UFC Fight Data Analysis

Report submitted in partial fulfillment of the requirement for the degree of

B.Tech

in

Computer Science & Engineering

[](http://www.google.co.in/url?url=http://egov.engineeringwatch.in/bhagwan-parshuram-institute-of-technology-new-delhi-north-west-rohini/&rct=j&frm=1&q=&esrc=s&sa=U&ei=Qcc4VaDfFcOUsAHi54DwDg&ved=0CBUQ9QEwAA&sig2=VGYgGxagdSfRMxejTdK0Yg&usg=AFQjCNFN993UjYXRhdtuExrXVrIGKtxPmw)

Under the supervision

of

Ms. Charu Gupta

by

Ujjwal Chhabra (07920802716)

Piyush Sharma (40220802716)

Department of Computer Science & Engineering

Bhagwan Parshuram Institute of Technology

PSP-4, Sec-17, Rohini, Delhi-89

MAY - 2020

**DECLARATION**

This is to certify that Report titled “UFC Fight Data Analysis”, is submitted by us in partial fulfillment of the requirement for the award of degree B.Tech. in Computer Science & Engineering to BPIT, GGSIP University, Dwarka, Delhi. It comprises of our original work. The due acknowledgement has been made in the report for using others work.

**Date: 14/5/2020** Ujjwal Chhabra (07920802716)

Piyush Sharma (40220802716)

**Certificate by Supervisor**

This is to certify that Report titled “UFC Fight Data Analysus” is submitted by Ujjwal Chabra (07920802716) & Piyush sharma (40220802716) in partial fulfillment of the requirement for the award of degree B.Tech in Computer Science & Engineering to BPIT, GGSIP University, Dwarka, Delhi. It is a record of the candidates own work carried out by them under my supervision. The matter embodied in this Report is original and has not been submitted for the award of any other degree.

**Date: 14/5/2020 Supervisor**

**Certificate by HOD**

This is to certify that Report titled “UFC Fight Data Analysus” is submitted by Ujjwal Chabra (07920802716) & Piyush sharma (40220802716) under the guidance of Mr. Piyush Bhardwaj in partial fulfillment of the requirement for the award of degree B.Tech in Computer Science & Engineering to BPIT, GGSIP University, Dwarka, Delhi. The matter embodied in this Report is original and has been dully approved for the submission.

**Date: 14/5/2020 Dr. Deepali Virmani**

**ACKNOWLEDGEMENT**

This Project report, while an achievement that bears my name, would not have been possible without the help of others. I am glad to take this opportunity to thank the people who helped me to make this work possible.

Firstly, I would like to sincerely thank my Supervisor Ms. Charu Gupta, for his able guidance. His suggestions, recommendations and comments were invaluable for this report. The report would not have been possible without his constant support and belief in my capabilities. I owe my gratitude to him for showing me the right direction that guided me to overcome the problems I faced during my report work. In spite of his busy schedule, he patiently listened to my doubts and shared his valuable knowledge. It has been my privilege to study at BHAGWAN PARSHURAM INSTITUTE OF TECHNOLOGY, NEW DELHI which is one of the leading technology colleges in India. I would like to express my warm gratitude to my family for their love and support. Finally, I would like to thank all my batch-mates for motivating me and helping me with sincerity and patience.

**Table of Contents**

Chapter 1 : Introduction……………………………………………………….. 1

Chapter 2 : System Analysis and Design

2.1 : Software Requirement Specifications (H/W and S/W

requirements)…………………………………………..…………3

2.2 : Flow Chart ……………………………………………….………25

Chapter 3 : Proposed Work (Your Algorithm)………………………..………..26

Chapter 4 : Implementation and Results………………………………..………29

Chapter 5 : Conclusion…………………………………………………………49

Chapter 6 : Future Work…………………………………………….………….50

REFERENCES

**List of Figures**

Fig. 4.1 Red vs blue team with draw & match cancelled in pie chart ………………………… 25

Fig. 4.2 Red vs blue fighter age ………………………………………………………………………………….26

Fig. 4.3 No of blue fighters in each age group …………………………………………………………….26

Fig. 4.4 No of red fighters in each age group ……………………………………………………………..27

Fig. 4.5 Red and blue fighters of age less than and greater than …………………………………27

Fig. 4.6 mean age difference of fighters ……………………………………………………………………..28

Fig. 4.7 Red and blue fighter height distribution …………………………………………………………28

Fig. 4.8 Red and blue fighter height difference on same plot ……………………………………..29

Fig. 4.9 Mean difference in height by numbers …………………………………………………………..29

Fig. 4.10 How the win happened ………………………………………………………………………………30

Fig. 4.11 How win happened by each age group ………………………………………………………..31

Fig. 4.12 Strikes landed by fighter when match ended by no contest and draw ………….31

Fig. 4.13 Significant number of strikes by blue and red fighter ……………………………………32

Fig. 4.14 Most popular country for MMA ……………………………………………………………………32

Fig. 4.15 Most popular Cities by blue fighters …………………………………………………………….33

Fig. 4.16 Fig. 4.16 Grappling reversals, grappling stand-ups and grappling takedowns landed in each category in round 1 ………………………………………………………………………….34

Fig. 4.17 Grappling reversals and takedown in weight classes

between 70 - 80 in round I …………………………………………………………………………………………34

Fig. 4.18 Clinch head strikes, Clinch leg strikes for 80-90 Category …………………………….35

Fig. 4.19 Round 1 vs Round 5 head strikes by fighters ………………………………………………..35

Fig. 4.20 Number of UFC events by country ……………………………………………………………….36

Chapter 1- Introduction

The Ultimate Fighting Championship (UFC) is an American mixed martial arts (MMA) promotion company based in Las Vegas, Nevada, which is owned and operated by Endeavor Group Holdings along with Silver Lake Partners, Kohlberg Kravis Roberts and MSD Capital via Zuffa, LLC.[3][4] It is the largest MMA promotion company in the world and features some of the highest-level fighters in the sport on its roster.[5] The UFC produces events worldwide that showcase twelve weight divisions (eight men's divisions and four women's divisions) and abides by the Unified Rules of Mixed Martial Arts.[6] As of 2020, the UFC has held over 500 events. Dana White has been UFC president since 2001. Under White's stewardship, the UFC has grown into a globally popular multi-billion-dollar enterprise.[7]

The first event was held in 1993 at the McNichols Sports Arena in Denver, Colorado.[8] The purpose of the early Ultimate Fighting Championship competitions was to identify the most effective martial art in a contest with minimal rules and no weight classes between competitors of different fighting disciplines like boxing, kickboxing, Brazilian Jiu-Jitsu, Sambo, wrestling, Muay Thai, Karate, Taekwondo, and Judo. In subsequent events, fighters began adopting effective techniques from more than one discipline, which indirectly helped create a separate style of fighting known as present-day mixed martial arts.[9] In 2016, UFC's parent company, Zuffa, was sold to a group led by William Morris Endeavor (WME–IMG), including Silver Lake Partners, Kohlberg Kravis Roberts and MSD Capital[10] for US$4.025 billion.[11]

With a TV deal and expansion in Australia, Asia, Europe,[12][13][14] and new markets within the United States, the UFC has increased in popularity, and has achieved greater mainstream media coverage; the promotion brought in a total revenue of US$609 million in 2015,[15] and its next domestic media rights agreement with ESPN was valued at $1.5 billion over a five-year term. [16]

--

In this project we’ve visualized the data to find if we can infer anything from all the fights that have happened so far and try to answer questions like, in which country is UFC most popular, what’s the difference in fights style in round I vs round V, how age affects the fighter, what are most popular ways fights end, how height makes a difference in the game, red vs blue fighter winning ratio, male vs female participation etc.

