# F1 피트스톱(Pit-Stop) 최적화 전략



# 프로젝트 개요

### F1에서는 어떻게 하면 이기는 건데? 😲

- 1. 우선 팀이 제일 빠른 차를 만든다
- 2. 드라이버가 실수없이 잘탄다
- 3. 팀이 전략을 잘 가져오고 잘 실행한다





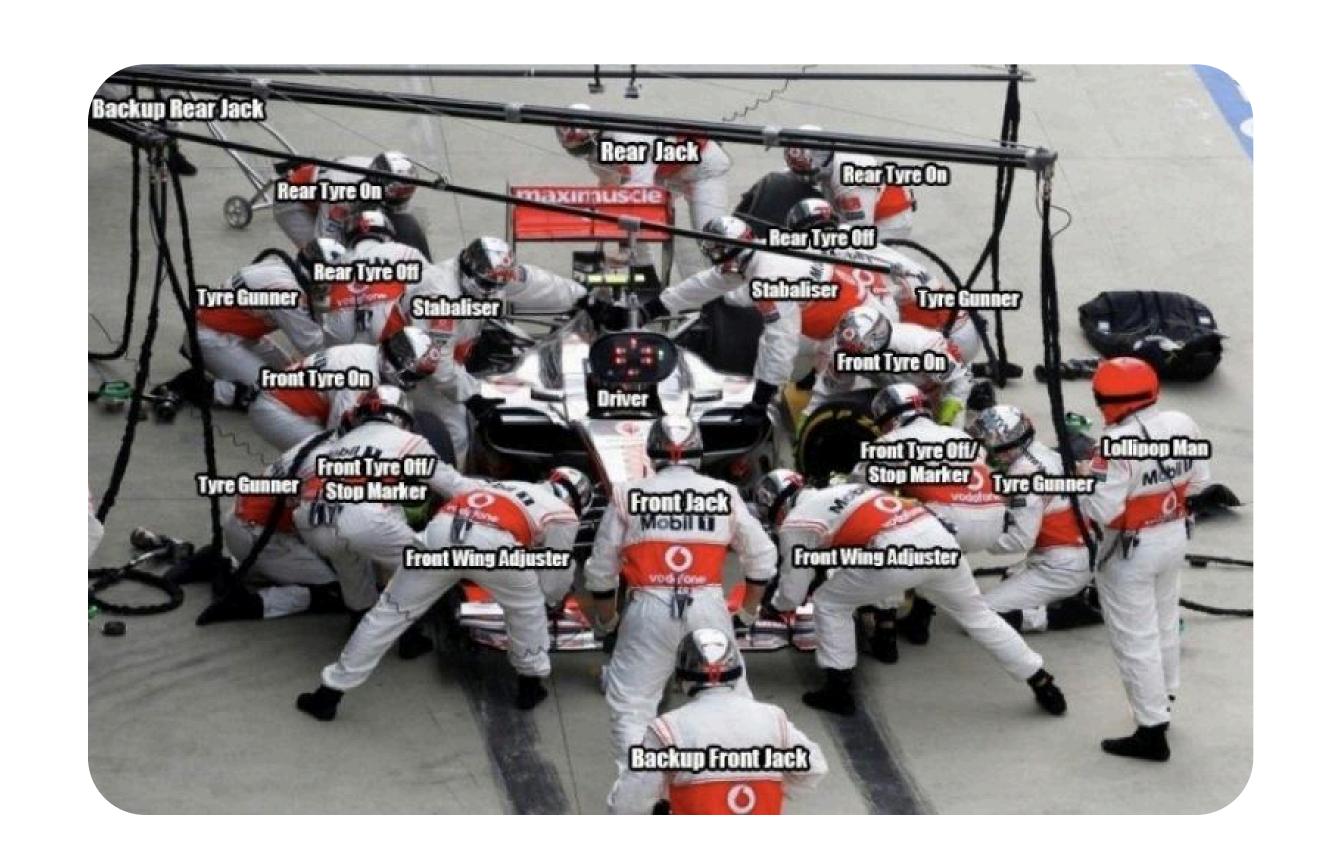
# 프로젝트 개요

### F1에서는 어떻게 하면 이기는 건데? 😲



- 1. 우선 팀이 제일 빠른 차를 만든다
- 2. 드라이버가 실수없이 잘탄다
- 3. 팀이 전략을 잘 가져오고 잘 실행한다







# 데이터셋설명

Ergast Developer API...

