Ultra Marathon Data Analysis Project Report

Presented By

Shripad Kulkarni

Data Analyst

Language	Python
Libraries	Pandas, Matplotlib, Seaborn
Web Application	Jupyter Notebook
Project Link	

July 2024

shripadkulkarni2212@gmail.com | +91 7378674911

Ultra Marathon Data Analysis Project Report

Introduction

According to Wikipedia, an ultramarathon, also called ultra distance or ultra running, is any footrace longer than the traditional marathon length of 42.195 kilometres (26 mi 385 yd). Various distances are raced competitively, from the shortest common ultramarathon of 31 miles (50 km) to over 200 miles (320 km). 50k and 100k are both World Athletics record distances, but some 100 miles (160 km) races are among the oldest and most prestigious events, especially in North America.

Problem Statement

This project involves analysing ultra marathon race data to gain insights into various factors influencing athlete performance. The dataset includes information on race events, athlete demographics, and performance metrics. The goal is to understand trends and patterns in the data and visualise these insights effectively.

Dataset Overview

The data in this file is a large collection of ultra-marathon race records registered between 1798 and 2022, providing a formidable long-term sample. (*Data Source*)

The dataset contains **7,461,226** ultra-marathon race records from **1,641,168** unique athletes. The following columns (with data types) are included:

- Year of event (int64)
- Event dates (object)
- Event name (object)
- Event distance/length (object)
- Event number of finishers (int64)
- Athlete performance (object)
- Athlete club (object)
- Athlete country (object)
- Athlete year of birth (float64)
- Athlete gender (object)
- Athlete age category (object)
- Athlete average speed (object)
- Athlete ID (int64)

The Event distance/length column describes the type of race, covering the most popular UM race distances and lengths, and some other specific modalities (multi-day, etc.):

- Distances: 50km, 100km, 50mi, 100mi
- Lengths: 6h, 12h, 24h, 48h, 72h, 6d, 10d

Steps Followed

1.Data Preprocessing

Importing libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
```

Loading data

```
df = pd.read_csv('D:/#DataAnalyst/Python Projects/New folder/TWO_CENTURIES_OF_UM_RACES.csv')
df
df.head(10)
```

Data details

```
df.info()
df.shape
df.dtypes
```

2.Data Cleaning

Events with the year 2020 in the USA of 50 km or 50 miles

```
df[(df['Event distance/length'].isin(['50km','50mi'])) & (df['Year of event'] == 2020) &
(df['Event name'].str.split('(').str.get(1).str.split(')').str.get(0) == 'USA')]
```

So the newly filtered data is named as df2 which is to be used further for data analysis

```
df2 = df[(df['Event distance/length'].isin(['50km','50mi'])) & (df['Year of event'] == 2020) &
(df['Event name'].str.split('(').str.get(1).str.split(')').str.get(0) == 'USA')]

df2.head(10)
```

Remove USA from event name

```
df2['Event name'].str.split('(').str.get(0)
df2['Event name'] = df2['Event name'].str.split('(').str.get(0)
df2.head(10)
```

Clean up athlete age

```
df2['athelete_age'] = 2020 - df2['Athlete year of birth']
df2
```

Remove h from athlete performance

```
df2['Athlete performance'] = df2['Athlete performance'].str.split(' ').str.get(0)
```

Clean up null values

```
df2.isna().sum()
df2[df2['athelete_age'].isna() == 1]
df2 = df2.dropna()
```

Check for duplicates

```
df2[df2.duplicated() == True]
```

Reset Index

```
df2.reset_index(drop = True)
```

Fix Data Types

```
df2.dtypes

df2['athelete_age'] = df2['athelete_age'].astype(int)
df2['Athlete average speed'] = df2['Athlete average speed'].astype(float)
df2.head()
```

Rename columns

```
df2 = df2.rename(columns = {'Year of event' : 'year',
'Event dates': 'race_day',
'Event name': 'race_name',
'Event distance/length': 'race_length',
'Event number of finishers': 'race_number_of_finishers',
'Athlete performance': 'athelete_performance',
'Athlete gender': 'athlete_gender',
'Athlete average speed': 'athlete_average_speed',
'Athlete ID': 'athelete_id'} )
```

Removing Unnecessary Columns

```
df2 = df2.drop(['Athlete club', 'Athlete country', 'Athlete year of birth', 'Athlete age category'], axis=1)
df2
```

3.Data Finalisation

Reorder Columns and Take this data as df3 (more refined and cleaned version than df and d2)