After running the whole visualization, we found out answers to all those questions. Our methodology was very simple, after basic preprocessing of data we jumped straight into data visualization. The data was easy to manipulated and plot. All missing values were replaced with mean. Later seaborn library was used to plot most of the graphs. It’s a matplotlib based library but provides more visually appealing charts.

Problem Statement

Mixed martial arts (MMA) is a full-contact combat sport that allows striking and grappling, both standing and on the ground, using techniques from other combat sports and martial arts. The Ultimate Fighting Championship (UFC) is an American mixed martial arts organization based in Las Vegas, Nevada and is the largest MMA promotion in the world and features the top-ranked fighters of the sport. Based in the United States, the UFC produces events worldwide that showcase twelve weight divisions and abide by the Unified Rules of Mixed Martial Arts. This is a highly unpredictable sport

Few things we will try to visualize:

* How's Age/Height related to the outcome?
* Most popular locations in UFC?
* Most popular way to win the fight?
* Comparing techniques used by fighters

Chapter 2- System Design and Analysis

2.1 Software and Hardware Requirement Specification

A Software Requirements Specification (SRS) - a requirements specification for a software system - is a complete description of the behavior of a system to be developed. Use cases are also known as functional requirements. In addition to use cases, the SRS also contains non-functional (or supplementary) requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

2.1.1 Software Requirements

·   Languages used: Python

·   Python versions: 3.6.X, 3.7.X.

·   Microsoft Excel

· Jupyter Notebook

2.1.2 Hardware Requirements

·       Processors: Intel Core i3 Processor

·       Disk space: 1GB

·       Operating Systems: Windows 7, Mac OS , Linux

Technologies Used

PYTHON

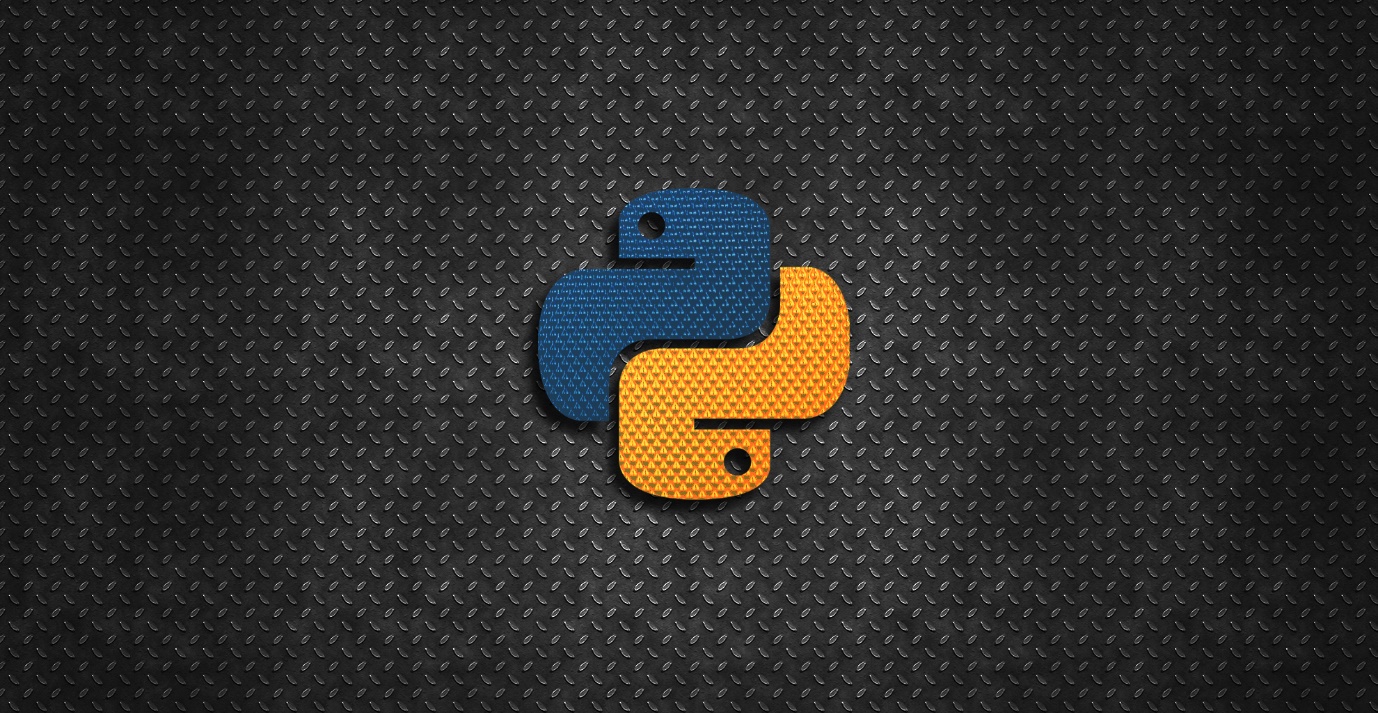


Fig 2.1. Python Logo

History of Python

Python is a widely used general-purpose, high-level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

**Let’s dig deeper –**  
In the late 1980s, history was about to be written. It was that time when working on Python started. Soon after that, Guido Van Rossum began doing its application based work in December of 1989 by at Centrum Wiskunde & Informatica (CWI) which is situated in Netherland. It was started firstly as a hobby project because he was looking for an interesting project to keep him occupied during Christmas. The programming language which Python is said to have succeeded is ABC Programming Language, which had the interfacing with the Amoeba Operating System and had the feature of exception handling. He had already helped to create ABC earlier in his career and he had seen some issues with ABC but liked most of the features. After that what he did as really very clever. He had taken the syntax of ABC, and some of its good features. It came with a lot of complaints too, so he fixed those issues completely and had created a good scripting language which had removed all the flaws. The inspiration for the name came from BBC’s TV Show – ‘Monty Python’s Flying Circus’, as he was a big fan of the TV show and also he wanted a short, unique and slightly mysterious name for his invention and hence he named it Python! He was the “Benevolent dictator for life” (BDFL) until he stepped down from the position as the leader on 12th July 2018. For quite some time he used to work for Google, but currently, he is working at Dropbox.  
The language was finally released in 1991. When it was released, it used a lot fewer codes to express the concepts, when we compare it with Java, C++ & C. Its design philosophy was quite good too. Its main objective is to provide code readability and advanced developer productivity. When it was released it had more than enough capability to provide classes with inheritance, several core data types exception handling and functions.