### [pit\_stops, lap\_times, races, results, circuits]

이름	수정한 날짜	유형	크기
circuits	2025-03-10 오후 9:36	Microsoft Excel 쉼	10KB
constructor_results	2025-03-10 오후 9:36	Microsoft Excel 쉼	215KB
constructor_standings	2025-03-10 오후 9:36	Microsoft Excel 쉼	310KB
constructors	2025-03-10 오후 9:36	Microsoft Excel 쉼	18KB
driver_standings	2025-03-10 오후 9:36	Microsoft Excel 쉼	864KB
drivers	2025-03-10 오후 9:36	Microsoft Excel 쉼	93KB
lap_times	2025-03-10 오후 9:36	Microsoft Excel 쉼	17,210KB
pit_stops	2025-03-10 오후 9:36	Microsoft Excel 쉼	434KB
qualifying	2025-03-10 오후 9:36	Microsoft Excel 쉼	455KB
races	2025-03-10 오후 9:36	Microsoft Excel 쉼	161KB
results	2025-03-10 오후 9:36	Microsoft Excel 쉼	1,682KB
seasons	2025-03-10 오후 9:36	Microsoft Excel 쉼	5KB
sprint_results	2025-03-10 오후 9:36	Microsoft Excel 쉼	25KB
status	2025-03-10 오후 9:36	Microsoft Excel 쉼	ЗКВ

```
데이터셋 병합을 위한 간단한 전처리
 불필요한 컬럼은 미리 전처리
[] races = races.drop(columns=["url", "fp1_date", "fp1_time", "fp2_date", "fp2_time", "fp3_date", "fp3_time", "quali_date", "quali_time", "spr
     circuits = circuits.drop(columns=["url"], errors='ignore')
     results = results.drop(columns=["positionText", "fastestLapTime", "fastestLapSpeed"], errors='ignore')
 중복되는 변수명 구분될 수 있도록 Rename
 [] races = races.rename(columns={"name": "race_name"})
     circuits = circuits.rename(columns={"name": "circuit_name", "location": "circuit_location", "country": "circuit_country"})
 key변수를 기준으로 JOIN 병합
[] df = (
        pit_stops
        .merge(lap_times, on=["raceld", "driverld", "lap"], how="left", suffixes=("_pits", "_laps"))
        .merge(races, on=["raceld"], how="left", suffixes=("", "_races"))
        .merge(results, on=["raceld", "driverld"], how="left", suffixes=("", "_results"))
         .merge(circuits, on=["circuitId"], how="left", suffixes=("", "_circuits"))
```



## 데이터셋설명

### Ergast Developer API...

#### 변수(컬럼) 분석 노트 변수명 변수 설명 변수 아이디어 각 레이스(그랑프리) 고유 ID 경기벌 테이터를 구분하는 기본 식벌자 raceld driverId 각 드라이버의 고유 ID 드라이버벌 피트스톱 패턴을 분석 현재 경기에서 해당 드라이버의 피트스톱 횟수 드라이버가 여러 번 피트스톱을 하는지 여부 분석 (일반적으로 1회, 당연히 적을수록 좋음) stop 해당 피트스톱이 발생한 랩 피트스톱이 주로 어느 랩에서 발생하는지 분석 피트스톱이 이루어진 시간 피트스톱이 경기 중 어느 시점에 발생하는지 확인 time\_pits 피트스톱 소요 시간이 짧을수록 순위에 미치는 영향 분석 피트스톱 소요 시간 (초 단위) duration 정밀한 분석을 위해 사용 가능 피트스톱 소요 시간 (밀리초 단위) milliseconds\_pits 피트스톱 직전 드라이버의 순위 피트스톱 전후 순위 변화 분석 position 해당 랩에서의 기록된 시간 타이어 마모 정도나 경기 진행 중 성능 변화 분석 time\_laps milliseconds\_laps 해당 랩에서의 기록된 시간 (밀리초) 정밀한 랩 타임 분석 경기가 열린 연도 연도벌 트렌드 변화 분석 시즌 내 경기 진행 상황과 피트스톱 전략 비교 해당 시즌에서 몇 번째 경기인지 round 서킷의 고유 ID 서킷별 피트스톱 전략 비교 circuitld 해당 레이스(그랑프리)의 이름 경기벌 데이터를 구벌하는 데 사용 race\_name 날씨 및 계절적 요인과의 관계 분석 경기가 열린 날짜 date 트랙 온도 및 날씨와 피트스톱 간의 관계 분석 경기가 시작된 시간 time 결과 태이블에서의 고유 ID 각 경기의 결과를 특정하는데 사용 resultid 팀벌 피트스톱 전략 비교 해닷 드라이버가 소속된 팀의 ID