```
df3 = df2[['race_day','race_name','race_length','race_number_of_finishers','athelete_id',
          'athlete_gender','athelete_age','athlete_average_speed','athelete_performance','year']]
```

This large dataset is converted into a simplified version with the following columns as per requirement:

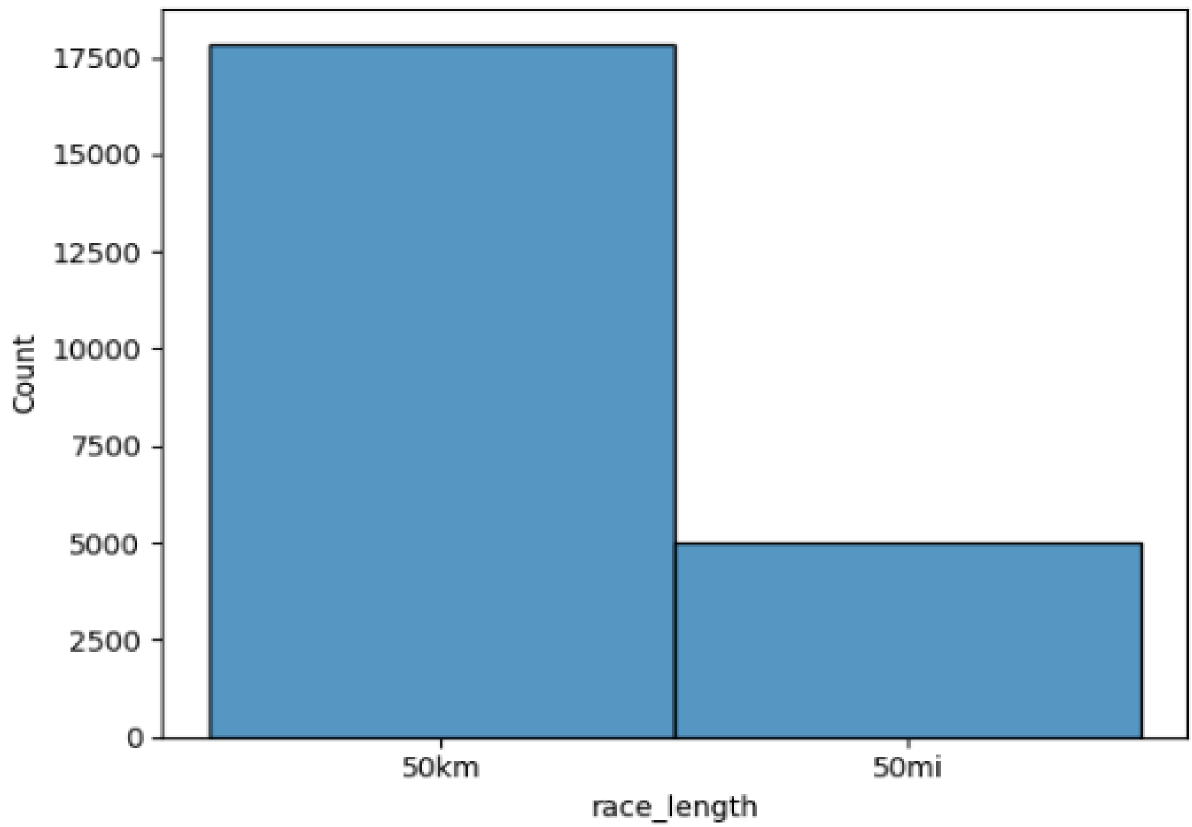
- **year**: Year of the race
- **race_day**: Day of the race
- **race_name**: Name of the race
- **race_length**: Length of the race
- **race_number_of_finishers**: Number of finishers in the race
- **athlete_performance**: Performance metrics of the athlete
- **athlete_gender**: Gender of the athlete
- **athlete_average_speed**: Average speed of the athlete (in miles per hour)
- **athlete_id**: Unique identifier for the athlete
- **athlete_age**: Age of the athlete

4.Data Visualisation and Key Insights

1. Histogram of Race Length vs No of Participants

[40]:
```
sns.histplot(df3['race_length'])
```

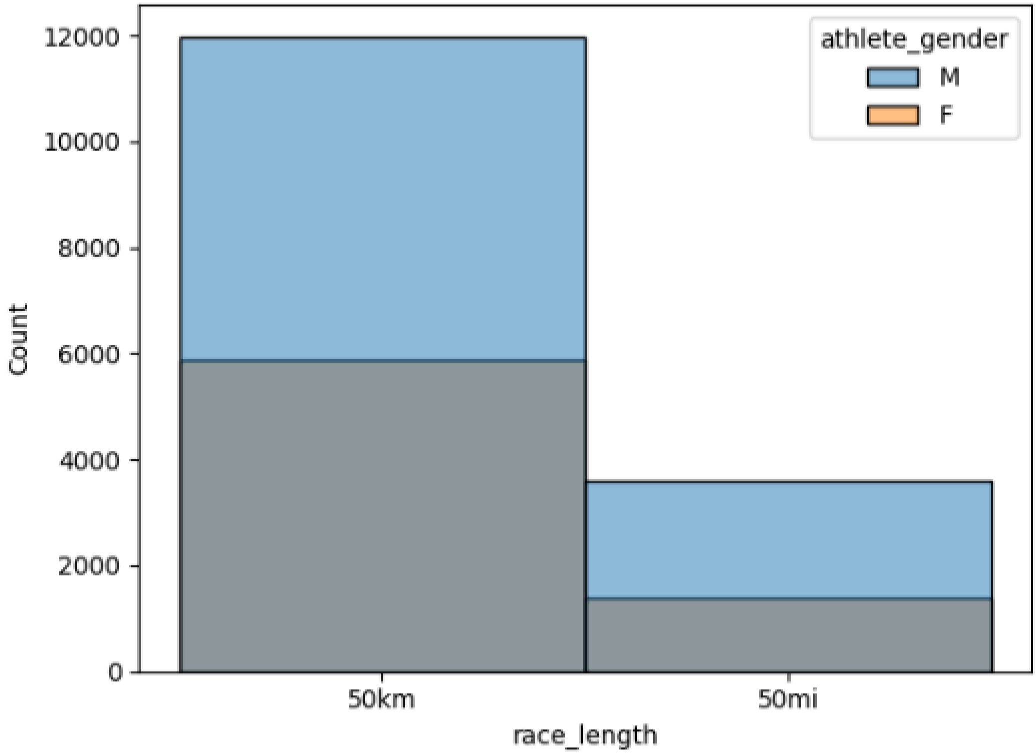
[40]: <Axes: xlabel='race_length', ylabel='Count'>



2. Histogram of Race Length as per Gender