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming[46] and metaobjects (magic methods)).[47] Many other paradigms are supported via extensions, including design by contract[48][49] and logic programming.[50]

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map, and reduce functions; list comprehensions, dictionaries, sets, and generator expressions.[51] The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.[52]

The language's core philosophy is summarized in the document *The Zen of Python* (*PEP 20*), which includes aphorisms such as:[53]

* Beautiful is better than ugly.
* Explicit is better than implicit.
* Simple is better than complex.
* Complex is better than complicated.
* Readability counts.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.[32]

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one—and preferably only one—obvious way to do it" design philosophy.[53] Alex Martelli, a Fellow at the Python Software Foundation and Python book author, writes that "To describe something as 'clever' is *not* considered a compliment in the Python culture."[54]

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the CPython reference implementation that would offer marginal increases in speed at the cost of clarity.[55] When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

An important goal of Python's developers is keeping it fun to use. This is reflected in the language's name—a tribute to the British comedy group Monty Python[56]—and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (from a famous Monty Python sketch) instead of the standard foo and bar.[57][58]

A common neologism in the Python community is *pythonic*, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called *unpythonic*.

### Statements and control flow

Python's statements include (among others):

* The assignment statement (token '=', the equals sign). This operates differently than in traditional imperative programming languages, and this fundamental mechanism (including the nature of Python's version of *variables*) illuminates many other features of the language. Assignment in C, e.g., x = 2, translates to "typed variable name x receives a copy of numeric value 2". The (right-hand) value is copied into an allocated storage location for which the (left-hand) variable name is the symbolic address. The memory allocated to the variable is large enough (potentially quite large) for the declared type. In the simplest case of Python assignment, using the same example, x = 2, translates to "(generic) name x receives a reference to a separate, dynamically allocated object of numeric (int) type of value 2." This is termed *binding* the name to the object. Since the name's storage location doesn't *contain* the indicated value, it is improper to call it a *variable*. Names may be subsequently rebound at any time to objects of greatly varying types, including strings, procedures, complex objects with data and methods, etc. Successive assignments of a common value to multiple names, e.g., x = 2; y = 2; z = 2 result in allocating storage to (at most) three names and one numeric object, to which all three names are bound. Since a name is a generic reference holder it is unreasonable to associate a fixed data type with it. However at a given time a name will be bound to *some* object, which **will** have a type; thus there is dynamic typing.
* The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).
* The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
* The while statement, which executes a block of code as long as its condition is true.
* The try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.
* The raise statement, used to raise a specified exception or re-raise a caught exception.
* The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.
* The def statement, which defines a function or method.
* The with statement, from Python 2.5 released on September 2006,[63] which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing Resource Acquisition Is Initialization (RAII)-like behavior and replaces a common try/finally idiom.[64]
* The pass statement, which serves as a NOP. It is syntactically needed to create an empty code block.
* The assert statement, used during debugging to check for conditions that ought to apply.
* The yield statement, which returns a value from a generator function. From Python 2.5, yield is also an operator. This form is used to implement coroutines.
* The import statement, which is used to import modules whose functions or variables can be used in the current program. There are three ways of using import: import <module name> [as <alias>] or from <module name> import \* or from <module name> import <definition 1> [as <alias 1>], <definition 2> [as <alias 2>], ....
* The print statement was changed to the print() function in Python 3.[65]

Python does not support tail call optimization or first-class continuations, and, according to Guido van Rossum, it never will.[66][67] However, better support for coroutine-like functionality is provided in 2.5, by extending Python's generators.[68] Before 2.5, generators were lazy iterators; information was passed unidirectionally out of the generator. From Python 2.5, it is possible to pass information back into a generator function, and from Python 3.3, the information can be passed through multiple stack levels.[69]

### Expressions

Some Python expressions are similar to languages such as C and Java, while some are not:

* Addition, subtraction, and multiplication are the same, but the behavior of division differs. There are two types of divisions in Python. They are floor division (or integer division) // and floating point/division.[70] Python also added the \*\* operator for exponentiation.
* From Python 3.5, the new @ infix operator was introduced. It is intended to be used by libraries such as NumPy for matrix multiplication.[71][72]
* From Python 3.8, the syntax :=, called as 'walrus operator' was introduced. It assigns values to variables as part of a larger expression.[73]
* In Python, == compares by value, versus Java, which compares numerics by value[74] and objects by reference.[75] (Value comparisons in Java on objects can be performed with the equals() method.) Python's is operator may be used to compare object identities (comparison by reference). In Python, comparisons may be chained, for example a <= b <= c.
* Python uses the words and, or, not for its boolean operators rather than the symbolic &&, ||, ! used in Java and C.
* Python has a type of expression termed a *list comprehension*. Python 2.4 extended list comprehensions into a more general expression termed a *generator expression*.[51]
* Anonymous functions are implemented using lambda expressions; however, these are limited in that the body can only be one expression.
* Conditional expressions in Python are written as x if c else y[76] (different in order of operands from the c ? x : y operator common to many other languages).
* Python makes a distinction between lists and tuples. Lists are written as [1, 2, 3], are mutable, and cannot be used as the keys of dictionaries (dictionary keys must be immutable in Python). Tuples are written as (1, 2, 3), are immutable and thus can be used as the keys of dictionaries, provided all elements of the tuple are immutable. The + operator can be used to concatenate two tuples, which does not directly modify their contents, but rather produces a new tuple containing the elements of both provided tuples. Thus, given the variable t initially equal to (1, 2, 3), executing t = t + (4, 5) first evaluates t + (4, 5), which yields (1, 2, 3, 4, 5), which is then assigned back to t, thereby effectively "modifying the contents" of t, while conforming to the immutable nature of tuple objects. Parentheses are optional for tuples in unambiguous contexts.[77]
* Python features *sequence unpacking* where multiple expressions, each evaluating to anything that can be assigned to (a variable, a writable property, etc.), are associated in the identical manner to that forming tuple literals and, as a whole, are put on the left hand side of the equal sign in an assignment statement. The statement expects an *iterable* object on the right hand side of the equal sign that produces the same number of values as the provided writable expressions when iterated through, and will iterate through it, assigning each of the produced values to the corresponding expression on the left.[78]
* Python has a "string format" operator %. This functions analogous to printf format strings in C, e.g. "spam=%s eggs=%d" % ("blah", 2) evaluates to "spam=blah eggs=2". In Python 3 and 2.6+, this was supplemented by the format() method of the str class, e.g. "spam={0} eggs={1}".format("blah", 2). Python 3.6 added "f-strings": blah = "blah"; eggs = 2; f'spam={blah} eggs={eggs}'.[79]
* Python has various kinds of string literals:
  + Strings delimited by single or double quote marks. Unlike in Unix shells, Perl and Perl-influenced languages, single quote marks and double quote marks function identically. Both kinds of string use the backslash (\) as an escape character. String interpolation became available in Python 3.6 as "formatted string literals".[79]
  + Triple-quoted strings, which begin and end with a series of three single or double quote marks. They may span multiple lines and function like here documents in shells, Perl and Ruby.
  + Raw string varieties, denoted by prefixing the string literal with an r. Escape sequences are not interpreted; hence raw strings are useful where literal backslashes are common, such as regular expressions and Windows-style paths. Compare "@-quoting" in C#.
* Python has array index and array slicing expressions on lists, denoted as a[key], a[start:stop] or a[start:stop:step]. Indexes are zero-based, and negative indexes are relative to the end. Slices take elements from the *start* index up to, but not including, the *stop* index. The third slice parameter, called *step* or *stride*, allows elements to be skipped and reversed. Slice indexes may be omitted, for example a[:] returns a copy of the entire list. Each element of a slice is a shallow copy.

