# ScrapStats: A Cloud Application

Thomas Sato
CS 529 Cloud App Development
Willamette University
December 2024

All code to produce this application can be found here:
`https://github.com/thomascsato/CS-529-MMA`

Note: Because SageMaker is expensive, the application will be shut down, but screenshots will be provided. Example demo is also included in GitHub.

## 1 Introduction

The sport of Mixed Martial Arts (MMA) has been rapidly growing ever since the 1990s, beginning in ancient forms from cultures in China and Greece to a fully sanctioned professional combat sport today. It is now one of the biggest sports internationally, with athletes spanning every hemisphere of the world. Here is how the competition works: Two athletes face each other to reveal which style of fighting is the best– Karate, Judo, Boxing, Jiu-Jitsu, etc. When a fighter is knocked out from strikes (punches and/or kicks) or taps out due to a submission (chokes/joint locks), that fighter loses and their opponent is declared the winner of the bout. Originally, one of most important promotions of the sport from the 90s was the Ultimate Fighting Championship (UFC), which is by far the most popular promotion today.

The topic of this paper will be the inception and development of an MMA cloud application that takes two athletes in the UFC and uses data to compare their stats, producing probabilities that one fighter or the other will win the fight. It is intended to be interactive, where users of the application can choose fighters to compare based off of previous bouts in the UFC, and see which fighter is predicted to win. The application will be called ScrapStats.

The target audience of the app is fans of the sport who are curious about a deeper look into comparisons between fighters, as well as people who are interested in sports betting. While of course no predictive model is perfect, it provides an alternative perspective of the data that may be of interest to some groups of people.

Platforms like DraftKings and other sports betting companies have an interest in analyses like these, and they have their own methods and algorithms that calculate betting odds for their customers. However, there is not much transparency on how the numbers are calculated. Public opinion is a big factor in these calculations as well. This application seeks to provide better analysis and display of statistics between two fighters, in a way that is easily digestible and accurate. It will employ machine learning techniques in order to create a win probability, so that the customer can compare fighters against one another. It would even include an option to compare fighters who are not fighting each other, or who are from different weight classes for example.

## 2 Scope of Project

There is room for a multitude of other secondary features as well, such as something like a fan forum, where fans of the sport can discuss their analyses of different fights and whatnot, and even going as far as to have an innovative draft league feature.

Revenue streams for this app would mainly come from advertisements, but there is potential for setting up a subscription service for premium features.

The data exists to be taken advantage of. The only reservation I have about this idea is considering continuously updateable data sources and API permissions. The data is not coming from myself, but it must come from a third party. There are datasets out there already that are updated, but for this to be a legitimate sustainable business idea, more thought must be given to the data source.

The scope of this project mainly focuses on the fundamentals of developing and structuring the cloud application rather than implementing all of these potential features. The main functionality on the front end will consist of users being able to choose two fighters, weight class, and gender, and click on a button to produce a prediction that displays on screen along with basic fighter statistics. The process of learning how to use a cloud provider effectively and produce (at minimum) a working final product hosted in the cloud is the goal of this project.

## 3 Cloud Provider Selection

One thing that I have found while doing research on Amazon Web Services, Microsoft Azure, and Google Cloud Platform, is that while there are differences in each platform, for a small scale project like this one, I do not believe there would be a large difference between the three.

The key differentiators for AWS is that it is the oldest cloud platform of the three, which means it has had time to develop itself into a mature ecosystem of services. The wide range of services that AWS offers is also a key in considering which model to choose. Being the most well-developed service, its products have global reach as well, which is important when considering it if you are a multinational enterprise.

Microsoft Azure is most well known for being very well integrated within the Microsoft ecosystem. This is its big strength, especially for enterprises that are already heavily dependent on Microsoft services. This provides a great advantage to those companies who are already well-invested in Microsoft.

My research seems to suggest that GCP is most well known for its artificial intelligence/machine learning and analytic capabilities. I have also seen that it is a cheaper option compared to AWS or Azure due to its relative newness in the market.

I have found that the similarities of all three platforms are more telling than their differences. It seems to me that all three platforms have a pay-as-you-go option for their services, as well as free tiers. In terms of reliability, security, and scalability, it seems to me that all cloud providers are sufficient with regard to these areas, and neither one seems to suffer more than the other. It was difficult for me to find many sources online that had consistent opinions with regard to one service over the others.

Despite GCP being a platform that seems to have great capability for machine learning, I would like to use AWS for this application. I believe that the reputation of AWS as the longest standing cloud provider and the most mature make it a promising choice for this project and that skills developed using AWS will be more useful when I graduate and move on to working in the industry. I believe that even though GCP is known for having AI/ML capabilities, AWS has machine learning capabilities that are just as good with a more mature platform to support it. Because MMA is a global sport, having that global reach aspect of AWS is a big advantage as well. Finally, from what I understood from my research, Microsoft Azure

would have been a fine option as well, but it is more geared towards established companies who already use Microsoft products.

