

Aplikacija za ekstrakciju podataka sa websajta DUVStatistics i njihova analiza i vizualizacija

Studijski program: Računarstvo i informatika

Modul: Softversko inženjerstvo

Predmet: Studijsko-istraživački rad

|  |  |
| --- | --- |
| Student: | Profesor: |
|  |  |
| Jovan Vukadinović,  broj indeksa 1654 | Prof. dr Valentina Nejković |

Niš, avgust 2024. godina

Sadržaj

[1. Uvod 3](#_Toc176008678)

[1.1. Projekti za ekstraktovanje podataka 4](#_Toc176008679)

[2. Korišćenje Beautiful Soup biblioteke 4](#_Toc176008680)

[2.1. Instaliranje biblioteke 4](#_Toc176008681)

[2.2. Parsiranje HTML teksta 5](#_Toc176008682)

[2.3. Parsiranje udaljenog HTML-a 6](#_Toc176008683)

[2.4. Razlika između find i find\_all 7](#_Toc176008684)

[2.5. Ekstrakovanje Linkova i slika 7](#_Toc176008685)

[2.6. Nalaženje tagova pomoću njihovih atributa 8](#_Toc176008686)

[2.7. Pronalaženje više tagova na osnovu svojstva 9](#_Toc176008687)

[3. Priprema za ekstrakciju podataka sa stranice DUVStatistics 9](#_Toc176008688)

[4. Kod ekstrakcije podataka sa stranice DUVStatistics 13](#_Toc176008689)

[4.1. Analiza podataka 16](#_Toc176008690)

[4.1.1. Distribucija prosečne brzine po kategoriji – Avg.Speed km/h vs. Cat. 17](#_Toc176008691)

[4.1.2. Odnosa između starosne dobi, trajanja performansi, i prosečne brzine takmičara - Performance vs. Avg.Speed km/h and Age 18](#_Toc176008692)

[4.1.3. Distribucija srednje brzine po dužini trke - Performance vs. Avg.Speed km/h 19](#_Toc176008693)

[4.1.4. Performanse prema nacionalnosti - Avg.Speed km/h by Nat. 20](#_Toc176008694)

[4.1.5. Performanse prema godini rođenja - Avg.Speed km/h by YOB 21](#_Toc176008695)

[4.1.6. Broj osvojenih prvih mesta po državama – Rank vs. Nat. 22](#_Toc176008696)

[4.1.7. Broj osvojenih prvih mesta po godini starosti – Rank vs. YOB. 23](#_Toc176008697)

[4.1.8. Broj osvojenih prvih mesta po polu – Rank vs. Gender za obe tipove trka 24](#_Toc176008698)

[4.1.9. Opadanje prosečne brzine sa starosnom dobi – Avg.Speed km/h vs. Age 25](#_Toc176008699)

[4.1.10. Heatmap korelacija između performansi, starosti i prosečne brzine – Performanse vs. Age vs. Avg.Speed km/h 26](#_Toc176008700)

[5. Zaključak 27](#_Toc176008701)

[6. Literatura 28](#_Toc176008702)

# Uvod

Razvoj tehnologije i masovna dostupnost interneta doveli su do potrebe za prikupljanjem podataka sa veb stranica, odnosno veb ekstrakcijom. Ovaj proces podrazumeva automatsko pretraživanje i izdvajanje informacija sa sajtova, što je omogućilo brojne primene u različitim industrijama. Prvi veb ekstraktori su razvijeni od strane programera pretraživača kao što su Google i AltaVista, čime su postavljeni temelji za pretraživačke sisteme kakve danas poznajemo. Ovi alati pretražuju gotovo ceo internet, skenirajući veb stranice, izvlačeći informacije i kreirajući indekse koji omogućavaju brzo i efikasno pretraživanje sadržaja.

Sa razvojem tehnologije, mogućnost kreiranja veb ekstraktora postala je dostupna svima, omogućavajući pojedincima i kompanijama da ciljano prikupljaju podatke sa odabranih veb sajtova i koriste ih u svrhu daljih analiza ili migracija podataka. Digitalna transformacija, kao savremeni trend u poslovanju, uključuje i omogućavanje pristupa podacima putem API-ja, što dodatno olakšava proces prikupljanja informacija. Međutim, i pored dostupnosti API-ja, potreba za veb ekstrakovanje i dalje postoji, jer mnoge kompanije ne pružaju API-je, već se oslanjaju na postojeće veb stranice.

Primene veb ekstraktovanja su brojne i variraju od jednostavnih zadataka, kao što je prikupljanje liste proizvoda, do kompleksnih projekata, kao što je migracija sadržaja sa jedne platforme na drugu. Ovi procesi značajno ubrzavaju rad i smanjuju mogućnost grešaka u odnosu na ručno prikupljanje podataka. Ipak, kreiranje efikasnog veb ekstraktor nije jednostavan zadatak zbog raznolikosti rasporeda i struktura veb sajtova, kao i zbog činjenice da se ti rasporedi često menjaju. Ove promene zahtevaju stalno prilagođavanje ekstraktora kako bi se osigurao kontinuiran pristup željenim podacima.

Cilj ovog rada je da istraži izazove i mogućnosti veb ekstraktovanje, kao i da prikaže konkretne primere primene ovog procesa u situaciji prikupljanja podataka sa sajta DUVStatistics. Kroz analizu različitih tehnika i alata, nastojaće se pružiti uvid u efikasan način prikupljanja podataka sa veb stranica, uz osvrt na specifične probleme sa kojima se susreću programeri pri izradi ektraktora.

Takođe će jedan deo biti posvećen analizi prikupljenih podataka, gde će grafički prikazi omogućiti jednostavan pregled i lakšu analizu. Postojaće mogućnost upoređivanja naših ličnih rezultata sa onima koje takmičari širom sveta ostvaruju, što će olakšati proces pripreme, jer ćemo znati u kojoj meri smo spremni za trku. Takođe ćemo moći da procenimo koliko se rezultati koje postižemo slažu sa takmičarima istog pola, godinama starosti, i nacionalnosti kao što je naša.

## Projekti za ekstraktovanje podataka

Postoji mnogo slučajeva u kojima možete primeniti ekstraktovanje sajtova. Neki od njih su očigledni, dok su drugi ekstremni slučajevi. Glavni razlog za kreiranje ekstraktora je da se izvuku informacije sa veb sajta. Te informacije mogu biti lista proizvoda koje prodaje neka kompanija, nutritivni podaci o namirnicama ili NFL rezultati iz poslednjih nekoliko godina. Većina ovih projekata predstavlja osnovu za dalju analizu podataka. Ručno prikupljanje svih ovih podataka je dug proces i sklon greškama.

Jedan projekat koji bi mogao nekome pasti na pamet jeste preuzimanje celog interenta. To u suštini nije nemoguće, ali biste naišli na mnogobrojne probleme. Ukoliko biste počeli od jednog sajta, preuzeli ga, izvučete i pratite sve linkove na toj stranici, i preuzmete nove sajtove takođe u jednom momentu biste preuzeli sav sadržaj. Međutim preporuka za ovakav projekat nije tako jednostavna, jer biste najverovatnije vrlo brzo naišli na problem nedovoljne memorije, ali ideja je zanimljiva.

Jedan od najtežih delova prikupljanja podataka sa veb sajtova je to što se sajtovi razlikuju. Ne mislimo samo na podatke, već i na izgled stranice. Teško je kreirati ekstraktor koji će odgovarati za sve sajtove jer svaki sajt ima drugačiji izgled, koristi različite (ili nikakve) HTML ID-ove za identifikaciju polja, i slično. A da stvar bude još komplikovanija, mnogi sajtovi često menjaju svoj izgled. Kada se to dogodi, vaš ekstraktor više ne funkcioniše kao ranije. U tim slučajevima, jedina opcija je da ponovo pregledate svoj kod i prilagodite ga promenama za ciljani sajt.

Nažalost, ne postoji direktan način koji će vam pomoći da kreirate ekstraktor koji uvek funkcioniše, ako želite da pišete specijalizovane ekstraktore podataka.

# Korišćenje Beautiful Soup biblioteke

Beautiful soup jeste python biblioteka, koja predstavlja jednu od najpoznatijih i najjednostavnijih alata koji se koriste u ekstrakciji podataka sa veba. Vrlo je jednostavna i dosta lako koristi, a takođe je i veoma popularna te ujedno pruža i veliki opseg već postojećih projekata koji nam mogu biti veoma dobar uvod u njeno korišćenje.

## Instaliranje biblioteke

Za početak je potrebno posedovati Python na računaru na kome planirate koristiti ovu biblioteku, potom korišćenjem pip-a koji se instalira prilikom instalacije pythona, izvršiti sledeću komandu:

pip install beautifulsoup4

U konzoli će se nakon uspešne instalacije ispisati da je sada biblioteka spremna za rad, te ju je moguće importovati u python projekte.

## Parsiranje HTML teksta

Osnovna upotreba biblioteke Beautiful Soup, koju ćete videti u svakom tutorijalu, jeste parsiranje i izvlačenje informacija iz HTML stringa. Ovo je osnovni korak, jer kada preuzmete sadržaj veb stranice, šaljete taj sadržaj biblioteci Beautiful Soup na parsiranje, ali nema ništa vidljivo ako prosledite promenljivu parseru.

Većinu vremena ćete raditi sa sledećim višelinijskim stringom:

example\_html = """

<html>

<head>

<title>Your Title Here</title>

</head>

<body bgcolor="#ffffff">

<center>

<img align="bottom" src="clouds.jpg"/>

</center>

<hr/>

Chapter 3 Using BeaUtifUl soUp

43

<a href="http:// site.com">Link Name</a> is a link to another site

<h1>This is a Header</h1>

<h2>This is a Medium Header</h2>

Send me mail at <a href="mailto:support@yourcompany.com">

support@yourcompany.com</a>.

