F3

Faculty of Electrical Engineering
Department of Computer Science

Bachelor's Thesis

Machine learning operations framework for predictions in sports

Tigran Oganesian

May 2023
Study Programme: Open Informatics
Field of Study: Software
Supervisor: Ing. Gustav Šír, Ph.D.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Oganesian** Jméno: **Tigran** Osobní číslo: **499048**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Software**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Framework pro provozní operace strojového učení ve sportech

Název bakalářské práce anglicky:

Machine learning operations framework for predictions in sports

Pokyny pro vypracování:

The subject of this bachelor thesis is to develop a well-designed software framework for efficient empirical testing of machine learning models in the domain of predictive sports analytics. The design should follow Machine Learning Operations (MLOps) [1] principles to allow for trackable, reproducible, modular, and efficient experimentation. The framework is to be tested with selected predictive models [2] and portfolio optimization strategies [3], applied across a range of sport data domains. The models, data, and strategies will be provided by the supervisor. The student is expected to:
1) Study up MLOps principles and review the predictive sports analytics domain.
2) Combine the existing resources in a joint SW framework suitable for efficient experimentation by integrating with existing MLOps solutions.
2) Set up a sound workflow evaluation protocol (validation, confidence intervals).
3) Perform a wide range of experiments across different settings in a well-organized fashion.
4) Analyze your results, discuss possible improvements.

Seznam doporučené literatury:

[1] What is MLOps? : https://neptune.ai/blog/mlops-what-it-is-why-it-matters-and-how-to-implement-it-from-a-data-scientist-perspective
[2] Hubáček, Ondřej, Gustav Šourek, and Filip železný. "Forty years of score-based soccer match outcome prediction: an experimental review." IMA Journal of Management Mathematics 33.1 (2022): 1-18.
[3] Matej, Uhrín, et al. "Optimal sports betting strategies in practice: an experimental review." IMA Journal of Management Mathematics 32.4 (2021): 465-489.
[4] MLFlow: An open source platform for the machine learning lifecycle https://mlflow.org/

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Gustav Šír, Ph.D. Intelligent Data Analysis FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **26.01.2023** Termín odevzdání bakalářské práce: **26.05.2023**

Platnost zadání bakalářské práce: **22.09.2024**

Ing. Gustav Šír, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____ _____

Datum převzetí zadání Podpis studenta

Acknowledgement / Declaration

I would like to express my heartfelt gratitude to the Czech Technical University for providing me with the opportunity to pursue my studies. The support offered by the university has played a crucial role in shaping my academic journey and enabling the successful completion of this thesis.

I am deeply indebted to my family for their unwavering support and encouragement throughout my journey at the university. Their belief in my abilities and their willingness to send me for further studies has been instrumental in my academic success.

I am also deeply grateful to my beloved girlfriend, Valeriia, who has been by my side every step of the way. Together, we have faced and overcome the challenges of our studies, providing each other with unwavering support and motivation.

Furthermore, I would like to extend a special thank you to my friends Bohdan and Mitya. Our frequent meetings in the library to tackle difficult study problems have not only enhanced my understanding but also made the learning experience enjoyable and memorable.

Last but not least, I am profoundly thankful to my supervisor, Gustav, whose guidance and expertise have been invaluable throughout the process of writing this thesis. His continuous support, valuable feedback, and timely assistance have significantly contributed to the quality and completion of this work.

I declare that I have prepared the submitted thesis individually and that I have listed all the information sources used in accordance with the Methodological Guideline on the observance of ethical principles in the preparation of an academic final thesis.

In ..... on .....

.....

Abstrakt / Abstract

Tato práce se zaměřuje na vývoj komplexního softwarového frameworku pro efektivní empirické testování modelů strojového učení v prediktivní sportovní analytice. Cílem práce je poskytnout možnost provádět sledovatelné, reprodukovatelné, modulární a efektivní experimenty dodržující zásady MLOps [1] a vylepšit současný systém pro predikci výsledků zápasů začleněním modulů určených pro optimalizaci sázek. Systém bude vyhodnocen pomocí různých modelů předpovědí [2] a strategií optimalizace portfolia [3] aplikovaných na různé oblasti sportovních dat.

Tato práce ukáže integraci stávajících zdrojů do jediného softwarového frameworku v souladu s řešeními MLOps, vytvoření robustního protokolu pro vyhodnocování workflow, širokou škálu experimentů za různých podmínek, analýzu výsledků a případná vylepšení. Tato práce umožní dosáhnout pokroku v chápání a používání metod strojového učení v prediktivní sportovní analytice a zároveň zdůrazní význam efektivního experimentování a reprodukovatelnosti.

Kromě toho má tato práce využít výpočetní možnosti počítačového clusteru k urychlení procesu experimentování a podrobně se zabývá praktickou implementací frameworku, včetně nezbytných softwarových požadavků na nastavení počítačového clusteru.

Klíčová slova: sport prediction, betting optimization, MLOps, MLFlow, computer cluster.

This thesis focuses on the development of a comprehensive software framework for efficient empirical testing of machine learning models in predictive sports analytics. The goal of the work is to provide the option to conduct trackable, reproducible, modular and efficient experiments adhering to MLOps [1] principles, and to improve the current framework for predicting match results by incorporating blocks responsible for betting optimization. The system will be evaluated using different prediction models [2] and portfolio optimization strategies [3] applied to different areas of sports data.

This work will demonstrate the integration of existing resources into a single software framework consistent with MLOps solutions, the creation of a robust workflow evaluation protocol, a wide range of experiments under various conditions, the analysis of results, and potential improvements. This work advances the understanding and application of machine learning methods in predictive sports analytics while emphasizing the importance of efficient experimentation and reproducibility.