In Python, a distinction between expressions and statements is rigidly enforced, in contrast to languages such as Common Lisp, Scheme, or Ruby. This leads to duplicating some functionality. For example:

* List comprehensions vs. for-loops
* Conditional expressions vs. if blocks
* The eval() vs. exec() built-in functions (in Python 2, exec is a statement); the former is for expressions, the latter is for statements.

Statements cannot be a part of an expression, so list and other comprehensions or lambda expressions, all being expressions, cannot contain statements. A particular case of this is that an assignment statement such as a = 1 cannot form part of the conditional expression of a conditional statement. This has the advantage of avoiding a classic C error of mistaking an assignment operator = for an equality operator == in conditions: if (c = 1) { ... } is syntactically valid (but probably unintended) C code but if c = 1: ... causes a syntax error in Python.

LIBRARIES USED IN OUR CODE

NumPy

|  |  |
| --- | --- |
| **NumPy** | |
|  | |
| **Original author(s)** | Travis Oliphant |
| **Developer(s)** | Community project |
| **Initial release** | As Numeric, 1995; as NumPy, 2006 |
| **Stable release** | 1.17.4 / 11 November 2019; 5 days ago[1] |
| **Repository** | * github.com/numpy/numpy |
| **Written in** | Python, C |
| **Operating system** | Cross-platform |
| **Type** | Numerical analysis |
| **License** | BSD |
| **Website** | www.numpy.org |
|  |  |

Fig. NumPy Introduction

**NumPy** (pronounced /ˈnʌmpaɪ/ (*NUM-py*) or sometimes /ˈnʌmpi/[2][3] (*NUM-pee*)) is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

PANDAS

|  |  |
| --- | --- |
|  | |
| **Original author(s)** | Wes McKinney |
| **Developer(s)** | Community |
| **Initial release** | 11 January 2008; 11 years ago |
| **Stable release** | 0.25.1[1] / 21 August 2019; 2 months ago |
| **Repository** | * github.com/pandas-dev/pandas |
| **Written in** | Python, Cython, C |
| **Operating system** | Cross-platform |
| **Type** | Technical computing |
| **License** | New BSD License |
| **Website** | pandas.pydata.org |

Fig. pandas Introduction

In computer programming, **pandas** is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.[2] The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.[3]

|  |  |
| --- | --- |
| **Matplotlib** | |
|  | |
| [Screenshot of Matplotlib plots and code](https://en.wikipedia.org/wiki/File:Mpl_screenshot_figures_and_code.png)  Screenshot of Matplotlib plots and code | |
| **Original author(s)** | John D. Hunter |
| **Developer(s)** | Michael Droettboom, *et al.* |
| **Initial release** | 2003; 16 years ago[1] |
| **Stable release** | 3.1.1 / 1 July 2019; 4 months ago[2] |
| **Preview release** | 3.2.0rc2 / 5 November 2019; 11 days ago |
| **Repository** | * github.com/matplotlib/matplotlib   [Edit this at Wikidata](https://www.wikidata.org/wiki/Q2985668#P1324) |
| **Written in** | Python |
| **Engine** | * cairo * Anti-Grain Geometry   [Edit this at Wikidata](https://www.wikidata.org/wiki/Q2985668#P408) |
| **Operating system** | Cross-platform |
| **Type** | Plotting |
| **License** | Matplotlib license |
| **Website** | matplotlib.org |

Fig. Matplotlib Introduction

**Matplotlib** is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged.[3] SciPy makes use of Matplotlib.

Matplotlib was originally written by John D. Hunter, has an active development community,[4] and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012,[5] and further joined by Thomas Caswell.[6][7]

Matplotlib 2.0.x supports Python versions 2.7 through 3.6. Python3 support started with Matplotlib 1.2. Matplotlib 1.4 is the last version to support Python 2.6.[8] Matplotlib has pledged to not support Python 2 past 2020 by signing the Python 3 Statement.[9]

2.2 Flow Chart

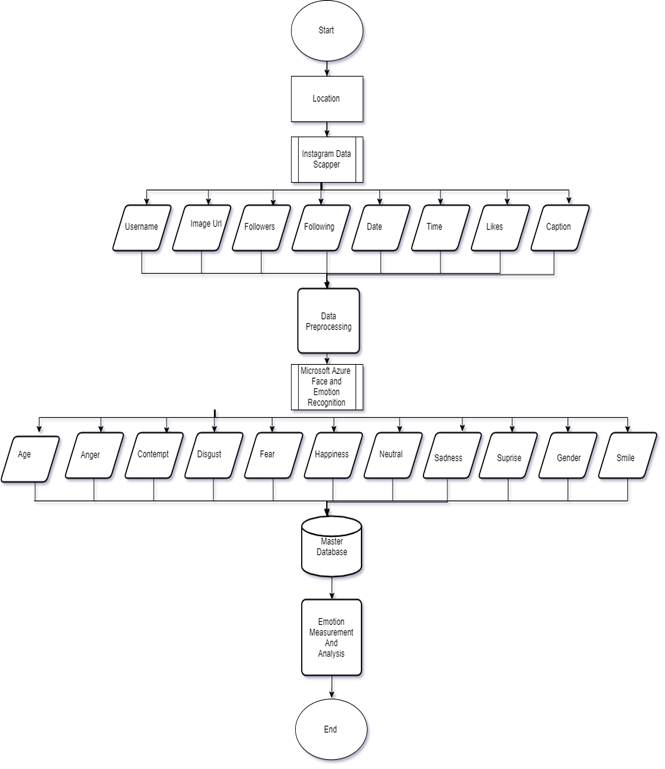


Fig 2.12. Flow chart

Chapter 3- Proposed Work

**Background**

Mixed martial arts (MMA) is a full-contact combat sport that allows striking and grappling, both standing and on the ground, using techniques from other combat sports and martial arts. The Ultimate Fighting Championship (UFC) is an American mixed martial arts organization based in Las Vegas, Nevada and is the largest MMA promotion in the world and features the top-ranked fighters of the sport. Based in the United States, the UFC produces events worldwide that showcase twelve weight divisions and abide by the Unified Rules of Mixed Martial Arts. This is a highly unpredictable sport

