UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DEVELOPMENT OF A DASHBOARD FOR SPORTS BETTING TIPSTERS ANALYSIS

JUAN PABLO SERRANO ZAPATERO
JULIO 2024

TRABAJO DE FIN DE GRADO

Título:	Desarrollo de un Cuadro de Mando para el Análisis de Tipsters de Apuestas Deportivas
Título (inglés):	Development of a Dashboard for Sports Betting Tipsters Analysis
Autor:	Juan Pablo Serrano Zapatero
Tutor:	Carlos Ángel Iglesias Fernández
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	——
Vocal:	——
Secretario:	——
Suplente:	——

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO FIN DE GRADO

DEVELOPMENT OF A DASHBOARD FOR SPORTS BETTING TIPSTERS ANALYSIS

Juan Pablo Serrano Zapatero

Julio 2024

Resumen

El mundo de las apuestas deportivas ha visto un crecimiento exponencial en el número de cuentas activas, especialmente entre los apostadores online, gracias a las facilidades que ofrecen las nuevas tecnologías para apostar en cualquier momento y en casi cualquier deporte y mercado de apuestas. En este contexto, la figura de los tipsters, pronosticadores expertos con un historial comprobado de estadísticas y métricas almacenadas en plataformas online, conocidas como comunidades de tipsters, ha cobrado gran relevancia. La rentabilidad demostrada por estos tipsters ha dado lugar a un nuevo concepto en el que las apuestas deportivas pueden considerarse una inversión, en lugar de solo un entretenimiento.

Dado este enfoque de inversión, la diversificación se presenta como una práctica recomendada. Por ello, se ha desarrollado una herramienta innovadora que permite a los usuarios crear una cartera de tipsters basada en sus preferencias individuales, como el número de apuestas diarias, los deportes en los que se apuesta y el nivel de riesgo deseado. Esta herramienta no solo selecciona a los tipsters recomendados para el usuario, sino que también ofrece un análisis profundo de la cartera creada gracias a un dashboard interactivo. Este dashboard, compuesto por múltiples gráficos y métricas divididos en varias secciones, facilita una comprensión integral y visual de la rentabilidad y el desempeño de la cartera.

En resumen, el objetivo de este proyecto ha sido desarrollar una herramienta que convierte la apuesta dportiva en una actividad de inversión estructurada, permitiendo a los usuarios gestionar y optimizar sus apuestas mediante la creación y análisis detallado de una cartera de pronosticadores personalizada.

Palabras clave: apuestas deportivas, tipsters, portfolio de inversión, análisis de datos, dashboard, gráficos interactivos, personalización de usuarios, Streamlit, visualización de datos

Abstract

The world of sports betting has seen exponential growth in active accounts, especially among online bettors, thanks to the facilities offered by new technologies to bet at any time and in almost any sport and betting market. In this context, the figure of tipsters, expert forecasters with a proven track record of statistics and metrics stored on tipster communities, has gained great relevance. The profitability demonstrated by these tipsters has led to a new concept in which sports betting can be considered an investment rather than just entertainment.

Given this investment approach, diversification is presented as a recommended practice. Therefore, an innovative tool has been developed that allows users to create a portfolio of tipsters based on their individual preferences, such as the number of daily bets, the sports on which they bet, and the desired level of risk. This tool selects the recommended tipsters for the user and offers an in-depth analysis of the portfolio created through an interactive dashboard. This dashboard comprises multiple graphs and metrics divided into various sections, which facilitates a comprehensive and visual understanding of the profitability and performance of the portfolio.

In summary, the objective of this project has been to develop a tool that turns sports betting into a structured investment activity, allowing users to manage and optimize their bets through the creation and detailed analysis of a customized portfolio of tipsters.

Keywords: sports betting, tipsters, investment portfolio, data analysis, dashboard, interactive charts, user customization, Streamlit, data visualization

Agradecimientos

A mi madre, Mayte, por todo el amor y cariño que me muestra día tras día. Por su apoyo incondicional en todos los momentos que he vivido a lo largo de mis años de carrera. Por felicitarme en los buenos momentos y apoyarme en los que no lo son tanto. Por el esfuerzo, sacrificio e ilusión que siempre ha mostrado por mí y mis hermanos. Gracias, Mamá.

A mi padre, Jorge, por ser el principal impulsor de que eligiera la senda ingenieril, cuyo camino he disfrutado y en el que he recibido cientos de aprendizajes que van mucho más allá de las telecomunicaciones. Por inculcarme una manera de pensar analítica y racional de la que tan orgulloso me siento. Por ser una persona ejemplar en infinitos ámbitos y ser uno de esos seres humanos que hacen avanzar a la sociedad. Por todas sus enseñanzas, habidas y por haber. Por mostrar siempre interés en que, tanto yo como mis hermanos, seamos las mejores personas posibles y tomemos las mejores decisiones. Y, por supuesto, por el apoyo, aliento y consejos durante mis años de carrera. Gracias, Papá.

A mi hermana, Sara, y a mi abuela, Tere, por ser dos personas esenciales en mi vida y que, cada día, me dan un motivo para ser un poco más feliz y estar agradecido de lo que me ha dado la vida. Gracias, Saru, Abuelita.

A María, por ser una persona tan especial en mi vida. Por quererme y apoyarme durante los años de carrera. Por siempre haberme mostrado interés y alentarme en que siguiera trabajando para sacar el grado adelante y seguir creciendo como persona. Por sacar lo mejor de mí. Gracias, Mery.

A los profesores y a la ETSIT (Escuela Técnica Superior de Ingenieros de Telecomunicaciones) por ofrecerme todas las herramientas y enseñanzas necesarias para completar este grado, desarrollarme profesionalmente y proporcionarme una manera de afrontar problemas aplicable a una infinidad de ámbitos en la vida.

A mi tutor, Carlos Ángel Iglesias, por su inestimable ayuda para el desarrollo de este proyecto. Por haberme animado a aventurarme con este proyecto tan relacionado con mis intereses actuales, y que nunca hubiera seleccionado de no ser por él. Por su atención y disponibilidad para solucionar dudas y ofrecerme su consejo. Por atenderme en las tutorías con la mejor de las actitudes y el apoyo mostrado en todas ellas. Gracias, Carlos.

Contents

List of Figures

CHAPTER 1

Introduction

1.1 Context

In this digital era, online sports betting has grown rapidly in popularity and drawn a wide range of bettors interested in the thrill and potential profits of sporting events [18]. New technologies have created opportunities that are primarily responsible for this phenomenon. Sports betting is now more accessible and attractive than ever thanks to features such as mobile betting, instant access to a wide selection of betting markets, flexibility in accessing accounts from any device, fast bet execution, and a wide choice of bets [12].

However, the paradox is that despite the growing popularity of sports betting, more than 75% of bettors in Spain lost money in 2022 [17]. Of the remaining percentage, only a few made significant profits. Only gamblers with excellent analytical skills and emotional abstraction manage to belong to the select group of winners [13]. This is where professional forecasters, known as tipsters, come into play. Through their experience and in-depth analysis, these experts create forecasts in those markets and matches where they consider the probability of victory more significant than what the bookmaker indicates, looking for long-term reward and profitability in sports betting [20]. Throughout their careers, tipsters are characterized by a history full of data and parameters that define their past performance, which is a valuable source of information to determine their potential profitability in the future.

Delegating bets to the expertise of tipsters tends to be a good practice, as they leverage their specialized knowledge and experience. However, this practice also introduces certain risks. A significant problem facing bettors who rely on tipsters is choosing the right ones. A portion of bettors who experience losses are those who fail to select successful tipsters, which occurs mainly for two reasons.

The first reason is to be fooled by scammers who sell their services as professional tipsters. These fraudsters promise impossible profits to their followers by sharing their bets argued with empty words instead of backing them up with significant and valuable data [7]. In addition, they usually lack past statistics to demonstrate their profitability. Some even misrepresent such statistics. Tipster communities have emerged to deal with these deceptions. These are platforms where tipsters upload their forecasts. The platform itself verifies and stores the statistics of each tipper, making it a fundamental tool for bettors to analyze the tipsters [9].

Another reason that can lead to money losses is the lack of competence and skills to correctly analyze the data provided by these communities, which define the performance of each tipster. Accurate analysis of tipster's past statistics, aiming to maximize the probability of future profitability, is a complex task that requires specific knowledge and tools [21]. Although sports betting has traditionally been considered a kind of entertainment, it can now become a new investment with the correct strategy and application of modern technologies [14]. Studying and implementing effective betting strategies and tools to help bettors make informed decisions is relevant and essential in this evolving landscape.

In this context, the interest in developing a project focused on creating a tool to form a portfolio of high-quality tipsters and visualizing data and statistics tailored to each user to maximize profits becomes evident. The next section explores the specific goals of this project.

1.2 Project goals

This project aims to build an interactive dashboard for users to form their custom portfolios of tipsters based on their interests. This could be daily activity levels (e.g., number of bets per day), favorite bookmakers, sports, or betting markets. For more novice players who cannot analyze tipster statistics, the dashboard will be a high-value tool, as it will be for those more experienced in the world of sports betting who seek to structure a solid tipster portfolio that will allow them to diversify their investments.

1.2.1 Specific Goals

- **User-Friendly Dashboard**: The dashboard will provide an interactive and highly intuitive tool so that any user, without requiring extensive knowledge in the field, can build their portfolio of quality tipsters according to their user needs.
- **Comprehensive Tracking and Analysis**: The application will be provided with interactive capabilities to provide users with different analyses and parameters that define the characteristics of their portfolio and of each tipster in it, as well as the ability to compare the results between them. This will be achieved thanks to the inclusion of different graphs and key performance indicators that will help users analyze and understand their investments from different points of view.
- **Historical Performance and Future Projections**: The tool will also offer an economic view, considering the user's initial investment and betting strategy. A temporal analysis will be performed to visualize and analyze past performance and expected outcomes in the future in terms of monetary values. In this way, users will be provided with features that will allow them to analyze the potential risks and benefits of the investment.

1.2.2 Data Collection and Processing

A web scraping module will be developed to support the dashboard to gather data from hundreds of tipsters from a tipster community (`pyckio.com`). This module will create an up-to-date dataset that contains all the necessary data for thorough analysis. The use of web scraping ensures that the data remains current and relevant, providing users with the latest insights into tipster performance.

1.2.3 Analytical Function Library

A comprehensive library of functions will be designed to work on the dataset, enabling different approaches and analyses of the collected data. This library will facilitate the extraction of meaningful insights and support various analytical needs, ensuring that the dashboard provides robust and versatile analysis capabilities.

1.3 Structure of this document

This section provides a brief overview of the chapters included in this document. The structure is as follows:

- **Introduction (Chapter 1)**: This chapter introduces the context of the project, the growth and challenges of online sports betting, the role of tipsters, and the main objectives of the project.
- **Theoretical Framework (Chapter 2)**: This chapter covers the key concepts related to sports betting and tipsters and explains the enabling technologies used in the project.
- **Architecture & High-Level Design(Chapter 3)**: This chapter details the global architecture of the system, the internal architecture of each of the modules and the interactions between them. It also provides an overview of the development process for the creation of the application.
- **Data Processing Detailed Design (Chapter 4)**: This chapter details the process followed for the creation of a dataset with tipster information and statistics.
- **Visualization Design (Chapter 5)**: This chapter provides a detailed overview of the process followed for the development of the visualization module and its components.
- **Case Study (Chapter 6)**: This chapter shows the final results of the dashboard created, where the layout of the page, its different sections and the functionalities it offers can be seen, based on a use case that represents the process followed by a user.
- **Conclusions and Future Work (Chapter 7)**: This chapter discusses the achievement of the project objectives, the impact and relevance of the work, and potential future improvements.

CHAPTER 2
Theoretical Framework

This chapter deals with the most important concepts and terms in the environment in which the project is developed. On the one hand, the theoretical scope of sports betting and the factors that make this project an application of interest are analyzed. For this purpose, the importance and relevance of tipsters and tipster communities are detailed in-depth, in addition to several high-relevance keywords. On the other hand, the technological environment that allows the development of the system is detailed, describing enabling technologies and their usefulness in the project.

2.1 Tipster & Tipsters Communities

Although sports betting, in which players place bets based on their predictions of how sporting events will turn out, has traditionally been considered a kind of enjoyment, it has now also become a new type of investment, using the correct strategy and application of modern technologies [14]. Some bettors have moved, from considering sports betting as entertainment to bring extra excitement to their favorite leagues and games, to applying a pure investment point of view, regardless of whether or not the event on which they have placed their bets generates interest for them.

Tipsters have become more prevalent in this setting of growing sports betting possibilities and a money-making perspective. These forecasting experts employ several techniques and methodologies, including statistical research, trend studies, player injury analysis, knowledge of game tactics, and other relevant elements, to create their forecasts [20]. They typically focus on particular sports or markets, which allows them to examine those events more thoroughly and identify value in the odds provided.

As mentioned in the introduction, it is crucial to remember that not all tipsters are equally successful, and it is the bettor's responsibility to identify which are the most dependable and lucrative. To assess a tipster, bettors typically look at their track record of accurate forecasts, hit rate, yield, Return On Investment (ROI), maximum dropdown (largest loss over a given period), and other Key Performance Indicators (KPIs) [21], the definitions of which are presented in the section below.

On the other hand, tipster communities have developed as a reaction to the difficulties and frauds that bettors face when selecting trustworthy tipsters [9]. These Internet platforms offer a forum for tipsters to exchange their projections, analyses, and data, providing bettors with an invaluable resource to assess their performance and make wise decisions.

The primary goal of these networks is to offer openness and verification in a world where credibility can be difficult to find. Bettors can access many tipsters with a unique prediction history, performance metrics, and user reviews. Using these data, they can evaluate and compare many tipsters before selecting one to follow.

One of these systems' most significant aspects is the verification of statistics. The tipsters' communities must gather, preserve, and present comprehensive data for each tipster. Using data-driven research, bettors can objectively assess a tipster's consistency and profitability over time.

Tipster communities frequently include tools and functionalities that help in tipster selection. Among these resources are filters for finding tipsters based on various criteria, such as market, sport, kind of bet, and odds range. Additionally, they can offer classes or rankings based on tipsters' past success. Examples of such platforms include Pyckio, Blogabet, or Tipsterland.

Within the project's scope, tipster forums are an essential tool for its development, as they are the main source of information to feed the database. The project also assumes that the final application inherits the reliability and verification of the processed data provided by the tipster communities. Therefore, they are an ideal option for performing data scraping.

2.2 Key Concepts of Sports Betting

A series of terms are defined in this section, whose correct understanding is essential to interpreting the context of the project, comprehending its purpose, and analyzing the results.

Bankroll:

The amount of money a bettor has and wants to invest in sports betting is called the bankroll. Proper financial account management is a crucial strategy to reduce the chances of big losses.

Betting Market:

The several types of wagers offered for a certain sporting event are referred to as betting markets. The money line, under/over bets, and handicap bets are the three most popular betting markets. Different bookmakers may or may not offer different betting markets for the same match. Therefore, it is common for a tipster to make his predictions by focusing only on using a single bookmaker to make it easier for his clients to follow his predictions without worrying about having an account on multiple existing bookmakers.

Bookmaker:

A bookmaker, also known as 'bookie', is an entity that enables its clients to place bets in sporting events. It sets the odds for the different outcomes and betting markets. Bet365, William Hill, and Sportium are good examples of well-known bookmakers in Spain.

