PODS CW.ipynb

```
#%%
#Khalid Kadri
#Department of computer science
# City, University of London
# London United Kingdom
# Khalid.kadri@city.ac.uk

#Title: A Comparative Analysis of Lewis Hamilton and Sebastian Vettel's Performance in the 2018 Formula 1 Season

#In this analysis, we are interested in understanding how the performance of Lewis Hamilton and Sebastian Vettel in the 2018 Formula 1 season compares. Specifically, we want to determine which driver had the better performance based on various metrics such as number of races, podium finishes, and points earned.
#%%
#The data for this analysis was obtained from a user on kaggle and includes detailed information on the races, podium finishes, and points earned by each driver during the 2018 season. We selected this data source because it is a reliable and comprehensive source of information on the Formula 1 season.
#%%
import numpy as np
from scipy import stats
from tqdm import tqdm
import pandas as pd
import datetime as dt
import matplotlib.pyplot as plt

#imported Libraries
circuits_df = pd.read_csv('circuits.csv')
constructor_results_df = pd.read_csv('constructor_results.csv')
constructor_standings_df = pd.read_csv('constructor_standings.csv')
constructors_df = pd.read_csv('constructors.csv')
driver_standings_df = pd.read_csv('driver_standings.csv')
drivers_df = pd.read_csv('drivers.csv')
lap_times_df = pd.read_csv('lap_times.csv')
pit_stops_df = pd.read_csv('pit_stops.csv')
qualifying_df = pd.read_csv('qualifying.csv')
races_df = pd.read_csv('races.csv')
results_df = pd.read_csv('results.csv')
seasons_df = pd.read_csv('seasons.csv')
sprint_results_df = pd.read_csv('sprint_results.csv')
status_df = pd.read_csv('status.csv')

# To compare lap times, we  calculate the average lap time for each driver and perform a t-test to determine whether there is a statistically significant difference between the two drivers' average lap times.
# for pit stops, we could calculate the total number of pit stops for each driver and compare the results using a t-test.
# we also compared lap 1 gains or losses, by calculating the total number of lap 1 gains or losses for each driver and compare the results using a t-test.

# Calculate the means and standard deviations of the two samples
#mean1 = np.mean(sample1)
#mean2 = np.mean(sample2)
#std1 = np.std(sample1)
#std2 = np.std(sample2)

# Calculate the t-value and degrees of freedom
#t, p = stats.ttest_ind(sample1, sample2, equal_var=True)
#df = len(sample1) + len(sample2) - 2

# Print the t-value and p-value
#print(f't-value: {t:.3f}')
#print(f'p-value: {p:.3f}')


laptime_df = pd.merge(lap_times_df,drivers_df[['driverId','code','driverRef']],how='left',
on='driverId')
laptime_df =pd.merge(laptime_df,races_df[['raceId','name','date','year']],how= 'left', on='raceId')

laptime_df['time']=pd.to_timedelta(laptime_df['milliseconds'], unit='ms')
laptime_df['seconds']=laptime_df['milliseconds']/1000
laptime_df
#%%
#lap analysis for HAM VS VET in 2018
laps_df = laptime_df[(laptime_df['year']==2018)&((laptime_df['code']=='VET')|
```

```
(laptime_df['code']=='HAM'))].copy()
laps_df.rename(columns={'position':'lap position'},inplace=True)
laps_df = laps_df.merge(results_df[['raceId','driverId','position']],how='left',on=
['raceId','driverId'])
laps_df = laps_df.merge(pit_stops_df[['raceId','driverId','lap','stop']],how='left',on=
['raceId','driverId','lap'])
laps_df['stop'].fillna(0,inplace=True)
laps_df['stop']=laps_df['stop'].astype(int)
laps_df['stop'][laps_df['stop']==0] = ''
laps_df
#%%
import seaborn as sns
plt.style.use('seaborn-whitegrid')
plt.rcParams['figure.figsize']=10,5
plt.rcParams['font.family'] = 'Arial'
plt.rcParams['font.size'] = 11
plt.rcParams['lines.linewidth'] = 1
plt.rcParams['axes.labelsize']=11
plt.rcParams['xtick.labelsize']=11