Furthermore, this work is to leverage the computational capabilities of a computer cluster to accelerate the experimentation process and discusses the practical implementation of the framework, including the necessary software requirements for setting up the computer cluster.

Keywords: sport prediction, betting optimization, MLOps, MLFlow, computer cluster.

Contents /

/ Figures

Chapter 1
Introduction

In today's fast-paced and interconnected world, the realm of sports betting has evolved into a global phenomenon, captivating millions of enthusiasts and presenting lucrative opportunities for individuals seeking to engage in predictive analysis and strategic decision-making. The emergence of online platforms, the proliferation of sports data, and advancements in technology have transformed the landscape of betting, necessitating innovative tools and frameworks to navigate this dynamic industry.

This thesis explores developing and implementing a framework that leverages the power of data-driven insights and sophisticated algorithms to enhance decision-making processes and optimize betting strategies in our contemporary world. By incorporating state-of-the-art technologies and analytical approaches, this framework aims to empower individuals with a comprehensive toolkit to navigate the complexities of the betting landscape and achieve favorable outcomes.

1.1 Goals

The project focuses on implementing MLOps practices, including experiment tracking, model deployment, and reproducibility using an open-source tool called MLflow[1]. Therefore, users can easily track their experiments and have a clear understanding of the model's performance, enabling them to improve the model continuously. Furthermore, the project aims to facilitate the migration of computationally-intensive workflows to the Research Center for Informatics cluster[2] (RCI cluster), which can handle large amounts of data and allow for faster computations. This will lead to an improvement in the efficiency and scalability of the workflow. The project also aims to integrate betting strategies into the existing sports analysis prediction system, providing users with an additional layer of insight into the predictions made by the system.

1.2 Structure

In order to effectively achieve these objectives, this thesis is structured into four primary sections. Chapter 2 of this thesis introduces the technical terms necessary for understanding the rest of the paper. Chapter 3 explores the current capabilities of the framework, while Chapter 4 showcases examples of workflows created with the framework. The pivotal Chapter 5 presents the new features added to the framework and provides experimental evidence.

[1] https://mlflow.org/
[2] http://rci.cvut.cz/

Chapter 2
Definitions

In order to facilitate understanding of the technical terms and concepts that will be discussed in my work, I will provide definitions of key terms that will be used frequently. These definitions will help clarify the meaning and context of the various components that are used in my work, including machine learning techniques used in sports analysis and betting.

2.1 Framework

Software frameworks give programmers strong resources to create customizable, error-free programs more efficiently. Because they include critical capabilities „out of the box“, software frameworks frequently speed up the development process [4]. Software frameworks help to ensure consistency and standardization across different applications and development teams. By providing a common set of tools, libraries, and best practices, frameworks can help to promote code reusability.

2.2 Workflow

Workflow is the basic tool for automating a process or processes. A process is any action that requires a series of steps to be automated by software [5]. Workflows can include multiple steps, each of which may involve different users, systems, or data sources. By automating these steps, workflows can help to reduce errors and improve efficiency, while also providing greater visibility and control over the process. Overall, workflows are a critical tool for businesses looking to improve efficiency, reduce errors, and increase visibility and control over their processes.

2.3 Model

In machine learning, a model refers to a mathematical representation or algorithm that is trained on data to make predictions or decisions, where each model can be configured to meet the specific requirements of an application [6]. Models are a fundamental tool for machine learning and data analysis, providing a powerful way to make predictions, uncover patterns, and gain insights into complex datasets. By selecting the right algorithm and configuring the model to meet the specific requirements of an application, developers can create models that deliver accurate and reliable results, helping to drive innovation and improve business outcomes.

2.3.1 Statistical models

Statistical models are mathematical frameworks that use statistical methods to describe and analyze data. They are used to make predictions, test hypotheses, and identify relationships between variables. A statistical model typically includes a set of

assumptions about the distribution of the data, and uses these assumptions to estimate the parameters of the model [7]. Statistical models can take many forms, including linear regression models, generalized linear models, and machine learning models.

The basic idea behind the Poisson distribution in football prediction is that goals scored by a team can be considered as independent random events occurring at a certain rate. The rate parameter represents the average number of goals scored by the team per match. The Poisson distribution then provides the probabilities of different goal counts based on this rate parameter [8]. To create a football prediction model using the Poisson distribution, historical data is analyzed to estimate the average goal-scoring rates for each team.

Therefore models based on the Poisson variable can be very effective in football. We will look at two models, which will be used in my work.

- **Bivariate Poisson**

  Based on the established assumption that the goals scored follow a Poisson distribution the probability of an away and a home goal in a game i is respectively given by

$$P(X_{xi} = x) = \frac{e^{-\lambda_a}(\lambda_a)^x}{x!}; \tag{2.1}$$

$$P(Y_{xi} = x) = \frac{e^{-\lambda_h}(\lambda_h)^x}{x!}. \tag{2.2}$$

  For $x = 0, 1, 2, ...$ provided $E(x) = \lambda_a = \lambda_h = Var(x)$ (equidispersion) where x is the number of goals scored by a home team or an away team, λ_a is the intensity of away goals scored per match, λ_h is the intensity of home goals scored per match, a is the away team, h is the home team.

  The Probability of a win, draw, or loss between the away team and home team for game i with a respective expected average score of and is the product of the respective Poisson distribution described as bivariate Poisson distribution [9]

$$P(X_{ai}, Y_{hi} = x) = \frac{e^{-\lambda_a}(\lambda_a)^{x_i}}{x!} \cdot \frac{e^{-\lambda_h}(\lambda_h)^{y_i}}{y!}. \tag{2.3}$$

- **Double Poisson**

  The Double Poisson model is based on the same principles as the bivariate Poisson model, but the bivariate Poisson model provides an improved fit for data that exhibits over-dispersion or correlation between the number of goals scored by each team in a football match. It is a popular model used in sports analytics to predict the number of goals scored in a football match and estimate the probabilities of different outcomes. [10].

