Project Description

Question: Who is the greatest F1 driver of all time?
The problem is that there is over 850+ drivers that have participated in over 70 years of formula 1 racing history. How do you pick one of them to be the best? How do you even big to understand them? Most fans, resort to incidents or anecdotes to make their case. They're biased by peak performances and recent history but don't acknowledge the history of the race. With this project, I try to demonstrate a simplistic solution to determining the greatest of all time, by comparing a set of chosen driver against particular range of seasons as well as circuits.
This naively determined GOAT (greatest of all time) is chosen and even his contributions to the F1 team are shown in terms of race points – Overall as well as compared to other drivers in the team.

Design Rationales

Dividing the page into containers to provide side by side statistics.
This was important for the comparative study. Once the small subset of drivers is chosen, they are compared in terms of summary statistics. These summary statistics when placed beside each other provide an intuitive comparative analysis in the viewer's mind.

Input multiselect and sliders
The number of drivers is really high. Multiselect lets the user type in a dropdown to select their racers. This allows them to keep an eye on what is selected as well as how many. The avid fan would not need an exhaustive list of the drivers and would only be concerned in comparing his top favorites or rival contenders.

Grouped Bar Charts : Comparing drivers in two instances
Bar charts provides intuitiveness in terms of heights of each bar. When you display them in groups, where is group refers to multiple cases, it becomes easy to understand the winner or the loser within each group.
For comparing statistics between drivers, this was what I found to be the most useful. I explored other options that included scatters, lines as well as some more fancier boxplots, but nothing provided the simple intuition that grouped bar charts allowed for.

Map with towering heights:
Using the maps with a tower over the race circuit allowed me to capture, again in terms of heights, the number of races, each driver had made on the circuit. The map itself, provides a unique identification of the location. It goes down to the latitude and longitude of the track and it is actually exciting for the user to see the location.

Line chart to show change in positions:
In terms of positions, I used a line chart, as it shows the improvement or deterioration of the performance for each driver. Multiple lines on the chart help visualize the improvement or deterioration from the start to the end of the race.

Heatmap to visualize all points for Goat
The best way to visualize all the races (over 200+) usually for a driver was a tough problem. But when I saw the Github commit history heatmap, it struck me that this could be beneficial to provide a broad overview of the performance over all the races. Hence, I created a heatmap, that based on the year and the racetrack plots the "heat" in terms of points achieved in that race. Especially happy with this summarization of large amounts of information.

Development Process

Data gathering: I found the datasets from Kaggle as well as http://ergast.com/mrd/ .
I understood the information on the site as well as Kaggle to figure out what each column meant. As there were about 12 CSV files, I also spent some time considering the relationships between them. That helped me to merge them and extract the useful information that I needed.

Solution Development: I framed the question that I wanted to answer and evaluated the datasets to understand what could be useful and how I could combine them to expose results beneficial to answer the questions. For framing the question, I just thought of something I would want to know about F1.

Data preprocessing: Given the datasets, I decided to create a smaller dataset with the information that I found relevant and useful. I performed this process using a jupyter notebook (in Repo) and exported a smaller subset with 25k rows and ~30 columns.

Design: The main focus was to ensure that we can perform a comparative study, given few contenders. I tried to design it in a way I found intuitive to visually as well empirically compare.
I drew out some rough wireframes to help have a visual frame of mind that would help with the coding process.

Code: This was by far the easiest of all, once I had everything planned out, coding it out was just as good as finding the right API documentation to help build the components. I used the streamlit's advice to keep the browser and editor in a split screen mode and saw the updates live as and when I made changes to the code. This helped especially in configuring as well as designing the components.

Deployment: I faced a lot of bugs while using streamlit and that were mostly due to the limited capability it provided. For deployment, I created a requirements.txt file so that the serverside could install any and all packages being used. I also tried to use streamlit secrets – a method to store your secret API keys and tokens. But given it was relatively new and lacked a lot of documentation or features required, I quickly gave up on it.

The project probably took me about 25+ hours for the whole process.