I believe that any one of the three have good merits for a project of this scale, and that it would be a successful idea no matter which one was chosen. All three seem to have the capability for a small scale machine learning application project, so I will choose AWS because it is the best-established of the three.

# 4 Compute Choice

In terms of cost management, serverless computing seems to be the most optimal for a term project. At scale, it may be more efficient to rely on other computing methods if there is a large amount of traffic to the application. For a small scale project like this one, however, since the computing resources are allocated by usage. Along with this, since the infrastructure is managed by the cloud provider, it will be easier to focus on perfecting the product rather than having to worry too much about infrastructural issues. This will save a lot of time in the development process in the long run. For the relatively large amount of data that will be used in this project, it is also beneficial to have serverless functions that efficiently process the data, which is another benefit to having infrastructure managed by the cloud provider.

AWS provides serverless computing services such as AWS Lambda that are useful for executing tasks in the back-end efficiently, such as querying databases in the cloud and invoking machine learning model end-points after a user hits the "Compare Fighters" button in the front-end of the application. Using this fully managed service without having to provision servers for this project simplifies the workflow and allows the developer of the application to focus less on the infrastructure and more on the actual development of the application.

# 5 Database

The data for this project comes from a web scraping Python script adapted from a public Kaggle project:
`https://www.kaggle.com/datasets/maksbasher/ufc-complete-dataset-all-events-1996-2024`

The script included used to generate the dataset used for this project is located in my Github:
`https://github.com/thomascsato/CS-529-MMA/blob/main/MMA_Fighter_Compare/src/mma_data_scraper.py`

Two datasets are used in the development of this project. The first dataset consists of every fight in the UFC from the first UFC event in 1993 until October 12, 2024, including the winner of the fight, weight class, fighter statistics like significant strikes during the match, and more. The second dataset consists of statistics of every UFC fighter who has competed in the promotion, including age, stance, average takedown accuracy, and more.

Both of these datasets are tabular, and are well structured for using a SQL database as opposed to a NoSQL database. The rows of the first dataset are individual fights, and the columns represent the statistics relating to that particular fight. The rows of the second dataset are fighters, and the columns represent statistics relating to the fighters. I decided to use AWS's Relational Database Service to set up a database in MySQL, and then access it during the execution of a Lambda serverless function. In a more mature rendition of this project, the database would be updated automatically every time a UFC event occurs, which is approximately weekly, but for the sake of simplicity in this project, the datasets will be static.

# 6 Network and Diagrams

Now that the main back-end components of the cloud provider, compute choice, and database are described, I would like to discuss the main architecture diagram of the application, shown in Figure 1.

Essentially, when the user of the application clicks on the "Compare Fighters" button once all the selections have been made, the application will send a HTTP POST request to an HTTP Web API in AWS, submitting data to the Lambda function handler that will then process the data.