Few things we will try to visualize:

* How's Age/Height related to the outcome?
* Most popular locations in UFC?
* Most popular way to win the fight?
* Comparing techniques used by fighters

2. Loading libraries and retrieving data

Not all python capabilities are loaded to your working environment by default. We would need to import every library we are going to use. We will choose alias names to our modules for the sake of convenience (e.g. numpy --> np, pandas --> pd)

3. Understanding the data

Dataset contains list of all UFC fights since 2013 with summed up entries of each fighter's round by round record preceding that fight. Created in the attempt to predict a UFC fight winner . Each row represents a single fight - with each fighter's previous records summed up prior to the fight. Blank stats mean its the fighter's first fight since 2013 which is where granular data for UFC fights.

We have about 895 columns, few important columns to note:

* BPrev: Previous fights by 'Blue' fighter
* B\_Age: Age of 'Blue' fighter
* B\_Height: Height of 'Blue' fighter
* B\_Weight: Weight of 'Blue' fighter
* B\_Location: Location of 'Blue' fighter
* B\_Hometown: Hometown of 'Blue fighter
* RPrev: Previous fights by 'Red' fighter
* R\_Age: Age of 'Red' fighter
* R\_Height: Height of 'Red' fighter
* R\_Weight: Weight of 'Red' fighter
* R\_Location: Location of 'Red' fighter
* R\_Hometown: Hometown of 'Red fighter
* Date: Date of the fight
* winby: How did the fighter win the fight (decision, submission KO etc.)
* winner: Who was the winner of the fight?

Apart from this, dataset contains all the techniques (punch, kicks, takedowns etc.) attempted and landed by the fighters in each round.

4. Missing Values

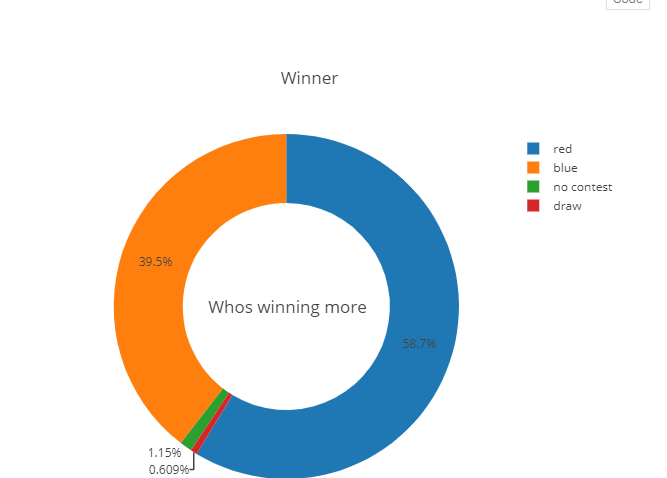
We overserve there are some missing values in our data. I know Age and Height are important features in any combat sport and they have handful of missing values.

We will address the missing values in age and height. We can simply delete rows with missing values, but usually we would want to take advantage of as many data points as possible. Replacing missing values with zeros would not be a good idea - as age 0 will have actual meanings and that would change our data.

Therefore a good replacement value would be something that doesn't affect the data too much, such as the median or mean. the "fillna" function replaces every NaN (not a number) entry with the given input (the mean of the column in our case). Let's do this for both 'Blue' and 'Red' fighters.

5. Data Visualization

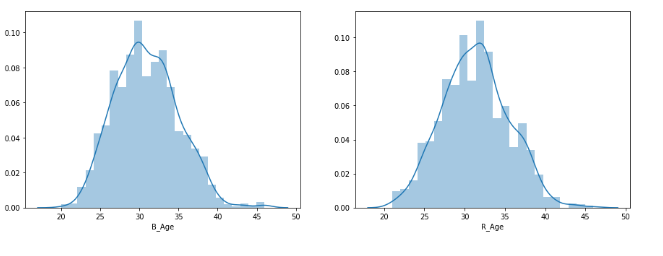
Let's start by looking who's winning more from our dataset:



**Fig. 4.1>**Red vs blue team with draw & match cancelled in pie chart

Let's talk about Age - a critical factor in any sport. We will start by looking at the distribution of Age from our dataset

Age is a big factor in any sport, moreover in MMA where you must have combination of strength, agility and speed (among other skills). These skills peak at 27-35 and fighter's fighting at this age should have higher likelihood of winning the fight. Let's validate by grouping age for Blue fighters who have won the fight.



**Fig. 4.2>**Red vs blue fighter age



**Fig. 4.3** No of blue fighters in each age group

33.0 138

29.0 134

32.0 128

27.0 120

31.0 112

28.0 106

34.0 106

26.0 72

35.0 67

Name: winner, dtype: int64

Clearly, most fights have been won by fighters in their late 20’s through early 30’s as they peak during this time and then lose strength, quickness and cardiovascular capacity

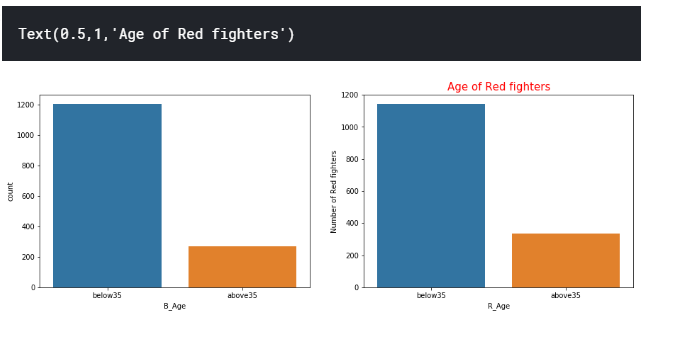
On the other hand, younger fighters do not develop peak strength till 27-28~ while older fighters are usually slower and more likely to lose. Let's check if this is true in our data. This time we will check for 'Red' fighters.



**Fig. 4.4** No of red fighters in each age group

: winner, dtype: int64

Looks like this is true. It makes me curious about the total number of Red and Blue fighters who are younger than 35.