<p>This is a paragraph!</p>

<p>

<b>This is a new paragraph!</b><br/>

<b><i>This is a new sentence without a paragraph break, in bold italics.</i></b>

<a>This is an empty anchor</a>

</p>

<hr/>

</body>

</html>

"""

A da biste kreirali drvo parsera korišćenjem Beautiful Soup, potrebno je napisati sledeće:

soup = BeautifulSoup(example\_html, 'html.parser')

Drugi argument u pozivu ove funkcije predstavlja određivanje koji parser će se koristiti. Ukoliko se ovaj argument ne navede, prikazaće se greška sa sledećim sadržajem:

UserWarning: No parser was explicitly specified, so I'm

using the best available HTML parser for this system

("html.parser"). This usually isn't a problem, but if you

run this code on another system, or in a different virtual

environment, it may use a different parser and behave

differently.

The code that caused this warning is on line 1 of the file

<stdin>. To get rid of this warning, change code that looks

like this:

BeautifulSoup(YOUR\_MARKUP)

to this:

BeautifulSoup(YOUR\_MARKUP, "html.parser")

Ovo upozorenje je jasno definisano i govori vam sve što treba da znate. Pošto možete koristiti različite parser-e sa bibliotekom Beautiful Soup, ne možete pretpostaviti da će se uvek koristiti isti parser. Štaviše, to može dovesti do neočekivanog ponašanja, na primer, usporavanja vaše skripte.

Nadalje se može koristiti promenljiva soup za navigaciju kroz HTML.

## Parsiranje udaljenog HTML-a

Beautiful Soup nije HTTP klijent, tako da ne možete slati URL-ove njemu za ekstrakciju. Možete pokušati ovako:

soup = BeautifulSoup('statistik.d-u-v.org', 'html.parser')

Prethodni kod rezultira upozorenjem kao što je ovo:

UserWarning: "http:// statistik.d-u-v.org" looks like a URL. Beautiful Soup is not an HTTP client. You should probably use an HTTP client like requests to get the document behind the URL, and feed that document to Beautiful Soup.

Da biste pretvorili udaljene HTML stranice u soup objekat, trebali biste koristiti biblioteku requests:

soup = BeautifulSoup(requests.get('statistik.d-u-v.org').text, 'html.parser')

## Razlika između find i find\_all

Dve metode se koriste veoma često u okviru ove biblioteke, a to su find i find\_all.

Razlika između njih leži u funkciji i tipu vraćenih podataka: find vraća samo jedan rezultat—ako više čvorova odgovara kriterijumima, vraća prvi. A vraća None ako ništa nije pronađeno. find\_all vraća sve rezultate koji odgovaraju datim argumentima kao listu i ova lista može biti prazna.

To znači, svaki put kada tražite tag sa određenim ID-jem, možete koristiti find, jer možete pretpostaviti da se ID koristi samo jednom na stranici. S druge strane, ako tražite prvo pojavljivanje taga, možete koristiti i find. Ako niste sigurni, koristite find\_all i prolazite kroz rezultate.

## Ekstrakovanje Linkova i slika

Osnovna funkcija skrejpera je da izvuče linkove sa veb sajta koji vode ka drugim stranicama ili drugim veb sajtovima.

Linkovi se nalaze u <a> tagovima, a odredište linka je u href atributu tih togova. Da biste pronašli sve <a> tagove koji imaju href atribut, možete koristiti sledeći kod:

links = soup.find\_all('a', href=True)

for link in links:

print(link['href'])

Pokretanjem ovog koda na prethodno predstavljenom HTML-u dobićete sledeći rezultat:

http://site.com

[mailto:support@yourcompany.com](mailto:)

Poziv metode find\_all uključuje argument href=True. Ovo govori Beautiful Soup da vrati samo one <a> tagove koji imaju href atribut. Ovo vam omogućava da direktno pristupite ovom atributu na rezultovanim linkovima bez provere da li postoji.

Možete dodati bilo koji atribut u metodu find\_all, i možete pretraživati tagove gde atribut nije prisutan.

Drugi najčešći slučaj upotrebe ekstraktora je ekstrakcija slika sa veb sajtova i njihovo preuzimanje ili samo skladištenje njihovih informacija, kao što su gde se nalaze, njihova veličina prikaza, alternativni tekst i još mnogo toga.

Kao što je slučaj sa ekstraktorom linkova, ovde možete koristiti metodu find\_all na soup objektu i specifikovati filter tagove.

images = soup.find\_all('img', src=True)

Traženje prisutnog src atributa pomaže da pronađete slike koje imaju nešto za prikaz. Naravno, ponekad je izvorni atribut dodat putem JavaScript-a, i morate uraditi neku vrstu obrnutog inženjeringa.

## Nalaženje tagova pomoću njihovih atributa

Ponekad morate pronaći tagove na osnovu njihovih atributa. Na primer, već smo identifikovali HTML blokove za zahteve preko njihovog class atributa.

Prethodni delovi su vam pokazali kako da pronađete tagove gde je atribut prisutan. Sada je vreme da pronađete tagove čiji atributi imaju određene vrednosti.

Dva slučaja upotrebe dominiraju ovom temom, a to su pretraga po id ili class atributima.

soup.find('p', id='first')

soup.find\_all('p', class\_='paragraph')

Možete koristiti bilo koji atribut u metodama find i find\_all. Jedina izuzetka je class jer je to ključna reč u Python-u. Međutim, kao što možete videti, možete koristiti class\_ umesto toga.

To znači da možete pretraživati slike gde je izvor clouds.jpg.

soup.find('img', src='clouds.jpg')

Možete koristiti i regularne izraze za pronalaženje tagova koji su specifične vrste, a njihovi atributi ih kvalifikuju kroz neki uslov. Na primer, svi image tagovi koji prikazuju GIF fajlove.

soup.find('img', src=re.compile('\.gif$'))

Štaviše, tekst tag-a je takođe jedan od njegovih atributa. To znači da možete pretraživati tagove koji sadrže određeni tekst (ili samo deo teksta).

soup.find\_all('p', text='paragraph')

soup.find\_all('p', text=re.compile('paragraph'))

Razlika između dva prethodna primera je u njihovom rezultatu. Pošto u primer HTML-u ne postoji paragraf koji sadrži samo tekst "paragraph," vraća se prazna lista. Drugi poziv metode vraća listu paragraf tagova koji sadrže reč "paragraph."

## Pronalaženje više tagova na osnovu svojstva

Prethodno ste videli kako da pronađete jedan određeni tip taga (<p>, <img>) na osnovu njegovih svojstava.

Međutim, Beautiful Soup vam nudi i druge opcije, tako da na primer, možete pronaći više tagova koji dele iste kriterijume. Pogledajte sledeći primer:

for tag in soup.find\_all(re.compile('h')):

print(tag.name)

Ovde pretražujete sve tagove koji počinju sa slovom h. Rezultat bi izgledao ovako.

html

head

hr

h1

h2

hr

Još jedan primer bi bio pronalaženje svih tagova koji sadrže tekst "paragraph."

soup.find\_all(True, text=re.compile('paragraph'))

Ovde koristite ključnu reč True da biste pronašli sve tagove. Ako ne navedete atribut da biste suzili pretragu, dobićete listu svih tagova u HTML dokumentu.

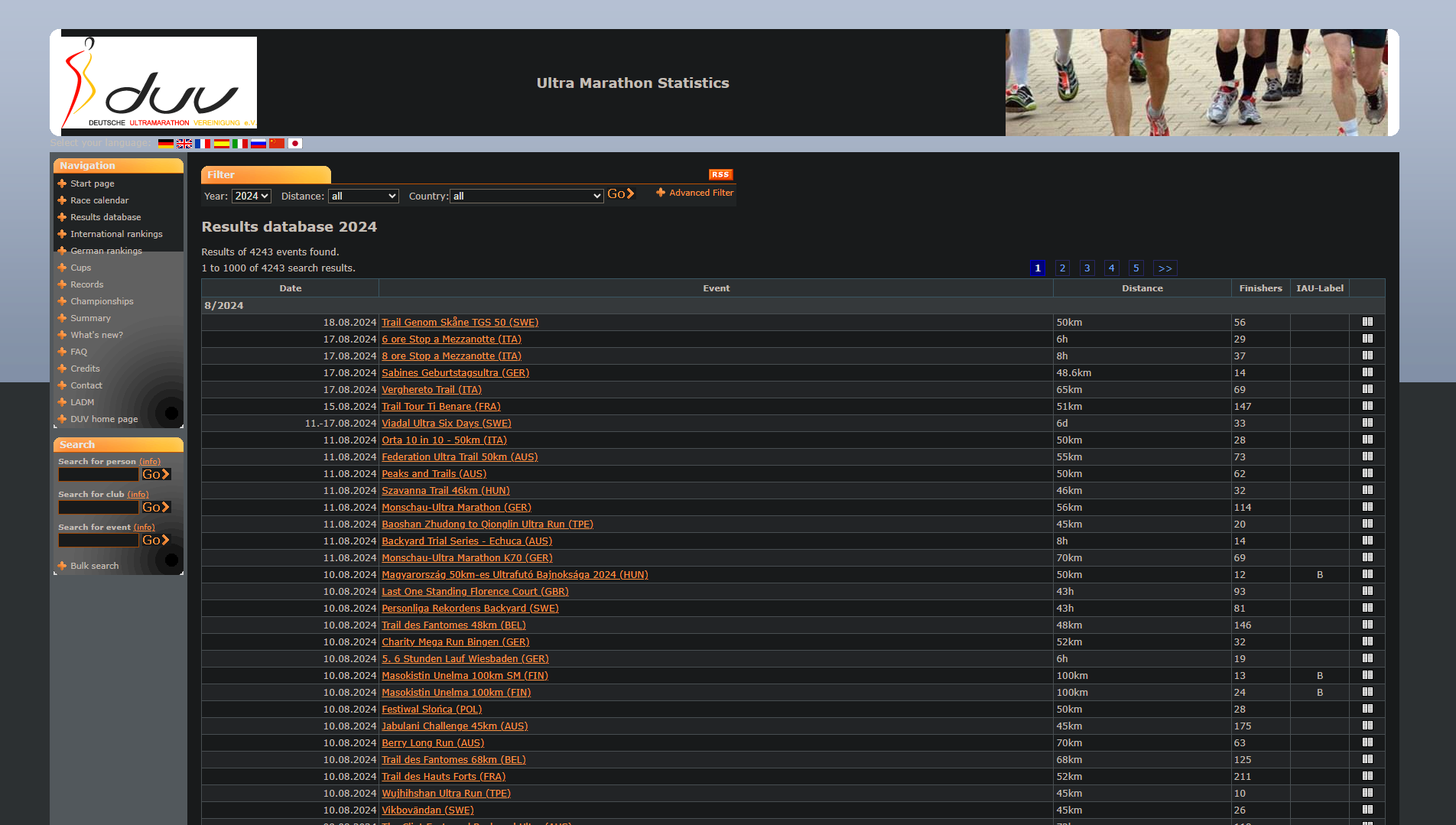
# Priprema za ekstrakciju podataka sa stranice DUVStatistics