Liquidity of a Market:

The amount of money available for a bettor to wager on the corresponding market at the corresponding odd is known as liquidity. High liquidity causes a high-profit maximum limit allowed by the bookmaker, and the variability of the odds over time is minimal. On the other hand, markets with lower liquidity are usually characterized by a low limit maximum profit and very unstable odds. Therefore, for the same profitability, it is always more interesting to follow a tipster who operates in liquid markets. Over/under goals in a Champions League final is a good example of a high liquidity market, while the number of corners in an Indian third-division football game would be a low liquidity one.

Odds:

The odds serve the purpose of a probability measure, indicating how big is the chance for an expected outcome. It calculates the amount the player will potentially win, depending on the amount wagered. Small odds indicate that something is more likely to happen and do not offer high profit, whereas big odds signal a low-probability event and, therefore, promise higher profit. For instance, placing a €10 bet over @1.7 odd results in gross earnings of €17 when you win, which is the original wager plus a profit of €7.

Pick:

A pick, or forecast, is a prediction about the result of a sports event made by a tipster. It comprises specific data, including the betting market, odds, and the stake to be placed.

ROI:

It is a typical metric for investment analysis. It can be computed by dividing net gains by the initial investment. It is a valuable tool for assessing the profitability and long-term efficacy of betting methods.

Stake:

A bettor's stake defines the money amount that should be placed on a certain pick. A pick made by a tipster is usually expressed as a numerical normalized value, commonly ranging from 1 to 10. It serves as an indicator of how much valuable a bet is in the eyes of the tipster.

Yield:

A performance metric called yield is used to assess a tipster's profitability. It is computed by dividing the entire amount of gross earnings by the total amount of money wagered. Whereas a negative yield denotes losses, a positive yield shows profits.

2.3 Enabling Technologies

To make possible the development and correct functioning of the final application, a series of programming languages, libraries, and other tools have been used. The functionality of which and the main reason for their use are presented in the following. They are classified according to their role in the project.

2.3.1 Programming Languages

2.3.1.1 Python

Python [22] is a programming language recognized for its simplicity and strength. It is an interpreted language, which implies that the code is processed directly, line by line, without being compiled. Python is also object-oriented; specifically, it contains a clear and readable syntax, making it understandable even for beginners who have never coded. A popular feature of Python is its vast set of standard libraries that include various modules and packages to satisfy any demand. It encourages code modularity and the completion of functions, reducing development time, and increasing productivity. Python is also known for its rapid app development pace, which is known as rapid application development. It is possible to successfully and rapidly create change and enhance code programs, especially because Python writing, testing, and debugging are straightforward. Finally, Python is utilized in various areas, such as data science, machine training, web development, program writing, editing, and standard automation. Due to all these reasons, completing this project is an optimal choice.

Python is the programming language mostly used in the project's development, both in the data munging and visualization modules. It has also been used to create various functions, such as designing graphs and processing data.

2.3.1.2 JavaScript

JavaScript [5] is one of the top programming languages for interactive websites, as, together with HTML and CSS, it is used in more than 90% of websites today. It provides users with a wide range of functionality, from simple interactions to database-driven applications, and it is well-known for its versatility and efficiency in handling complex tasks.

JavaScript has been useful for operating Crawlee, a web scraping and browser automation JS library to create a Comma Separated Value (CSV) file with all the tipsters' statistics and valuable data.

2.3.2 Web Scraping

2.3.2.1 Crawlee

Crawlee [1] is a comprehensive and sophisticated tool for browser automation and web scraping. It was designed to make it easy for developers to build sturdy, active crawlers. Crawlee demonstrates its adaptability and power through its cohesive interface, which can be used in HTTP-based scraping and a fully-featured browser. Depending on the project's distinct requirements, the developer can instantly use simple HTTP requests or work with a full-fledged browser, benefitting from JavaScript rendering and screen scraping in some difficult situations. This flexibility addresses various web scraping scenarios, from retrieving information from fixed websites to interacting with vibrant, device-dependent web applications. Another powerful point of Crawlee is its constant queue management for URLs, which allows breadth and depth integration. This element is crucial to ensure that the crawl can be resumed reliably during a system power failure or reboot. Crawlee is featured in tabular and historical data-applicable storage to promote easy connection to local storage systems. Finally, Crawlee is featured by its automatic scalability feature, which guarantees optimal use of the CPU and memory through automatic adjustments of the crawler at runtime. Crawlee also improves the crawl ability stage using hooks to enable developers to extend the crawl as per their project requirements. The command-line interface enables rapid project initiation and, as such, automatic scraping templates eliminate manual configuration hurdles that are useful for time-constraint developers. In addition, error management and support related to retry operations are crucial to a reliable web scraper. Implementing TypeScript and the strong typing and generics in Crawlee enhance production expertise by reducing frequent mistakes and maintainable coding. JavasScript and how Crawlee presents structures simplify developers' integration and continuity handling of the scraper.

Crawlee has been an essential tool for developing the dataset on which the final application runs. Thanks to this library, all the necessary statistics, and data from hundreds of tipsters on Pyckio.com, a huge tipster community, have been extracted from the website and stored.

2.3.2.2 Playwright

Playwright [3] is a robust tool for automating browser tests, specially designed for any browser, platform, and language. It is built for Chromium, WebKit, and Firefox; for Windows, Linux, and macOS, it can be used locally and in continuous integration environments, both headless and headed. Playwright Application Program Interfaces (APIs) are available in TypeScript, JavaScript, Python, .NET, and Java. This library is used to write tests that are difficult to stabilize automatically using a 'point-in-time' wait automatically. Ensure interactivity or readiness of the elements before proceeding. Playwright provides numerous ways to address the failure, such as screenshots, videos, execution traces, etc. The playwright architecture performs tests out-of-process from the browser and handles multiple tabs, sources, and users. Every test runs with an isolated browser context comparable to a new browser profile, sharing the authentication state between tests. Additionally, it offers tools for solving failures, such as the Playwright inspector and the trace viewer.

Playwright has been a highly useful tool in this project since, together with Crawlee, it has allowed to interact correctly with pyckio.com, as this website's content is loaded dynamically with JavaScript, which often makes it difficult to track and extract data successfully.

2.3.2.3 DevTools Chrome

Chrome DevTools [8] is a set of web development tools directly integrated into the Google Chrome browser. It offers developers a variety of ways to inspect, debug, and optimize their web pages and applications. One of the most valuable features from the perspective of web scraping is its ability to inspect and edit the Document Object Model (DOM) in real-time. This enables developers to navigate and understand the structure of pages, locate specific elements, and test CSS selectors directly in the browser. DevTools automatically shows the HTML source code of an element when inspected and allows the editing of attributes and styles. This has proven very useful and discrete, offering an interactive way to refine CSS selectors for scraping scripts.

Chrome DevTools provides sophisticated resources to debug JavaScript, which is essential for scraping more interactive and dynamic content. For this reason, this tool has made it possible to correctly analyze the DOM of the tipster community to select the data, through CSS selectors, and extract them with Crawlee.

2.3.3 Front-End Development

2.3.3.1 Streamlit

Streamlit [11] is a Python library that aims to help developers quickly and effectively build and share web applications focusing on data science or machine learning. Unlike other tools involved in the development of websites, Streamlit is aimed at people who are strong in machine learning and data science. It does not require the end user to grasp front-end technology like HTML, JavaScript, or CSS. This way, data scientists can concentrate on their models and analysis work without worrying about making web interfaces from scratch. With just a few lines, Streamlit provides a neat and intuitive API that allows you to create good-looking, efficient user interfaces. It is perfect for small, targeted data applications. It is compatible with most Python libraries for data analytics, such as pandas, matplotlib, seaborn, plotly, Keras, and PyTorch. This lets people visualize their data in an attractive, uniform style. Streamlit gives the user several pre-designed components, such as charts and widgets. These can be easily adapted to the specific needs of each project, making it easy to add functions to web applications without writing a whole load of new code.

In the project environment, Streamlit has been the main tool in the visualization module, thanks to its facility to design the page layout, insert widgets, and draw graphics.

2.3.3.2 Plotly

Plotly [4] is an open-source library available for several programming languages, including Python, whose main functionality is designing and constructing various graphs useful for various environments, such as scientific or financial analysis. It is characterized by creating highly customizable graphs that can interact with the user once built.

In this project, Plotly has provided the dashboard with different graphs, such as line graphs, bar graphs, or pie graphs, allowing bettors to analyze different aspects of their portfolios and tipsters.

2.3.4 Data Processing and Analysis

Pandas [15] has been the main technology used for data processing. It is a Python-written library used for data manipulation and analysis, among other functionalities. It is widely used in various fields, including finance, neuroscience, economics, and web analytics. The authors of Pandas explicitly state that it is built on top of a powerful data structure, the DataFrame. This data structure is crucial for efficient data manipulation; it is a multidimensional array with an integrated index system that allows quick and fast data retrieval. Pandas can easily read, write, and convert data from various formats, including CSV files, excel, and SQL databases. It also has tools for alignment, missing data, and slicing, hence indexing. Additionally, it reads large amounts of data and can perform transformation or aggregation activities since it can also group data. Pandas perform great when merging data and combining several datasets and can handle multidimensional data. In conclusion, Pandas is the go-to library for data analysis and manipulation where speed is warranted.

Pandas has been of great value when dealing with the dataset generated after the data extraction process, allowing to analyze the data, generate new data, and design different DataFrames.

CHAPTER 3

Architecture and High-Level Design

In this chapter, the architecture and different layers of the application it is discussed, whose correct understanding is essential to comprehend how the tool has been built and how it works. First, a general view of the application architecture is presented to understand the complete data flow through the different modules, from its extraction from the Web browser to its presentation on the screen for each end user. The following sections describe each component, including the files that make up it and the connection and utility between them. Several architecture and data flow diagrams accompany all this to visualize and internalize the processes intuitively.

3.1 Introduction

The development of this project has been based on achieving a robust and data-rich dataset and an intuitive and interactive dashboard for visualization. To ensure a structured and orderly development of the application, a high-level outline of the design and architecture of the application was first made. In this way, it was possible to obtain a clear conception of each module's process flow and behavior to manage the potential complexity of the data collection and final visualization processes.

Figure 3.1 shows a flow chart that chronologically illustrates the different stages of project development. Provides a clear and concise overview of the entire process.

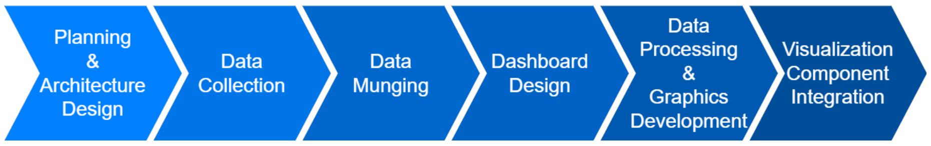

Planning & Architecture Design
Data Collection
Data Munging
Dashboard Design
Data Processing & Graphics Development
Visualization Component Integration

Figure 3.1: Development Process Chronology

As a development environment, using Visual Studio Code has been essential since it is an excellent text editor compatible with the languages used and has different support features for the tools and technologies mentioned previously. Therefore, it has been used to develop the three modules, each developed in an independent environment, manually importing the output from the previous module to the new one to be developed.

The component with the most intricate task involved developing the Visualization and User-Interaction module. Its mission is to integrate the graphics design, the DataFrames treatment, the application's visual design, and the user-interaction features to deploy an application that results in a friendly user experience.

The following sections of this chapter offer a detailed view of the system architecture, commenting on the internal structure of each of the modules that compose it, its technologies, scripts, files and key parts for its correct operation.

3.2 System Architecture Overview

This section introduces the system architecture. Since the data and graphs displayed in the final dashboard result from the treatment and processing of the extracted data, it is crucial to take a rich and quality data source. For this reason, the first step in the system's design is to select a tipster community of guarantees, such as Pyckio.com. This web platform is an ideal tool to connect the system, as it has a database of hundreds of tipsters, on which it is possible to perform a deep analysis of their results, with both temporary and overall statistics.

The architecture of the application is divided into three main modules. The first one is the Data Collection Module, which, from an HTTP connection with Pyckio.com, performs a web scraping process on all the URLs of the personal pages of each tipster. Using the Crawlee and Playwright libraries, the data in each web page's DOM (Document Object Model) is extracted to create a CSV file with the raw data of all tipsters. Secondly, the

Data Munging Module makes use of the Pandas library to import this CSV to make a data cleaning and transformation, as well as to generate some new data of value from them to, finally, build a new CSV file that conforms a dataset suitable for a correct analysis and treatment of the data in the last module. Finally, the Visualization & User-Interaction Module deals with generating the dashboard for the final user. This module is the most complex since it combines user-interaction tasks through Streamlit, a second stage of data processing, with Pandas, graph generation with Plotly, and the design and visualization of the dashboard. Therefore, it takes the dataset extracted from the Data Munging Module to generate the final tool that allows bettors to explore and analyze the data of their personalized tipster portfolio to place their bets under quality criteria and make informed decisions.

The system's general architecture is shown in Fig. 3.2. Provides an intuitive overview of the structure and data flow of the system. The system processes tipster community data and provides end-users with personalized dashboards. All this process is carried out in three modules: Collection, Munging, and Visualization). Later sections provide a more detailed view of each of them.

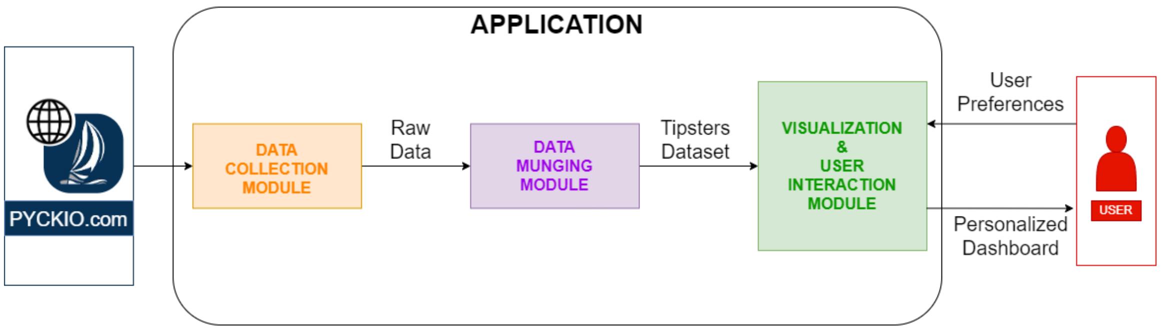


Figure 3.2: High-Level General Architecture of the Application

3.3 Data Collection Module

The Data Collection Module is in charge of extracting all the information from pyckio.com tipsters. To achieve this, it automates the process of visiting each URL of the personal page of each tipster and extracts the necessary statistics for further analysis. Crawlee and Playwright are essential tools for an effective and efficient process.

The module consists of four main components, `main.js`, `routes.js`, `config.js`, and `selectors.js`, which work together to achieve the purpose. The following details the function of each in the module structure.

- `main.js` is the file coordinating the process. It initializes a crawler with Playwright technology whose initial URL is the main page of the tipsters, which you can access through a link to all the tipsters of the community, and associates a router, defined in `routes.js`, which manages how the scraping process is performed depending on the URL visited. Finally, once the data is collected in JSON format and stored in a folder in the workspace, a CSV file containing all this data is generated.

- `routes.js` implements the data scraping logic. When the crawler visits the main page discussed above, it collects all the tipster URLs and adds them to a queue for further processing. If the visited URL is a tipster's personal page, it extracts its statistics using CSS selectors, contained in `selectors.js`, to obtain the DOM values from the HTML. A JSON file containing all the parameters is generated and saved for each tipster in the corresponding folder.