2.3.2 Elo rating

The Elo system was invented by Arpad Elo, a Hungarian-born American physics professor, as an improvement on the chess rating system. The system assigns a numerical

rating to each player, with the rating changing after each game based on the outcome and the rating of the opponent. The Elo system is widely used in sports analytics and has been adapted to various contexts, including team ratings and league rankings [11].

The Elo system was invented as an improvement on the chess rating system, which is now used to calculate the relative skill levels of players in various sports, such as football, basketball, and tennis [10]. The expected match outcomes for the home team and the away team are in the range of zero to one and can be calculated as follows

$$E^H = \frac{1}{1 + c^{(R^A - R^H)/d}}; \tag{2.4}$$

$$E^A = 1 - E^H, \tag{2.5}$$

where R^A is the ratings of the away team, R^H is the ratings of the home team, c and d are the metaparameters of the model.

2.3.3 Berrar

The Berrar method is a statistical approach that involves creating a classification model to predict the outcome of football matches based on various features, such as the teams' rankings, previous match results, and other relevant factors. The method was designed to provide an accurate and reliable prediction model that could be used in the sports industry to improve betting strategies or help coaches make informed decisions [12]. This method has been shown to be effective in predicting the outcome of football matches, and its success has led to its use in other sports as well.

The method was first presented as a solution to the task of the 2017 Soccer Prediction Challenge [13]. The data set provided for this challenge contained a wide range of variables related to both teams' previous performances, including the number of goals scored, shots taken, corners won, and more. The challenge was to use this data to develop accurate models that could predict the outcome of future matches.

2.4 MLOps

MLOps[1] practices are essential for organizations that are focused on scaling up their machine learning initiatives. MLOps brings together a variety of practices that help data scientists and operations teams collaborate efficiently. The MLOps process includes continuous integration and delivery, automated testing, version control, and monitoring of machine learning models in production. By using MLOps, data scientists can focus on developing and improving models, while operations teams can ensure that these models are reliable, scalable, and secure [1]. By providing a standardized framework for managing ML models, MLOps helps to streamline the entire ML development and deployment process, allowing organizations to focus on creating value through data-driven insights and actions.

[1] Machine Learning Operations

2.5 MLflow

MLflow[2] is an open-source software platform that provides a complete solution for managing the entire machine learning development cycle. It allows data scientists to efficiently manage their experiments by keeping track of all the parameters, code, and data used in a particular experiment. This makes it easier to reproduce results, collaborate with team members, and compare different models [14]. MLflow enables data scientists to streamline their machine learning workflows and focus on creating better models.

2.6 Parameters

In the field of machine learning, the term `parameters` refers to the internal variables of a model that are learned during the training process by adjusting the model to minimize the difference between its predicted output and the actual output. Once trained, these parameters are used to make predictions on new data [15].

2.7 Hyper-parameters

Hyper-parameters are different from parameters in that they are not learned from the data, but are set before training the model. Hyper-parameters determine how the learning algorithm will adapt or learn from the data, and can significantly affect the performance of the resulting model [16]. The optimal values of hyper-parameters can vary depending on the dataset and the specific learning task, so it is important to carefully tune them in order to achieve the best possible performance of the model.

2.8 Optimization

Optimization refers to the process of finding the optimal values of parameters within a model to minimize a certain cost function. These parameters are learned during the training phase using the available data. The process of optimization involves selecting an appropriate optimization algorithm.

2.9 Hyper-parameter optimization

Hyper-parameter optimization is the process of finding the optimal set of hyper-parameters for a machine learning algorithm to achieve the best performance on a given task. Hyper-parameters are adjustable parameters that are not learned from data during the training process, but rather set prior to training [17]. The optimal values of these hyper-parameters can significantly affect the performance of the machine learning model, and hyper-parameter optimization methods aim to find the best combination of hyper-parameters through a search over a predefined space of possible values.

[2] https://mlflow.org/

2.10 Metrics

In machine learning, a metric is a performance measurement used to evaluate the quality of a machine learning model. Metrics are used to assess how well the model is performing and how closely it is approximating the true function that it is trying to learn. The choice of metric is dependent on the specific problem and the goals of the project [18]. Metrics are often used to compare different models or to assess the performance of a single model over time. It is important to choose appropriate metrics that are aligned with the goals of the project and that provide a meaningful evaluation of the model's performance.

2.11 Ranked Probability Score

The ranked probability score (RPS) gives the measurement tool for a soccer match outcome by measuring the discrepancy between the predicted probabilities and the actual outcomes. It is used to evaluate the accuracy of predictive models for soccer matches and can provide insights into the reliability of these models. The ranked probability score is defined as

$$RPS = \frac{1}{r-1} \sum_{i=1}^{r-1} (\sum_{j=1}^{i} (p_j - a_i))^2, \tag{2.6}$$

where r is the number of possible outcomes (here, r=3 for win, draw, and loss), $p = (p1, p2, p3)$ is the vector of predicted probabilities for win (p1), draw (p2), and loss (p3), with p1+p2+p3=1. And $a = (a1, a2, a3)$ is the vector of the real, observed outcomes for win, draw, and loss, with a1+a2+a3=1.

2.12 Computer cluster

Computer clusters are powerful computing systems that consist of multiple interconnected computers or nodes working together as a single entity. They are designed to handle computationally intensive tasks and large-scale data processing that would be challenging for a single machine [19].