Kako bismo počeli ekstrakciju podataka, potrebno je prvo dobro proučiti strukturu stranice odnosno razmotriti njen izgled i razumeti komponente od kojih se ona sastoji. Tako da je za početak potrebno otvoriti stranicu DUVStatistics i analizirati je viuzelno.

Možemo primetiti da postoji glavna podela stranice na header koji u ovom slučaju predstavlja sliku i naziv stranice, potom se ispod njega nalazi glavni deo stranice koji će biti od glavnog interesa. Taj glavni deo strane je podeljena na levi deo koji predstavlja neki vid navigacije, gde se nalaze prečice za različite stranice kao što su početna trana, kalendar trka, Nemačka rankiranja itd. Osim te navigacije može se primetiti i glavni sadržaj. Prvi stavka odozgo na dole koju vidimo jeste traka za odabir filtera, tu od ponuđenih opcija imamo sledeće sledeće:

* Godina: u izboru su sve godine od 1798 kada je početa da se beleži statistika pa do danas.
* Distanca: ponuđene su neke distance kao što su 50km, 100km, 50mi kao i vremenske distance u pogledu trajanja trka na primer 6h, 12h itd ukoliko je trka tog tipa.
* Država: ukoliko nas zanima određena država taj odabir možemo uneti ovde.

Potom klikom dugmeta Go možemo primeniti ove filtere i dobiti rezultate koji ga zadovoljavaju.

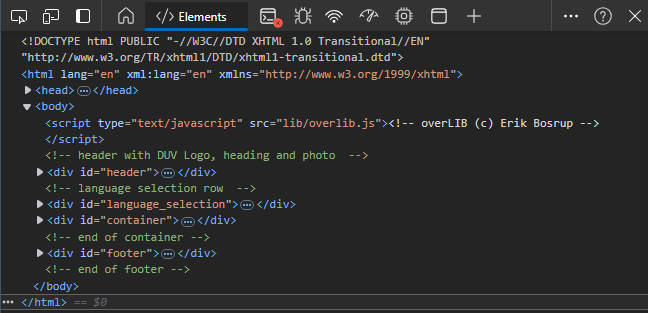


Slika 1. Izgled početne strane za pregled rezultata trka

Kada odaberemo određeni filter i prikažemo one podatek koji ga zadovoljavaju, oni su prikazani u odeljku ispod. Taj odeljak predstavlja neki vid tabele, a kolone koje ta tabela poseduje su:

* Date: Datum početka trke.
* Event: Naziv događaja, odnosno trke.
* Distance: Dužina trke izražena u razdaljini ili vremenu.
* Finishers: Broj takmičara koji su uspešno završili trku.
* IAU-Label: Od 2008. Godine, uvedene su oznake od strane National Athletics Federation.

Ukoliko na stranici kliknemo desni klik i odaberemo inpect element, otvoriće se prozor u okviru pretraživača koji nam prikazuje sve dostupne elemente sa veb stranice. A tu možemo prepoznati one koje sadrže naše podatke od interesa. Pogledajmo kako to izgleda na primeru.



Slika 2. Prikaz podele elemenata na stranici DUVStatistics

Primetimo nekoliko glavnih elemenata, to su <head>, <body> i <div>. Od našeg glavnog interesa će biti div elementi u okviru body taga. Vidimo da svi div elementi imaju svoji ID, najznačajniji za ekstrakciju podataka u okviru projekta se nalaze u divu sa ID-jem container, jer upravo on sadrži tabelu sa rezultatima trka.



Slika 3. Prošireni prikaz elementa sa ID-jem container

Kada proširimo prikaz elementa container, otvoriće se dodatni HTML tagovi, ukoliko proučimo strukturu stranice dublje, primetićemo da se nalaze tri HTML elementa <table>. U okviru Edge pretraživača, kao i u većini ostalih, postoji opcija označavanja elementa kada kursor stavimo na neki od tagova u ovom nizu. Time možemo testirati koja tabela će zapravo predstavljati onu koja sadrži podatke za prikupljanje. A to je u ovom primeru tabela sa ID-jem „Resultlist”.

Prisetimo se strukture HTML taga table, a ona izgleda ovako:

* <table>: Glavni element koji definiše tabelu.
* <thead>: Opcioni element koji grupiše redove tabele koji sadrže naslove kolona.
* <tr> (table row): Definiše red u tabeli.
* <th> (table header): Koristi se za definisanje naslova kolone, obično se nalazi unutar <thead> elementa. Tekst unutar <th> je obično podebljan i centriran.
* <tbody>: Opcioni element koji grupiše redove sa podacima u tabeli.
* <td> (table data): Definiše ćeliju unutar reda, gde se prikazuju podaci.

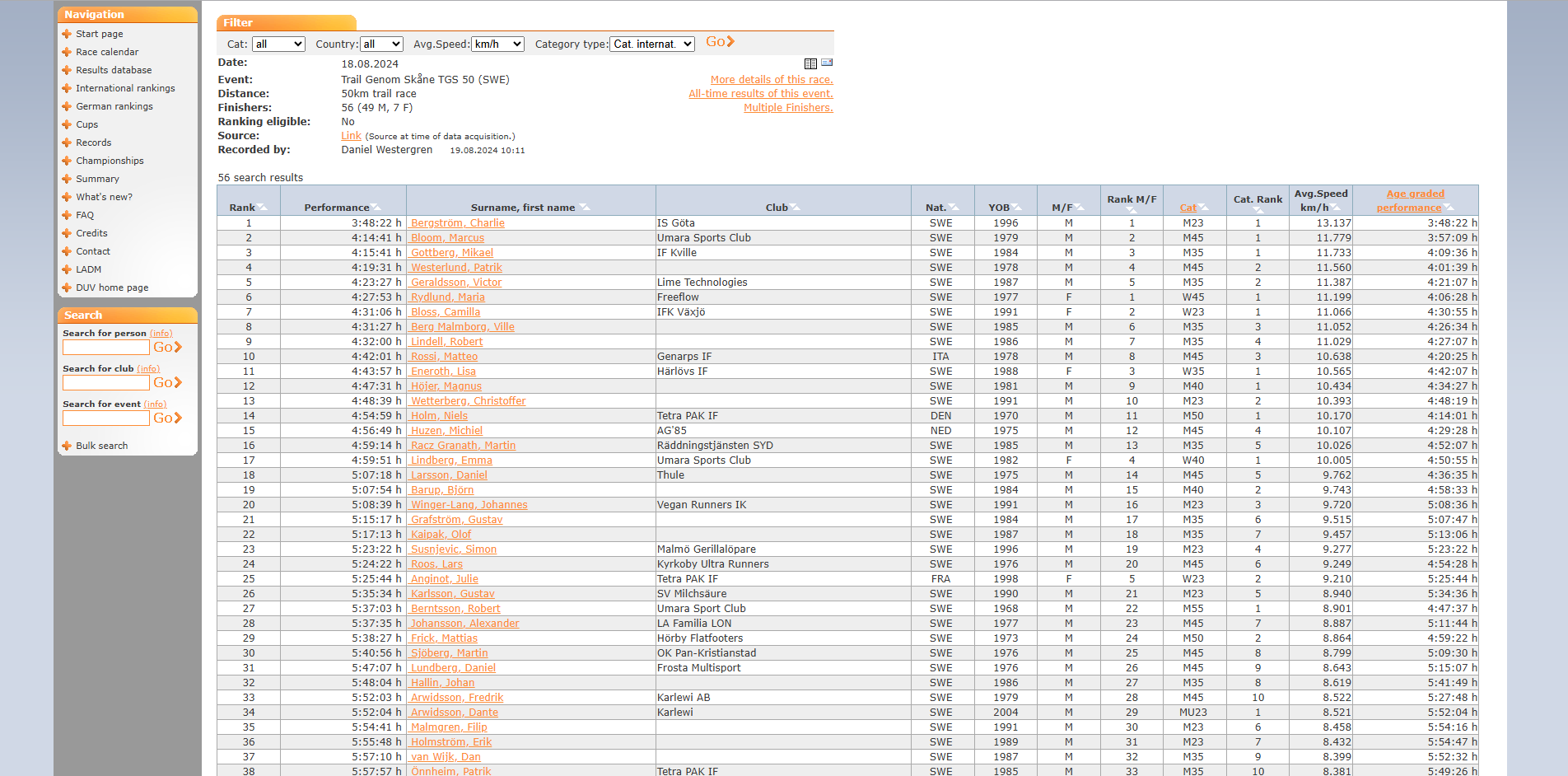
Sada kada proširimo tabelu sa ID-jem Resultlist, videćemo da se ona upravo sastoji iz ovih delova. <thead> element će sadržati sve one kolone koje želimo estraktovati za podatke, ukoliko pogledamo koje <th> elemente sadrži ovaj element primetićemo sledeće Date, Event, Distance, Finishers kao i IAU-Label.