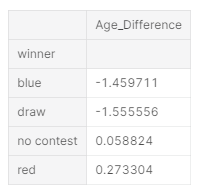


**Fig. 4.5** Red and blue fighters of age less than and greater than 35

of Red fighters')

Interestingly, most fighters are below 35. MMA is a brutal sport for older guys and can leave them with lifelong injuries.

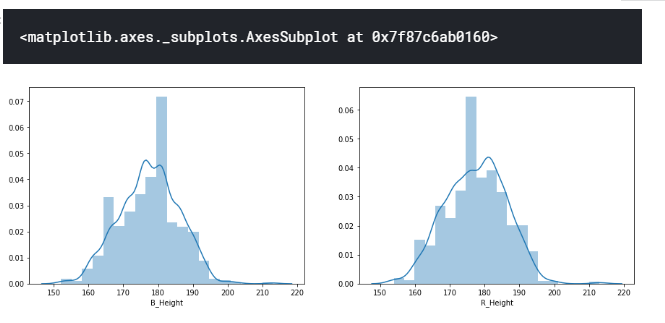
Lastly, let's look at the mean difference



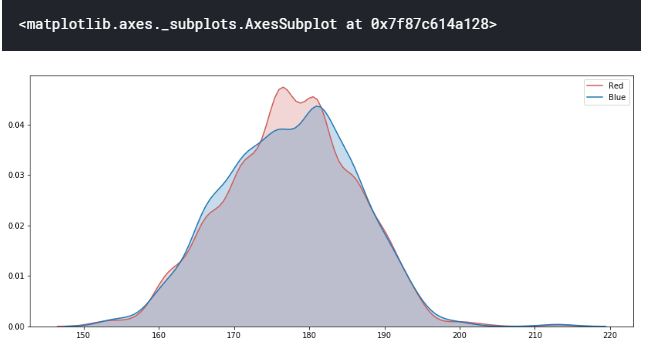
**Fig. 4.6>**Mean age difference of fighters

Age matters, and **youth is a clear advantage.**

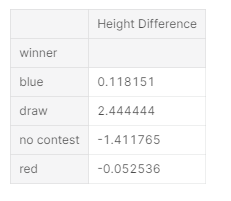
Height is also a major advantage in MMA as it means more the height more is the reach, meaning - taller fighter can attack from a distance keeping themselves safe from the hitting zone. Let's start by looking at the distribution of height:



**Fig. 4.7** Red and blue fighter height distribution



**Fig. 4.8** Red and blue fighter height difference on same plot



**Fig. 4.9** Mean difference in height by numbers>

Taller fighter has an advantage and, on average, wins. Of course, unless you are Rocky fighting Drago ;)

Now, let's talk about how the fighters are winning. The three most popular ways to win in an MMA fight are:

**1. DEC:** Decision (Dec) is a result of the fight or bout that does not end in a knockout in which the judges' scorecards are consulted to determine the winner; a majority of judges must agree on a result. A fight can either end in a win for an athlete, a draw, or a no decision.

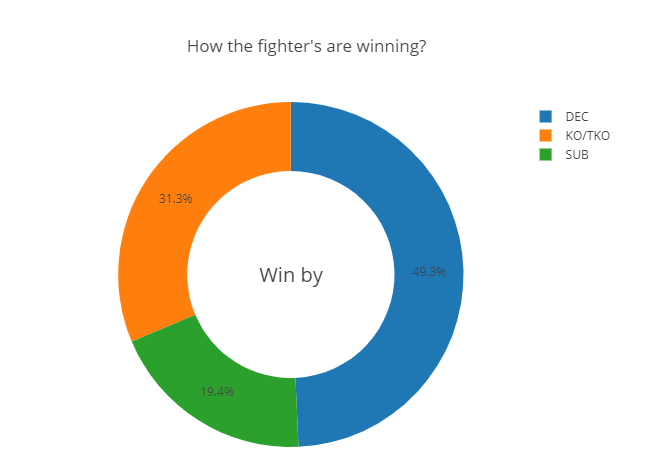
**2. SUB:**also referred to as a "tap out" or "tapping out" - is often performed by visibly tapping the floor or the opponent with the hand or in some cases with the foot, to signal the opponent and/or the referee of the submission

**3. KO/TKO:** Knockout (KO) is when a fighter gets knocked out cold. (i.e.. From a standing to not standing position from receiving a strike.). Technical Knockout (TKO) is when a fighter is getting pummelled and is unable to defend him/herself further. The referee will step in and make a judgement call to end it and prevent the fighter from receiving any more unnecessary or permanent damage, and call it a TKO.

DECKO/TKOSUBHow the fighter's are winning?Win by

So most fights are going to the judges. Second most popular way is Knockout and the Technical KO.

Let's check how this is distibuted with respect to Age for 'Red' fighters.

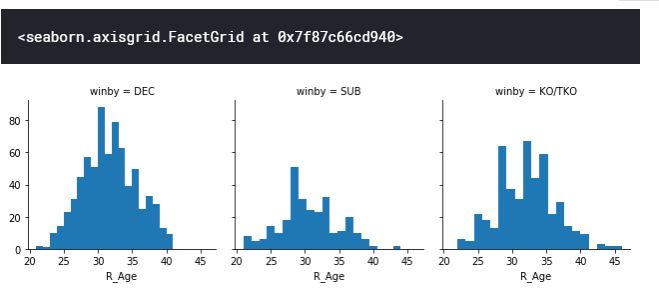


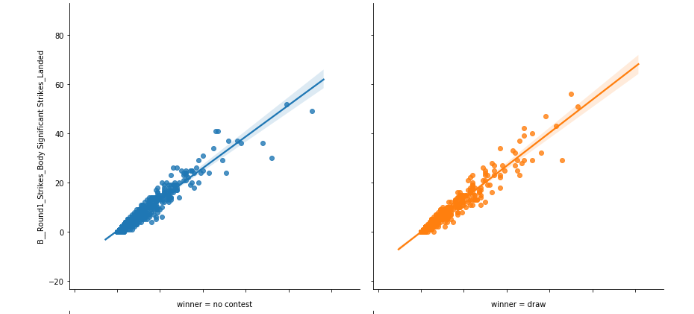
**Fig. 4.10** How the win happened

0x7f87c66cd940>

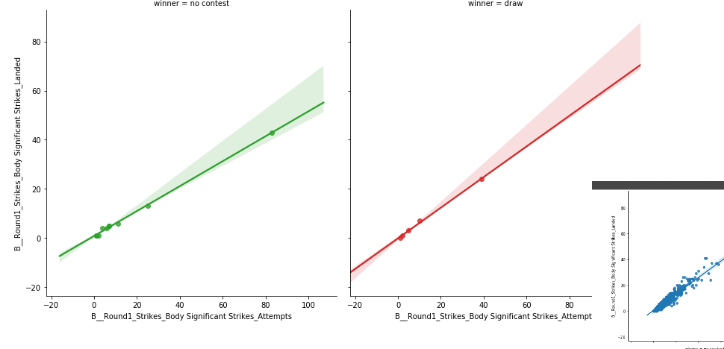
MMA is a complex sport, in a sense it is the only sport where defense and offense could be done in the same movement. Hitting someone is a risk as it leaves you open for your opponent to counter. However, the *bigger the risk, the greater the reward*. More offensive attempts you make should mean more you land on your opponent (and with right skills and power - more chance you have to win the fight).