#### df.info() <class 'pandas.core.frame.DataFrame'> RangeIndex: 11371 entries, 0 to 11370 Data columns (total 36 columns): Column Non-Null Count 11371 non-null raceld int64 11371 non-null driverId int64 11371 non-null int64 stop 3 11371 non-null int64 lap 11371 non-null time\_pits object 11371 non-null duration object 11371 non-null int64 milliseconds\_pits position 11371 non-null int64 time\_laps object 11371 non-null milliseconds\_laps 11371 non-null int64 11371 non-null int64 year 11371 non-null round int64 circuitld 11371 non-null int64 11371 non-null object race\_name



### 데이터전처리

- 불필요한 변수 삭제: position\_results, Ing, lat, alt 등
- 데이터 타입 변환: milliseconds\_pits, final\_position, fastestLap 등의 컬럼을 숫자형으로 변환
- 결측값 처리: \N값 → 보통 리타이어(경기 중 탈락)한 드라이버나 경기를 완료하지 못한 드라이버에게 할당

#### 중복 및 필요없는 컬럼 데이터 타입 변환 (범주 -> 수치) 결측값 확인 및 제거 [29] # position\_results 와 positionOrder가 겹침 [31] # milliseconds\_pits(피트 스탑 타임-밀리초)와 final\_position은 숫자로 변환되어야 한다! # 수치형인 positionOrder만 남김 # 서킷(위치, 경도 등) 필요없음 db['time\_pits'] = pd.to\_numeric(db['time\_pits'], errors='coerce') db.info() db = db.drop(columns=['position\_results']) db['milliseconds\_pits'] = pd.to\_numeric(db['milliseconds\_pits'], errors='coerce') db = db.drop(columns=['lng']) db = db.drop(columns=['lat']) 🚁 <class 'pandas.core.frame.DataFra # position -> 최종 순위로 구분하기 위해 final\_position으로 변수명 변경 db = db.drop(columns=['alt']) RangeIndex: 11371 entries, 0 to 1 db['final\_position'] = pd.to\_numeric(db['position'], errors='coerce') print(db.columns) db['number'] = pd.to\_numeric(db['number'], errors='coerce') Data columns (total 33 columns): Index(['raceld', 'driverld', 'stop', 'lap', 'time\_pits', 'duration', db['fastestLap'] = pd.to\_numeric(db['fastestLap'], errors='coerce') # Column Non-Null I 'milliseconds\_pits', 'position', 'time\_laps', 'milliseconds\_laps',