Od velikog interesa za analizu u ovom projektu jeste kolona sa nazivom Event, ona će sadržati link ka novoj stranici na kojoj se mogu naći statistike učesnika te trke. Potrebno je da estraktor podataka otvori svaku stranicu koja se nalazi u toj koloni, sa koje će izvlačiti podatke za svakog učesnika trke pojedinačno.

Ukoliko otvorimo neki od tih linkova videćemo strukturu svake strane koja prikazuje informacije o učesnicima te trke. Ukoliko ponovo analiziramo stranicu po ugledu na prethodno objašljen način za početnu stranu, videćemo da je sadržaj manje više isti. Takođe postoji podela na levi i desni deo, desni deo je podeljen na gornji odnosno filter i donji odnosno tabelu od interesa. Upravo u toj tabeli se ponovo nalaze informacije koje treba ekstraktovati i koje će služiti za analizu podataka.

Tabela sadrži 12 kolona za svakog od takmičara, a to su sledeće kolone:

* Rank: Broj reda u tabeli, sortiran po redosledu vrednosti iz kolone Performance rastuće.
* Performance: Učinak takmičara u trci, odnosno vreme potrebno da se završi trka ukoliko se radi o trci na dužinu. Ili vrednost dužine koju je takmičar pretrčao za maksimalno vreme trke.
* Surname, first name: Imena i prezimena takmičara.
* Club: Naziv kluba ukoliko takmičar pripada nekom, odnosno prazno ukoliko ne.
* Nat.: Nacionalnost takmičara, tojest država koju predstavlja.
* YOB: Skraćenica za year of birth, u prevodu godina rođenja takmičara.
* M/F: Oznaka da li je takmičar muškog ili ženskog pola.
* Rank M/F: Mesto koje je takmičar osvojio u trci u okviru pola kome pripada.
* Cat: Kategorija starosti sportiste.
* Cat Rank: Mesto koje je takmičar osvojio u tric u okviru starosne kategorije kojoj pripada.
* Avg.Speed km/h: Srednja brzina takmičara izražena u kilometrima na sat.
* Age graded performance: Kategorizacija sportskih rezultata prema starosti omogućava fer poređenje učinka između različitih starosnih grupa i polova korišćenjem matematičkih modela i faktora koji uzimaju u obzir smanjenje performansi s godinama.



Slika 4. Prikaz stranice za pribavljanje informacije o učesnicima trke

# Kod ekstrakcije podataka sa stranice DUVStatistics

Sada kada su objašnjeni koraci ekstraktovanja podataka, videćemo na primeru koda za pribavljanje podataka sa stranice DUVStatistics. Kod je napisan u jeziku Python, a za pisanje koda i pokretanje je korišćen VS Code editor.

1. Biblioteke:

import requests

from bs4 import BeautifulSoup

import pandas as pd

import time

requests: Koristi se za slanje HTTP zahteva na veb stranice i preuzimanje njihovog sadržaja.

BeautifulSoup: Koristi se za parsiranje HTML sadržaja i lako navigaciju kroz strukturu stranice.

pandas: Koristi se za strukturisanje i manipulaciju prikupljenim podacima, posebno za skladištenje podataka u DataFrame objekte i snimanje u CSV fajlove.

time: Koristi se za pauziranje između zahteva kako bi se izbeglo preopterećenje servera.

1. Funkcija scrape\_table\_from\_url(url):

# Function to scrape a table from a given URL

def scrape\_table\_from\_url(url):

    response = requests.get(url)

    response.raise\_for\_status()

    soup = BeautifulSoup(response.text, 'html.parser')

    table = soup.find('table', id='Resultlist')

    if table is None:

        print(f"Table with ID 'Resultlist' not found at {url}.")

        return pd.DataFrame()

    # Initialize lists to hold data

    headers = []

    rows = []

    # Extract headers

    header\_row = table.find('tr')

    headers = [th.text.strip() for th in header\_row.find\_all('th')]

    # Extract data rows

    for row in table.find\_all('tr')[1:]:

        columns = row.find\_all('td')

        columns = [col for col in columns if 'KategorieHeader' not in col.get('class', [])]

        rows.append([col.text.strip() for col in columns])

    # Create a DataFrame

    return pd.DataFrame(rows, columns=headers)

Ova funkcija preuzima HTML sadržaj sa prosleđene URL adrese i koristi BeautifulSoup da izvuče tabelu sa ID-om Resultlist.

Prvo, funkcija proverava da li tabela postoji. Ako ne postoji, vraća prazan DataFrame i ispisuje poruku.

Ako tabela postoji, izdvaja zaglavlja (<th>) iz prvog reda tabele i koristi ih kao nazive kolona za DataFrame.

Zatim prolazi kroz sve redove tabele (osim zaglavlja), izdvaja podatke iz ćelija (<td>) i skladišti ih u listu rows.

Na kraju, vraća DataFrame sa prikupljenim podacima.

1. Glavni deo koda:

# URL of the main page

main\_url = 'https://statistik.d-u-v.org/geteventlist.php'

# Send a GET request to the website

response = requests.get(main\_url)

response.raise\_for\_status()  # Check if the request was successful

# Parse the HTML content

soup = BeautifulSoup(response.text, 'html.parser')

# Find the main table

main\_table = soup.find('table', id='Resultlist')

if main\_table is None:

    print("Main table with ID 'Resultlist' not found.")

else:

    # Initialize a DataFrame to hold all data

    all\_data = pd.DataFrame()

    # Extract links and scrape each

    for row in main\_table.find\_all('tr')[1:]:  # Skip header row

        cells = row.find\_all('td')

        links = [cell.find('a') for cell in cells if cell.find('a')]

        for link in links:

            href = link.get('href')

            full\_url = requests.compat.urljoin(main\_url, href)  # Construct full URL

            print(f"Scraping data from {full\_url}")

            # Scrape data from the linked page

            data\_df = scrape\_table\_from\_url(full\_url)

            # Append to the main DataFrame

            all\_data = pd.concat([all\_data, data\_df], ignore\_index=True)

            # Optional: Sleep to avoid too many requests too quickly

            time.sleep(1)  # Sleep for 1 second between requests

Prvo, šalje GET zahtev na glavnu URL adresu (main\_url) da preuzme HTML sadržaj glavne stranice.

HTML sadržaj se parsira pomoću BeautifulSoup, a zatim se traži glavna tabela sa ID-om Resultlist.

Ako glavna tabela nije pronađena, ispisuje se poruka o grešci.

Ako je tabela pronađena, kod kreira prazan DataFrame all\_data koji će sadržati sve podatke sa svih stranica.

Prolazi kroz sve redove tabele (osim zaglavlja), pronalazi linkove (<a> tagove) unutar ćelija i izdvaja njihov href atribut.

Za svaki pronađeni link, konstruiše se puna URL adresa i poziva se funkcija scrape\_table\_from\_url da preuzme podatke sa te stranice.

Prikupljeni podaci sa svake stranice se dodaju u glavni DataFrame all\_data.

Da bi se izbegao prevelik broj zahteva u kratkom vremenskom periodu, kod pravi pauzu od 1 sekunde između svakog zahteva pomoću time.sleep(1).

1. Snimanje podataka:

    # Save all collected data to a CSV file

    all\_data.to\_csv('all\_scraped\_data.csv', index=False)

    print("All data has been successfully scraped and saved to 'all\_scraped\_data.csv'.")

Na kraju, svi prikupljeni podaci se snimaju u CSV fajl pod nazivom all\_scraped\_data.csv.

## Analiza podataka

Analiza ovih podatak nam daje uvid u to koji tip takmičara je osvojio najveći broj trka, u pogledu pola, godina starosti, nacionalnosti itd. Grafikoni koji su iscrtani u nastavku nam na najjednostavni način omogućavaju vrlo pregledan i jasan opis podataka koji su ekstraktovani sa ovog sajta.

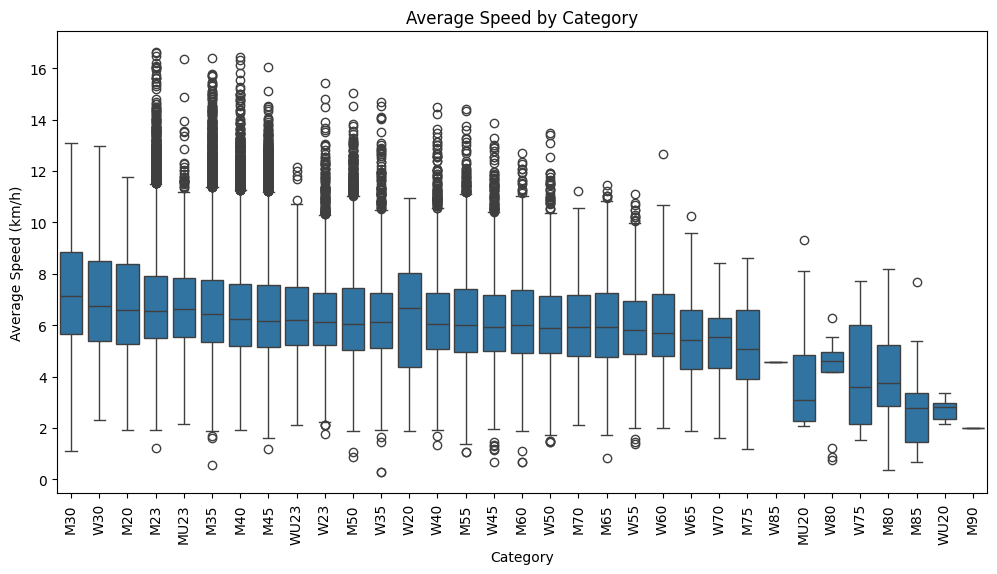
Ova analiza i statistika nam daje mogućnost da ocenimo naše lične rekorde i time se sa lakoćom možemo uporediti sa ostalim takmičarima širom sveta. Time se možemo bolje pripremiti za trku u zavisnosti od toga gde se ona održava ili videti koji tip trke nam više odgovara

Takođe ekstraktovani podaci su podeljeni na dve grupe, jedna predstavlja one trke koje se odnose na distancu, a druga na one koje predstavljaju vremenske trke. Te nam omogućava dublju podataka.