Let's see if this is true with our data.

 **Fig. 4.11** How win happened by each age group



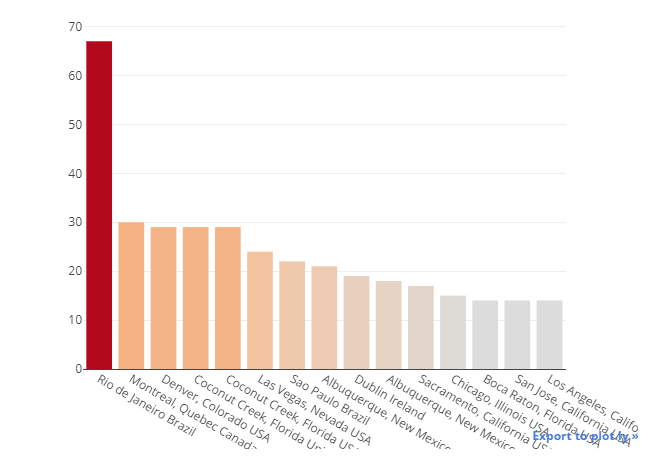
**Fig. 4.12** Strikes landed by fighter when match ended by no contest and draw

0x7f87c670dcc0>

**Fig. 4.13** Significant number of strikes by blue and red fighter

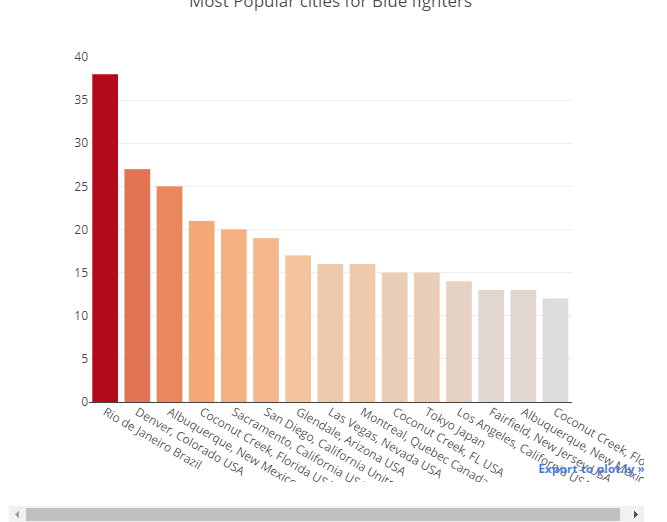
Attempts and strikes landed are, as expected, perfectly linear.

Now, let's look at the location and find out most popular countries



**Fig. 4.14** Most popular country for MMA

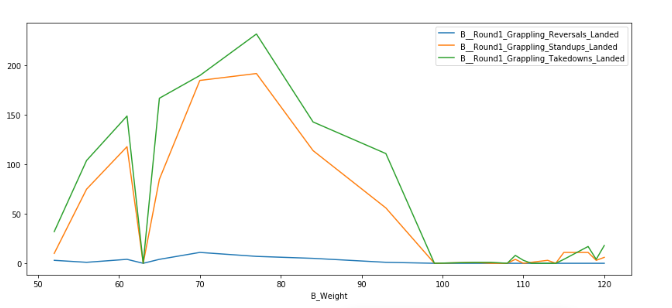
Most Popular cities for Blue fighters



**Fig. 4.15** Most popular Cities by blue fighters

MMA seems to be most prominent in Brazil and USA. Infact, MMA is second most popular sport after Soccer in Brazil. I wonder if it is due to ancient Brazilian Jiu-Jitsu?

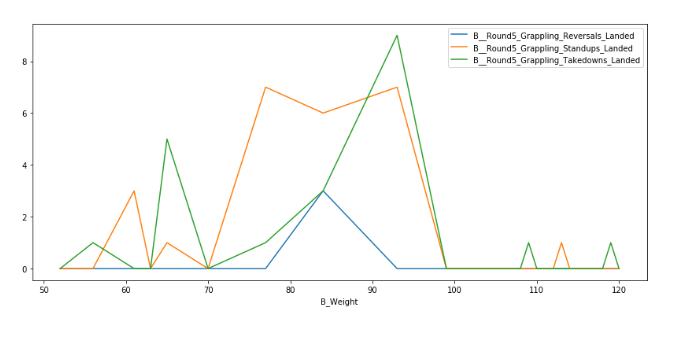
Now, let's look at the Grappling reversals, grappling stand-ups and grappling takedowns landed in different weight categories in**Round 1**



**Fig. 4.16** Grappling reversals, grappling stand-ups and grappling takedowns landed in each category in round 1

There are very few Grappling reversals but high number of Grappling takedowns that were landed. More specifically weight classes between 70 - 80 prefer takedowns during Round 1.

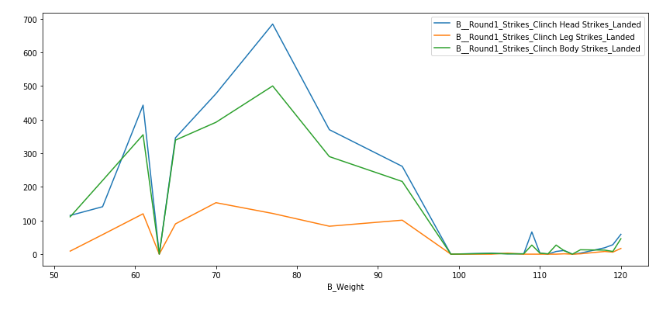
Let's compare the same for Round 5



**Fig. 4.17** Grappling reversals and takedown inweight classes between 70 - 80 in round I

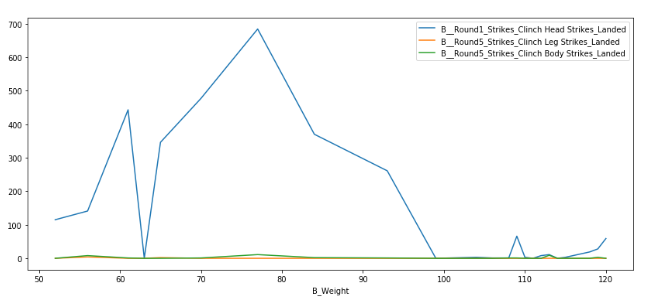
Interestingly, grappling reversals increase for fighters between weight 80-90, while takedowns have decreased in the lighter weight groups.