- In addition, two configuration files export their contents to the above components. On the one hand, `config.js` contains a series of configuration constants, with various values such as the number of tipsters to extract information from or the number of last months to obtain statistics. On the other hand, `selectors.js` contains the values of the CSS selectors obtained by inspecting the DOM through Chrome DevTools.

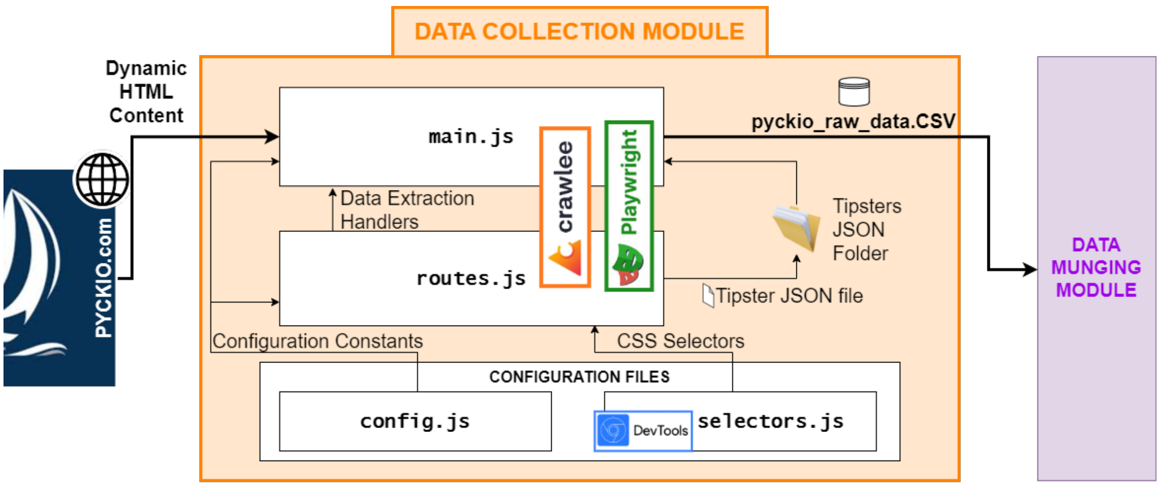


Figure 3.3: Data Collection Module Architecture

Fig. 3.3 shows the internal architecture diagram of the Data Collection Module, the data flow and the existing connections between the components can be seen. `main. js` is the only element connected to the outside, since it takes care of both the input, by connecting to pyckio.com, and the output, by generating the final CSV file that the Data Mungling Module imports. However, all parts are crucial in ensuring data extraction is effective, accurate, and adapted for the following processes.

3.4 Data Munging Module

The Data Munging Module is an essential, yet straightforward, component of the architecture. Its main purpose is to import the raw data collected from the previous module, clean and transform these data, and generate new derived metrics. This module significantly enhances the project's scalability, especially with the future aim of incorporating data from multiple tipster communities. The Data Munging Module ensures seamless integration into the subsequent Visualization and User-Interaction module by consistently normalizing and preparing these data.

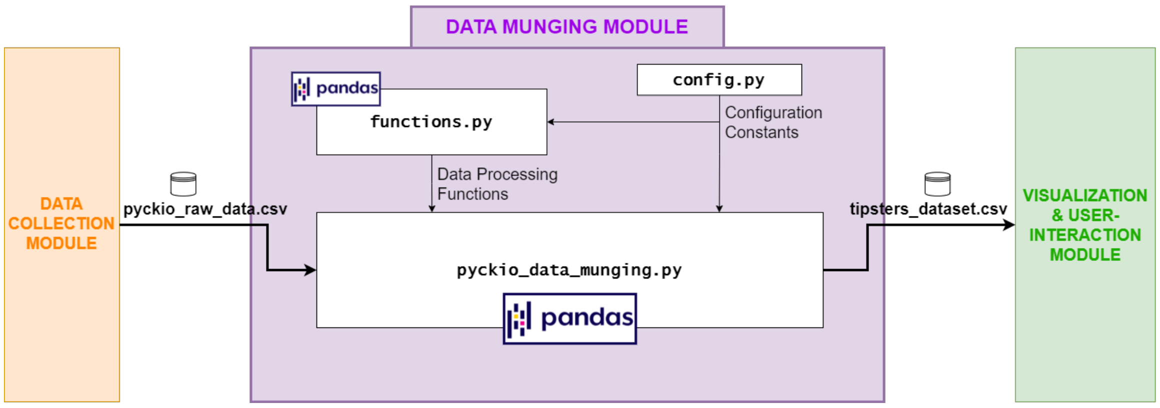


Figure 3.4: Data Munging Module Architecture

The heart of this module lies the `pyckio_data_munging.py` script. This script uses Pandas to import the `pyckio_raw_data.csv` file generated by the Data Collection Module, converting it into a DataFrame. It then performs a series of data cleaning and transformation processes, such as handling missing values, normalizing data formats, and generating new data fields derived from the existing ones. The final output is a cleaned and enriched dataset, saved as `pyckio_dataset.csv`, ready for use in the Visualization and User-Interaction module.

For readability and scalability, `pyckio_data_munging.py` leverages `functions.py` and `config.js`. The functions.py file contains a collection of functions designed to handle various data processing tasks, streamline the main script, and facilitate future expansions.

Meanwhile, `config.js` includes configuration constants that ensure the data munging process adheres to predefined parameters, making it adaptable and consistent.

Figure 3.4 illustrates the workflow of the Data Munging Module. It highlights the flow of data from the initial import of raw data through the transformation processes to the final output, showcasing how `pyckio_data_munging.py`, `functions.py`, and `config.js` interact to produce a refined dataset. It provides a clear overview of the Data Munging Module's structure and operations, emphasizing its role in preparing the data for the final Visualization and User-Interaction phase. The modular design ensures the system remains flexible and scalable, ready to handle future enhancements and additional data sources.

3.5 Visualization & User-Interaction Module

The Visualization & User-Interaction Module is the final stage of the system architecture. It is the module with more complexity and content of the application. It is responsible for presenting on screen an intuitive and interactive dashboard for the end users, the bettors. To do so, it imports the `tipsters_dataset.csv` file from the previous module, with the cleaned and processed data ready for analysis, in addition to user preferences, to build and draw a series of tables, graphs, and metrics with which the user can analyze and parameterize his tipster portfolio from different perspectives.

The structure of the module consists of the following components:

- `app.py` in the main element as it handles the inputs and outputs of the module and defines the layout and content of the final application. It makes use of Streamlit to, on the one hand, collect user preferences thanks to several widgets from the library and, on the other hand, compose, generate, and distribute different visual components such as graphs, tables, or metrics from DataFrames, so it also makes use of the Pandas library. To fulfill its mission, app.py interacts with two valuable files, `data_processing.py` and `plotting.py`.
- `data_processing.py` is a library of functions that are invoked from `app.py`. Generally, these functions take as a parameter a DataFrame and return another DataFrame, resulting from the filtering or processing of the original one, which allows for examination of the portfolio from different points of view, such as temporal analysis, overall statistics, or financial studies. One of the key functions of this component is to process the original DataFrame, imported from the data munging module, together with the user preferences, once they are defined, to build a reduced version

of the original DataFrame, which contains exclusively the data of the tipsters that make up the user's particular portfolio.

- `plotting.py` is the element whose purpose is the construction of figures that the dashboard presents on screen. For this purpose, this file defines a series of functions that are invoked from `app.py`. Normally, they receive as a parameter a DataFrame, generated thanks to `data_processing.py`, and return a figure, such as a line chart, bar chart, or pie chart.

- `config.py` file provides configuration constants for the rest of the module components. Furthermore, in`app.py`, it provides the values of the `column_config` parameter of the `st.DataFrame()` method from Streamlit defines the layout of the tables displayed on the dashboard.

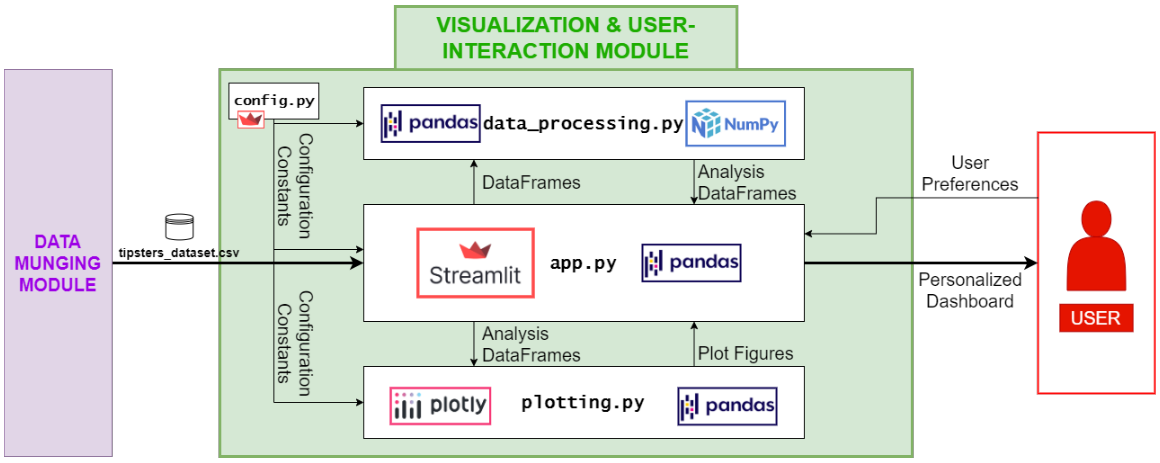


Figure 3.5: Visualization & User-Interaction Module Architecture

Figure 3.5 offers a visual representation of the architecture of the Visualization & User-Interaction Module by observing the technologies used, the data flow, and the internal interaction between the components and the outside. It shows how `app.py` makes use of `data_processing.py` and `plotting.py` to render the dashboard to which the user has access. The multi-component design allows for flexibility and scalability of the module, optimizing the integration of new features and system enhancements in the future.

CHAPTER 4

Data Processing Detailed Design

This chapter deals with the whole process up to the generation of the final tipster dataset. It details, therefore, both the data extraction procedure from the web page and the subsequent wrangling process to format the data and prepare the dataset for the visualization module. In this way, the different technologies used and the work methodology to be appreciated.

4.1 Data Collection Process

The data collection process is a critical phase for the project's development, since it involves creating a robust, accurate, and updated database on which the rest of the application is built and fed. This process involves navigating to a highly reputable tipster community, Pyckio.com, visiting each tipster's personal page, and storing all the data and statistics that have been considered necessary for a detailed analysis. This ensures that the portfolios received by the final users are composed of forecasters studied in detail.

The initial setup was based on a suitable environment configuration after a study of the enabling technologies needed to perform the data collection process successfully. Using VSCode for file generation and directory structuring, Crawlee and Playwright import scripts were created, and libraries allowed efficient and robust data extraction from a dynamically loaded JavaScript web page such as Pyckio. In addition, using the Chrome browser, which implements the DevTools tool, would allow the correct page inspection.

Next, the work done in each of the necessary substeps in the data collection process it is detailed, from the reasons for selecting pyckio.com as a tipster community, to creating the CSV file with all the data.

4.1.1 Tipster Community Selection

As a first step in the data collection process, an analysis was performed in many tipster communities that could be potentially useful for this purpose. Finally, Pyckio.com was selected as the tipster community from which the data would be extracted for several reasons. In the first place, the prestige and good references about the credibility and verification of the statistics of the tipsters that compose it would provide reliability to the project's database. Pyckio stores a large amount of data and statistics, allowing for a detailed analysis from different perspectives on hundreds of tipsters distributed internationally.

Another of the main reasons this tipster community has been selected and differentiates it from the others is that its tipsters operate in highly liquid markets, which also means they are accessible from the vast majority of bookmakers. This is a great advantage for end users, as it is useful for beginners who bet small amounts and have an active account in a single bookmaker and those who bet large amounts in different bookies. In addition, the algorithm that generates portfolios has fewer limitations that would reduce the number of valid tipsters for analysis.

Finally, in addition to the classic statistics and metrics in most tipster communities, such as yield, hit rate, or number of picks, Pyckio offers many other data about the tipster's historical performance, which can be useful in extending the application's functionalities.

4.1.2 Data Scraping

Data scraping is the most important part and the heart of this process. It is the process of extracting the necessary data from the DOM of the Pykcio.com pages. The extraction and crawl logic is defined through the two main files of the data collection module: `main.js` and `routes.js`.

To fully understand how these files work, it is important to understand three basic concepts of the Crawlee library.

- **Request:** This is a JavaScript class whose instances determine where the crawler has to move. This class's most important and basic parameter is a URL, in which a defined operation extracts the desired data. In the particular case of the project, an instance of Request is generated for each of the personal pages of each tipster since each one is contained in a different URL.

- **RequestQueue:** Since the crawling process involves navigation between different pages, the concept of RequestQueue becomes necessary. As its name indicates, it is a queue that stores all the generated requests. A RequestQueue needs at least one Request containing an initial URL, usually named start URL, which indicates the initial page on which the crawler will operate. It can also be initialized with more Requests whose URLs will be visited when the URL processing is finished. However, one of Crawlee's most powerful features is the ability to dynamically add new Requests to the queue from links on the visited page. In the project's scope, the request queue is initialized exclusively with a start URL containing links to each tipster page.

- **RequestHandler:** This is a user-defined function that indicates which processes to follow given a given Request or set of them. It can perform various actions, such as extracting data from the page, saving it, adding new requests to RequestQueue, etc. Normally, as in the case of this project, a router file is usually created where the different handlers are defined for each type of request.

Once these concepts have been introduced, it is easy to understand the overall operation of the module and each file. The file `main.js` takes care of the initial configuration of the crawler. It configures a PlaywrightCrawler, capable of dealing with the dynamic data loading characteristic of a JavaScript web page, such as Pyckio, and associates the router defined in `routes.js` as its requestHandler, whose operation is detailed in the next subsection. The initial configuration includes defining the Pyckio page where all the existing tipsters are included as startURL, the only one with which the requestQueue is initialized. The rest of the Requests are dynamically added by traversing this initial page.

Both startURL and other constants and value configurations are defined in the `config.js` file, whose existence provides the module with compartmentalization and ease when making different adjustments in the scraping process.

Listing 4.1: Exported constants from config.js

```
export const CONFIG = {
  SCROLL_TIMEOUT: 30,
  DATASET_ROW_SIZE: 700,
    MAX_MONTHS_TO_SCRAP: 12,
    START_URL: ["https://pyckio.com/i/#!rankings"]
};
```

Next, the functionality of `routes.js` as the module's requestHandler is detailed, distinguishing the process into two phases: one in which the different URLs are stored and the other in which the information of each tipster is extracted.