U nastavku su data 10 grafikona, i svaki od njih predstavlja drugačije poređenje podataka, neki se odnose na pol, neki na starost, dok neki uzimaju sve podatke u obzir.

Tekstualni podaci kao što su imena takmičara i nazivi klubova nisu bili od velikog značaja za analizu.

### Distribucija prosečne brzine po kategoriji – Avg.Speed km/h vs. Cat.



Slika 5. Distribucija prosečne brzine po kategoriji

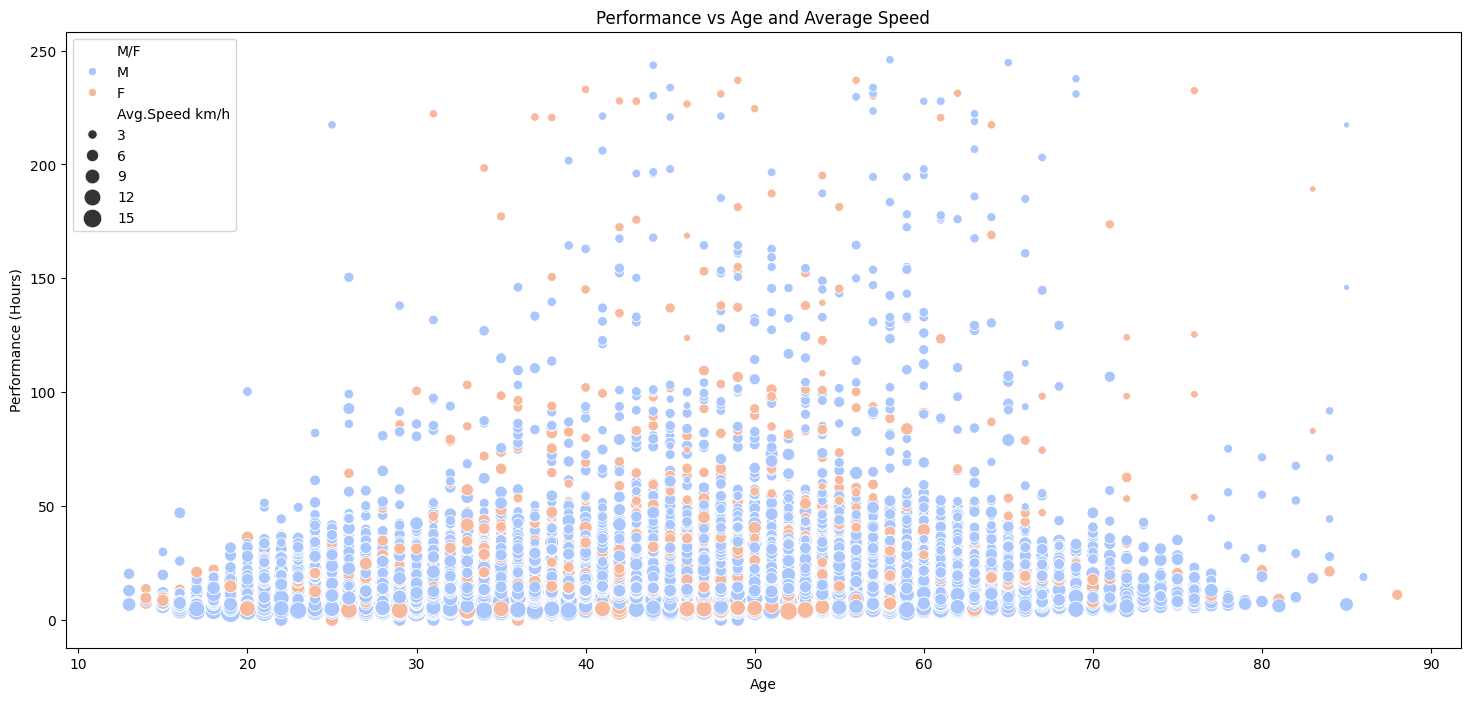
**Šta radi:**

* Prosečna brzina (Avg.Speed km/h) je izračunata za svaku kategoriju (Cat). Kategorije su sortirane u opadajućem redosledu prosečne brzine. Ovo znači da su kategorije sa najvećim prosečnim brzinama prikazane levo, dok su one sa manjim prosečnim brzinama desno.
* Box: Prikazuje interkvartilni opseg (IQR), tj. raspon između 25. i 75. percentila prosečnih brzina. Središnji deo kutije prikazuje median (srednja vrednost).

**Zašto je važno:**

* Raspodela brzina po kategorijama: Grafikon omogućava uvid u kako se prosečna brzina razlikuje između različitih kategorija takmičenja. Kategorije sa većim prosečnim brzinama su prikazane levo, dok su one sa manjim prosečnim brzinama desno.
* Varijabilnost unutar kategorija: Boxplot prikazuje koliko se prosečne brzine razlikuju unutar svake kategorije. Kategorije sa većim rasponom ili više ekstremnih vrednosti mogu imati veću varijabilnost u brzini.
* Poređenje između kategorija: Grafikon omogućava upoređivanje performansi između različitih kategorija, pomažući u identifikaciji koje kategorije imaju najviše i najniže prosečne brzine.
* Ekstremne vrednosti: Prikaz ekstremnih vrednosti može otkriti kategorije sa neobično visokom ili niskom brzinom u poređenju sa ostatkom svoje grupe.

### Odnosa između starosne dobi, trajanja performansi, i prosečne brzine takmičara - Performance vs. Avg.Speed km/h and Age



Slika 6. Odnosa između starosne dobi, trajanja performansi, i prosečne brzine takmičara

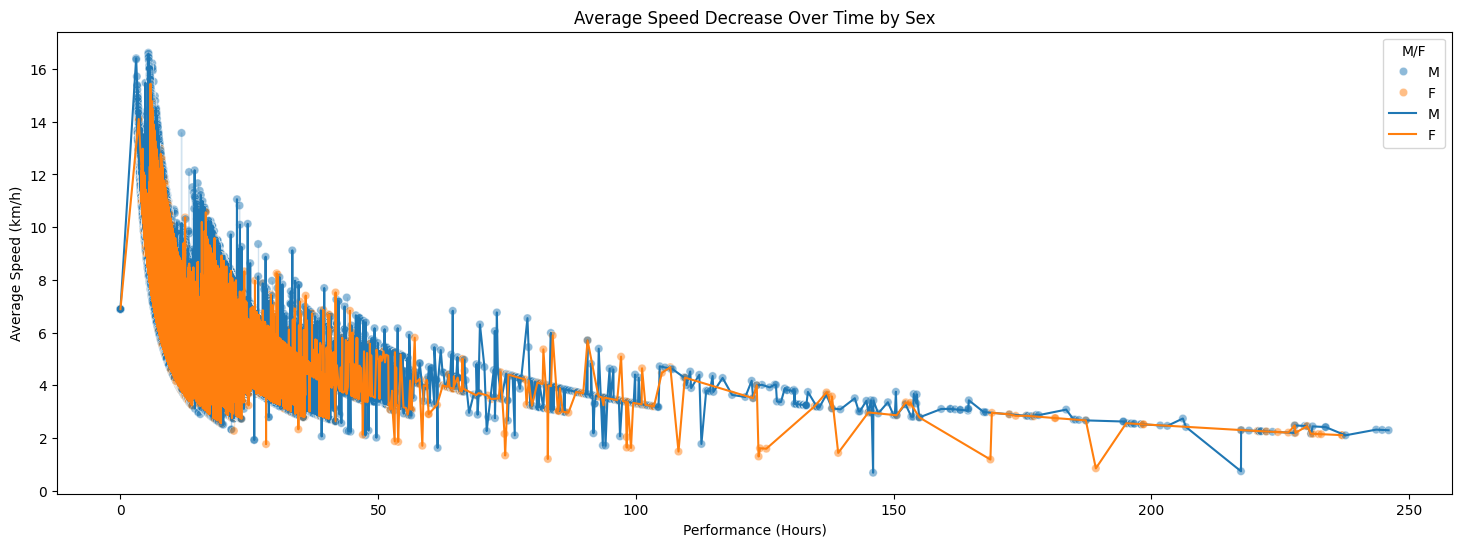
**Šta radi:**

* X-osa (Starosna Dob): Ova osa prikazuje starosnu dob takmičara, izračunatu kao 2024 - godina rođenja (YOB). Osa se proteže od mlađih takmičara na levoj strani do starijih takmičara na desnoj strani.
* Y-osa (Performanse u Satu): Ova osa prikazuje trajanje performansi u satima, gde je 'Performance' iz 'hh:mm' formatiran u decimalne sate. Osa se proteže od kraćih trajanja performansi na donjoj strani do dužih trajanja na gornjoj strani.
* Veličina Tačaka (Prosečna Brzina): Veličina svake tačke u grafikonu predstavlja prosečnu brzinu (Avg.Speed km/h). Veće tačke ukazuju na veće prosečne brzine, dok manje tačke predstavljaju niže prosečne brzine.
* Boje Tačaka (Pol): Boje tačaka pokazuju pol takmičara, pri čemu je svaka boja dodeljena jednom polu (npr., coolwarm paleta može koristiti boje poput plave za muškarce i crvene za žene).