Chapter 3
Sports prediction framework summary

Developed as a master's thesis project by a CTU[1] student [20], the Sports prediction framework is a tool tailored for sports analysis. Its primary objective is to provide users with the necessary tools and resources to conduct meaningful experiments, enhance their predictive models, and make data-driven decisions with confidence.

The framework is built on top of the Python programming language, which makes it accessible and easy to use for both novice and experienced users. Moreover, the framework leverages the latest machine learning techniques to enable users to build accurate and robust predictive models that can be used to analyze and predict the outcomes of sporting events.

At the core of the Sports prediction framework is the creation and running of workflows, which enable users to specify the parameters and models for their experiments and generate predictions based on the data provided. These workflows are designed to be modular and customizable, which means that users can easily modify and adapt them to suit their specific needs and requirements. Moreover, the framework provides users with a range of evaluation metrics, such as the Rank Probability Score (RPS) and accuracy, which can be used to assess the performance of their models and optimize their strategies.

In my work, I will focus on extending the capabilities of the Sports prediction framework by integrating the experiment tracking system and betting strategies [3], as well as by optimizing the workflow for scalability and performance using the computer cluster. To achieve this, I will need to provide a detailed description of the framework's purpose, functionality, and components, as well as develop new modules and workflows that can be integrated seamlessly into the existing framework.

3.1 Framework functionality

The Sports prediction framework allows the user to create their unique workflows depending on the selected sport and the model to be used in the calculation. An important feature of these workflows is the convenient way to make modifications to them. For example, to enable the output of the input parameters on the command line, it's sufficient to use an already implemented function called *print_parameters*, which works for any of the new workflows.

The data stored in the PostgreSQL[2] database can be used not only for training machine learning models but also for performing data analysis and generating insights. For example, a workflow created in this framework can use a table with over 200,000 records of football matches since the year 2000. The Sports prediction framework supports the use of XGBoost machine learning algorithm and various rating and statistical models, including the Elo Rating system, Double Poisson, Bivariate Poisson, and others.

[1] Czech Technical University
[2] https://www.postgresql.org/

The workflow created by the Sports prediction framework is composed of several blocks that work together to produce predictions and evaluation metrics. By breaking down the workflow into these individual blocks and understanding the order in which they are executed, we can gain a better understanding of the inner workings of the Sports prediction framework and how it can be used to generate accurate predictions for sports analysis and betting.

3.2 Core blocks of the Sports prediction framework

The framework comprises several core blocks, each serving a specific purpose in the betting process. These blocks include data loading, data transforming, model training, prediction generation, betting optimization, and evaluation. The order in which these blocks are executed is critical to the success of the workflow, as each block depends on the output of the previous block.

This chapter will explore each of these blocks in detail, examining their role in the framework and discussing best practices for their implementation.

In the diagram below you can see the order in which each block is executed and the relationship between the blocks in the Sports prediction framework.

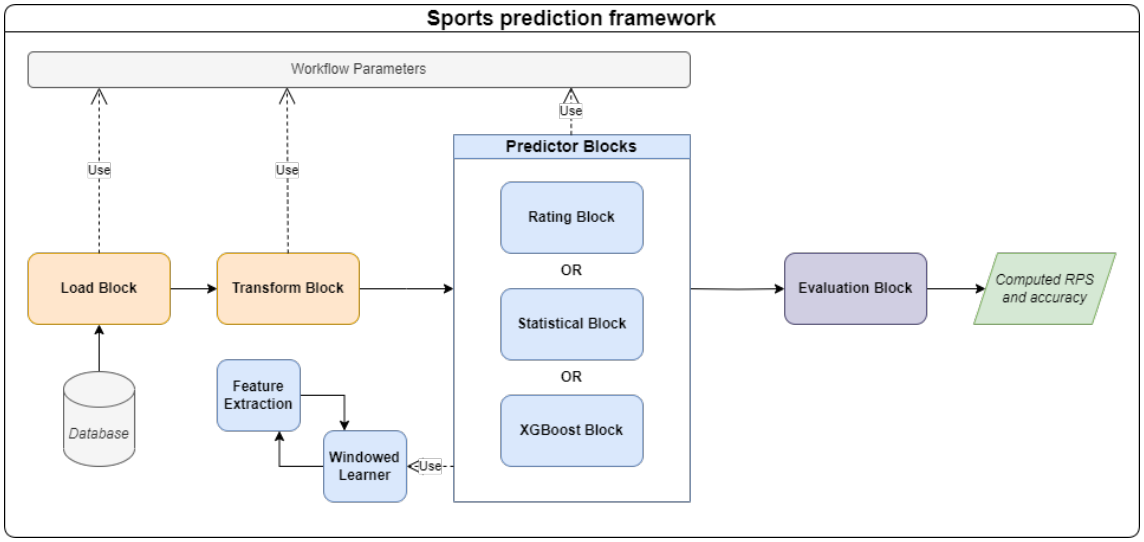

Figure 3.1. The Sports prediction framework workflow diagram

This section will provide a comprehensive overview of each block that constitutes the Sports prediction framework. Each block plays a crucial role in the overall functionality of the system, and understanding how they work together is essential for developing a successful and profitable betting strategy. We will explore each block's specific features and capabilities, as well as their interdependence, to give a complete picture of the framework. By the end of this chapter, you will have a solid understanding of the components that comprise the Sports prediction framework and be well-equipped to design and implement your own workflow.

3.2.1 Load Block

The Load Block is a crucial component in the Sports prediction framework, designed to retrieve input data from various external sources and load them into the framework for processing. One of the primary uses of the Load Block is to extract data from a PostgreSQL database that houses vast amounts of information related to various sports, including football, hockey, baseball, and many others.