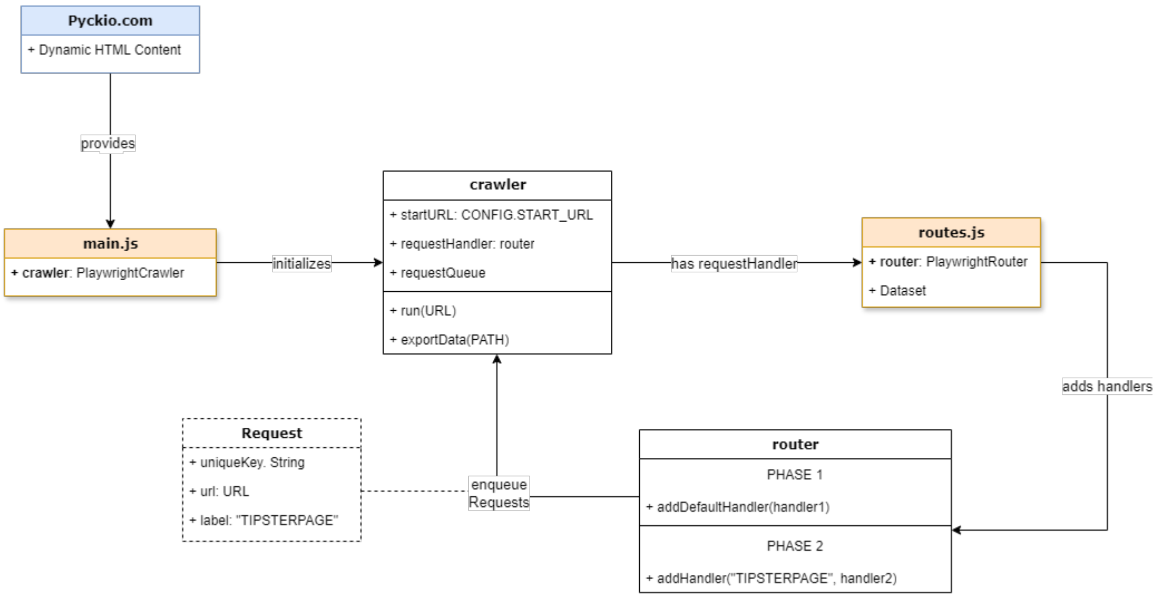


Figure 4.1: Data Scraping Workflow

4.1.2.1 Scraping Phases

As mentioned above, `routes.js` deals purely with the scraping logic depending on the URL visited, which can be the main page where all tipsters are located or the personal page of a tipster. A handler is defined for these two possible situations, so the scraping process can be divided into two phases. The first one, corresponding to the work done on the general page of tipsters, deals with navigating through the whole page extracting all the URLs that lead to the personal pages, creating their corresponding Request and adding them to the RequestQueue. The second phase operates on these personal pages, extracting the data of each tipster.

Fig. 4.1 shows a class diagram with Unified Modeling Language (UML) notation that illustrates the workflow in the module. It shows the relationship between the main files, `main.js` and `routes.js`, and the main objects they use and allow their correct operation.

All data extracted in any phase are obtained from the DOM, using Cascade Style Sheet (CSS) selectors. For this reason, and to facilitate the readability and modification of the code, it was decided to create the file `selectors.js`, where the selectors associated with each data to be extracted are defined.

Having understood the general operation of the scraping process, each of its phases is now explained in more detail.

Listing 4.2: Exported constants from selectors.js

```
export const SELECTORS = {
  TIPSTER_URL: "tbody tr td a",
  SPORT: ".panel-heading i img:nth-child(2)",
  MONTH_NPICKS: "#time-stats tbody tr:not(.global) > td:nth-child(5)",
  // other selectors...
};
```

Phase 1: Tipsters URLs Extraction

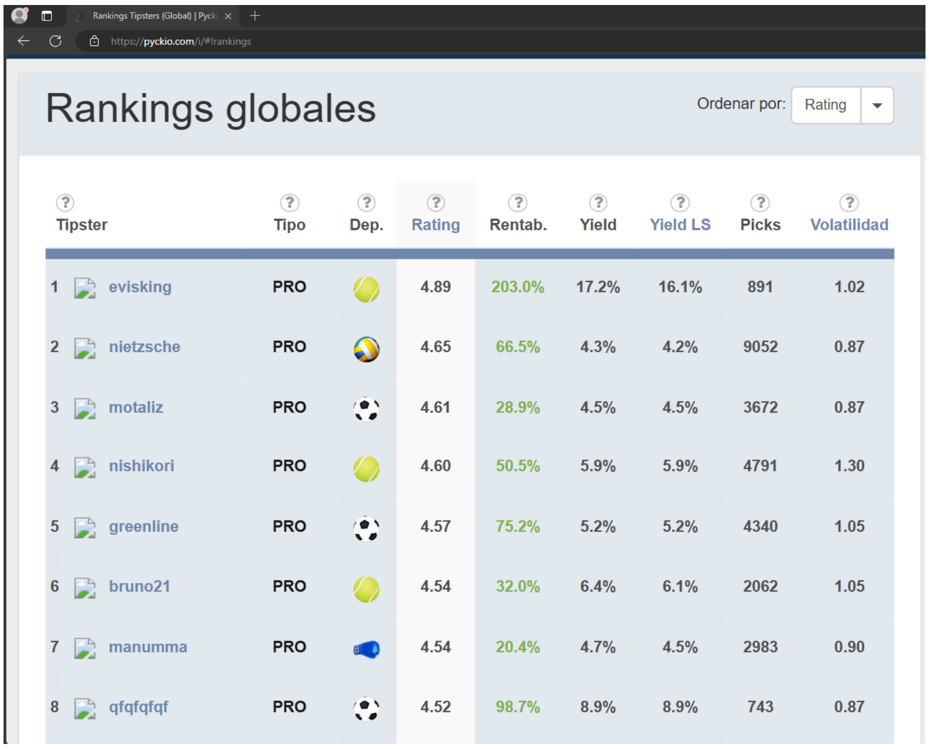
Pyckio offers a tab on its website, the ranking page, where all the platform's tipsters are presented on the same page. When this page is loaded for the first time, only the tipsters that enter the screen are loaded, while the rest are loaded as you scroll down.

For this reason, as a first step in obtaining each tipster's URL, it is necessary to use Playwright's infinite scroll method, which dynamically loads all the rows of tipsters. As its name indicates, the method simulates a scroll down the page under the configuration indicated in its parameters, such as `timeoutSecs`, which measures the time in seconds for which the scroll is being performed, or `waitforSecs`, which defines the maximum waiting time before stopping the process if no new content is loaded.

Listing 4.3: Script for Dynamic Row Loading on the Rankings Page using Infinite Scroll

```
  await playwrightUtils.infiniteScroll(page, {
    timeoutSecs: CONFIG.SCROLL_TIMEOUT,
    waitForSecs: 10,
  });
```

Once the desired scroll has been performed, it is ensured that all the rows of tipsters are in the DOM, where each tipster URL is stored, so the next step involves extracting those paths and adding them to the queue. To do this, by using the appropriate CSS selector, a loop runs through all HTML elements whose `href` attribute contains the URL


Rankings Tipsters (Global) | Pyck
https://pyckio.com/i/#!rankings

Rankings globales

Ordenar por: Rating

	Tipster	Tipo	Dep.	Rating	Rentab.	Yield	Yield LS	Picks	Volatilidad
1	evisking	PRO	🎾	4.89	203.0%	17.2%	16.1%	891	1.02
2	nietzsche	PRO	🏐	4.65	66.5%	4.3%	4.2%	9052	0.87
3	motaliz	PRO	⚽	4.61	28.9%	4.5%	4.5%	3672	0.87
4	nishikori	PRO	🎾	4.60	50.5%	5.9%	5.9%	4791	1.30
5	greenline	PRO	⚽	4.57	75.2%	5.2%	5.2%	4340	1.05
6	bruno21	PRO	🎾	4.54	32.0%	6.4%	6.1%	2062	1.05
7	manumma	PRO	🥊	4.54	20.4%	4.7%	4.5%	2983	0.90
8	qfqfqfqf	PRO	⚽	4.52	98.7%	8.9%	8.9%	743	0.87

Figure 4.2: Pyckio.com Tipster Ranking Page

of the tipster, generates a Request, and adds it to an array containing all those created. Each Request, in addition to the `url` and its `uniqueKey`, has a `label` parameter with the value 'TIPSTERPAGE', which indicates that this page has to be treated with a requestHandler specifically defined for those Requests defined with this label.

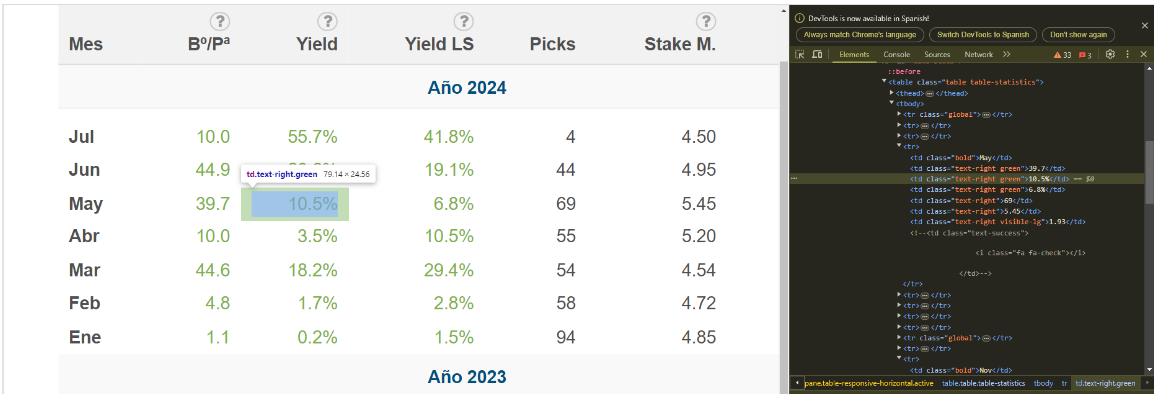
Listing 4.4: Script for Extracting and Generating Requests for Tipsters' Personal Page URLs

```
const requests = [];
for (const urli of await page.locator(SELECTORS.TIPSTER_URL).all()) {
  if (requests.length >= CONFIG.DATASET_ROW_SIZE) break;
  const urlrelativa = await urli.getAttribute("href");
  const urlabsoluta = "https://pyckio.com/i/" + urlrelativa;
  const request = new Request({
    url: urlabsoluta,
    uniqueKey: urlabsoluta,
    label: "TIPSTERPAGE",
  });
  requests.push(request);
}
if (requests.length < CONFIG.DATASET_ROW_SIZE) {
  log.warning(
        `Insufficient scroll to fill the dataset. Dataset expected size: ${
            CONFIG.DATASET_ROW_SIZE}. URLs extracted: ${requests.length}`
      );
});
};
```

During this process, the DATASET_ROW_SIZE value imported from the config.js file also comes into play. This constant indicates the desired size of the dataset, i.e., the number of tipsters to be processed. Once the loop has added this limit of requests, it stops loading new tipsters. In case the number of tipsters loaded after the scroll process is less than the DATASET_ROW_SIZE defined, a warning log pops up on the console informing of the event and indicating how many requests were expected to be created and how many have actually been created.

Phase 2: Tipster Statistics Extraction

During phase 2, scraping is performed on the particular pages of each tipster, where all the statistics and the value data are found to create the D. Inspecting the DOM using Chrome DevTools is important during this process.



Mes	Bº/Pª	Yield	Yield LS	Picks	Stake M.
		Año 2024			
Jul	10.0	55.7%	41.8%	4	4.50
Jun	44.9	[illegible]	19.1%	44	4.95
May	39.7	10.5%	6.8%	69	5.45
Abr	10.0	3.5%	10.5%	55	5.20
Mar	44.6	18.2%	29.4%	54	4.54
Feb	4.8	1.7%	2.8%	58	4.72
Ene	1.1	0.2%	1.5%	94	4.85
		Año 2023			

```
::before
<table class="table table-statistics">
  <thead>⋯</thead>
  <tbody>
    <tr class="global">⋯</tr>
    <tr>⋯</tr>
    <tr>⋯</tr>
    <tr>
      <td class="bold">May</td>
      <td class="text-right green">39.7</td>
      <td class="text-right green">10.5%</td> == $0
      <td class="text-right green">6.8%</td>
      <td class="text-right">69</td>
      <td class="text-right">5.45</td>
      <td class="text-right visible-lg">1.93</td>
      <!--<td class="text-success">

                    <i class="fa fa-check"></i>

                </td>-->
    </tr>
    <tr>⋯</tr>
    <tr>⋯</tr>
    <tr>⋯</tr>
    <tr>⋯</tr>
    <tr class="global">⋯</tr>
    <tr>⋯</tr>
    <tr>
      <td class="bold">Nov</td>
```

td.text-right.green 79.14 × 24.56
DevTools is now available in Spanish!
Always match Chrome's language
Switch DevTools to Spanish
Don't show again
Elements
Console
Sources
Network
33
3
pane.table-responsive-horizontal.active
table.table.table-statistics
tbody
tr
td.text-right.green

Figure 4.3: Using Chrome DevTools to Inspect a Tipster's Personal Page on Pyckio.com

To accomplish this task, a requestHandler is defined to handle Requests with label = 'TIPSTERPAGE'. This handler performs a series of processes to extract the information, supported by the different CSS selectors.

On the one hand, several simple data are extracted whose reference in the DOM is immediate, such as the sport where a tipster works.

Listing 4.5: Scraping Simple Data: Sport

```
const sport = await page
.locator(SELECTORS.SPORT)
.getAttribute("data-original-title");
```

On the other hand, more complex processes involve iteration over several DOM elements to extract datasets stored in arrays, such as extracting each value corresponding to the number of picks made during the last months.

Listing 4.6: Scraping datasets: Last Months' Number of Picks

```
const rawLastMonthsnPicks = [];
for (let i = 1; rawLastMonthsnPicks.length < nmonths; i++) {
const nPicksi = await page
.locator(SELECTORS.MONTH_NPICKS)
.nth(i)
.textContent();
rawLastMonthsnPicks.push(nPicksi);
}
const lastMonthsnPicks = rawLastMonthsnPicks.map(Number);
```

Finally, once all the necessary values have been saved, they are merged into a single results object, ready to be saved,

Listing 4.7: Generating the Results Object

```
const results = {
name: tipstername,
url: url,
sport: sport,
globalYield: globalYield,
// other values...
};
```

After this process, the object is transformed to the JavaScript Object Notation (JSON) format, a procedure detailed in the next section; the Request object associated with the tipster just analyzed is removed from request queue, and the next tipster in the queue is analyzed.

During the data consolidation and storage process, all information extracted during the scraping process is collected to store and structure all tipster data and statistics. To carry out this task, it is essential to use Crawlee's Dataset class, which allows both to generate

JSON files with the information of each tipster and, after processing all of them, to generate a CSV file that gathers all the data.

- **Dataset class:** The Dataset class represents a structured data store where each stored object has the same attributes. In the project's scope, it generates a Dataset where the results of each scraping on a personal tipster page are stored.
- **Dataset.open():** It is a method of the Dataset class used to open a given dataset or generate a new one, in case the one included as a parameter does not exist. It returns a promise resolved in an instance of the Dataset class.
- **Dataset.pushData():** This method stores, in JSON format, an object or an array of objects in the dataset on which the method is invoked.
- **Dataset.exportToCSV():** This method saves the complete contents of the dataset in a CSV file.
- **PlaywrightCrawler.getDataset():** This is a method of the PlaywrightCrawler class that retrieves the specified dataset.

First, in the `routes.js` file, a dataset, `tipstersdataset`, is defined using the `open()` method of the Dataset class.

Listing 4.8: Defining the Dataset in routes.js

```
const tipstersdataset = await Dataset.open("tipstersdataset");
```

After creating the result object, the handler dedicated to processing the tipsters' pages uses this object to add a JSON file to `tipstersdataset` with the `pushData()` method.

Listing 4.9: Adding a JSON File to the Dataset in routes.js

```
await tipstersdataset.pushData(results);
```

Finally, the dataset is complete when all the Requests from the `requestQueue` have been processed. In the `main.js` file, the dataset is imported with the `getDataset()` method of PlaywrightCrawler and converted to CSV with the `exportToCSV()` method of the dataset. At the end of this process, the module has fulfilled its function, and the newly created file is ready to be processed by the next module.