```
[41]: sns.histplot(df3, x = 'race_length', hue = 'athlete_gender')
```

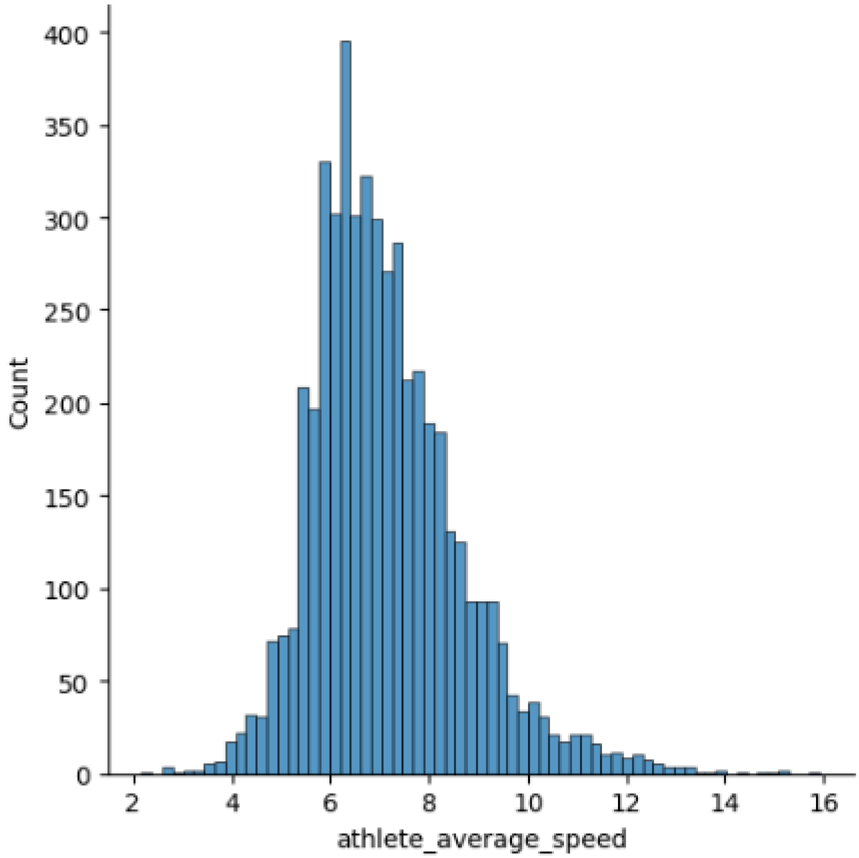
```
[41]: <Axes: xlabel='race_length', ylabel='Count'>
```



3. Distribution of Average Speed of Athletes

[42]:
```
sns.displot(df3[df3['race_length'] == '50mi' ] ['athlete_average_speed'])
```

[42]: <seaborn.axisgrid.FacetGrid at 0x18b722f2360>

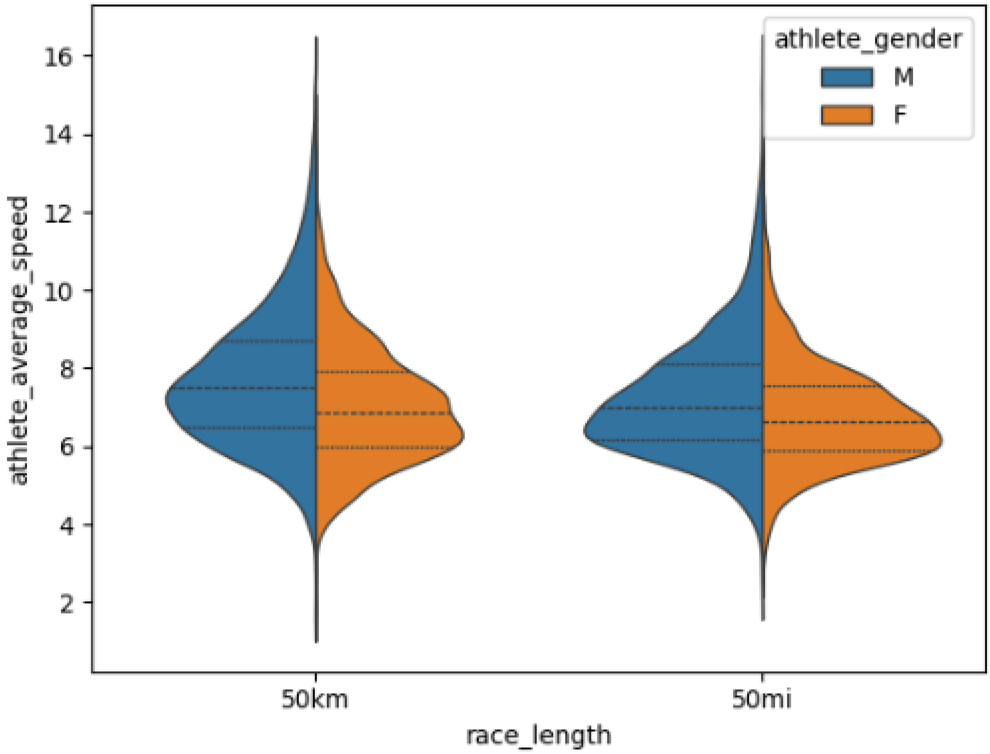


shripadkulkarni2212@gmail.com | +91 7378674911

4. Distribution of Athletes as per their Average Speed and Race Length