**Zašto je važno:**

* Trend Performansi po Starosnoj Dobi: Grafik pokazuje kako se trajanje performansi menja u zavisnosti od starosne dobi. Može se uočiti da li stariji ili mlađi takmičari generalno imaju duže ili kraće trajanje performansi.
* Prosečna Brzina po Starosnoj Dobi: Veličina tačaka daje uvid u prosečnu brzinu takmičara u različitim starosnim grupama. Ako su veće tačke prisutne u određenim starosnim grupama, to može sugerisati da takmičari iz tih grupa postižu veće prosečne brzine.
* Razlike po Polu: Boje tačaka omogućavaju upoređivanje performansi između muškaraca i žena. Ovaj deo grafikona može pomoći u identifikaciji potencijalnih razlika u performansama između polova.
* Opšti Uvidi: Analizom raspodela veličina tačaka i njihovih boja može se dobiti uvid u to koje starosne grupe i polovi imaju najveće ili najmanje prosečne brzine, kao i kako se trajanje performansi menja u različitim starosnim grupama..

### Distribucija srednje brzine po dužini trke - Performance vs. Avg.Speed km/h



Slika 7. Distribucija srednje brzine prema dužini trke

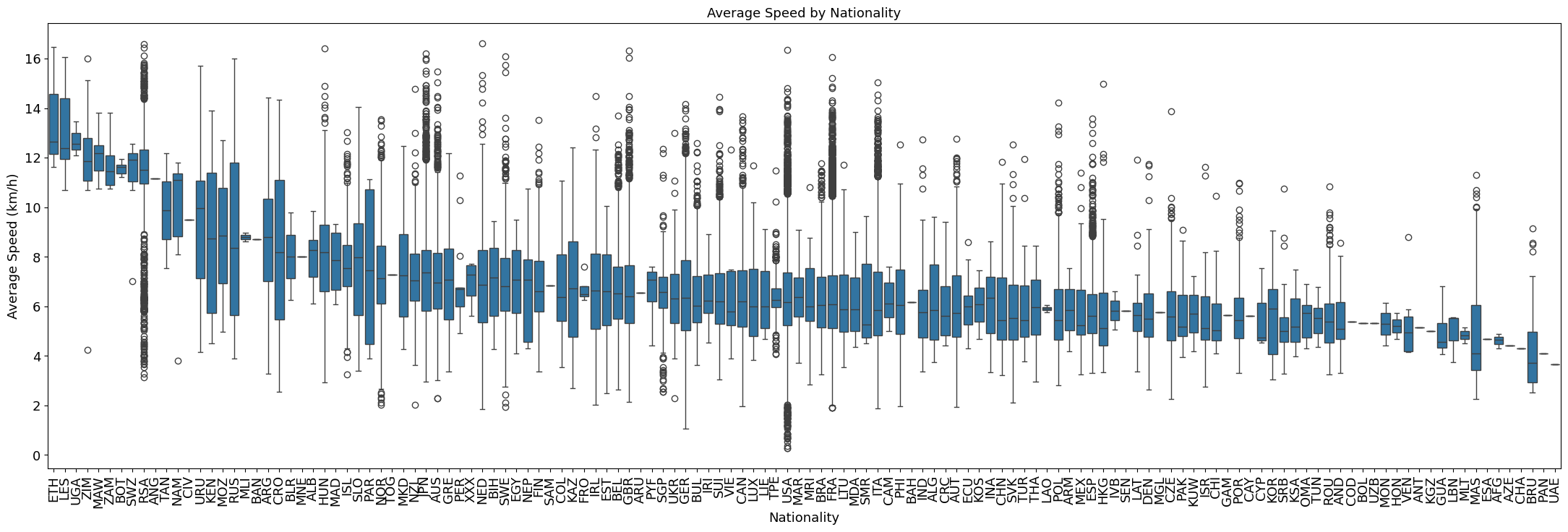
**Šta radi:**

* Ovaj graf prikazuje odnos između sati i performansi, uz razdvajanje po polovima (M/F).
* Scatterplot pokazuje kako srednja brzina prilagođene polovima variraju sa dužinom trke.
* Lineplot se koristi za prikaz trenda prosečne brzine u odnosu na vreme trajanja.

**Zašto je važno:**

* Trendovi za različite polove: Linije na grafikonu prikazuju kako se prosečna brzina menja sa povećanjem vremena trajanja performansi za muškarce i žene. Može se videti da li postoji razlika u načinu na koji vreme trajanja performansi utiče na brzinu između polova.
* Raspodela brzina: Scatter plot omogućava uvid u raspodelu prosečnih brzina u zavisnosti od vremena trajanja performansi za sve takmičare.
* Povezanost između vremena i brzine: Analizom grafikona možete uočiti da postoji opadajući trend u prosečnoj brzini kako se vreme trajanja trke povećava i da se ovaj trend razlikuje za muškarce i žene.
* Ekstremne tačke: Individualne tačke mogu otkriti izuzetne slučajeve gde su takmičari sa vrlo dugim ili vrlo kratkim vremenom trajanja performansi postizali izuzetne brzine.

### Performanse prema nacionalnosti - Avg.Speed km/h by Nat.



Slika 8. Ostvarene performanse prema nacionalnosti

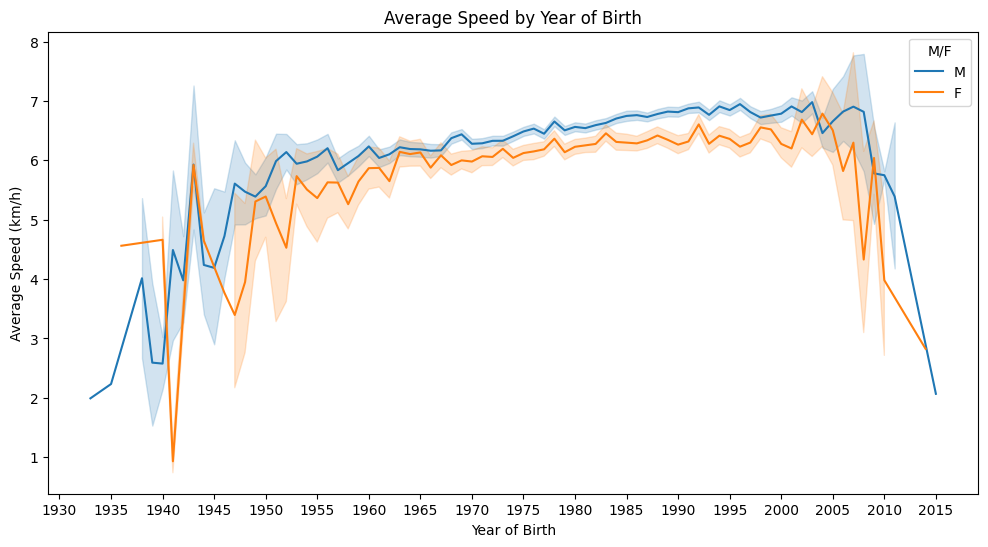
**Šta radi:**

* Ovaj graf prikazuje prosečnu brzinu učesnika razvrstanu po nacionalnosti (Nat.). Predstavljena je u opadajućem redosledu po proseku prosečne brzine svake nacionalnosti.
* Boxplot omogućava pregled medijane, interkvartilnog opsega i potencijalnih izuzetaka u prosečnoj brzini za svaku nacionalnost.

**Zašto je važno:**

* Raspodela brzina po nacionalnostima: Grafikon omogućava uvid u to kako prosečne brzine variraju između nacionalnosti. Nacionalnosti sa većim prosečnim brzinama imaju viši položaj na y-osi.
* Varijabilnost unutar nacionalnosti: Boxplot prikazuje koliko se prosečne brzine razlikuju unutar svake nacionalnosti. Nacionalnosti sa većim rasponom ili više ekstremnih vrednosti mogu imati veću varijabilnost u brzini.
* Ekstremne vrednosti: Prikaz ekstremnih vrednosti može otkriti nacionalnosti sa neobično visokom ili niskom brzinom u poređenju sa ostatkom svoje grupe.
* Poređenje između zemalja: Sortiranjem nacionalnosti prema prosečnoj brzini, lako je uporediti performanse različitih zemalja i identifikovati koje zemlje su najuspešnije u pogledu prosečne brzine.

### Performanse prema godini rođenja - Avg.Speed km/h by YOB



Slika 9. Performanse prema godini rođenja

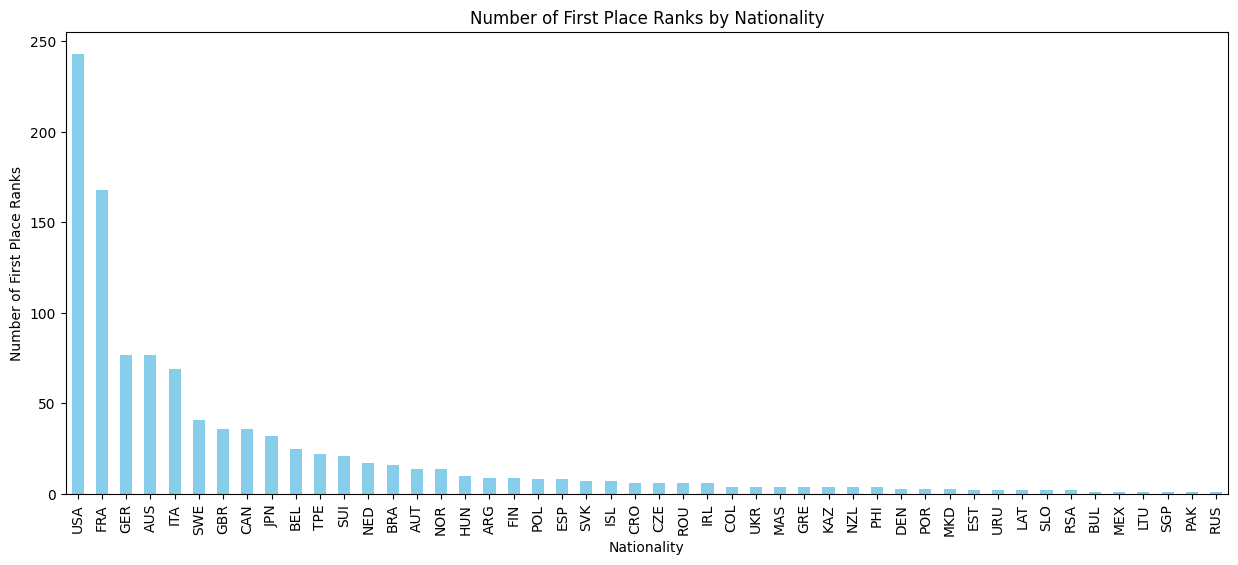
**Šta radi:**

* Ovaj graf prikazuje kako prosečna brzina varira u zavisnosti od godine rođenja (YOB), uz razdvajanje po polovima (M/F).
* Lineplot omogućava pregled trenda prosečne brzine u različitim godištima. Predstavljenih u intervalima od 5 godina.
* Plava linija predstavlja muški pol, a narandžasta ženski.