Listing 4.10: Converting the Dataset to CSV File in main.js

```
const tipstersdataset = await crawler.getDataset("tipstersdataset");
await tipstersdataset.exportToCSV("./storage/pyckio_raw_data.csv")
```

4.2 Data Munging Process

Developing this second system module is a simple but essential process for the project. The objective is to ensure that the data imported from the visualization module is clean and correctly processed for its subsequent use and treatment.

This goal has been achieved using the Python language and, mainly, the Pandas library. To maintain the modularity and improve the readability of the code, it has been compartmentalized into three files: `data_munging_module.py`, `functions.py`, and `config.py`, whose functionality and development are detailed later.

Before creating these files, Jupyter Notebook was a valuable tool for the development phase, providing an interactive environment in which it was easy to analyze, test, and visualize the transformations performed on the data imported from the previous module. The use of this technology for development is now specified.

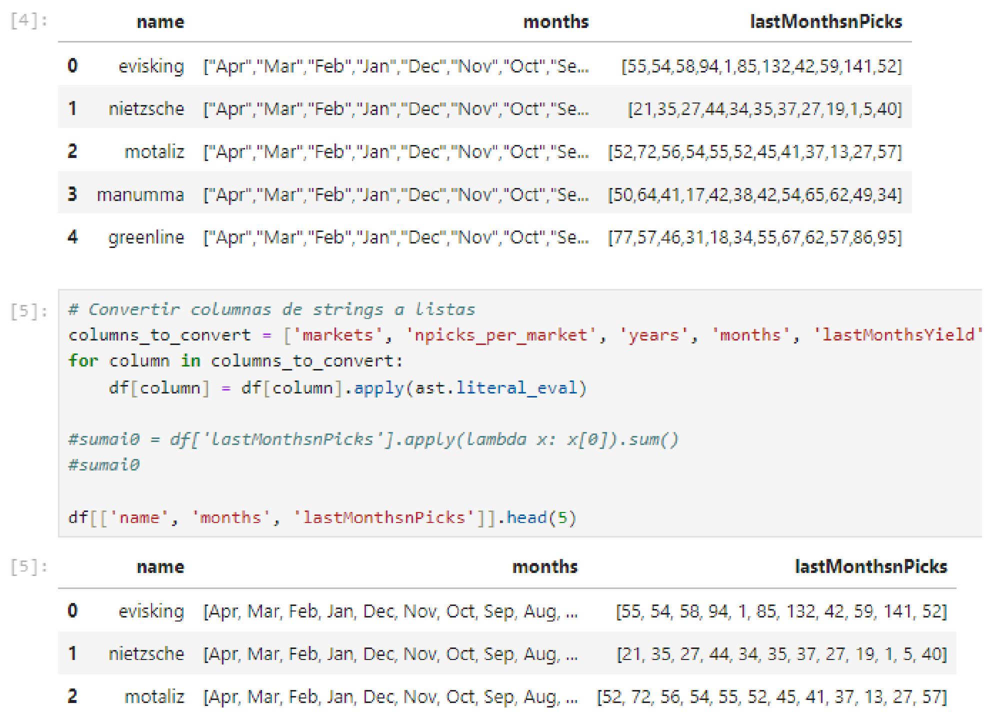
Much of the development of the Data Munging Module has been done on the environment provided by Jupyter Notebook, given its ease of debugging and the ability to run different sections of the code separately. In this process, a single `.ipynb` file was created that included configuration constants and functions, as well as the transformations in the data and their commands to test those changes.

The CSV file imported and converted into a Pandas DataFrame was not the same as the one generated in the Data Collection Module, but a mock file corresponding to a reduced version of it, with only the information of the first 20 tipsters included. Working on this mock-up would allow for faster iterations of the data, reduce waiting time, and facilitate error detection and analysis of the results.

Listing 4.11: Reading the mock CSV file.

```
df = pd.read_csv('pyckio_raw_data_reduced.csv')
```

Many tests have been carried out on the data conversions to define which ones would be necessary to adapt the final dataset for export. An example is the conversion of data that were originally arrayed and transformed into strings since CSV files cannot store this type of data in the List type to be able to work on them.



[4]:

	name	months	lastMonthsnPicks
0	evisking	["Apr","Mar","Feb","Jan","Dec","Nov","Oct","Se...	[55,54,58,94,1,85,132,42,59,141,52]
1	nietzsche	["Apr","Mar","Feb","Jan","Dec","Nov","Oct","Se...	[21,35,27,44,34,35,37,27,19,1,5,40]
2	motaliz	["Apr","Mar","Feb","Jan","Dec","Nov","Oct","Se...	[52,72,56,54,55,52,45,41,37,13,27,57]
3	manumma	["Apr","Mar","Feb","Jan","Dec","Nov","Oct","Se...	[50,64,41,17,42,38,42,54,65,62,49,34]
4	greenline	["Apr","Mar","Feb","Jan","Dec","Nov","Oct","Se...	[77,57,46,31,18,34,55,67,62,57,86,95]

[5]:

```
# Convertir columnas de strings a listas
columns_to_convert = ['markets', 'npicks_per_market', 'years', 'months', 'lastMonthsYield'
for column in columns_to_convert:
    df[column] = df[column].apply(ast.literal_eval)

#sumai0 = df['lastMonthsnPicks'].apply(lambda x: x[0]).sum()
#sumai0

df[['name', 'months', 'lastMonthsnPicks']].head(5)
```

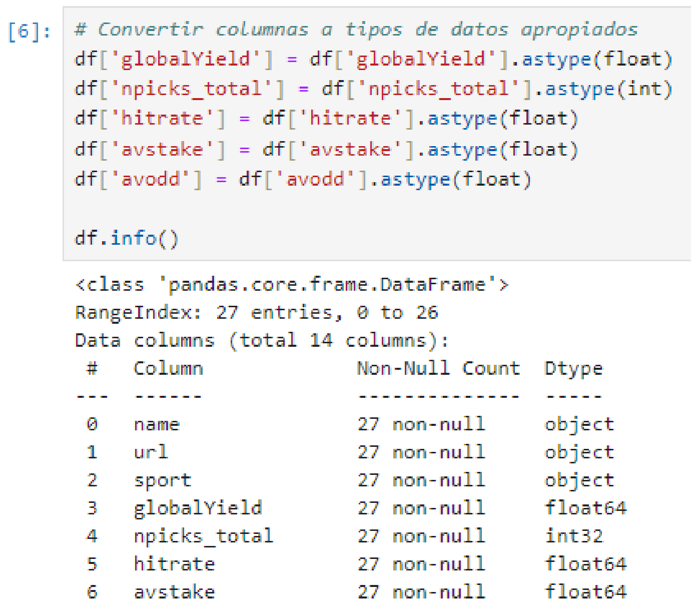
[5]:

	name	months	lastMonthsnPicks
0	evisking	[Apr, Mar, Feb, Jan, Dec, Nov, Oct, Sep, Aug, ...	[55, 54, 58, 94, 1, 85, 132, 42, 59, 141, 52]
1	nietzsche	[Apr, Mar, Feb, Jan, Dec, Nov, Oct, Sep, Aug, ...	[21, 35, 27, 44, 34, 35, 37, 27, 19, 1, 5, 40]
2	motaliz	[Apr, Mar, Feb, Jan, Dec, Nov, Oct, Sep, Aug, ...	[52, 72, 56, 54, 55, 52, 45, 41, 37, 13, 27, 57]

Figure 4.4: Conversion of string data to lists

Another example is to use the notebook to check the data type in each DataFrame column before proceeding with the transformations.

Once all desired procedures are performed successfully on the DataFrame and ready to be used in the final module of the system, the effective code was taken from the `.ipynb` file and distributed among the three files mentioned above: `pyckio_data_munging.py`, `functions.py`, and `config.py` to form the module in its final form. These files are described in Appendix C.



```
[6]: # Convertir columnas a tipos de datos apropiados
     df['globalYield'] = df['globalYield'].astype(float)
     df['npicks_total'] = df['npicks_total'].astype(int)
     df['hitrate'] = df['hitrate'].astype(float)
     df['avstake'] = df['avstake'].astype(float)
     df['avodd'] = df['avodd'].astype(float)

     df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27 entries, 0 to 26
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   name              27 non-null     object
 1   url               27 non-null     object
 2   sport             27 non-null     object
 3   globalYield       27 non-null     float64
 4   npicks_total      27 non-null     int32
 5   hitrate           27 non-null     float64
 6   avstake           27 non-null     float64
```

Figure 4.5: Verification of data types stored in the DataFrame

CHAPTER 5

Visualization Design

5.1 Overview

This chapter deals with the development of the last of the system modules, which uses the dataset generated from the previous processes to present the data in a clear and accessible way, taking into account the preferences indicated by the user. Therefore, this development process is a fundamental part that determines the final quality and usefulness of the system.

This process has been divided into different stages. First, the Streamlit environment was prepared, and the application's layout was defined in `app.py`, so the dashboard had a intuitive structure. Subsequently, most of the data analysis and plotting logic was developed, distributing the code between two files dedicated to these tasks: `data_processing.py` and `plotting.py`. Finally, all the work on the Streamlit file was integrated, enabling the user interaction features.

5.2 Streamlit Setup & Layout Design

This section discusses the process followed for designing, structuring, and styling the visual interface of the Streamlit application. This process gave us a clear idea of the desired appearance of the application and the contents of the dashboard for its further development. First, a layout mockup was developed, and then the idea was translated into code using the components offered by Streamlit. Each of these steps is discussed in more detail in the following.

5.2.1 Mockup Design

For the design of the application's user interface, which consists of the dashboard and the user preferences configuration panel, a schematic mockup was created to serve as a template before its actual implementation in the code.

The mockup, in Figure 5.1 shows the key components of the design and layout distribution. The user preferences are placed in a drop-down sidebar, where different widgets, such as sliders and selects, would be presented for adjustment. The rest of the page has a fixed header with the application's name and a brief description. Just below and occupying most of the screen is the dashboard itself. A navigation bar allows access to the different sections, and different graphs, tables, and metrics are presented in each.

5.2.2 Streamlit Widgets & Layout Components

Once the application's layout was defined, the Streamlit API was studied to select the components and widgets that best suit the needs. The Streamlit library offers a wide variety of proprietary and community-made elements, so it was easy to implement a design similar to the one shown in the mockup.

Below is a description of the main elements used in this development phase, divided between those used for the layout and those used to display interactive widgets to modify user preferences.

5.2.2.1 Layout Elements

Streamlit offers different options for controlling how elements are arranged on the screen. These elements generally behave as containers and have been used mainly for the dashboard

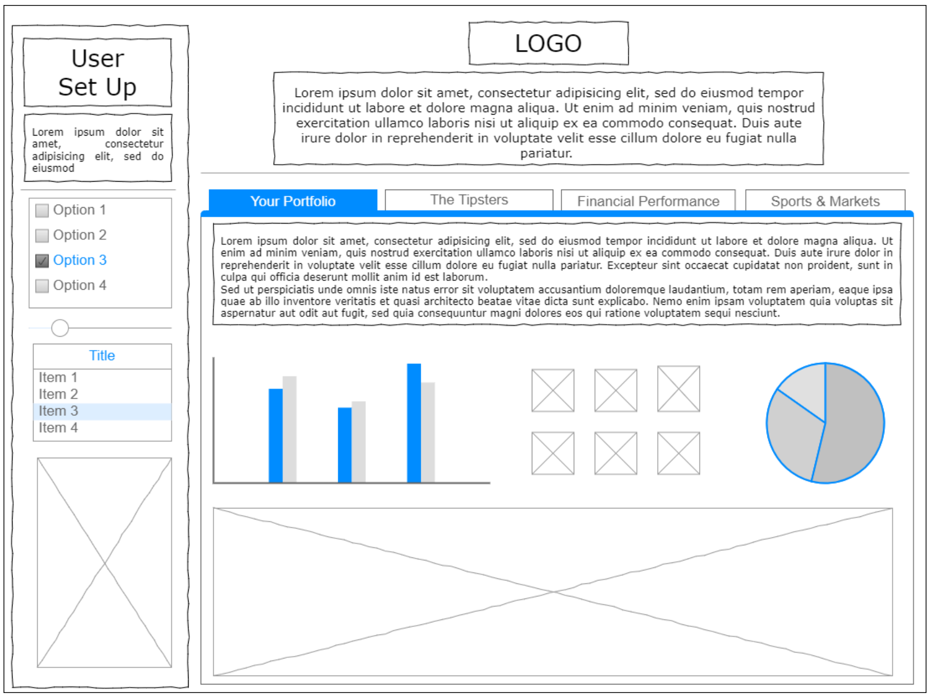

User Set Up
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
Option 1
Option 2
Option 3
Option 4
Title
Item 1
Item 2
Item 3
Item 4
LOGO
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Your Portfolio
The Tipsters
Financial Performance
Sports & Markets
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt.

Figure 5.1: Mockup Design of the Application Layout

layout.

- **st.container:** Inserts an invisible container in the application, which allows adding several elements containing their expansion on the page and maintaining the desired layout. It has been used to display metrics or group graphs with widgets that allow customization, such as multi-selects to select the tipsters to analyze.

- **st.columns:** Inserts several containers in a row, placed side by side, in which multiple elements can be added. With its parameters, the number of columns, the width of each column, or the spacing between columns can be modified.

- **st.tabs:** Inserts separate tabbed containers, allowing users to navigate between sections with related content. It has divided the dashboard into four main sections: Your Portfolio, The Tipsters, Financial Performance, and Sports&Markets.

- **st.sidebar:** Inserts a deployable sidebar, pinned to the left, on which you can include different elements. It is an ideal element on which to add the input widgets. It has been used to group user preferences. Collapsing the sidebar allows the user to focus their attention on the dashboard by having more screen space.

- **st.expander:** Inserts a container in the app that the user can expand or collapse to display its content. It has been used in the sidebar to display input widgets that allow advanced portfolio configuration.

These elements allow interaction with the user, as the selected or inserted value can be stored and used in other parts of the code. Generally, this widget is displayed on the sidebar as a customization component of user preferences.

- **st.slider:** Displays a slider widget, enabling it to define its maximum and minimum values, step interval, or format of its values, among other parameters. It has been used to define several customization parameters, such as the number of daily picks the user wishes to place.
- **st.radio:** Displays a radio button widget, with which the user can mark a single selection from several previously defined ones. It has been used, for example, to select whether or not to customize the sports on which to bet.
- **st.multiselect:** Displays a multi-select widget, with which the user can select one, several, or none of a series of previously defined options. It has been used, among other uses, to select the sports the user wants to bet on.
- **st.selectbox:** Displays a select widget, where the user can choose one of the options displayed when clicking on the widget. It has been used for the user to define the bookmaker on which he bets.

The `app.py` file is the core of this module as it deals with both the visualization and the interaction with the user, so it is in this file where the Streamlit library is imported and, therefore, where the actual implementation of the mockup design and the components described is carried out. Next, it details how this process was carried out in the code.

First, the dashboard is sectioned under the application header using `st.tabs`.

Listing 5.1: Layout of the main content tabs

```python
tab1, tab2, tab3, tab4 = st.tabs(["Your Portfolio", "The Tipsters", "
    Financial Performance", "Sports & Markets"])
with tab1:
    # tab1 content...
with tab2:
    # tab2 content...
with tab3:
     # tab3 content...
with tab4:
     #tab4 content...
```

Subsequently, particularly in each tab, an internal layout is made for each one, considering the graphs and data to be included. This structured layout would be provisional since small adjustments were made for their correct visualization after developing graphs and tables.