```
[43]: sns.violinplot( data = df3, x = 'race_length' , y = 'athlete_average_speed',
                 hue = 'athlete_gender', split = True , inner = 'quart' , linewidth = 1 )
```

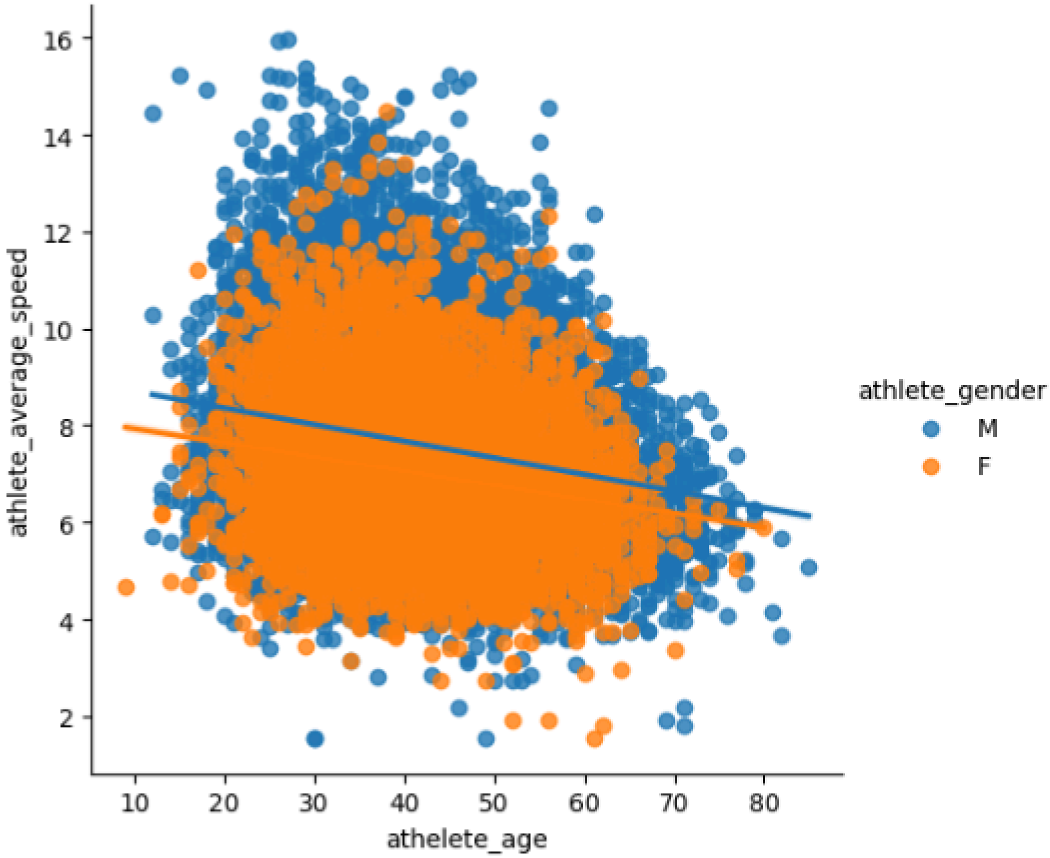
```
[43]: <Axes: xlabel='race_length', ylabel='athlete_average_speed'>
```



5. Distribution of Athlete as per their Average Speed and Age