**Zašto je važno:**

* Trendovi kroz vreme: Grafikon pokazuje kako se prosečna brzina takmičara menja u zavisnosti od godine rođenja. Možete videti da li su takmičari rođeni u određenim periodima postizali veće ili manje prosečne brzine.
* Razlike između polova: Linije na grafikonu omogućavaju poređenje prosečne brzine muškaraca i žena u različitim godinama rođenja. Može se uočiti da li postoji konzistentna razlika između polova u performansama ili da li se ta razlika menja kroz vreme.
* Uticaj starosti: Pošto godina rođenja indirektno odražava starost takmičara, grafikon može ukazati na to kako starost može uticati na performanse, naročito kada se upoređuju različite generacije takmičara.

### Broj osvojenih prvih mesta po državama – Rank vs. Nat.

****

Slika 10. Broj osvojenih prvih mesta po državama

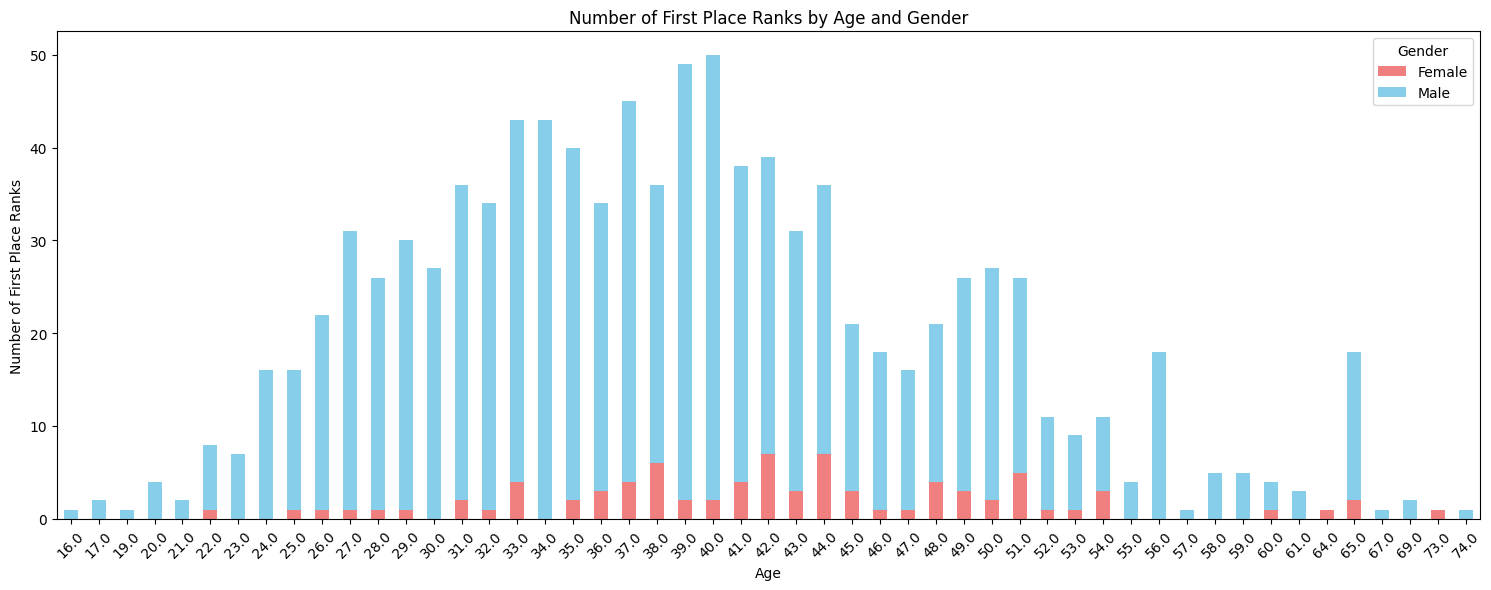
**Šta radi:**

* Prikazivanje podataka: Koristi bar grafikon za prikaz broja prvih mesta po nacionalnosti, gde svaka traka predstavlja jednu nacionalnost, a visina trake predstavlja broj prvih mesta.
* Nacionalnosti su sortirane prema broju osvojenih prvih mesta, od zemlje sa najviše prvih mesta do zemlje sa najmanje.
* Ovo sortiranje omogućava lako prepoznavanje koja zemlja ima najuspešnije takmičare u smislu broja osvojenih prvih mesta.

**Zašto je važno:**

* BarDominantne zemlje: Grafikon jasno pokazuje koje zemlje imaju najviše osvojenih prvih mesta. Na primer, ako jedan stubić značajno nadmašuje ostale, to ukazuje na dominaciju te zemlje u osvajanju prvih mesta.
* Poređenje između zemalja: Ovaj grafikon omogućava lako poređenje uspeha različitih zemalja. Na osnovu visine stubića, možete uočiti koje zemlje su generalno uspešnije u takmičenjima.
* Ukupna distribucija: Grafikon pruža uvid u to koliko je prva mesta podeljeno među različitim nacionalnostima, što može ukazivati na globalnu raspodelu uspeha u takmičenju.
* Trendovi po zemljama: Ako se na grafikonu vidi da određeni broj zemalja ima sličan broj osvojenih prvih mesta, to može ukazivati na sličan nivo konkurencije među tim zemljama.

### Broj osvojenih prvih mesta po godini starosti – Rank vs. YOB.

****

Slika 11. Broj osvojenih prvih mesta po godini starosti

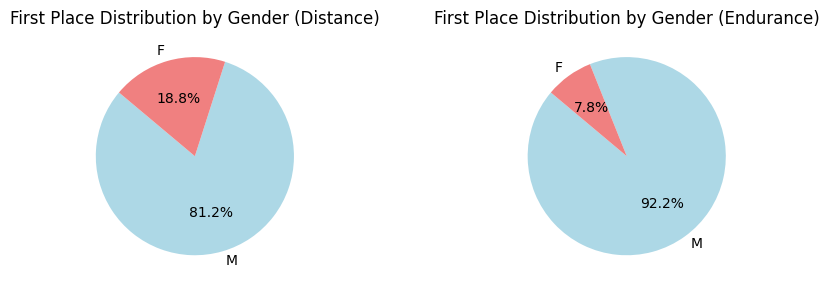
**Šta radi:**

* X-osa (Age): Prikazuje godine starosti takmičara koji su osvojili prvo mesto.
* Y-osa (Number of First Place Ranks): Prikazuje broj osvojenih prvih mesta u svakoj starosnoj grupi.
* Boja: Crvena boja predstavlja osvojena prva mesta od strane ženskog pola, dok plava boja označava broj osvojenih prvih mesta od strane muškog pola.

**Zašto je važno:**

* Distribucija prvih mesta po starosnim grupama: Grafikon pokazuje koje starosne grupe su najuspešnije u osvajanju prvih mesta. Na primer, možete videti da li su mlađi ili stariji takmičari uspešniji.
* Uporedba između polova: Unutar svake starosne grupe, možete uporediti uspešnost muškaraca i žena u osvajanju prvih mesta. Na primer, možete videti da li jedan pol dominira u određenim starosnim grupama.
* Ukupna dominacija: Ako neki stubić ima mnogo veći deo jedne boje u poređenju sa drugom, to ukazuje na dominaciju tog pola u osvajanja prvih mesta u toj starosnoj grupi.
* Starosni trendovi: Ovaj grafikon može pokazati kako starost utiče na šanse za osvajanje prvog mesta i da li postoji specifična starosna grupa koja dominira među takmičarima.
* Sa grafikona možemo zaključiti različite činjenice. Vidimo da prosečna starost za oba pola koja osvaja najveći broj prvih mesta jeste izmedju 39 i 40 godina.