#### 4.Track



# EDA 요약 (1)

#### 변수 간 상관관계 확인 (양/음)

1.00

- 0.75

- 0.50

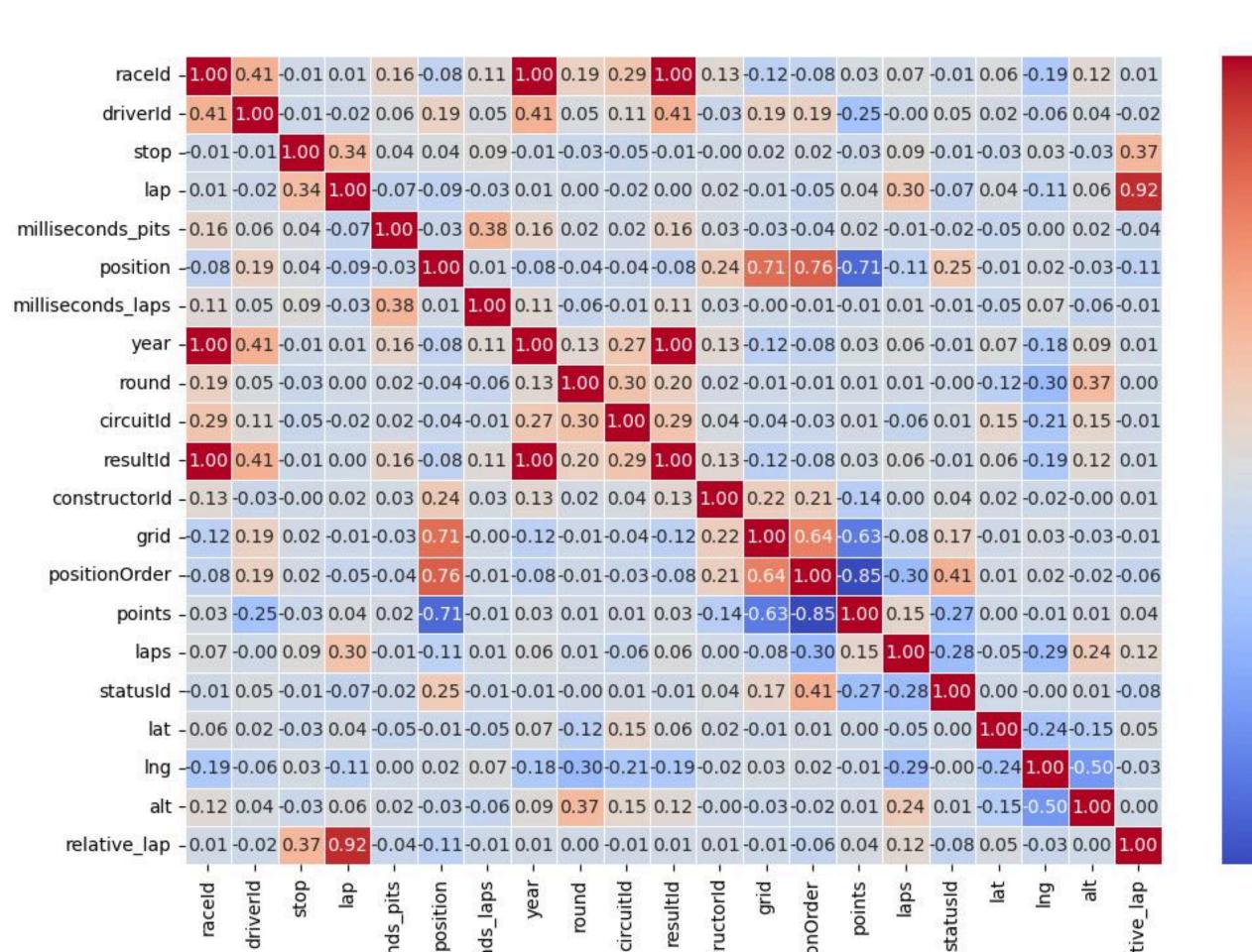
- 0.25

- 0.00

-0.25

-0.50

- -0.75



출발 위치(grid)와 최종 순위(positionOrder): 0.64

출발 위치가 뒤 일수록 (grid 값이 클수록) 최종 순위가 더불리할 가능성이 큼 (강한 양의 상관관계)