```
[44]: sns.lmplot(data = df3 ,x = 'athelete_age', y = 'athlete_average_speed', hue = 'athlete_gender')
```

```
[44]: <seaborn.axisgrid.FacetGrid at 0x18b662bf440>
```



6. Difference in speed for the 50km, 50 miles male to female

```
df3.groupby(['race_length','athlete_gender'])['athlete_average_speed'].mean()
```

```
race_length  athlete_gender
50km         F                 7.053849
             M                 7.721989
50mi         F                 6.820359
             M                 7.240810
Name: athlete_average_speed, dtype: float64
```

7. What age group are the best in the 50 miles Race

(20+ races minimum) (show first 15 entries)

[48]:
```
result = (
    df3.query('race_length == "50mi"')
    .groupby('athelete_age')['athlete_average_speed']
    .agg(['mean', 'count'])
    .sort_values('mean', ascending=False)
    .query('count > 19')
    .head(15)
)

print(result)
```

```
                  mean  count
athelete_age
29            7.889707    123
23            7.698600     50
28            7.587792     96
30            7.529574    141
25            7.524000     79
38            7.452283    198
36            7.438932    162
31            7.390492    122
26            7.387836     73
42            7.380146    185
24            7.368435     69
35            7.348609    174
34            7.327699    166
21            7.315595     37
33            7.303633    128
```

8. What age group are the worst in the 50 miles Race

(20+ races minimum) (show first 15 entries)

[49]:
```
result = (
    df3.query('race_length == "50mi"')
    .groupby('athelete_age')['athlete_average_speed']
    .agg(['mean', 'count'])
    .sort_values('mean', ascending=True)
    .query('count > 19')
    .head(15)
)