In particular, the database for football contains a wealth of data such as the date of the match, the teams that played the game, and the final result. The Load Block reads this information and organizes it in a way that is easily accessible to other components of the framework. By using the Load Block to gather data from the PostgreSQL database, users can ensure that they have access to the most up-to-date information.

3.2.2 Transform Block

The Transform Block is used to preprocess and transform the data retrieved by the Load Block, making it more suitable for training machine learning models. This block applies various data transformations, such as cleaning, feature engineering, normalization, and scaling.

For example, the Transform block can filter raw input data from the Load block by date. It can also convert the Time and Season columns, which are written in the format *dd.mm* and *yyyy*, respectively, to the Date column in the much more usable format *dd.mm.yyyy*. It can also remove missing or irrelevant data, perform feature selection, and derive new features from the existing ones.

Overall, the Transform Block plays a critical role in preparing the data for machine learning algorithms. Once the Transform Block has processed the data, the resulting transformed data can be passed on to the next stage of the pipeline.

3.2.3 Predictor Blocks

Predictor Blocks are used to train and generate predictions using machine learning models. These blocks take data from the Transform Block as input and use it to train a model. The trained model is then used to generate predictions for future data.

The predictor blocks used in the Sports prediction framework are Rating Block, Statistical Block, and XGBoost Block. The Rating Block generates team or player ratings based on historical performance, while the Statistical Block uses statistical models to analyze data and make predictions. The XGBoost Block is a machine learning algorithm that uses gradient boosting to make predictions. All of these blocks can be used to create models for predicting the outcome of sporting events.

Each predictor block in the Sports prediction framework can use a variety of models to generate predictions. For instance, the Rating Block in the framework uses the Elo and Berrar models to generate ratings and probabilities for different events. Similarly, the Statistical Block uses models such as Double Poisson and Bivariate Poisson to make predictions based on statistical analysis of historical data. The XGBoost Block uses a gradient boosting framework to train models on large datasets.

3.2.4 Evaluation Block

The Evaluation Block in the Sports prediction framework is used to evaluate the performance of the prediction model by comparing its output to the actual outcomes of the events being predicted. This block typically receives the output from the Predictor Block and compares it to the actual results, which are usually obtained from the Load Block.

It uses accuracy and Rank Probability Score (RPS) to evaluate the performance of the predictive model. The accuracy measures how often the model's predictions are correct, while the Rank Probability Score measures how well the model's predicted probabilities match the actual probabilities. These evaluation metrics help determine the effectiveness and reliability of the predictive model and can guide further improvements and optimizations.

3.2.5 Workflow Parameters File

In the Sports prediction framework, the Workflow Parameters File is a configuration file that specifies the parameters and settings of the entire workflow. This file allows the user to customize the workflow and adjust its behavior according to specific requirements.

The Workflow Parameters File 3.2 is in a JSON format and contains different sections that define the parameters for each step of the workflow, such as the Load Block, Transform Block, Predictor Block, and Betting Optimization Block.

```
{
    'load_block': {
        ...
    },
    'transform_block':{
        ...
    },
    'statistical_block': {
        'learner':{
            'data_selector': {
                'train_selectors': {
                    ...
                },
                'test_selectors': {
                    ...
                }
            }
        }
    }
}
```

Figure 3.2. Example of the structure of the Workflow Parameters file

Some of the parameters that can be specified in the Workflow Parameters File include the data source location, the machine learning algorithms to use and the hyper-parameters to tune.

Using a Workflow Parameters File allows for a more streamlined and efficient workflow as it eliminates the need for the user to manually specify the same parameters for each step of the workflow. Additionally, it provides a level of flexibility and customization, as the user can easily modify the parameters in the file to experiment with different settings and configurations.

3.3 Hyper-parameter optimization

After selecting the most suitable models for the workflow, the next step is to determine which model will perform the best for the specific task. This is where hyper-parameter

optimization comes in as the solution to this problem. Hyper-parameter optimization is the process of finding the best combination of hyper-parameters for a machine learning algorithm that produces the best results.

Automatic hyper-parameter optimization significantly improves the performance of the machine learning algorithm. [21] There are a variety of frameworks for such purposes, but we will be considering two of them, which are the Optuna[3] and the Ray Tune[4] optimization frameworks. Through the use of these frameworks, the performance of the selected models can be further improved.

3.3.1 Optuna optimization framework

Optuna is a hyper-parameter optimization framework that defines the process of hyper-parameter optimization as a way of finding the best set of hyper-parameters that maximize or minimize a given function [22]. The process involves creating a study, which is a container for the optimization process, and then evaluating an objective function over multiple trials. The results of each trial are recorded, and the framework uses these results to guide the selection of the next set of hyper-parameters to evaluate[23].

In Optuna, each trial is a single execution of the objective function with a specific set of hyper-parameters. The framework tries to minimize the objective function by iteratively selecting new hyper-parameters for each trial based on the results of previous trials. The result of optimizing with Optuna is a study that contains the best set of hyper-parameters found during the optimization process, as well as the results of all trials conducted.

The number of trials that Optuna will run can be specified by the user using the n-trials parameter. This parameter controls the number of trials that will be executed during the optimization process. By adjusting the value of n-trials, the user can balance the trade-off between the quality of the hyper-parameter selection and the computational resources required for the optimization process. Overall, Optuna is a powerful tool for optimizing hyper-parameters in machine learning models, and its flexibility and ease of use make it a popular choice for many data scientists and machine learning engineers.

3.3.2 Ray Tune optimization framework

Ray Tune is a hyper-parameter optimization framework that uses the concept of a trial, which is defined as a single training cycle with a resolved initial configuration of hyper-parameters, similar to Optuna [24]. In Ray Tune, an experiment is a collection of trials that is managed by Tune using one of the task scheduling algorithms, such as HyperBand, Asynchronous Successive Halving Algorithm, and Population Based Training.