최종 순위(positionOrder)와 포인트(points): -0.85

순위가 낮을수록(=더 앞선 순위일수록) 높은 포인트

출발 위치(grid)와 포인트(points): -0.63

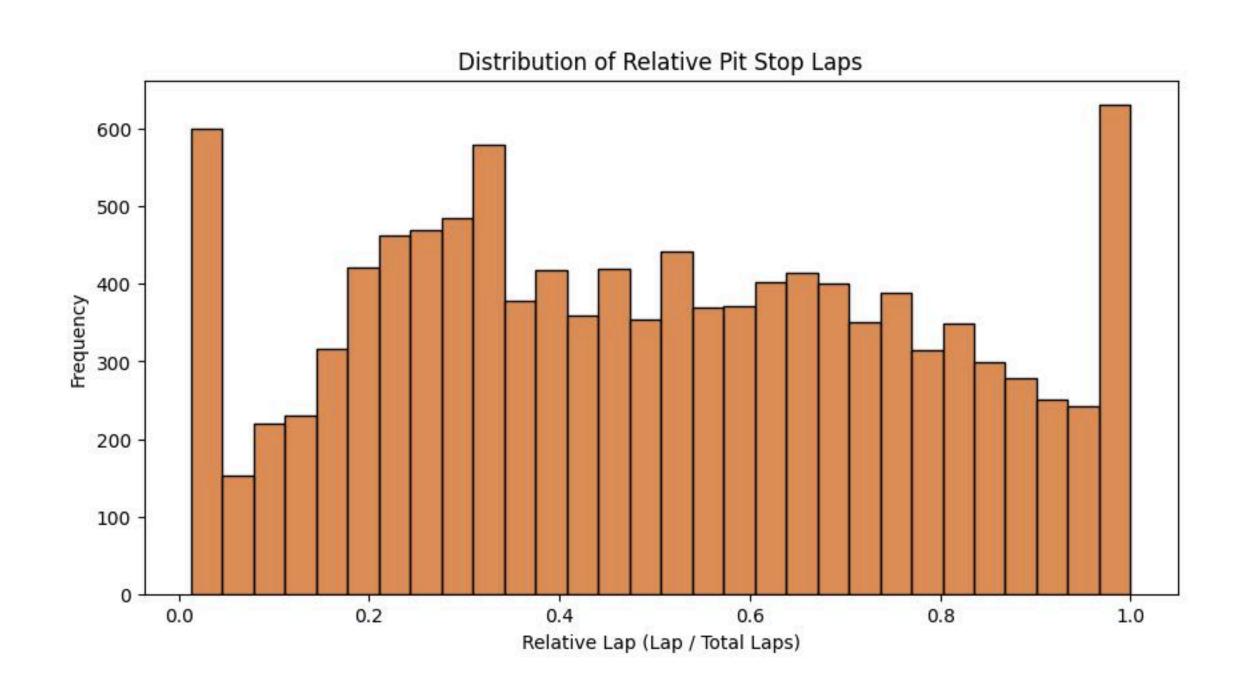
위 최종 순위와 마찬가지로 출발 위치가 좋을수록(낮을 수록) 경기에서 더 많은 포인트를 획득할 가능성이 높음