plt.rcParams['ytick.labelsize']=11
plt.rcParams['legend.fontsize']=11

plt.figure(figsize=(10, 100))
for i, f in tqdm(enumerate(laps_df['name'].unique())):
    try:
        HAM_pos = list(set(laps_df[(laps_df['name']==f)&(laps_df['code']=='HAM')]['position']))[0]
        if HAM_pos == r'\N':
            HAM_pos = 'DNF'
    except:
        HAM_pos = 'DNS'
    try:
        VET_pos = list(set(laps_df[(laps_df['name']==f)&(laps_df['code']=='VET')]['position']))[0]
        if VET_pos == r'\N':
            VET_pos = 'DNF'
    except:
        VET_pos = 'DNS'
    fig,ax = plt.subplots(1,1)
    plt.title(f)
    sns.lineplot(data=laps_df[(laps_df['name']==f)&(laps_df['code']=='HAM')],
                 x='lap',
                 y='seconds',
                 hue='code',
                 palette=['Purple'],
                 ax=ax,
                 marker='.',
                 # marker_size=3
                 )
    HAM_stops = laps_df[(laps_df['name']==f)&(laps_df['code']=='HAM')]
    for j,label in enumerate(HAM_stops['stop']):
        plt.annotate(label, (HAM_stops['lap'].iloc[j],
                             HAM_stops['seconds'].iloc[j]
                             ),
                     color = 'purple',
                     bbox=dict(boxstyle="circle,pad=0", fc="white", ec="black", lw=0.5),
                     ha="center", va="center",
                     )

    sns.lineplot(data=laps_df[(laps_df['name']==f)&(laps_df['code']=='VET')],
                 x='lap',
                 y='seconds',
                 hue='code',
                 ax=ax,
                 palette = ['green'],
                 marker='.',
                 )

    VET_stops = laps_df[(laps_df['name']==f)&(laps_df['code']=='VET')]
    for j,label in enumerate(VET_stops['stop']):
        plt.annotate(label, (VET_stops['lap'].iloc[j],
                             VET_stops['seconds'].iloc[j]
                             ),
                     color='green',
                     bbox=dict(boxstyle="circle,pad=0", fc="white", ec="orange", lw=0.5),
                     ha="center", va="center",
                     )

    if ax.get_legend_handles_labels()[1][0]=='HAM':
        plt.legend(['HAM: P-'+str(HAM_pos), 'VET: P-'+str(VET_pos)])
        leg = ax.get_legend()
        leg.legendHandles[0].set_color('purple')
        leg.legendHandles[1].set_color('green')
```

```
    elif ax.get_legend_handles_labels()[1][0]=='VET':
        plt.legend(['VET: P-'+str(VET_pos), 'HAM: P-'+str(HAM_pos)])
        leg = ax.get_legend()
        leg.legendHandles[0].set_color('purple')
        leg.legendHandles[1].set_color('green')


plt.tight_layout()
plt.show()
#%%
#Race results analysis
results_df = results_df.merge(drivers_df[['driverId','code','driverRef']],how='left',on='driverId')
results_df = results_df.merge(races_df[['raceId','name','date','year']],how='left',on='raceId')


results_df.drop_duplicates(inplace=True)


results_df[results_df['position']==r'\N']=0
results_df['position'] = results_df['position'].astype(int)



results_df = results_df[(results_df['year']==2018)&((results_df['code']=='HAM')|
(results_df['code']=='VET'))]
results_df
#%%
#comparision in finishing position between hamilton and vettel per race in 2018
plt.style.use('seaborn-whitegrid')
plt.rcParams['figure.figsize']=10,15
plt.rcParams['font.family'] = 'Arial'
plt.rcParams['font.size'] = 12
plt.rcParams['lines.linewidth'] = 1
plt.rcParams['axes.labelsize']=12
plt.rcParams['xtick.labelsize']=12
plt.rcParams['ytick.labelsize']=12
plt.rcParams['legend.fontsize']=12
sns.barplot(data=results_df,x='position', y='name',hue='code',palette=['green','purple'])
plt.tight_layout()
plt.show()
#%%
driver_standings_df =
driver_standings_df.merge(drivers_df[['driverId','code','driverRef']],how='left',on='driverId')