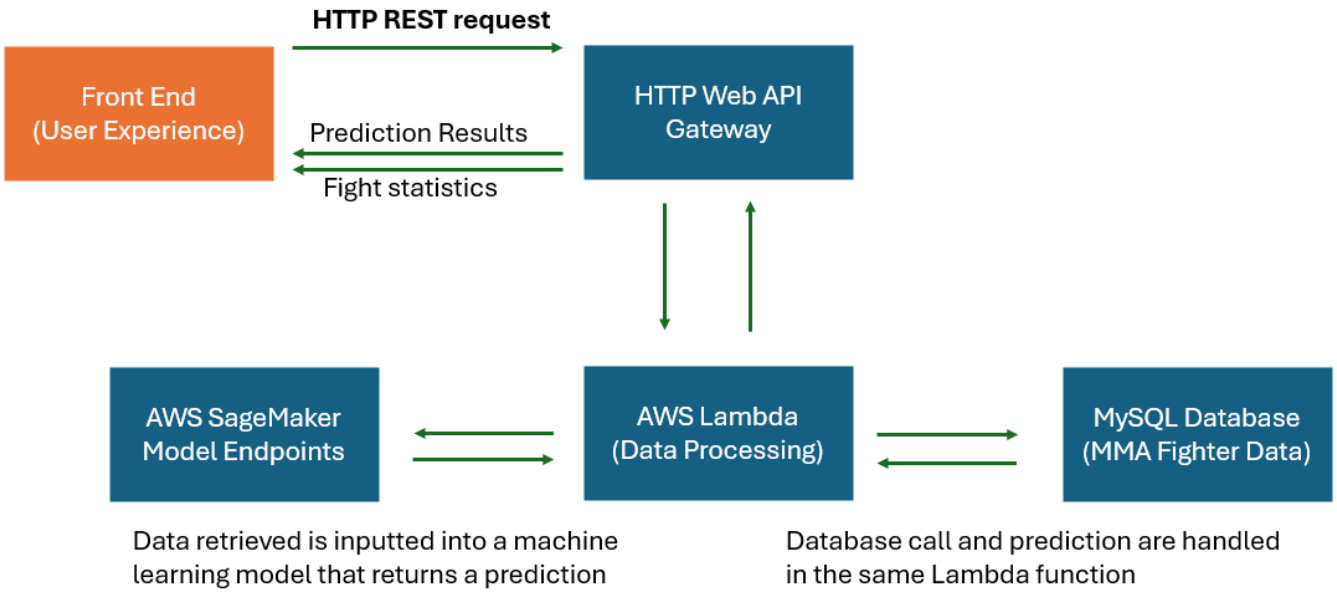


Figure 1: Architecture diagram of ScrapStats

In terms of communication protocols, since REST is best suited for resource-based operations like retrieving data, it is the best use case for this application. It is simple and does not require any real time communications, so it is very suitable for this use case. SOAP, gRPC, WebSocket, and Webhooks are all great for more specialized applications with more specific use cases, but for this one, the simple data request to data query, and then application to the model is the best for this MMA data comparison application. The complexity of SOAP and gRPC are not as suitable for a simple application like this one, and it does not require real time communication or an event driven architecture like WebSocket and Webhooks. For this reason, REST is most appropriate.

After the Lambda function receives the call with the data, it will query the database for the fighters specified by the end user. Then, based on the data that was queried, an endpoint for a machine learning model in AWS SageMaker will be invoked. The SageMaker endpoints will return the win probabilities back to AWS Lambda, which will then pass through the HTTP Web API gateway, making its way back to the front end of the application to display for the user. The specifics of the SageMaker implementation will be described in the next section, but this architecture diagram lays out the basis for the ScrapStats application.

# 7 Predictive Model Implementation

SageMaker is AWS's machine learning service that allows users to deploy machine learning models and integrate them within other AWS services. In this implementation, model endpoints are deployed that can be invoked with input data in order to quickly produce a prediction on a trained model.

As mentioned previously, there are two types of data that are used in this project: Fight data and fighter data. This is an important distinction to make because fight data is data that is recorded *after* the fight has

happened. Fighter data is historical data that is an aggregate of all past fights. There are a few methods that could be implemented in order to produce a win probability prediction based on this data. One way is that, since we are intending to create, in essence, a fight that has not happened yet, we first predict statistics for that fight before predicting an outcome of the fight. The data we will base these predictions on is the aggregated fighter statistics. Then, once those fight statistics are predicted, the historical fighter statistics in conjunction with the predicted fight statistics are used to predict a win probability. Essentially, there are two models in this particular methodology of the project: One to predict post-fight statistics of the two fighters, and one to then predict the win probability of each of the fighters.

For simplicity's sake, a Random Forest Regressor algorithm will be used to predict all of the post-fight statistics, most of which are numeric, and then a Random Forest Classifier will be used to predict win probabilities. There are many different models that could potentially be used to predict win probabilities; the point of this project is not to optimize and perfect the algorithm used to predict, but is more to have it work well enough for it to be used in a cloud application. With more time and effort poured into the actual algorithm would come a more sophisticated model.

# 8 Front-End Application

In the front end of the application, users will choose two fighters from a drop down menu, along with a weight class and gender, then will click on a button to display the statistics and the win probabilities on screen. See Figure 2 for the front end view.


Figure 2: Front end view of ScrapStats

Using this front end, it becomes very easy to select the fighters and weight class that they will be fighting at, and then view their probabilities of winning. The data from UFC 309, which occurred on November 16, 2024, was not included in the training dataset for the fighters or the fights. Therefore, as a sort of testing or validation dataset, fights from UFC 309 were inputted into the model using the application to see how well

the model performed. Results are shown in Figure 3 below, with the green indicating a correct prediction, and gray indicating an incorrect prediction.

| | | | |
|---|---|---|---|
| Jon Jones | 96 | 4 | Stipe Miocic |
| Charles Oliveira | 68 | 32 | Michael Chandler |
| Bo Nickal | 52 | 48 | Paul Craig |
| Viviane Araujo | 46 | 54 | Karine Silva |
| Mauricio Ruffy | 74 | 26 | James Llontop |
| Jonathan Martinez | 16 | 84 | Marcus McGhee |
| Jim Miller | 58 | 42 | Damon Jackson |
| Marcin Tybura | 64 | 36 | Jhonata Diniz |
| Mickey Gall | 58 | 42 | Ramiz Brahimaj |
| Bassil Hafez | 10 | 90 | Oban Elliott |
| Veronica Hardy | 22 | 78 | Eduarda Moura |

Figure 3: Win probabilities from ScrapStats on UFC 309

## 9 Conclusion

In the end, this of course is not a publication-ready product, and lots more features need to be implemented for this application to be marketable. Features to consider would be some sort of authorization and sign-up/sign-in system to create an account for ScrapStats in order to have more individualized functionalities. Some ideas to go along with that are a forum where people can discuss predictions from the application or a personal database that stores users' predictions for future reference. Some more advanced data visualizations to go along with the statistics, and a more polished front-end display will enhance the user experience. A continuously updated database will be a key feature as well, so the model can retrain itself based on the most recent data.

All of the potential expansions on the project being said, this application was a great introduction to application development in the cloud, and all the communication between the front and back end services work to create a functional application.