Listing 5.2: Container and column layout within a tab

```python
with tab1:
    # tab1 intro...
    t1c1 = st.container(border=True)
    t1col11, t1col21, t1col31, t1col41, t1col51 = t1c1.columns(5)
    with t1col11:
         # metrics, graphs, tables...
         t1c2 = st.container(border=True)
         t1col12, t1col22 = t1c2.columns(2, gap='large')
             #rest of the code...
```

Finally, the sidebar and the dropdown element mentioned above were created to display a container with advanced settings, with `st.expander`.

Listing 5.3: Sidebar configuration with user preference widgets

```python
with st.sidebar:
   # user preferences widgets...
   with st.expander('Show advanced options'):
       #advanced configuration widgets...
```

After defining the sidebar in the application layout, different types of input widgets were included, and the parameters were configured and associated with variables to develop the user interaction features later.

Listing 5.4: Sidebar widgets for user preferences and customization options

```python
st.subheader('Activity Level')
npicks_slider = st.slider(
        # slider parameters...
st.subheader('Betting Markets')
sport_selector_toggle = st.radio(
        # widget configuration...
if sport_selector_toggle == "Personalized":
    sport_selector = st.multiselect(
        # widget configuration...
```

The integration of visualization components with data processing is detailed in Appendix C.

CHAPTER 6

Case Study

This chapter's objective is to present the project's final results and demonstrate the application's main functionalities and uses. For this purpose, it is approached as a case study in which the different steps a user performs from the first contact with the application are presented.

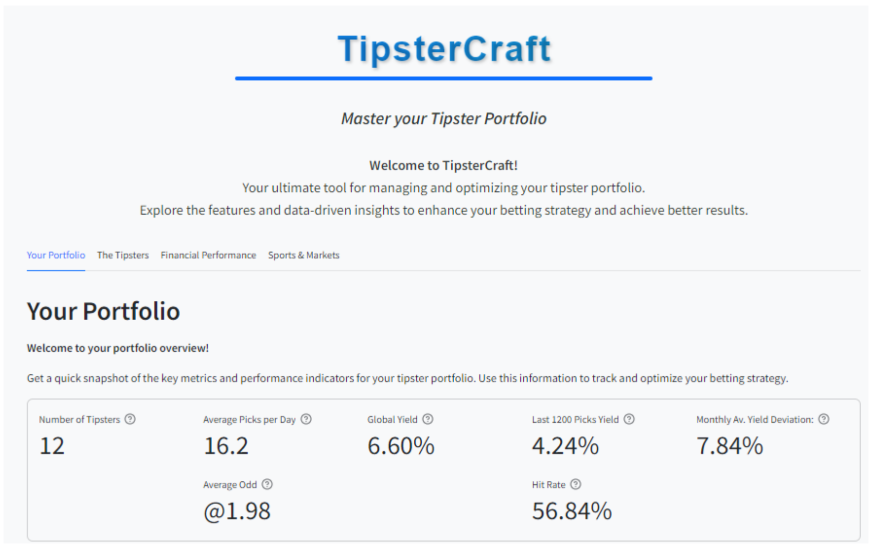
6.1 Introduction

In this case study, it is assumed that a bettor, interested in knowing the best tipsters that fit his needs, enters the application. First, he accesses the page where all the customization parameters show their default values. Subsequently, he enters his personal preferences, and the portfolio is updated to meet the specified requirements. Finally, the user navigates between the tabs to analyze the results.

Each of the following sections details these steps and shows figures to visualize the information and data provided by the dashboard.

6.2 Initial Access

When the user accesses the application for the first time, the page displays the header, which includes the logo, slogan, and a brief explanation of the tool. The user preferences are set to the default values, so a portfolio is loaded and determined through these values. In this process, the first tab is the active one. Figure 6.1 shows the initial state of the app.


TipsterCraft

Master your Tipster Portfolio

Welcome to TipsterCraft!
Your ultimate tool for managing and optimizing your tipster portfolio.
Explore the features and data-driven insights to enhance your betting strategy and achieve better results.

Your Portfolio | The Tipsters | Financial Performance | Sports & Markets

Your Portfolio

Welcome to your portfolio overview!

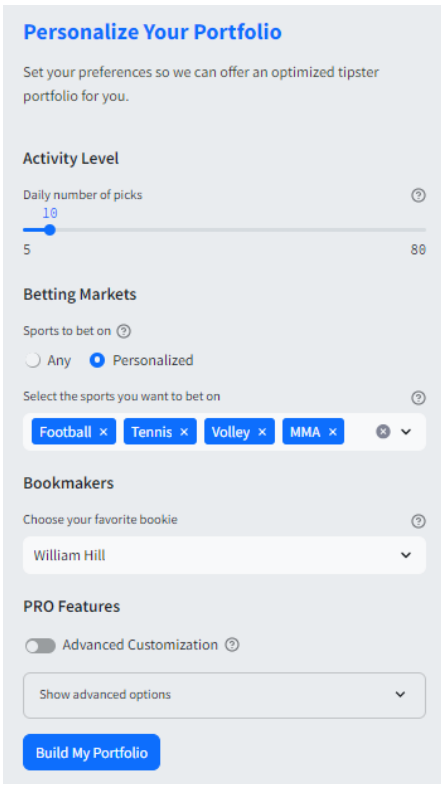
Get a quick snapshot of the key metrics and performance indicators for your tipster portfolio. Use this information to track and optimize your betting strategy.

Number of Tipsters: 12
Average Picks per Day: 16.2
Global Yield: 6.60%
Last 1200 Picks Yield: 4.24%
Monthly Av. Yield Deviation: 7.84%
Average Odd: @1.98
Hit Rate: 56.84%

Figure 6.1: Initial view of the application.

6.3 User Preferences Settings

The next step in the process of using the tool is to configure the user preferences. To do this, there is a drop-down sidebar on the left side of the page where different parameters can be adjusted, as shown in Figure 6.2. In this case, the user is not an expert in the field, so he prefers not to enable the advanced features, so that they remain unchanged, as shown in Figure 6.3.


Personalize Your Portfolio

Set your preferences so we can offer an optimized tipster portfolio for you.

Activity Level

Daily number of picks: 10 (5 – 80)

Betting Markets

Sports to bet on: Any [✓ Personalized]

Select the sports you want to bet on: Football, Tennis, Volley, MMA

Bookmakers

Choose your favorite bookie: William Hill

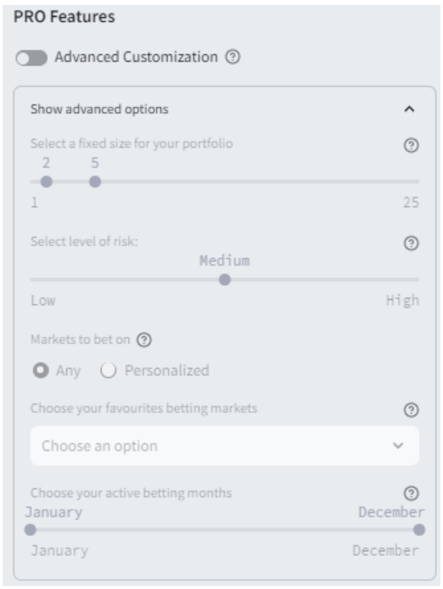
PRO Features

Advanced Customization

Show advanced options

Build My Portfolio

Figure 6.2: Sidebar settings for customizing default values.


PRO Features

Advanced Customization

Show advanced options

Select a fixed size for your portfolio: 2 – 5 (1 – 25)

Select level of risk: Medium (Low – High)

Markets to bet on: [✓ Any] Personalized

Choose your favourites betting markets: Choose an option

Choose your active betting months: January – December (January – December)

Figure 6.3: Sidebar PRO features.

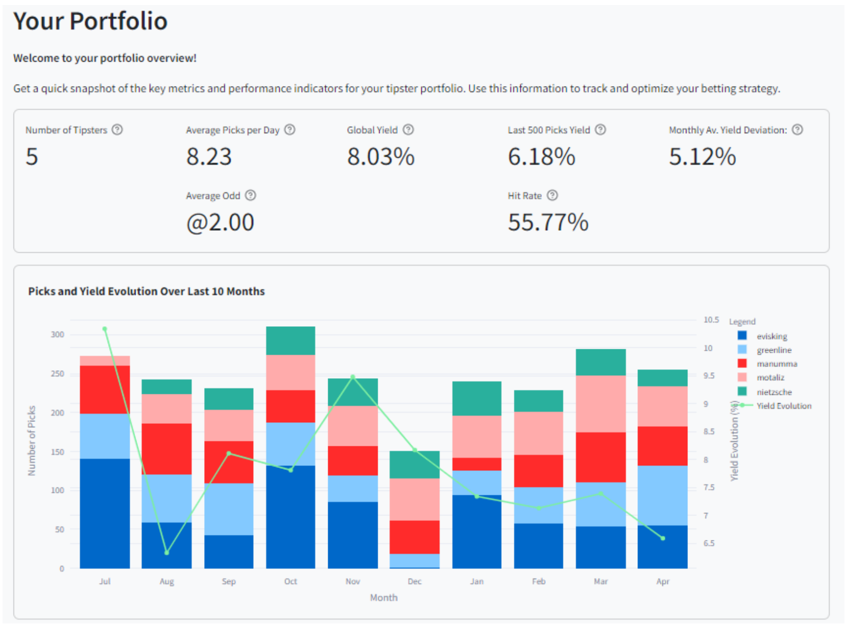
6.4 Navigation through Tabs

The user can then start to analyze and observe the behavior and characteristics of their newly created customized portfolio. To do this, they visit each of the tabs.

First, an overview of the portfolio's characteristics is provided through metrics and graphs, in the Your Portfolio tab, as shown in Figure 6.4.

Once inspected, the user accesses the next tab, The Tipsters, where you can study the tipsters that make up the portfolio in more detail. At the beginning of this tab, you can see a table with the main statistics of the tipter, a link to his personal page on Pyckio.com and a pie chart showing the influence of each one. This is represented in Fig. 6.5. There are some graphs where the user can select the tipsters to analyze, as shown in Figure 6.6.

The next tab is Financial Performance. First, two widgets are displayed with which the user selects the bank available for investment and betting strategy. The tool then calculates and displays the recommended stake per bet, as shown in Figure 6.7. Scrolling down, the user can see a graph showing the profit obtained given the bank and stake per bet and the



Your Portfolio

Welcome to your portfolio overview!

Get a quick snapshot of the key metrics and performance indicators for your tipster portfolio. Use this information to track and optimize your betting strategy.

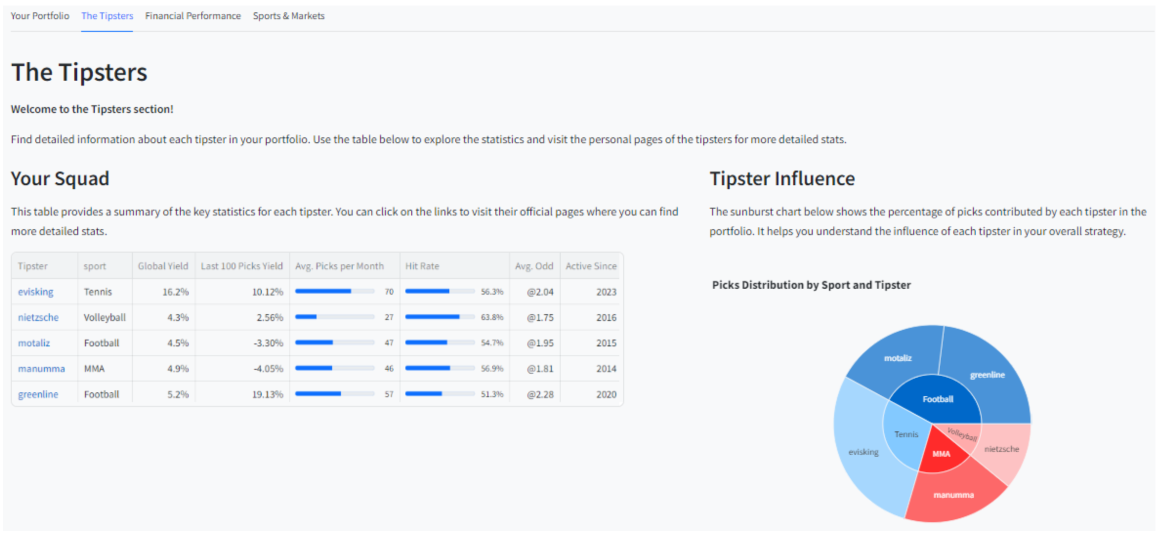
Number of Tipsters	Average Picks per Day	Global Yield	Last 500 Picks Yield	Monthly Av. Yield Deviation:
5	8.23	8.03%	6.18%	5.12%
	Average Odd		Hit Rate	
	@2.00		55.77%	

Picks and Yield Evolution Over Last 10 Months

Figure 6.4: Main view of *Your Portfolio* tab.

expected future profit as a function of the number of months, as shown in Figure 6.8.

Finally, the user accesses the Sport & Markets tab, where he can analyze the type of picks and the sport on which his portfolio is based, observing both the particular returns and the influence of the same, as shown in Figure 6.9.



The Tipsters

Welcome to the Tipsters section!

Find detailed information about each tipster in your portfolio. Use the table below to explore the statistics and visit the personal pages of the tipsters for more detailed stats.

Your Squad

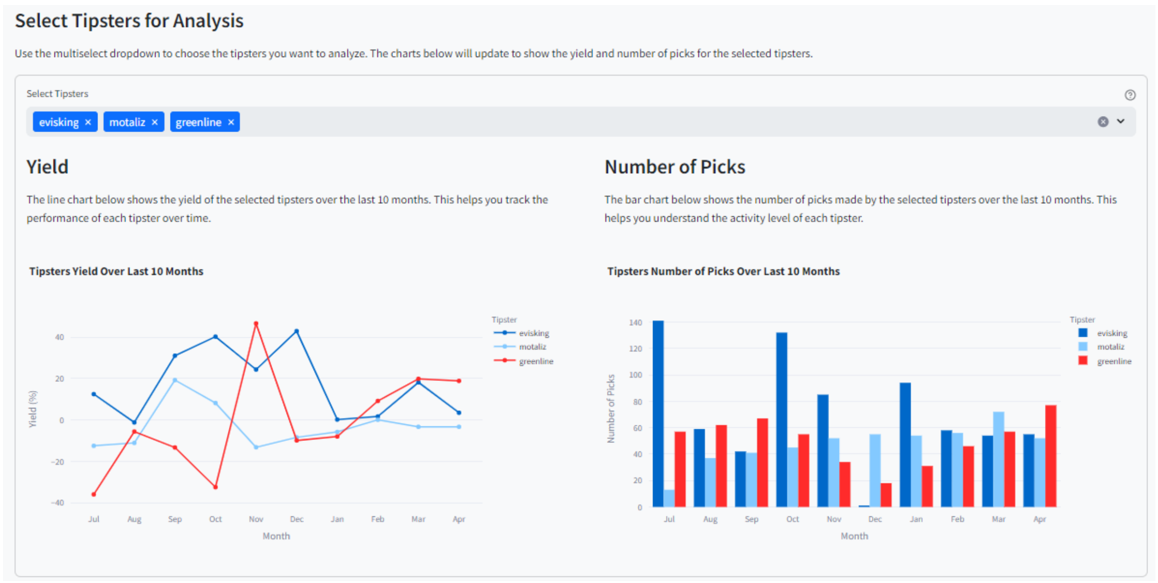
This table provides a summary of the key statistics for each tipster. You can click on the links to visit their official pages where you can find more detailed stats.