print(result)
```

```
                  mean  count
athelete_age
60            6.031692     26
62            6.273438     32
61            6.289480     25
63            6.524500     30
58            6.604298     57
59            6.609589     73
50            6.633587    150
57            6.642727     66
53            6.684775     89
56            6.703678     59
52            6.725713    108
48            6.754140    114
49            6.809021    142
43            6.816789    166
54            6.850232     69
```

9. Seasons for the data -> Slower in summer than winter?

Spring 3-5 ¶

Summer 6-8

Fall 9-11

Winter 12-2

For all races

[53]:
```
df3.groupby('race_season')['athlete_average_speed'].agg(['mean','count']).sort_values ('mean', ascending = False )
```

[53]:

	mean	count
race_season		
Spring	7.608495	3078
Winter	7.524490	9934
Fall	7.403170	7389
Summer	6.806785	2433

50 milers only

[54]:
```
df3.query('race_length == "50mi"').groupby('race_season')['athlete_average_speed'].agg(['mean','count']).sort_values ('mean', ascending = False )
```

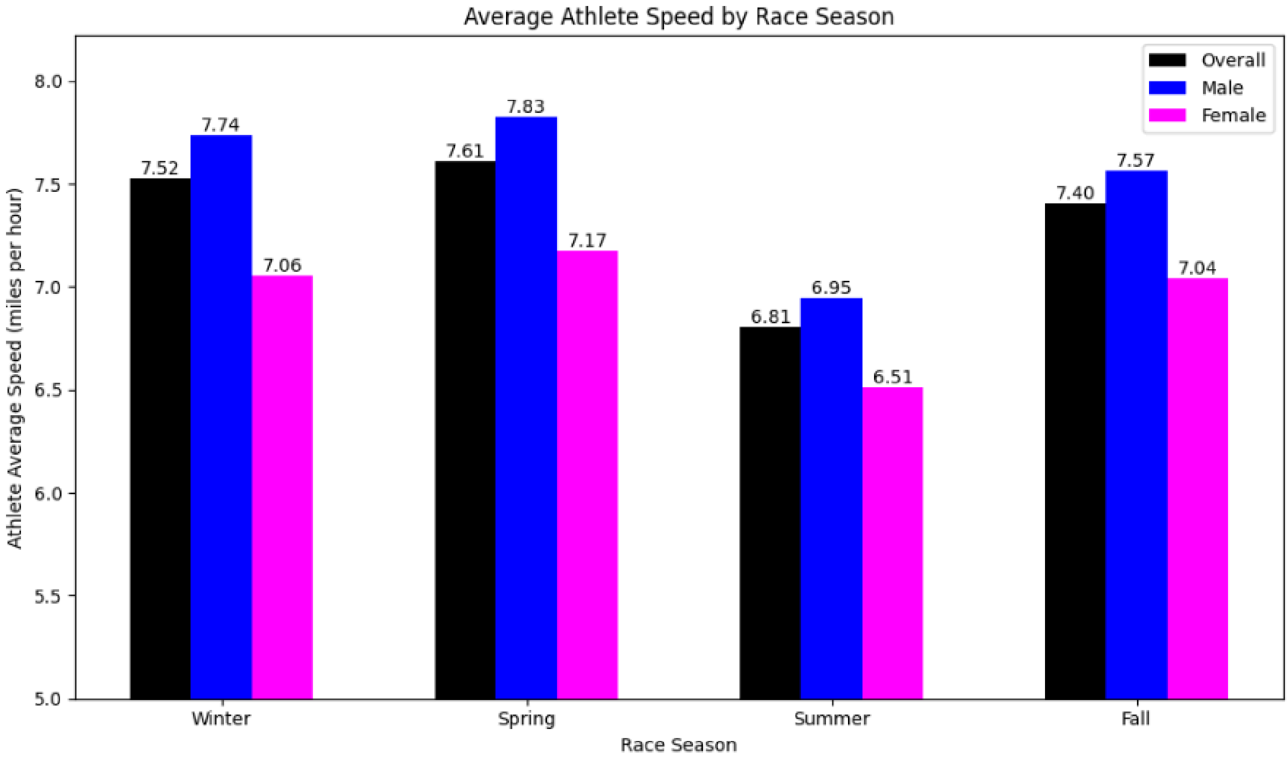
[54]:

	mean	count
race_season		
Fall	7.542307	1836
Spring	7.084466	831
Winter	6.990570	1553
Summer	6.434160	758

10. Average Athlete Speed by Race Season (Overall, Male, Female)

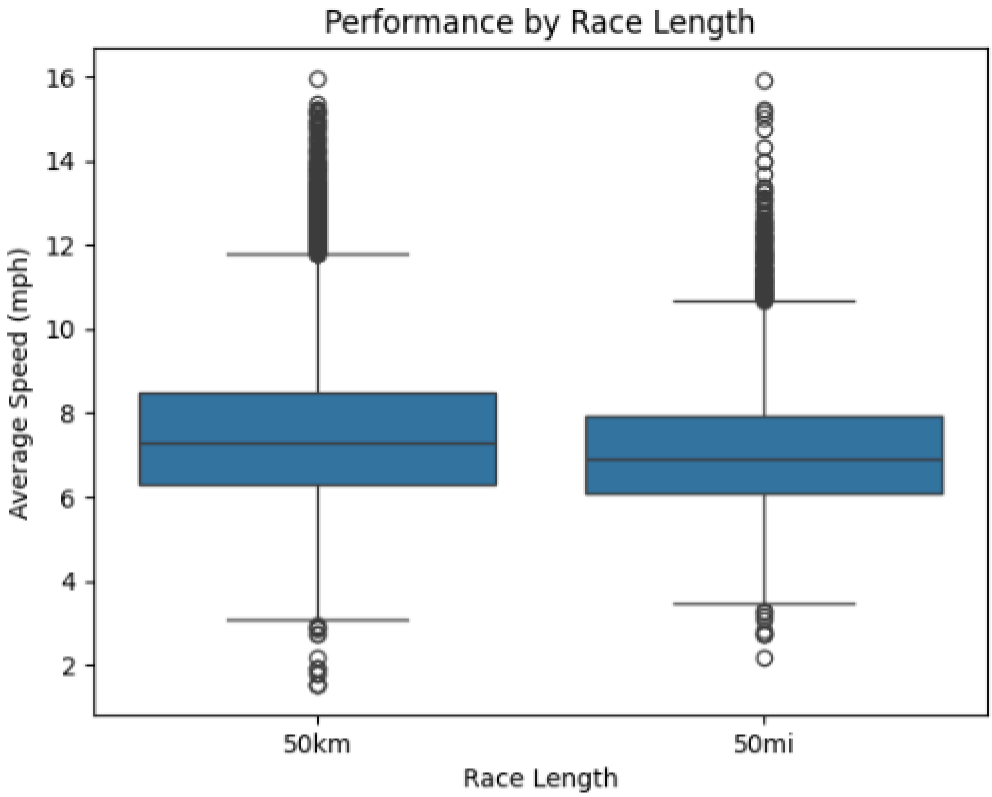
```
def calculate_mean_speeds(data):
    season_speed_avg = data.groupby('race_season')['athlete_average_speed'].mean().reindex(['Winter', 'Spring', 'Summer', 'Fall'])
    return season_speed_avg