### Broj osvojenih prvih mesta po polu – Rank vs. Gender za obe tipove trka



Slika 12. Broj osvojenih prvih mesta po polu takmičara

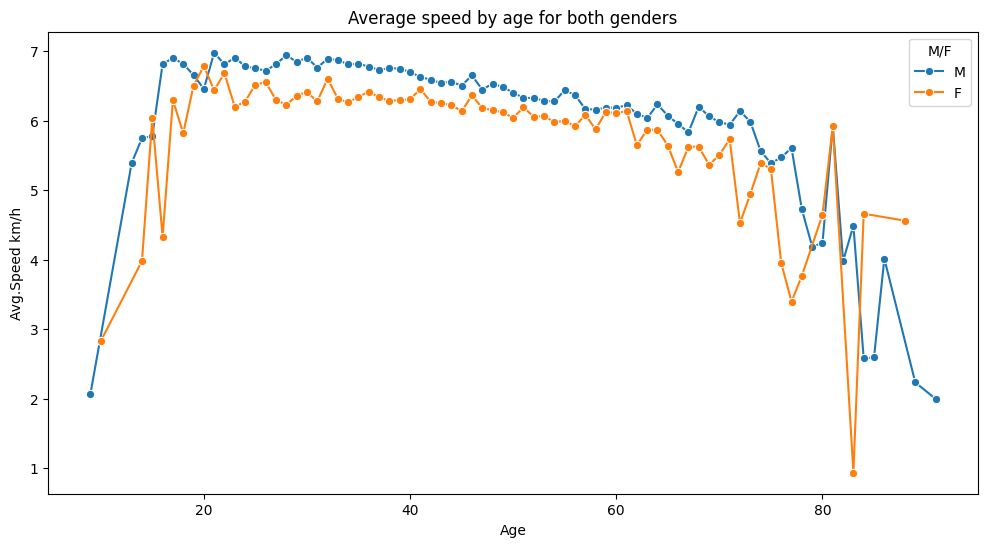
**Šta radi:**

* Prvi pie chart prikazuje raspodelu prvih mesta u disciplini "Distance".
* Drugi pie chart prikazuje raspodelu prvih mesta u disciplini "Endurance".
* Oznake "M" i "F" na pie chart-ovima predstavljaju polove: muškarce (M) i žene (F).

**Zašto je važno:**

* Uporedba između polova: Ovi grafici omogućavaju da odmah uočimo koliko su muškarci i žene bili uspešni u osvajanju prvih mesta u obe discipline. Na osnovu raspodele, možemo videti da li je jedan pol dominantan u osvajanju prvih mesta u određenoj disciplini.
* Razlike između disciplina: Možemo uporediti pie chart-ove da bismo uočili razlike između disciplina "Distance" i "Endurance". Na primer, ako jedna disciplina ima mnogo veću zastupljenost jednog pola u osvajanju prvih mesta u odnosu na drugu, to može ukazivati na specifične fizičke ili taktičke prednosti za taj pol u toj disciplini.
* Vizualni uvid u podatke: Ovi pie chart-ovi pružaju brz i intuitivan vizualni pregled raspodele prvih mesta, što omogućava lako uočavanje osnovnih trendova bez potrebe za dubinskom analizom numeričkih podataka.

### Opadanje prosečne brzine sa starosnom dobi – Avg.Speed km/h vs. Age



Slika 13. Opadanje prosečne brzine sa starosnom dobi takmičara

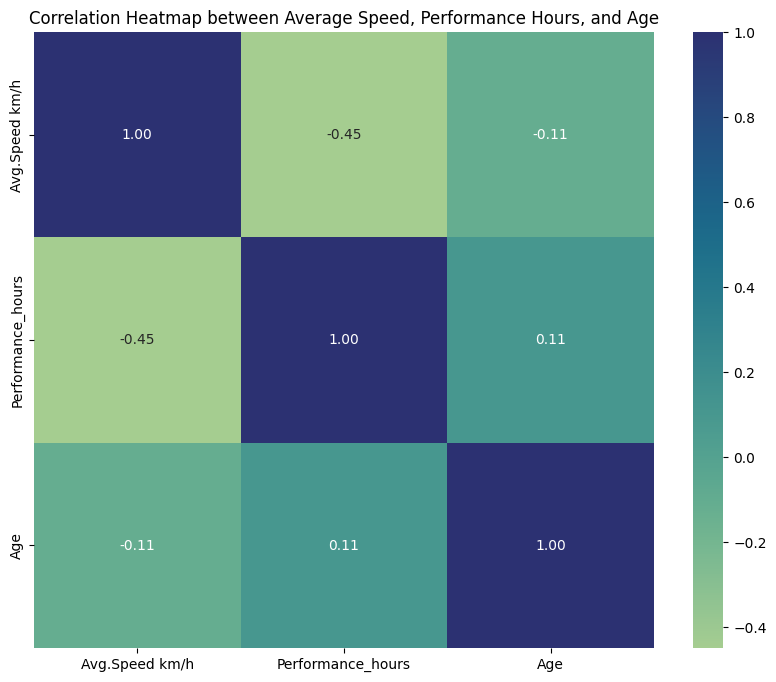
**Šta radi:**

* X-osa (Age): Predstavlja starosnu dob takmičara, svaka tačka na x-osi prikazuje određenu starosnu grupu.
* Y-osa (Avg. Speed km/h): Predstavlja prosečnu brzinu takmičara u kilometrima na sat (Avg.Speed km/h). Ovo je prosečna vrednost za sve takmičare unutar određene starosne grupe.
* Boja: Boja linija predstavlja da li se odnosi na muški ili ženski pol.

**Zašto je važno:**

* Graf omogućava direktno poređenje prosečne brzine između muškaraca i žena u različitim starosnim grupama. Moguće je uočiti da li postoji razlika u brzini između polova i da li se ta razlika menja sa starošću.
* Graf pokazuje da li prosečna brzina raste, opada ili ostaje stabilna kako takmičari stare. Na osnovu linija, može se uočiti kako starosna dob utiče na performanse takmičara za svaki pol.
* Ako postoje neočekivane oscilacije u prosečnoj brzini za određene starosne grupe, one će biti vidljive na grafu i mogu ukazivati na specifične faktore koji utiču na performanse tih starosnih grupa.

### Heatmap korelacija između performansi, starosti i prosečne brzine – Performanse vs. Age vs. Avg.Speed km/h



Slika 14. Heatmap korelacija između performansi, starosti i prosečne brzine

**Šta radi:**

* X-osa i Y-osa (Varijable): Obe ose prikazuju numeričke varijable (Avg.Speed km/h, Performance\_hours, i Age). Na preseku svake varijable prikazana je vrednost korelacije između te varijable i drugih.
* Vrednosti korelacije: Svaka ćelija u heatmap prikazuje koeficijent korelacije između dve varijable. Vrednosti se kreću od -1 do 1.
* Bojanje: boje u heatmap prikazuju snagu korelacije. Topli tonovi (poput crvene) označavaju jaču pozitivnu korelaciju, dok hladni tonovi (poput plave) označavaju jaču negativnu korelaciju.

**Zašto je važno:**

* Odnos varijabli: Ovaj grafikon omogućava identifikaciju kako se različite varijable međusobno odnose. Na primer, možete videti kako se prosečna brzina koreliše sa trajanjem performansi ili starosnom dobi.
* Prepoznavanje pattern-a: Vizualizacija korelacija može pomoći u prepoznavanju značajnih veza između varijabli, što može biti korisno za dalje analize ili modelovanje.
* Vizualizacija snage korelacija: Prikazuje koliko su varijable povezane, što može pomoći u odabiru relevantnih varijabli za analizu ili modelovanje.

# Zaključak

U procesu analize podataka sa veb stranice DUVStatistics, fokus je bio na preciznom prikupljanju i strukturiranju informacija koje su relevantne za dalje analize. Prvo je pažnja bila usmerena na identifikaciju ključnih elemenata stranice, kao što su tabele koje sadrže podatke o učesnicima i događajima. Korišćenjem BeautifulSoup biblioteke, podaci su uspešno ekstraktovani i smešteni u strukturiran format, što je omogućilo dalje analitičke korake.

Nakon ekstrakcije podataka, usledila je faza obrade i analize. Ovde je korišćenje biblioteke pandas bilo od suštinskog značaja, jer je omogućilo manipulaciju prikupljenim podacima, kao što su čišćenje nepotpunih ili nekonzistentnih vrednosti i transformaciju podataka u odgovarajući format za analizu. Na primer, podaci o učesnicima su grupisani po polu, godini rođenja i prosečnoj brzini, što je pružilo uvid u performanse različitih grupa učesnika na događajima.

Jedan od ključnih koraka u analizi bio je identifikacija obrazaca u podacima, kao što su varijacije u prosečnoj brzini u zavisnosti od starosne dobi i pola učesnika. Takođe, primenjene su napredne statističke metode kako bi se analizirali faktori koji najviše utiču na performanse, što je omogućilo dublje razumevanje rezultata.

Vizualizacija rezultata, pomoću biblioteka kao što su matplotlib i seaborn, omogućila je jasniji prikaz obrazaca u podacima. Na primer, boxplot grafici su korišćeni za poređenje performansi učesnika iz različitih zemalja, što je pružilo dodatne uvide o tome kako država iz koje takmičari potiču može uticati na rezultate.

Zaključci dobijeni iz ovih analiza pružili su ne samo kvantitativne podatke o performansama učesnika, već i kvalitetne uvide koji mogu biti korisni za buduće istraživanje i donošenje odluka. Na primer, identifikovani su faktori koji značajno utiču na uspeh u određenim disciplinama, što može poslužiti kao osnova za dalju optimizaciju treninga ili organizaciju događaja.

Ovakav pristup analizi podataka, koji kombinuje automatizovano prikupljanje, statističku analizu i vizualizaciju, omogućava ne samo efikasnu obradu velikih količina podataka, već i generisanje novih saznanja koja mogu imati značajan uticaj na praksu i teoriju u datoj oblasti.

# Literatura

[1] - „Beautifoul Soup documentation“ - https://tedboy.github.io/bs4\_doc

[2] - „Web Scraping with Python: Collecting More Data from the Modern Web“ - Ryan Mitchell

[3] - „Seaborn documnetation“ - https://seaborn.pydata.org/

[4] - „Data Wrangling with Python“ - Jacqueline Kazil, Katharine Jarmul

[5] - „Pandas documentation“ - https://pandas.pydata.org/docs/index.html

[6] - „Python Data Science Handbook“ - Jake VanderPlas

[7] - „Data Visualization: A Practical Introduction“ - Kieran Healy