Tipster	sport	Global Yield	Last 100 Picks Yield	Avg. Picks per Month	Hit Rate	Avg. Odd	Active Since
evisking	Tennis	16.2%	10.12%	70	56.3%	@2.04	2023
nietzsche	Volleyball	4.3%	2.56%	27	63.8%	@1.75	2016
motaliz	Football	4.5%	-3.30%	47	54.7%	@1.95	2015
manumma	MMA	4.9%	-4.05%	46	56.9%	@1.81	2014
greenline	Football	5.2%	19.13%	57	51.3%	@2.28	2020

Tipster Influence

The sunburst chart below shows the percentage of picks contributed by each tipster in the portfolio. It helps you understand the influence of each tipster in your overall strategy.

Figure 6.5: Main view of *The Tipsters* tab.



Select Tipsters for Analysis

Use the multiselect dropdown to choose the tipsters you want to analyze. The charts below will update to show the yield and number of picks for the selected tipsters.

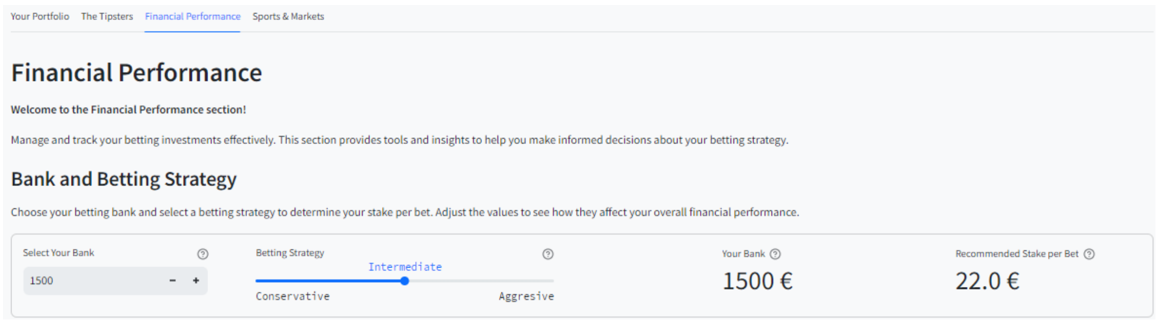
Yield

The line chart below shows the yield of the selected tipsters over the last 10 months. This helps you track the performance of each tipster over time.

Number of Picks

The bar chart below shows the number of picks made by the selected tipsters over the last 10 months. This helps you understand the activity level of each tipster.

Figure 6.6: Tipsters last month's analysis charts.



Financial Performance

Welcome to the Financial Performance section!

Manage and track your betting investments effectively. This section provides tools and insights to help you make informed decisions about your betting strategy.

Bank and Betting Strategy

Choose your betting bank and select a betting strategy to determine your stake per bet. Adjust the values to see how they affect your overall financial performance.

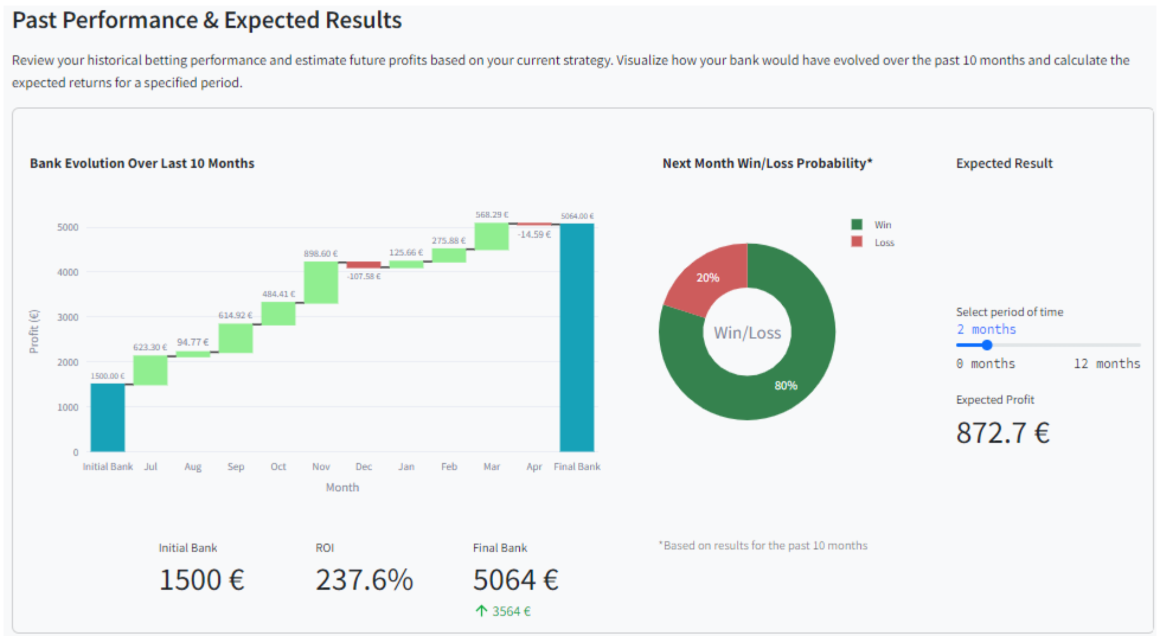
Select Your Bank: 1500

Betting Strategy: Intermediate (Conservative – Aggresive)

Your Bank: 1500 €

Recommended Stake per Bet: 22.0 €

Figure 6.7: Bank and Stake set up.



Past Performance & Expected Results

Review your historical betting performance and estimate future profits based on your current strategy. Visualize how your bank would have evolved over the past 10 months and calculate the expected returns for a specified period.

Bank Evolution Over Last 10 Months

Next Month Win/Loss Probability*

Expected Result

Select period of time
2 months
0 months 12 months

Expected Profit
872.7 €

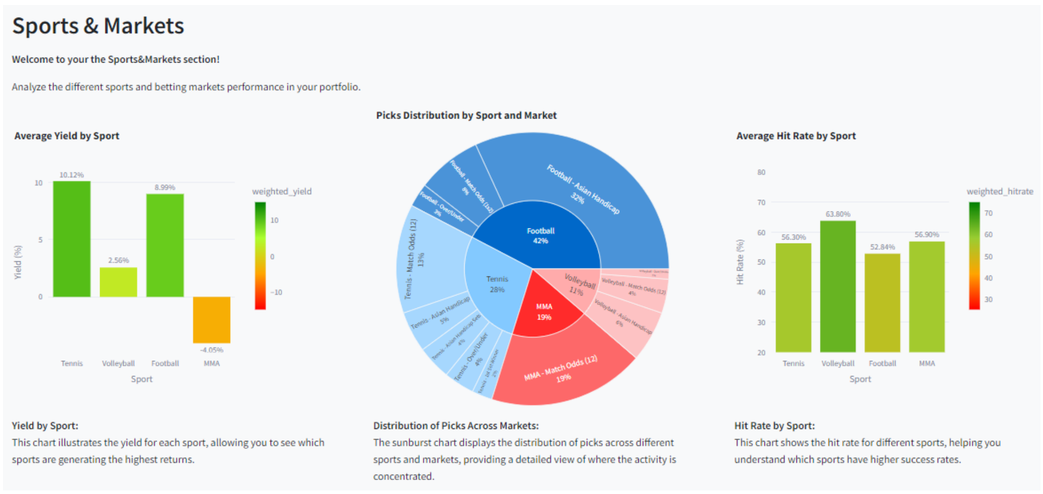
Initial Bank
1500 €

ROI
237.6%

Final Bank
5064 €
↑ 3564 €

*Based on results for the past 10 months

Figure 6.8: Past performance and expected results.



Sports & Markets

Welcome to your the Sports&Markets section!

Analyze the different sports and betting markets performance in your portfolio.

Average Yield by Sport

Picks Distribution by Sport and Market

Football 42%
Football - Asian Handicap 32%
Football - Match Odds (1x2) 6%
Football - Over/Under 3%
Tennis 28%
Tennis - Match Odds (12) 13%
Tennis - Asian Handicap 5%
Tennis - Asian Handicap Sets 4%
Tennis - Over/Under 4%
Tennis - 1st Set Winner 2%
MMA 19%
MMA - Match Odds (12) 19%
Volleyball 11%
Volleyball - Match Odds (12) 4%
Volleyball - Asian Handicap 6%

Average Hit Rate by Sport

Yield by Sport:
This chart illustrates the yield for each sport, allowing you to see which sports are generating the highest returns.

Distribution of Picks Across Markets:
The sunburst chart displays the distribution of picks across different sports and markets, providing a detailed view of where the activity is concentrated.

Hit Rate by Sport:
This chart shows the hit rate for different sports, helping you understand which sports have higher success rates.

Figure 6.9: Main view of the *Sports & Markets* tab.

CHAPTER 7

Conclusions and Future work

This chapter describes the conclusions extracted from this project and thoughts on future work.

7.1 Conclusions

To conclude this thesis, the developed system and its purpose are recapitulated. The primary motivation behind this project was to create a tool that empowers bettors to make informed decisions based on a comprehensive analysis of tipster statistics. The system is designed to generate a portfolio that diversifies the risk of investments and allows users to observe and analyze their portfolios' graphs, tables, and metrics. This analysis enables users to determine whether the portfolio aligns with their needs and understand the methodology followed by each tipster.

The application's user-friendly interface caters to many bettors, from novices with limited analytical skills to experienced bettors requiring advanced customization options. It is designed to accommodate users with varying levels of daily availability, allowing them to specify the number of picks they wish to make daily. This flexibility ensures that casual bettors and those with more time to dedicate to betting can benefit from the application.

Throughout the development of this project, Streamlit was utilized to represent results, and Python was employed to deploy the server and execute the necessary calculations. The system architecture is layered, ensuring that each component functions correctly and contributes to the overall efficacy of the application.

One of the significant benefits of this application is its educational component. Users can learn about the crucial parameters that determine a tipster's profitability by interacting with the dashboard. This educational aspect helps users make better-informed decisions and enhances their understanding of the betting landscape.

This tool can significantly impact how bettors approach their strategies as technology advances. The increasing interest in artificial intelligence and data analysis highlights the importance of such tools in various domains, including sports betting. Analyzing historical data, identifying trends, and predicting future outcomes can revolutionize betting strategies and outcomes.

Moreover, the application is designed for easy deployment and use across various platforms. All extracted data is stored in a standardized format, facilitating potential integration with other intelligent systems. Streamlit's cloud capabilities further enhance the application's accessibility and security, making it easier to manage and deploy.

In conclusion, this project aims to provide a comprehensive, user-friendly tool that empowers bettors to make data-driven decisions. By offering both basic and advanced customization options, the application caters to a diverse audience, ensuring that all users can benefit from informed betting strategies. The combination of educational and practical components makes this application a valuable resource for anyone involved in sports betting.

7.2 Achieved goals

This project has successfully achieved several specific goals, contributing to developing a comprehensive and user-friendly tool for informed betting decisions. The following points summarize the key accomplishments:

- **Formation of a Robust Dataset:** A web scraping module was successfully developed to gather data from hundreds of tipsters on pyckio.com. This module ensured that the dataset remained current and relevant by continuously updating the latest tipster performance data. This up-to-date dataset provided the foundation for thorough and accurate analysis.

- **User-Friendly Dashboard:** One of the primary goals was to create an interactive and intuitive dashboard. This was achieved by designing a user interface allowing users of varying expertise levels to build and manage their tipster portfolios easily. The dashboard provides essential information in a clear and accessible manner, ensuring that users can make informed decisions without requiring extensive knowledge in the field.

- **Comprehensive Tracking and Analysis:** The application includes interactive capabilities that offer users multiple analyses and parameters defining the characteristics of their portfolio and each tipster within it. Users can analyze and understand their investments from different perspectives by incorporating graphs and KPIs. This comprehensive tracking and analysis enable users to effectively compare results and identify trends.

- **Historical Performance and Future Projections:** An economic perspective was provided by considering users' initial investments and betting strategies. The tool performs temporal analyses to visualize and analyze past performance and project future outcomes in monetary terms. This feature allows users to assess potential risks and benefits, helping them make well-informed investment decisions.

- **Analytical Function Library:** A comprehensive library of functions was designed to work on the collected dataset. This library facilitates the extraction of meaningful insights and supports various analytical needs. It ensures that the dashboard provides robust and versatile analysis capabilities, empowering users to explore different approaches and gain deeper understanding of their investments.

- **Enhanced User Interaction and Customization:** The application offers advanced customization options, including adjusting stakes per tipster to equalize their influence on the portfolio's profitability. This feature allows users to tailor their betting strategies according to their preferences and risk tolerance, enhancing the tool's utility.

7.3 Future work

In the future, several enhancements and expansions can be made to the system to improve its functionality and effectiveness. The following points outline potential areas for development.

- **Addition of New Crawlers:** To expand the scope of data collection, new crawlers could be developed to extract information from additional tipster communities. This data would be unified by the data munging module, ensuring a comprehensive and consolidated dataset for analysis.

- **Dataset Expansion:** Increasing the size of the dataset and extending its temporal range would enhance the analysis. This expansion could include more detailed data, such as the monthly price of each tipster and any fixed costs associated with the portfolio, providing a deeper understanding of financial commitments.
- **Real-time Data Update Integration:** Reconnecting the visualization module with the data collection module would enable real-time updates of portfolio information. This integration would allow users to dynamically monitor their investments' evolution and make more informed decisions based on the latest data.
- **Machine Learning Applications:** Implementing machine learning techniques to analyze the results of different portfolios or potential portfolios over time could reveal patterns associated with high-yield portfolios. This analysis could help predict future performance and optimize investment strategies.
- **Stake Adjustment Service:** Adding an extra page to the application that offers a stake adjustment service by tipster would be highly beneficial. This service would aim to equalize the influence of each tipster on the final profitability by recommending the amount to be wagered based on the user's bank. This feature would provide personalized advice to users, helping them optimize their betting strategies.

APPENDIX A

Impact of this project

This appendix reflects, quantitatively or qualitatively, on the possible impact...

A.1 Social impact

The social impact of our betting portfolio management tool extends across various aspects of the betting community and beyond. This tool bridges data analysis and decision-making, promoting informed betting practices and reducing the risks associated with uninformed gambling.

Firstly, the tool democratizes access to advanced analytical capabilities, enabling bettors of all experience levels to make informed decisions. Novice bettors can use the intuitive interface to adjust basic parameters, while experienced bettors can delve deeper into the data for more nuanced insights. This inclusivity fosters a more knowledgeable and responsible betting community.

The tool educates users on the critical factors that influence betting success by providing detailed analyses and visualizations of tipster performance and portfolio metrics. This educational component helps bettors understand the importance of thorough analysis, reducing the likelihood of impulsive and potentially harmful betting behavior.

Moreover, the application encourages transparency and accountability among tipsters. By compiling and presenting comprehensive data on tipster performance, the tool enables users to make informed choices about which tipsters to follow, promoting a more trustworthy betting ecosystem.

This tool's potential to diversify and mitigate betting risks can lead to more sustainable betting practices. By allowing users to create portfolios that spread risk across multiple tipsters, the tool helps reduce the financial strain of betting, contributing to better mental and financial health for individuals involved in betting activities.

In summary, our betting portfolio management tool enhances the betting experience through advanced data analysis and contributes positively to the social landscape by promoting informed, responsible, and sustainable betting practices.