Compared to Optuna, Ray Tune is a more comprehensive optimization tool that offers a wide range of customization options for each trial. For example, Ray Tune supports distributed training across multiple nodes, and it provides various built-in schedulers that can terminate poorly performing trials early to speed up the overall optimization process.

Furthermore, Ray Tune allows users to define custom search spaces for hyper-parameters, including both continuous and categorical variables. Overall, Ray Tune is a powerful tool for hyper-parameter optimization that can significantly improve the performance of machine learning algorithms [25].

[3] https://optuna.org/
[4] https://docs.ray.io/en/latest/tune/index.html

Chapter 4
Available workflows of the Sports prediction framework

4.1 Workflows structure

Most of the workflows in the framework are similar in structure, which allows the user to easily switch between the different workflows, understand their purpose and easily make the required changes. The structure of a workflow in the Sports prediction framework typically consists of the following blocks:

- **Load Block**
  Reads input data from external sources, such as a database.

- **Transform Block**
  Filters and transforms raw input data from the Load Block, preparing it for analysis.

- **Predictor Block**
  Contains statistical and rating models used for predicting outcomes.

- **Evaluation Block**
  Evaluates the performance of the predictor models using metrics such as accuracy and Rank Probability Score (RPS).

4.2 Basic workflows

In order to enhance our comprehension of the workflow in the Sports prediction framework, we will take a closer look at several fundamental workflows that have already been implemented using the framework.

4.2.1 Rating

The Rating workflow is a simple example of the workflow created using the Sports prediction framework. It is aimed at making predictions based on rating models like Elo or Berrar. Each block in the Rating workflow loads its parameters separately from the *ratingParams* file. The *Parameters* class constructor is used to initialize the parameters with the values from the file. The *ratingParams* file is written in JSON[1] format for easy readability and convenience.

For example, the *LoadBlock* class, responsible for creating a wrap from the database, has its own set of parameters defined in the *ratingParams* file 4.1. These parameters

[1] JavaScript Object Notation.

include the filter for the table, the class that will wrap the data, and the name of the database. In the case of the Rating workflow, the *FootballWrapper* class is used to obtain all football match results from the *Bundesliga*[2] using the *bet* database.

```
'load_block': {
        'filter':Equal("Lge", "GER1"),
        'wrapper':{
            'class':FootballWrapper,
            'Football':{
                'football':{
                    'PSQLSchemaName':"isdb",
                    'DB_name':"bet"
                }
            }
        }
    }
```

Figure 4.1. Parameters for the LoadBlock

Similarly, the *TransformBlock* class has its own set of parameters defined in the *ratingParams* file. These parameters 4.2 determine how the data wrap from the previous step will be transformed. In the case of the Rating workflow, an identifier is added for each football team to enable further analysis.

```
'transform_block':{
        'names_to_ids':True
    }
```

Figure 4.2. Parameters for the TransformBlock

The *RatingBlock* class has its own set of parameters defined in the *ratingParams* file. These parameters 4.3 determine how the main computation block will operate. The compute function takes the transformed data wrap as input and returns a column with the probabilities calculated by the block.

```
'rating_block': {
        ...
        'learner': {
            'data_selector': {
                ...
            }
        }
    }
```

Figure 4.3. Parameters for the RatingBlock

[2] The Bundesliga is a professional association football league in Germany.

Finally, by sending the resulting column of probabilities to the *compute* function of the *EvaluationBlock* class, we will get the output accuracy and the Rank Probability Score (RPS) for the model involved.

In summary, each block in the Rating workflow loads its parameters separately from the ratingParams file, and these parameters are used to determine how the block operates within the workflow.

Presented below is a code representation 4.4 of the steps involved in the Rating workflow.

```
params = Parameters(params=ratingParams.params)
wrap = LoadBlock(parameters=params).compute()
wrap = TransformBlock(parameters=params).compute(wrap)
rating = RatingBlock(parameters=params)
prob = rating.compute(wrap)
metrics = EvaluationBlock().compute(prob)
```

Figure 4.4. Rating workflow steps

4.2.2 Statistical

The Statistical workflow, which is another basic workflow in The Sports prediction framework, shares similar functionality with the Rating workflow. However, instead of using the Elo and Berrar models, it solely utilizes statistical methods such as the Bivariate Poisson or Double Poisson models. Users have the ability to select the desired model and its corresponding parameters by modifying the 'model' variable within the *statisticalParams* class. In this way, the Statistical workflow provides flexibility in terms of model selection, as users can choose the most appropriate statistical model for their specific use case.

```
'statistical_block': {
        ...
        'learner':{
            'model':{'class': BivariatePoisson}
            ...
        }
        ...
    }
```

Figure 4.5. Statistical workflow model

4.2.3 Optimization

In addition to the workflows, which perform computations based on a model specifically defined by the user, it was necessary to create optimizing workflows that attempt to find the best possible model or other hyper-parameters for the given data set. For this purpose, there is the Optimization workflow that is based on the principles of Hyper-parameter optimization. It is possible to use one of the two previously mentioned workflows, either the Optuna or the Ray Tune.