# EDA 요약 (2)

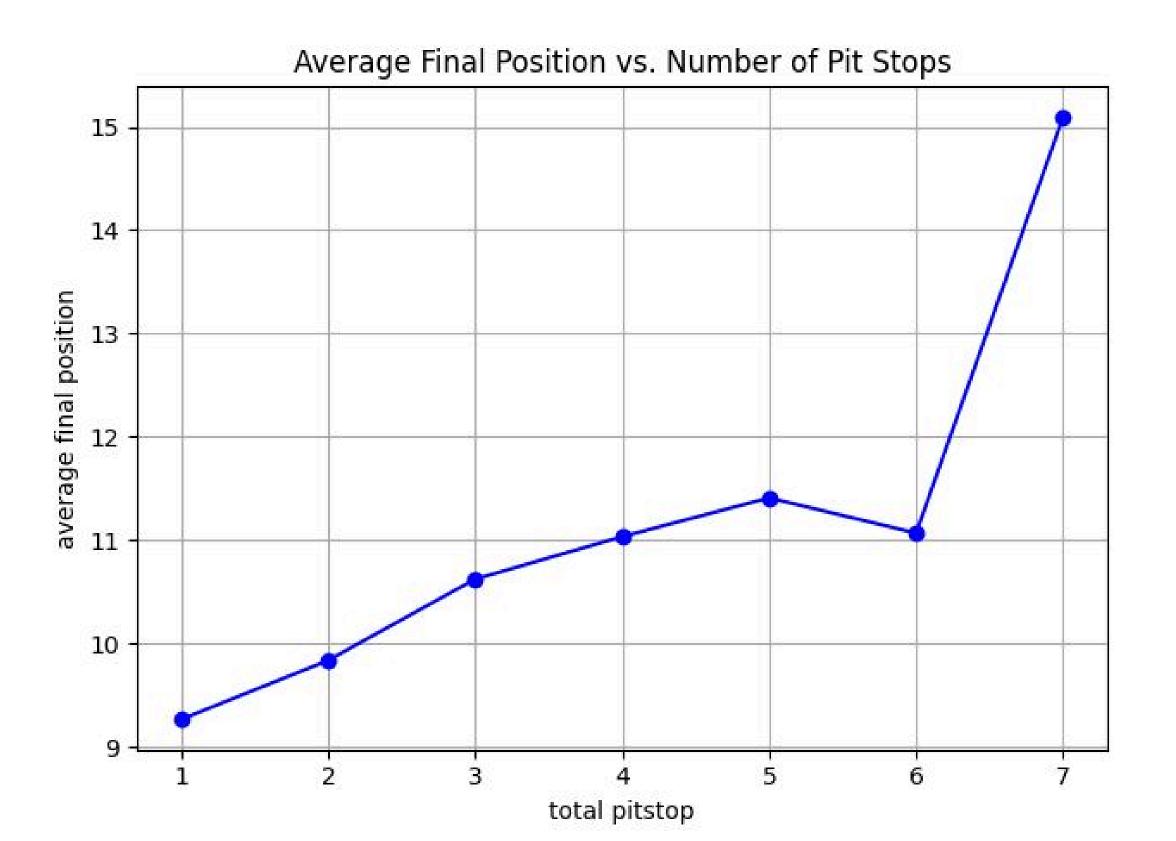
### 경기별 총 랩 수에서 Pit Stop이 이루어지는 랩 분포

- (피트스톱 랩) / (해당 경기 전체 랩 수)
- 각 경기별 총 랩수를 기준으로 상대적 랩수를 계산



### Pit Stop횟수와 최종 순위 관계

• 더 많은 피트스톱을 한 드라이버는 일반적으로 레이스에서 성적이 떨어지기 쉬움





# Feature Engineering



- 총 피트스톱 횟수: 더 많은 피트스톱을 한 드라이버는 일반적으로 레이스에서 성적이 떨어지기 쉬울 것
- 평균 피트스톱 시간: 피트스톱 시간이 긴 드라이버는 상대적으로 더 낮은 성적을 기록할 가능성이 높을 것
- 피트스톱 타이밍(이르거나/늦은): 이르거나/늦은 피트스톱을 한 드라이버가 어떤 성적을 보일지 확인



# Feature Engineering

### 총 피트 스톱 횟수

# ● import scipy.stats as stats # 피트스톱 횟수 - 평균 최종 순위 간의 피어슨 상관관계 계산 correlation, p\_value = stats.pearsonr(stop\_vs\_position['total\_stops'], stop\_vs\_position['avg\_position']) print(f"Pearson's correlation coefficient: {correlation}") print(f"P-value: {p\_value}") Pearson's correlation coefficient: 0.8513638408884768 P-value: 0.015075696139683445

### 피트 스톱 시점

```
import scipy.stats as stats
# 피트스톱 횟수 - 평균 최종 순위 간의 피어슨 상관관계 계산
correlation, p_value = stats.pearsonr(pit_data[['earliest_stop_lap']], pit_data['final_position'])
print(f"Pearson's correlation coefficient: {correlation}")
print(f"P-value: {p_value}")

Pearson's correlation coefficient: -0.16270207306283974
P-value: 2.679904638759774e-68
```

Pearson's Correlation Coefficient: 0.107 P-value: 3.98e-30 (0에 수렴)