A.2 Economic Impact

The potential economic impact of our betting portfolio management tool is significant, influencing various stakeholders within the betting industry and beyond. The tool offers economic benefits to end users, driving innovation and efficiency in the betting ecosystem.

The tool provides a means for end users to optimize their betting strategies and maximize their returns on investment. By enabling bettors to make data-driven decisions and manage their portfolios effectively, the tool increases the likelihood of achieving consistent profitability. This can attract more users to betting platforms, increasing the overall volume of bets.

Additionally, the tool supports tipsters and tipster communities by providing a direct link to the tipster's main page within the tipster community. This feature not only promotes the use of the tipster community but also encourages users to subscribe to the services of the tipsters included in their portfolios. This increased visibility and accessibility can lead to higher demand for their services and greater engagement within the community.

Overall, our betting portfolio management tool enhances the betting experience for individual users and drives economic growth and efficiency within the broader betting industry. By promoting informed decision-making and transparency, the tool supports the development of a more robust and economically viable betting ecosystem.

A.3 Environmental Impact

The project does not significantly change the environment. The main environmental consideration is the energy consumption associated with the development and maintenance of the system. The tool relies on cloud-based infrastructure and data centers to process and store the extensive datasets required for analysis. These data centers consume significant amounts of electricity for running and cooling servers, contributing to greenhouse gas emissions and environmental degradation.

Efforts to optimize data center energy efficiency are crucial to mitigate the environmental impact. Techniques such as using renewable energy sources, implementing more efficient cooling systems, and enhancing server utilization can help reduce the system's overall energy consumption.

In summary, while this project's primary environmental impact is the energy consumption of its server infrastructure, steps can be taken to minimize this footprint.

A.4 Ethical Impact

The primary ethical considerations of this project revolve around privacy and the responsible use of data. Although the project utilizes publicly available data, there remains a significant concern about public awareness and consent regarding data sharing. Most users are unaware of the implications of publicizing their data. Consequently, greater education on the importance and value of privacy is needed to ensure users understand how their data might be used.

Furthermore, the ethical impact also includes the potential job displacement caused by automation. While the system automates previously manual tasks, it simultaneously necessitates human oversight for maintenance and data analysis. This shift represents a transformation rather than eliminating jobs, requiring new skills and roles to manage and interpret the system's outputs.

APPENDIX B

Economic budget

B.1 Introduction

In this appendix, a comprehensive economic budget to implement this project is developed. The main components of this budget is detailed in the following sections, covering physical resources, human resources, software licenses, and taxes.

B.2 Physical resources

The budget for the physical devices necessary for the development of this project is mainly composed of a computer with the following minimum requirements.

- **CPU**: Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz
- **RAM**: 16 GB
- **Disk**: 500 GB

B.3 Human Resources

In this section, we calculate the cost of human resources required for the development and maintenance of the system. The project involves 12 ECTS (European Credit Transfer and Accumulation System), with each credit equivalent to 27 hours of work. Thus, the total work required is $12 \times 27 = 324$ hours.

Based on the average internship salary offers observed:

- Full-time internship: 1170 €/month
- Part-time internship (15 hours/week): 400 €/month

Assuming a rate of 400 € per month for part-time work, and considering an average of 4 weeks per month, we get a weekly rate of approximately 100 €. Given that part-time involves 15 hours per week, the hourly rate is $\frac{100 \text{ €}}{15 \text{ hours}} \approx 6.67$ €.

Therefore, for 324 hours, the total cost of human resources is:

$$324 \text{ hours} \times 6.67 \text{ €} \approx 2162.28 \text{ €}$$

B.4 Licenses

All the software tools utilized in this project are open-source and freely available. Consequently, the cost of software licenses is zero.

B.5 Taxes

When the final product is sold to a company, the sale is subject to a tax of 15% of the product's price, as defined in Spanish law's Statute 4/2008.

B.6 Conclusion

The total cost of developing this project, which includes physical and human resources, is approximately 3362.28 €. This budget ensures a comprehensive understanding of the financial requirements for its successful implementation.

APPENDIX C

Reference manual

This section describes the data processing and visualization modules.

C.1 Data processing modules

Each module component will now be described. In their current state, both the module and these files are relatively simple and perform basic functions on the data. However, they are designed with the application's scalability in mind, considering a potential increase in complexity and functionalities.

pyckio_data_munging.py

This is the module's main file, and the other two facilitate its readability. First, import the CSV file generated in the Data Collection module with the read_csv() pandas method.

Listing C.1: Importing necessary libraries and reading the raw data file

```python
import pandas as pd
import ast
```

```
from functions import *
from config import PYCKIO_RAW_DATA_FILE, PYCKIO_PROCESSED_DATA_FILE

df = pd.read_csv(PYCKIO_RAW_DATA_FILE)
```

It then converts the data columns to their desired type, as shown in the previous section, and generates new valuable metrics from other data, relying on functions defined in functions.py.

Listing C.2: Adding last100picksyield column

```
df['lastnpicksyield'] = df.apply(lambda row: functions.
    get_last_100_picks_yield(row['lastMonthsnPicks'], row['lastMonthsYield'
    ]), axis=1)
```

Finally, it generates the CSV file with the processed data, which is ready for use by the system's final module.

functions.py

This file is responsible for storing all the auxiliary functions of a certain complexity that facilitate data processing and the creation of new data. Therefore, these functions improve the readability of the main code and allow for greater modularity.

Listing C.3: Function to calculate the yield of the last NPICKS_YIELD picks

```
def get_last_npicks_yield(monthly_npicks: List[int], monthly_yields: List[
    float]) -> float:
    left_picks = config.NPICKS_YIELD
    last_npicks_yield = 0.0
    for i in range(len(monthly_npicks)):
        if left_picks <= 0:
            break
        weight = min(left_picks, monthly_npicks[i]) / 100.0
        last_100_picks_yield += weight * monthly_yields[i]
        left_picks -= monthly_npicks[i]
    return last_npicks_yield
```

config.py

Following the above-mentioned line of work, the config.py file allows efficient and scalable system development. In this case, the file contains the configuration constants in the module, ensuring that critical parameters are consistent and easy to modify.

Listing C.4: Configuration constants in config.py

```
# data storage
DATA_PATH = 'data/'
PYCKIO_RAW_DATA_FILE = 'pyckio_raw_data.csv'
PYCKIO_PROCESSED_DATA_FILE = 'pyckio_dataset.csv'

# dataframe configuration
NPICKS_YIELD: 100
```

C.2 Visualization modules

C.2.1 DataFrames Design for Visualization

In this section, we will discuss the creation of various dataframes that allow us to focus on the analysis of the portfolio and the tipsters from different perspectives. The definitions of the functions that will be explained in this section are provided in `data_processing.py`, which is the main file for this section. The `data_processing.py` file makes extensive use of the `pandas` and `numpy` libraries to perform these processes efficiently.

In `data_processing.py`, several functions are defined to develop the dataframes necessary for our analysis. Below is an overview of these functions:

- `generate_tipsters_portfolio:` : This is the application's core function. It is the one that deals with, given the dataframe with all the tipsters for which we have data, and the user preferences, whose values are collected in the widgets, to develop a reduced dataframe where only the data of the tipsters that make up the user's portfolio are included. This is done through an algorithm that sorts the complete df according to a score value that defines each tipster among the eligible ones.
- `create_portfolio_df`: From the dataframe generated by the previous function, called custom_tipsters_df, it generates a single row DataFrame containing the main metrics of the portfolio, i.e., without particular data for each tipster.

- `create_tipsters_df`: It generates a column-reduced version of custom_fulldata_df, suitable for presentation in table format in 'The Tipsters' tab.
- `create_timestats_df`: It breaks down the values related to the time statistics of the tipsters in the portfolio so that in the new dataframe, a single tipster has as many columns under his name as months have been extracted. Each row shows a tipster's statistics in a given month and year.
- `create_markets_df`: In a similar way to the previous function, it breaks down the different betting markets in which each tipster operates and indicates the values that define him, such as his yield or the percentage of picks he dedicates to that betting market.

The `data_processing.py` file also defines several auxiliary functions. These functions help to obtain new metrics and valuable data or handle more complex definitions that improve code readability. Some examples of these auxiliary functions include `get_stake_to_bank_ratio`, `mean_yield_deviation`, `load_data` (for loading the CSV from the data munging module), and `generate_dataframes`, which internally calls most of the dataframe generation functions and is invoked from `main.py` to utilize them.

Listing C.5: Structure of `generate_dataframes` function

```
def generate_dataframes(tipsters_fulldata_df):
    tipsters_df = create_tipsters_df(tipsters_fulldata_df)
    reduced_tipsters_df = create_reduced_tipsters_df(tipsters_fulldata_df)
    portfolio_df = create_portfolio_df(tipsters_fulldata_df)
    timestats_df = create_timestats_df(tipsters_fulldata_df)
    markets_df = create_markets_df(tipsters_fulldata_df)
    return tipsters_df, reduced_tipsters_df, portfolio_df, timestats_df,
        markets_df
```

C.2.2 Chart Creation and Design

The main file for this section is `plotting.py`, which defines functions to generate a wide variety of charts and graphics based on a dataframe provided as a parameter. Therefore, it utilizes the dataframes created in `data_processing.py`. The `app.py` file calls these functions to render them in the dashboard subsequently. To achieve this, `plotting.py` imports the `plotly.graph_objects` and `plotly.express` libraries.

In this section, we include some of the functions responsible for chart creation and design:

- `create_npicks_yieldevol_chart`: From the timestats_df and portfolio_df, it generates a graph composed of two superimposed charts. A bar chart, where each bar represents the number of monthly picks of the tipsters, is stacked in each month so that you can see the total number of picks of the portfolio. The other is a line chart showing the evolution of the portfolio's yield over the last months.

- `plot_waterfall_chart`: It generates a waterfall chart that allows one to analyze the evolution of the user's bank over the last months if he had placed all the bets of the tipsters of his custom portfolio.

- `create_tipster_activity_piechart`: It provides a piechart that allows an analysis of the influence of each tipster on the portfolio, calculated as the average number of monthly picks of a tipster concerning the average monthly total of the portfolio.

- `create_sport_markets_sunburst`: creates a sunburst chart to analyze a specific sport or market sector. The inner ring shows the sport, while the external ring shows the market associated with that sport.

C.2.3 Component Integration

This section will show the graphs and charts resulting from these functions, where the results will be presented.

The `app.py` file is the main driver to integrate the data processing and plotting functionalities described in the previous sections. This file obtains dataframes using the functions from `data_processing.py` and utilizes these dataframes to generate and render various charts on the pre-designed layout.

First, `app.py` imports the necessary modules:

Listing C.6: Importing Modules

```python
import data_processing as dp
import plotting as plt
```

Next, the full dataset is loaded into a dataframe using the `load_data` function from the data munging module:

Listing C.7: Loading the Dataset

```
df = dp.load_data('./data/tipsters_dataset.csv')
```

Subsequently, a custom dataframe for the user's portfolio is generated.

The `generate_tipsters_portfolio` function creates a reduced dataframe from the original, including only the tipsters that belong to the portfolio. Then, the main dataframes needed for visualization are generated:

Listing C.8: Generating Dataframes

```
custom_tipsters_df = dp.generate_tipsters_portfolio(df, user_preferences)
tipsters_df, reduced_tipsters_df, portfolio_df, timestats_df, markets_df =
    dp.generate_dataframes(custom_tipsters_df)
```

With the dataframes ready, various charts are created using the functions from `plotting.py`. These functions take the appropriate dataframes as input parameters to generate the desired visualizations:

Listing C.9: Creating Charts

```
portfolio_npicks_yieldevol_fig = plt.create_npicks_yieldevol_chart(
    timestats_df, portfolio_df)
```

Finally, the charts are included in the defined layout for the dashboard. For instance, the following code snippet shows how a chart is added to a specific column in the layout:

Listing C.10: Including a Chart in the Layout

```
col1, col2, col3 = st.columns(3, gap='large')
with col1:
    st.plotly_chart(yield_by_sport_chart)
```

This integration in `app.py` ensures that the data processing and visualization components work seamlessly together, providing a coherent and interactive user experience on the dashboard.

Bibliography

[1] Apify. Crawlee - apify, 2024. (Accessed: 2024-05-19).

[2] ApuestasDeportivas.com. Historical data of a tipster, 2023. (Accessed on 28/06/2024).

[3] M. Bansal, M. A. DAR, and M. M. Bhat. Data ingestion and processing using playwright. *Authorea Preprints*, 2023.

[4] E. Dabbas. *Interactive Dashboards and Data Apps with Plotly and Dash: Harness the power of a fully fledged frontend web framework in Python–no JavaScript required*. Packt Publishing Ltd, 2021.

[5] D. Flanagan. *JavaScript: The definitive guide: Activate your web pages*. " O'Reilly Media, Inc.", 2011.

[6] D. Forrest and R. Simmons. Forecasting sport: the behaviour and performance of football tipsters. *International journal of Forecasting*, 16(3):317–331, 2000.

[7] B. R. González. *Apuestas deportivas online: Claves para ganar apostando*. Punto de Lectura, 2013.

[8] Google. Chrome DevTools, 2024. /Accessed: 2024-05-19).

[9] A. Gruettner, T. Wambsganss, and A. Back. From data to dollar: using the wisdom of an online tipster community to improve sports betting returns. *European Journal of International Management*, 15(2-3):314–338, 2021.

[10] Inbetsment. Tipsters - inbetsment, 2024. Available at `https://inbetsment.com/es/tipsters`. (Accessed on 2/7/2024).

[11] M. Khorasani, M. Abdou, and J. Hernández Fernández. Streamlit basics. In *Web Application Development with Streamlit: Develop and Deploy Secure and Scalable Web Applications to the Cloud Using a Pure Python Framework*, pages 31–62. Springer, 2022.

[12] E. A. Killick and M. D. Griffiths. In-play sports betting: A scoping study. *International Journal of Mental Health and Addiction*, 17:1456–1495, 2019.

[13] H. Lopez-Gonzalez, A. Estévez, and M. D. Griffiths. Controlling the illusion of control: A grounded theory of sports betting advertising in the uk. *International Gambling Studies*, 18(1):39–55, 2018.

[14] H. Lopez-Gonzalez and M. D. Griffiths. Understanding the convergence of markets in online sports betting. *International Review for the Sociology of Sport*, 53(7):807–823, 2018.

[15] S. Molin. *Hands-On Data Analysis with Pandas: A Python data science handbook for data collection, wrangling, analysis, and visualization.* Packt Publishing Ltd, 2021.

[16] NumPy. What is numpy?, 2024. Accessed: 2024-05-19.

[17] Ordenación del Juego. Informe del jugador en línea, 2023. Available at `https://www.ordenacionjuego.es/es/informe-jugador-online` (Accessed on 2/07/2024).

[18] Ordenación del Juego. Mercado de juego online estatal, 2023.

[19] Python Software Foundation. Blurb: Python's design philosophy, 2024. Accessed: 2024-05-19.

[20] B. J. Riley, L. Li, D. Plevin, and M. Baigent. Betting on australian rules football: Can expert tipsters beat randomness? *Journal of Gambling Studies*, 39(4):1537–1546, 2023.

[21] TipsterTrust. Cómo analizar correctamente las estadísticas de un tipster, 2023. `https://tipstertrust.com/como-analizar-correctamente-las-estadisticas-de-un-tipster` (Accessed: 21/05/2024). (Accessed on 2/07/2024).

[22] G. Van Rossum et al. Python programming language. In *USENIX annual technical conference*, volume 41, pages 1–36. Santa Clara, CA, 2007.