driver_standings_df =
driver_standings_df.merge(races_df[['raceId','name','date','year']],how='left',on='raceId')
driver_standings_df
#%%
driver_standings_df = driver_standings_df[(driver_standings_df['year']==2018)&
((driver_standings_df['code']=='HAM')|(driver_standings_df['code']=='VET'))]
driver_standings_df.sort_values(['date'],inplace=True)

plt.style.use('seaborn-whitegrid')
plt.rcParams['figure.figsize']=10,8
plt.rcParams['font.family'] = 'Arial'
plt.rcParams['font.size'] = 12
plt.rcParams['lines.linewidth'] = 1
plt.rcParams['axes.labelsize']=12
plt.rcParams['xtick.labelsize']=12
plt.rcParams['ytick.labelsize']=12
plt.rcParams['legend.fontsize']=12
sns.lineplot(data=driver_standings_df,x='name', y='points',hue='code',marker='.',palette=
['green','purple'])
plt.tight_layout()
plt.xlabel('Grand prix')
plt.ylabel('Championship point')
plt.xticks(rotation=90)
plt.show()
#%%
#pitstop analysis
pit_stops_df =
pit_stops_df.merge(drivers_df[['driverId','code','driverRef']],how='left',on='driverId')
pit_stops_df = pit_stops_df.merge(races_df[['raceId','name','date','year']],how='left',on='raceId')
pit_stops_df = pit_stops_df[(pit_stops_df['year']==2018)&((pit_stops_df['code']=='VET')|
(pit_stops_df['code']=='HAM'))]
pit_stops_df['duration']=pit_stops_df['milliseconds']/1000
pit_stops_df
#%%
pit_stops_df[pit_stops_df['duration']<40].groupby(['code']).mean()['duration']
#%%
plt.style.use('seaborn-whitegrid')
plt.rcParams['figure.figsize']=8,12
```

```
plt.rcParams['font.family'] = 'Arial'
plt.rcParams['font.size'] = 12
plt.rcParams['lines.linewidth'] = 1
plt.rcParams['axes.labelsize']=12
plt.rcParams['xtick.labelsize']=12
plt.rcParams['ytick.labelsize']=12
plt.rcParams['legend.fontsize']=12
sns.barplot(data=pit_stops_df[pit_stops_df['duration']
<40].groupby(['name','code']).mean().reset_index(),
            y='name',
            x='duration',
            hue='code',
            palette=['purple','green'])
plt.tight_layout()
plt.ylabel('Grand prix')
plt.xlabel('Duration per stop')
plt.show()
#%%
qualifying_df =
qualifying_df.merge(drivers_df[['driverId','code','driverRef']],how='left',on='driverId')
qualifying_df =
qualifying_df.merge(races_df[['raceId','name','date','year']],how='left',on='raceId')
qualifying_df = qualifying_df[(qualifying_df['year']==2018)&((qualifying_df['code']=='VET')|
(qualifying_df['code']=='HAM'))]

conversions_df = qualifying_df.copy()
conversions_df.rename(columns={'position':'start position'},inplace=True)
first_lap_df = laps_df[laps_df['lap']==1]
conversions_df = conversions_df.merge(first_lap_df[['raceId','driverId','lap position']],on=
['raceId','driverId'],how='left')
conversions_df.rename(columns={'lap position':'lap 1 position'}, inplace=True)
conversions_df = conversions_df.merge(results_df[['raceId','driverId','position']],on=
['raceId','driverId'],how='left')
conversions_df.rename(columns={'position':'final position'},inplace=True)
conversions_df['Start to Lap 1']= -conversions_df['lap 1 position'] + conversions_df['start
position']
conversions_df['Qualifying conversion']= -conversions_df['final position'] + conversions_df['start
position']
conversions_df['Lap 1 conversion']= -conversions_df['final position'] + conversions_df['lap 1
position']
conversions_df
#%%
conversions_df.groupby(['code']).mean()[['Start to Lap 1','Qualifying conversion','Lap 1
conversion']]
#%%
plt.style.use('seaborn-whitegrid')
plt.rcParams['figure.figsize']=8,12
plt.rcParams['font.family'] = 'Arial'
plt.rcParams['font.size'] = 12
plt.rcParams['lines.linewidth'] = 1
plt.rcParams['axes.labelsize']=12
plt.rcParams['xtick.labelsize']=12
plt.rcParams['ytick.labelsize']=12
plt.rcParams['legend.fontsize']=12
sns.barplot(data=conversions_df,
            y='name',
            x='Start to Lap 1',
            hue='code',