# Calculate mean speeds
overall_mean = calculate_mean_speeds(df3)
male_mean = calculate_mean_speeds(df3[df3['athlete_gender'] == 'M'])
female_mean = calculate_mean_speeds(df3[df3['athlete_gender'] == 'F'])

# Plotting
fig, ax = plt.subplots(figsize=(10, 6))

bar_width = 0.2
seasons = ['Winter', 'Spring', 'Summer', 'Fall']
index = range(len(seasons))

# Bar positions
overall_bar = [i - bar_width for i in index]
male_bar = index
female_bar = [i + bar_width for i in index]

# Plot bars
bars_overall = ax.bar(overall_bar, overall_mean, width=bar_width, color='black', label='Overall')
bars_male = ax.bar(male_bar, male_mean, width=bar_width, color='blue', label='Male')
bars_female = ax.bar(female_bar, female_mean, width=bar_width, color='magenta', label='Female')

# Add value labels on top of bars
for bars in [bars_overall, bars_male, bars_female]:
    for bar in bars:
        height = bar.get_height()
        ax.text(bar.get_x() + bar.get_width() / 2.0, height, f'{height:.2f}', ha='center', va='bottom')
```

```
# Setting labels, title, legend, and grid
ax.set_xlabel('Race Season')
ax.set_ylabel('Athlete Average Speed (miles per hour)')
ax.set_title('Average Athlete Speed by Race Season')
ax.set_xticks(index)
ax.set_xticklabels(seasons)
ax.legend()