Let’s look similar data for Clinch head strikes, Clinch leg strikes and Body strikes for Round 1



**Fig. 4.18>** Clinch head strikes, Clinch leg strikes for 80-90 Category

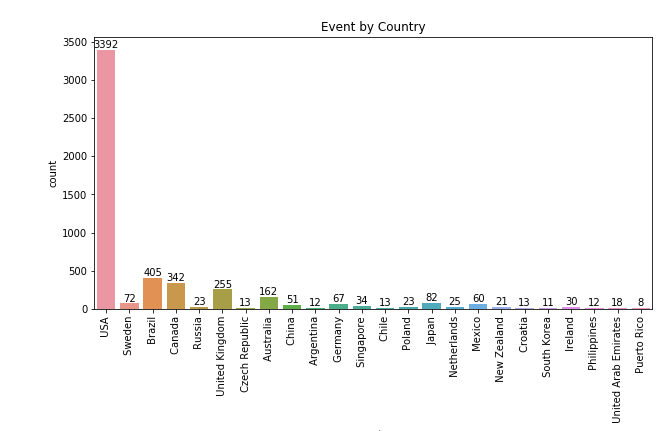
Fighters prefer to land more head strikes during round 1, let's compare this with what happens in Round 5:

ro

**Fig. 4.19** Round 1 vs Round 5 head strikes by fighters

By Round 5, fighters (who are now worn-out) are hardly landing any leg and body strike. They are still landing good amount of Head strikes. This makes sense as the fight is coming to an end and instead of depending on the judges, they want to go for a Knock out.

FC become popular since 2011 and have the most event happened on 2014



**Fig. 4.20** Number of UFC events by country **>**

Chapter 5 - Conclusion

With this study we found out that age and height matters in this sport. As well as certain fighters from certain locations are faster than other. That may be due to presence of a certain club.

Chapter 6- Future Work

In the future, several potential directions will be focused to explain related research questions. One direction is about data fusion. Although only Instagram photos are employed in the experiment, this study can be further improved with diverse data sources such as surveys, photos from Facebook, Twitter and other social networking sites. A data-synthesis-driven method might provide varied perspectives of human emotions. And the mix of text- based and facial expression-based emotion extraction methods may enhance the conﬁdence of the ﬁnal output. Another direction is to explore fundamental factors impacting human emotions. As we propose the framework for Place Emotion research, we will focus more on spatial analysis of the emotion patterns. Human emotions at diﬀerent scales will be compared to revisit the scale eﬀect in geography. And diﬀerent groups of people, as suggested by existing studies will be explored to ﬁgure out deeper insights on inﬂuential factors of human emotions. Moreover, diﬀerent place types as well as spatial units from diﬀerent scales, including points of interest, census blocks, neighbourhoods, and communities will be combined to examine the geographic patterns and socioeconomic linkages of human emotions. One speciﬁc research taking a limited number of places but with more environmental and socioeconomic factors to be examined can be conducted to enrich the understanding of place-based emotions

##### References

[1] De Silva, L. C., & Hui, S. C. (2003, December). Real-time facial feature extraction and emotion recognition. In *Fourth International Conference on Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint* (Vol. 3, pp. 1310-1314). IEEE.

[2] Kaur, M., Vashisht, R., & Neeru, N. (2010). Recognition of facial expressions with principal component analysis and singular value decomposition. *International Journal of Computer Applications*, *9*(12), 36-40.

[3] Kartikay Sharma, Siddharth Bhasin, Piyush Bharadwaj (2019, September). A Worldwide Analysis of Cyber Security and Cyber Crime using Twitter. In International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-8 Issue-6S3

[4] Waterloo, S. F., Baumgartner, S. E., Peter, J., & Valkenburg, P. M. (2018). Norms of online expressions of emotion: Comparing Facebook, Twitter, Instagram, and WhatsApp. *new media & society*, *20*(5), 1813-1831.

[5] Verma, A., Malla, D., Choudhary, A. K., & Arora, V. (2019, February). A Detailed Study of Azure Platform & Its Cognitive Services. In *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)* (pp. 129-134). IEEE.

[6]  Jung, S. G., An, J., Kwak, H., Salminen, J., & Jansen, B. J. (2018, June). Assessing the accuracy of four popular face recognition tools for inferring gender, age, and race. In *Twelfth International AAAI Conference on Web and Social Media*.

[7] You, Q., Luo, J., Jin, H., & Yang, J. (2016, February). Building a large-scale dataset for image emotion recognition: The fine print and the benchmark. In *Thirtieth AAAI conference on artificial intelligence*.

[8] Khanal, S. R., Barroso, J., Lopes, N., Sampaio, J., & Filipe, V. (2018, June). Performance analysis of Microsoft's and Google's Emotion Recognition API using pose-invariant faces. In *Proceedings of the 8th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion* (pp. 172-178).

[9] Kang, Y., Jia, Q., Gao, S., Zeng, X., Wang, Y., Angsuesser, S., ... & Fei, T. (2019). Extracting human emotions at different places based on facial expressions and spatial clustering analysis. *Transactions in GIS*, *23*(3), 450-480.

[10] Schaeffer, S. E. (2007). Graph clustering. *Computer science review*, *1*(1), 27-64.

[11] Pardàs, M., & Bonafonte, A. (2002). Facial animation parameters extraction and expression recognition using Hidden Markov Models. *Signal Processing: Image Communication*, *17*(9), 675-688.

[12] Boy, J. D., & Uitermark, J. (2017). Reassembling the city through Instagram. *Transactions of the Institute of British Geographers*, *42*(4), 612-624.

[13] Rowley, H. A., Baluja, S., & Kanade, T. (1998). Neural network-based face detection. *IEEE Transactions on pattern analysis and machine intelligence*, *20*(1), 23-38.

[14] H. A. Rowley, S. Baluja and T. Kanade, "Rotation invariant neural network-based face detection," *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.98CB36231)*, Santa Barbara, CA, 1998, pp. 38-44.

[15] Bhardwaj, P., Gautam, S., & Pahwa, P. (2018). A novel approach to analyze the sentiments of tweets related to TripAdvisor. Journal of Information and Optimization Sciences, 39(2), 591-605.

[16] Bhardwaj, P., Gautam, S., Pahwa, P., & Singh, N. (2016). Characteristics and challenges of big data. Int. J. Recent Innov. Trends Comput. Commun, 5, 187-190.

[17] Bhardwaj, P., Gautam, S., & Pahwa, P. (2017). Opinion Mining and Sentiment Analysis of Travel Websites through Twitter. International Journal of Applied Engineering Research, 12(22), 12431-12439**.**

[18] Sharma, T., Kumar, S., Yadav, N., Sharma, K., & Bhardwaj, P. (2017, February). Air-swipe gesture recognition using OpenCV in Android devices. In 2017 international conference on algorithms, methodology, models and applications in emerging technologies (ICAMMAET) (pp. 1-6). IEEE.