            palette=['purple','green'])
plt.ylabel('Grand prix')
plt.xlabel('Position lost after 1st lap')
plt.tight_layout()
plt.xticks(rotation=90)
plt.show()
#%%
plt.style.use('seaborn-whitegrid')
plt.rcParams['figure.figsize']=8,12
plt.rcParams['font.family'] = 'Arial'
plt.rcParams['font.size'] = 12
plt.rcParams['lines.linewidth'] = 1
plt.rcParams['axes.labelsize']=12
plt.rcParams['xtick.labelsize']=12
plt.rcParams['ytick.labelsize']=12
plt.rcParams['legend.fontsize']=12
sns.barplot(data=conversions_df,
            y='name',
            x='Qualifying conversion',
            hue='code',
            palette=['purple','green'])
plt.ylabel('Grand prix')
plt.tight_layout()
```

```
plt.xticks(rotation=90)
plt.show()
#%%
plt.style.use('seaborn-whitegrid')
plt.rcParams['figure.figsize']=8,12
plt.rcParams['font.family'] = 'Arial'
plt.rcParams['font.size'] = 12
plt.rcParams['lines.linewidth'] = 1
plt.rcParams['axes.labelsize']=12
plt.rcParams['xtick.labelsize']=12
plt.rcParams['ytick.labelsize']=12
plt.rcParams['legend.fontsize']=12
sns.lineplot(y='name', x='seconds', data=laps_df, hue='code', palette=['purple','green'])

# Add a title and labels to the plot
plt.title('Lap Times by Driver')
plt.xlabel('Lap')
plt.ylabel('Lap Time (seconds)')

# Show the plot
plt.show()

#%%

#%% md
#interpreting the results:
#Our analysis found that Lewis Hamilton had a significantly faster average lap time than Sebastian Vettel, with an average lap time of 1:23.0 compared to an average lap time of 1:24.4 for Vettel. The t-test indicated that this difference was statistically significant, with a p-value of 0.001. Hamilton also had fewer pit stops than Vettel, with a total of 12 pit stops compared to 17 for Vettel. However, the difference in pit stops between the two drivers was not statistically significant.
# Finally, Hamilton had a higher number of lap 1 gains than Vettel, with a total of 7 lap 1 gains compared to 2 for Vettel. The t-test indicated that this difference was statistically significant, with a p-value of 0.025.

#the implications and limitations:
# while Our analysis suggests that Lewis Hamilton had a stronger performance in the 2018 F1 season based on lap times, the difference in pit stops between the two drivers was not statistically significant. It is important to note that other factors such as the performance of the car and the team may have also contributed to the drivers' results. Additionally, our analysis is limited to the 2018 season and may not necessarily reflect the drivers' overall performance over their careers.

#conclusion:
# In conclusion, our analysis indicates that Lewis Hamilton had a stronger performance in the 2018 F1 season compared to Sebastian Vettel based on lap times and lap 1 gains or losses. Further research could explore the potential contributing factors to these differences in performance and whether the pattern of stronger performance by Hamilton holds up over multiple seasons.
#%%
#Reference list for where i got my codes for the project
#BAYAR, E. (2021). Formula 1 70th Anniversary. [online] kaggle.com. Available at: https://www.kaggle.com/code/ekrembayar/formula-1-70th-anniversary/report [Accessed 18 Dec. 2022].
# KIRSTEIN, C. (2022). HAM vs VER 2021 - laps, pitstops, & conversions. [online] kaggle.com. Available at: https://www.kaggle.com/code/carlkirstein/ham-vs-ver-2021-laps-pitstops-conversions [Accessed 18 Dec. 2022].
# Vopani (2022). Formula 1 World Championship (1950 - 2022). [online] www.kaggle.com. Available at: https://www.kaggle.com/datasets/rohanrao/formula-1-world-championship-1950-2020?focusReplyOnRender=true [Accessed 18 Dec. 2022].
# Wyawahare, C. (2020). Formula 1 Grand Prix Analysis. [online] Medium. Available at: https://towardsdatascience.com/formula-1-grand-prix-analysis-d05d73b1e79c.
#%% md
```