score'] = (df_A['Driving Score'].iloc[-1] + df_B['총점']) / 2

# 두 데이터프레임을 결합
df_final = pd.concat([df_A, df_B], axis=1)

# 결합된 데이터프레임을 엑셀 파일로 내보내기
excel_filename_final = "Final_analysis.xlsx"
df_final.to_excel(excel_filename_final, index=True, header=True)

print(f"최종 데이터가 성공적으로 내보내졌습니다. 파일명: {excel_filename_final}")
```

import

불러오기

의거하여 점수 산출

불러오기

산출

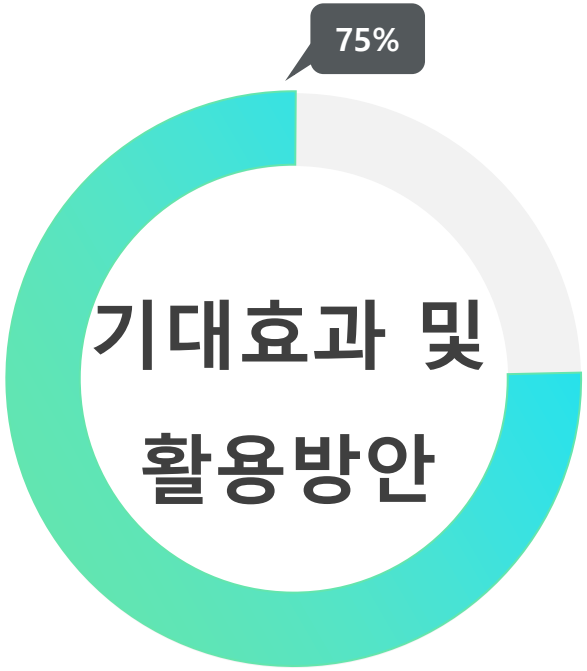
하나의 데이터로 통합

프로그램 설명

프로그램 코드 설명 – 최종 분석 코드 실행 결과(엑셀)

	Time.1	Current Speed	Speed Change	Blink Status	Blink Count	Rapid Acceleration Count	Rapid Deceleration Count	Driving Score	과거 사고 전적	연령대	면허 취득 기간	사고 전적 점수	연령대 점수	면허 취득 기간 점수	총점	Final Score
0	00:00	56	-2	0	0	0	0	80	네	24	2	70	60	50	62.5	71.25
1	00:01	68	12	1	1	1	0	80								
2	00:02	58	-10	0	1	1	1	80								
3	00:03	46	-12	0	1	1	2	80								
4	00:04	26	-20	0	1	1	3	80								
5	00:05	19	-7	0	1	1	3	80								
6	00:06	24	5	1	2	1	3	80								
7	00:07	17	-7	0	2	1	3	80								
8	00:08	8	-9	0	2	1	4	80								
9	00:09	4	-4	1	3	1	4	80								
10	00:10	10	6	1	3	1	4	80								
11	00:11	17	7	1	3	1	4	80								

해당 프로젝트 AI 모델 활용 방안



CONTENTS

해당 프로젝트 AI 모델 활용 방안

카셰어링 업계 리스크 감소 기대

인센티브/패널티 부과 등으로 인한 긍정적 효과 기대

A

보험료 차등 적용

미숙 운전자에게 추가 비용 적용

B

고급/신차 우선 배차

숙련된(점수가 높은) 운전자 인센티브 제공

C

안전운전 캠페인

개개인 맞춤형 제안으로 더욱 안전한 도로 만들기

마무리