For the workflows that use the hyper-parameter optimization, the models to be compared must be specified in advance. In the example of Optimization workflow, it can be done in the *ratingParams* file provided in JSON format. This can be done by changing the *hyper_parameters* variable. The following example 4.6 shows that two models will be compared, the EloRating model and the Berrar model.

```
hyper_parameters = {
        'rating_block': {
            'learner': {
                'feature_extraction':{
                    'model': {
                        'values': [
                            {'class': EloRating},
                            {'class': Berrar}
                            ]
                    }
                }
            }
        }
    }
```

Figure 4.6. Optimization workflow hyper-parameters

Below are the steps of the Optimization workflow 4.7 in the form of code.

```
params = Parameters(params=ratingParams.params)
hyper_parameters = ratingParams.hyper_parameters
wrap = LoadBlock(parameters=params).compute()
wrap = TransformBlock(parameters=params).compute(wrap)
rating = RatingBlock(parameters=params)

#OPTUNA
result = OptunaMetaOptimizer(rating, hyper_parameters).compute(wrap)

#RAY TUNE
result = RayTuneMetaOptimizer(rating, hyper_parameters).compute(wrap)
```

Figure 4.7. Optimization workflow steps

4.3 Advanced workflows

In the previous part, the most basic of the trivial workflows were demonstrated, with which the process of creating and running each workflow could be investigated. The following section will include a few more complex and advanced workflows, which are all based on the same principles as the previously shown examples to some extent. These workflows are more sophisticated, requiring additional time to complete their computations due to their complexity. Despite their greater complexity, these workflows provide more in-depth and refined data analysis, which is critical for a variety of applications.

4.3.1 PoissonFootball

The PoissonFootball and Rating workflows share many similarities in their basic structure, but there are some fundamental differences that set them apart.

The most significant difference between the two workflows is the choice of the model used for calculations. Rating workflow relies on Elo or Berrar methods to predict outcomes, while PoissonFootball uses the Double Poisson model instead. All these configurations 4.8, including the selection of data and model used, can be viewed in the *poissonParams* file.

The training method of this workflow utilizes a specific range of data between the dates of 2000-08-11 and 2005-08-10. On the other hand, the test method uses a different set of data that is only taken from the date 2005-08-11 and onwards. This ensures that the trained model is tested with data that it has not previously seen, thus providing a more accurate evaluation of the model's performance on unseen data.

```
'learner':{
        'model':{'class': DoublePoisson},
        'data_selector': {
            'train_selectors': {
                'window_scope': {
                    'col': "Date",
                    'start': extract_ordinal("2000-08-11"),
                    'size': extract_diff("2000-08-11","2005-08-10"),
                    'stride': extract_delta("365days")
                }
            },
            'test_selectors': {
                'window_scope': {
                    'col': "Date",
                    'start': extract_ordinal("2005-08-11"),
                    'size': extract_delta("365days"),
                    'stride': extract_delta("365days")
                }
            }
        ...
```

Figure 4.8. PoissonFootball workflow parameters

4.3.2 XGBoost

XGBoost[3] is a scalable and effective implementation of Friedman's (2001) gradient boosting methodology [26]. It provides the ability to parallelize the computation on Windows but also on Linux [27]. That is a big advantage for our project as we will be using a cluster on which we will run these types of workflows.

The XGBoost workflow follows a similar process as other basic workflows. After loading and initializing the parameters from the *xgboostParams* file, the wrapping and

[3] The abbreviation XGBoost stands for eXtreme Gradient Boosting package

transformation phase follows, as with any other basic workflow. In this case, however, a more complex transformation is used, where in addition to converting names to identifiers, several wrappers are merged.

Once the wrappers are connected, the main computational block, which is XGBBlock, is initialized according to the parameters specified in the *xgboostParams* file.

The individual configuration for XGBoost can be found in the file *xgboostParams*. It includes separate sections for setting the parameters used in the optimization process with Optuna and Ray Tune. It is possible to set various parameters that affect the behavior of the hyper-parameter optimization process. For example, the number of trials, which determines how many combinations of hyper-parameters will be tested during the optimization process. Additionally, the type of metric to optimize for, the searching algorithm to use, and the scheduler, which determines how resources are allocated during the optimization process, can all be specified in the configuration file.

As an example below you can see the current configuration 4.9 of the workflow, where the number of trials for Optuna and Ray Tune optimization process is 100, and the metric is Rank Probability Score (RPS).

```
'xgblock': {
     'optimization':{
        'score_function':compute_RPS,
        'parameters':{
            'ray_tune': {
                'num_samples': 100,
                'metric': "compute_RPS",
                'mode': "min",
                'search_alg': OptunaSearch(),
                'scheduler': ASHAScheduler(),
                'resources_per_trial': {"cpu": 1, "gpu": 0},
            },
            'optuna': {
                'n_trials': 100,
                'direction': "maximize"
            }
        }
    ...
```

Figure 4.9. XGBoost workflow parameters

The XGBoost workflow is particularly powerful thanks to its ability to optimize its hyper-parameters using the advanced techniques provided by the Ray Tune and Optuna workflows.

Hyper-parameters can be set in the *xgboostParams* file 4.10, which can be modified by the user to specify the ranges of values to be explored during the tuning process. By using the configuration file, the user can set constraints on the search space, such as limits on the number of trials, or the ranges of values to be explored for each hyper-parameter. In this way, the hyper-parameters can be tuned efficiently, and the resulting model can achieve better accuracy or other performance metrics.

```
hyper_parameters={
    'xgblock': {
        'learner': {
            'model': {
                'xgboost': {
                    'xgb_params': {
                        'learning_rate': {
                            'lb':0,
                            'ub':1,
                            'precision':0.01
                        },
                        'n_estimators': {
                            'lb':10,
                            'ub':10000,
                            'precision':1
                        },
                        ...
```