# Set y-axis limit to start from 5
ax.set_ylim(bottom=5)

plt.tight_layout()
plt.show()
```


Average Athlete Speed by Race Season
Overall
Male
Female
7.52
7.74
7.06
7.61
7.83
7.17
6.81
6.95
6.51
7.40
7.57
7.04
8.0
7.5
7.0
6.5
6.0
5.5
5.0
Athlete Average Speed (miles per hour)
Winter
Spring
Summer
Fall
Race Season

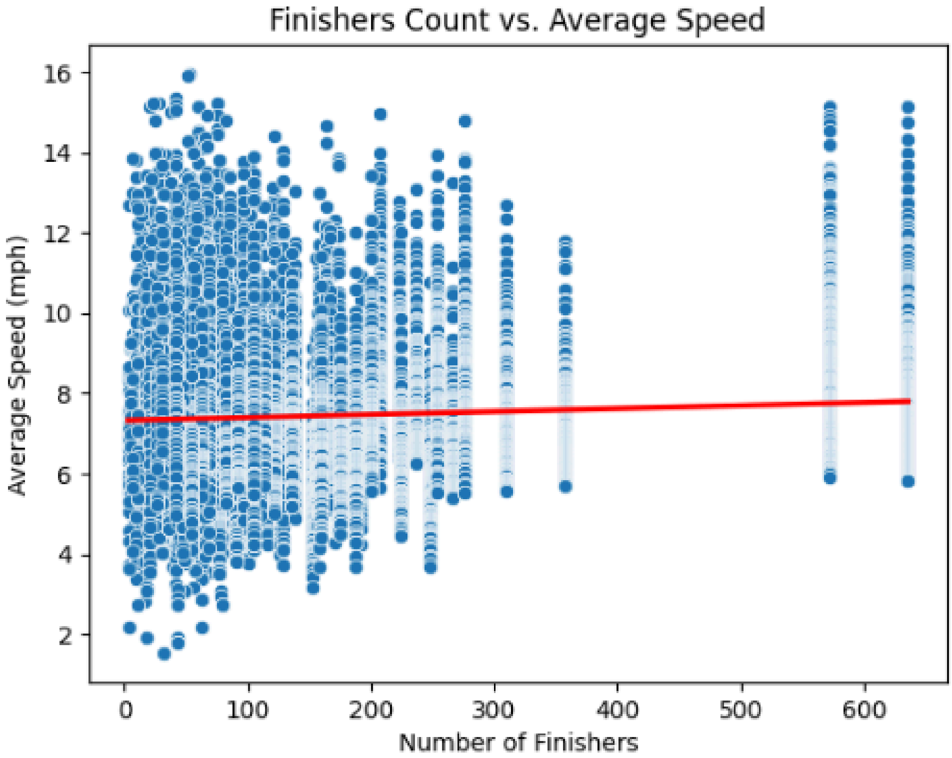
11. Performance by Race Length

[112]:
```
import seaborn as sns
sns.boxplot(x='race_length', y='athlete_average_speed', data=df3)
plt.xlabel('Race Length')
plt.ylabel('Average Speed (mph)')
plt.title('Performance by Race Length')
plt.show()
```



12. Finishers Count vs. Performance

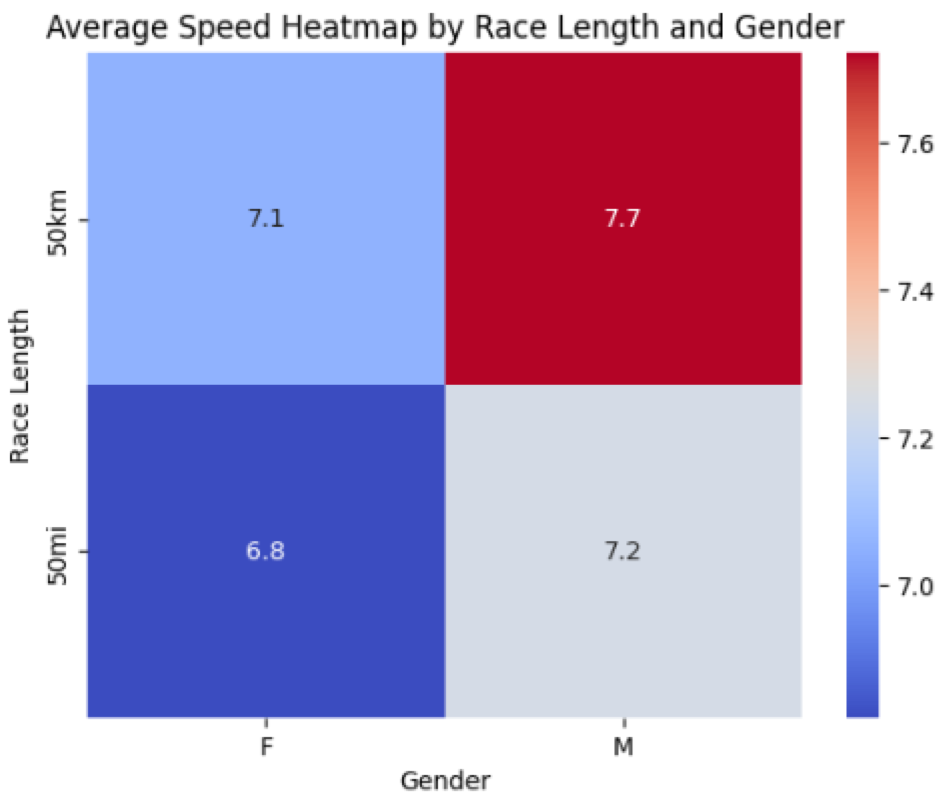
[122]:
```
sns.scatterplot(x='race_number_of_finishers', y='athlete_average_speed', data=df3)
sns.regplot(x='race_number_of_finishers', y='athlete_average_speed', data=df3, scatter=False, color='red')
plt.xlabel('Number of Finishers')
plt.ylabel('Average Speed (mph)')
plt.title('Finishers Count vs. Average Speed')
plt.show()
```



shripadkulkarni2212@gmail.com | +91 7378674911

13. Heatmap of Performance

[121]:
```
import numpy as np
pivot_table = df3.pivot_table(values='athlete_average_speed', index='race_length', columns='athlete_gender', aggfunc=np.mean)
sns.heatmap(pivot_table, annot=True, cmap='coolwarm')
plt.xlabel('Gender')
plt.ylabel('Race Length')
plt.title('Average Speed Heatmap by Race Length and Gender')
plt.show()
```



Key Learnings

- Exploratory Data Analysis (EDA):

  Conducted comprehensive EDA to uncover trends and patterns in the data.

- In-Depth Understanding of Tools:

  Mastered data manipulation and visualisation using Pandas, Matplotlib, and Seaborn.

- Data Filtering and Extraction:

  Applied advanced filtering techniques to isolate specific data subsets for analysis.

- Advanced Analytical Techniques:

  Utilised pivot tables, query functions, and group by operations for in-depth analysis.

- Project Documentation and Reporting:

  Developed detailed project documentation and reporting skills.

- AI Integration:

  Leveraged AI tools like ChatGPT to enhance insights and analysis.

- Graphical Analysis:

  Created and interpreted various graphs, including bar graphs, scatterplots, box plots, and heatmaps.

shripadkulkarni2212@gmail.com | +91 7378674911