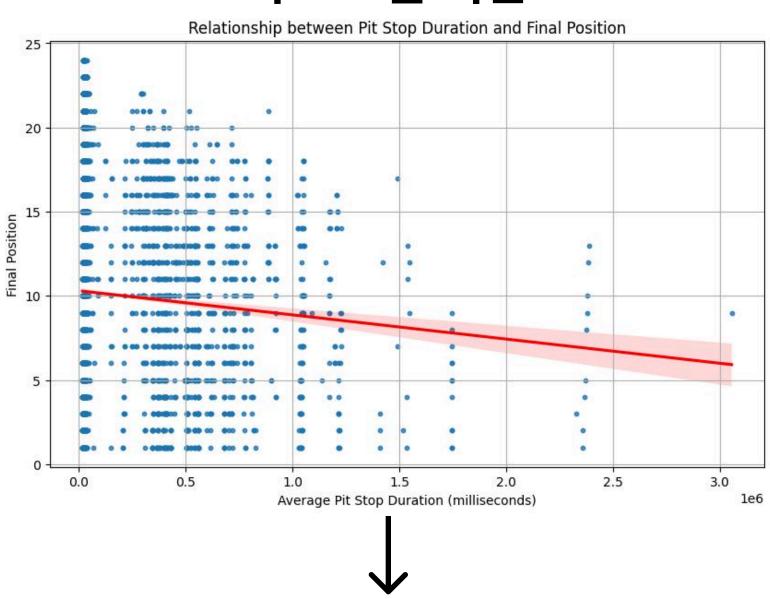
총 피트스톱 횟수가 많을 수록 최종 경기 순위가 안좋아질 상관관계가 있지만 그 관계는 매우 약하며 (Pearson=0.107) 이는 통계적으로 유의미함 (p-value=0) Pearson's Correlation Coefficient: -0.162

P-value: 3.67e-68 (0에 수렴)



피트스톱이 더 일찍 이루어질수록 최종 순위가 약간 더 좋을 가능성이 있지만,(Pearson=-0.16) 그 관계는 매우 약하며 통계적으로 유의미하다.

### 피트 스톱 시간



평균 피트스톱 시간이 짧을수록 더 좋은 성적을 기록하는 경향, 이는 기술 발전에 따라 점점 단축 되어 가는 중 (맥라렌 - 1.80초)



# 결과요약및인사이트

### 주요 분석 결과 & 인사이트:

- 1. 피트스톱 횟수와 최종 순위 간의 관계:
  - 상관관계: 피트스톱 횟수와 최종 순위 간에는 약한 양의 상관관계가 있음 (상관계수: 0.11, p-value < 0.05).</li>
  - 해석: 피트스톱을 더 많이 한 드라이버들이 평균적으로 성적이 더 나쁜 경향이 있지만, 그 관계는 미미함. 따라서, 피트스톱 횟수
     가 많다고 무조건 성적이 떨어진다고 단정 짓기는 어려움. 전략적인 다중 피트스톱이 유리한 결과를 가져올 수도 있음.
- 2. 이른 피트스톱 랩과 최종 순위 간의 관계:
  - ◇ 상관관계: 가장 이른 피트스톱 랩과 최종 순위 간에는 약한 음의 상관관계가 있음 (상관계수: -0.08, p-value < 0.05).</li>
  - 해석: 이른 피트스톱이 성적에 미치는 영향은 약하지만 긍정적인 경향을 보이며, 언더컷 전략을 통해 더 좋은 순위를 얻을 가능성이 있음.
- 3. 피트스톱 시간과 최종 순위:
  - **분석**: 평균 피트스톱 시간이 짧을수록 더 좋은 성적을 기록하는 경향이 있음. 이는 시대 흐름에 따라 점점 단축되어 가는 중임



내구성 약함

느림 내구성 강함

## 향후계획및개선방향

피트스톱 전략에서 보다 다양한 변수의 결합 필요성

서킷의 특성, 날씨, 타이어 전략 등 다양한 외부 요인 고려해야 할 것