Figure 4.10. XGBoost workflow hyper-parameters

Below are the steps of the **XGBoost** workflow 4.11 in the form of code.

```
params = Parameters(params=xgboostParams.params)
hyper_parameters = xgboostParams.hyper_parameters
wrapper = LoadBlock(parameters=params).compute()
wrapperids = NamesToIds(parameters=params).compute(wrapper)
wrapperrnd = RoundFeature().compute(wrapper)
wrapper = Merger().compute([wrapperrnd,wrapperids])
xgbf = XGBFeaturesBlock()
wrapper = xgbf.compute(wrapper)
xgb = XGBBlock(parameters=params)
analysis = RayTuneMetaOptimizer(xgb, hyper_parameters).compute(wrapper)
```

Figure 4.11. XGBoost workflow steps

Chapter 5
Framework for analyzing efficiency of predictive sports markets

In this chapter, I will provide a comprehensive overview of the significant changes and improvements that I have implemented in the Sports prediction framework. These changes have brought about a substantial transformation in the project, introducing new functionality and enhancing its capabilities. As a result, the framework has been given a new name to reflect its expanded scope and focus, now known as the `Framework for analyzing efficiency of predictive sports markets.`

These new features allow users to optimize their betting strategies, improve the accuracy of their predictions, streamline the entire betting process, and easily log and track the values of various parameters used in the training and evaluation processes.

We will explore some of the latest additions and how they can be used to achieve better outcomes in the world of sports betting. We will discuss the implementation of logging parameters using MLflow, the introduction of new blocks for betting strategies optimization, and a major refactoring of the project to improve its overall structure and code maintainability. Additionally, we will delve into the utilization of external computing resources through the RCI cluster to handle complex and time-consuming workflows.

5.1 New blocks added to framework

Compared to the Sports prediction framework, the current version of the framework introduces two new blocks, namely the Betting Optimization Block and the Profit Evaluation Block, alongside the existing core blocks: Load Block, Transform Block, Predictor Block, and Evaluation Block.

This chapter will explore each of these new blocks in more detail, examining their role in the framework.

In the diagram 5.1 you can see the order in which each block is executed and the relationship between the blocks in the Framework for analyzing efficiency of predictive sports markets.

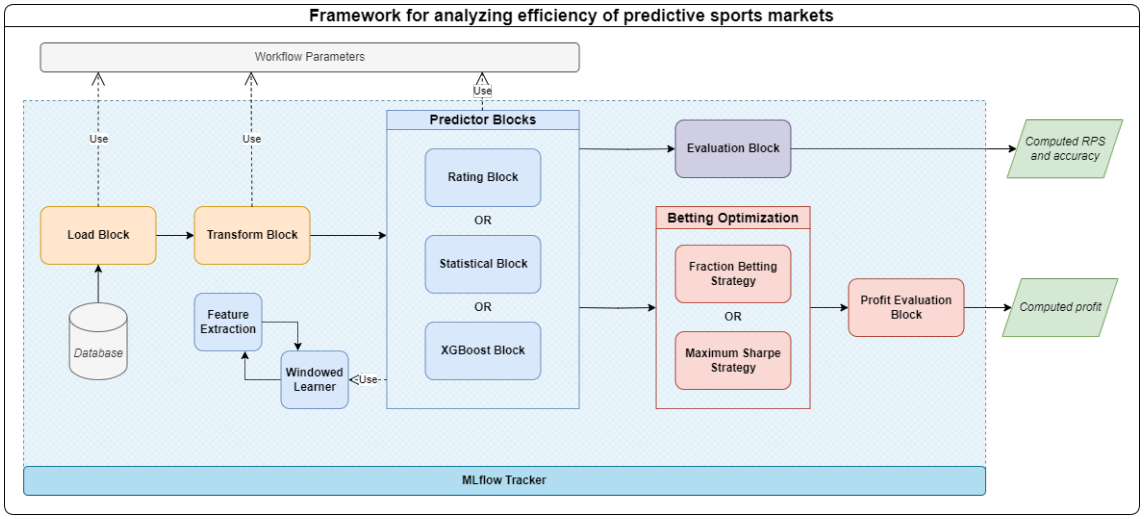

Figure 5.1. The Framework for analyzing efficiency of predictive sports markets workflow diagram

5.1.1 Betting Optimization Blocks

The Betting Optimization Blocks aim to optimize the betting strategy. This block receives the predictions from the Predictor Block and uses them to generate bets with the aim of maximizing the return on investment.

This type of block takes into account various factors such as the available budget, odds offered by bookmakers, and the predicted probability of the event occurring. The block uses these inputs to determine the optimal stake to place on each bet, considering factors such as risk tolerance and expected return.

There are two techniques used in the Betting Optimization Block to determine the optimal betting strategy, each of which will be described in the section below.

Fraction Strategy Block

Fraction Betting Strategy is a betting technique that involves placing wagers on the outcome of a particular event while using a fractional amount of the total bankroll for each bet. The size of each wager is determined as a fraction of the remaining bankroll, which allows for a more conservative approach to betting and reduces the risk of significant losses.

The idea behind this strategy is to avoid betting too much of the available bankroll at any one time, which could lead to substantial losses if a bet is unsuccessful. By placing smaller, fractional bets, the overall risk is reduced, and a losing streak will have less of an impact on the remaining bankroll [28].

For example, if a bettor has a bankroll of $1000 and decides to use a fraction of 1/10, then the maximum wager for the first bet would be $100. If the first bet is unsuccessful, the remaining bankroll would be $900, and the maximum wager for the second bet would be $90 (1/10th of $900). This process continues, with the maximum wager for each subsequent bet calculated as a fraction of the remaining bankroll.

The Fraction Betting Strategy is often used by novice bettors or those who prefer a more conservative approach to betting.

- **Max Sharpe Strategy Block**

  The Max Sharpe betting strategy is a popular method used in sports betting to maximize profits. It involves calculating the Sharpe ratio for each available bet and selecting the one with the highest ratio.

  The Sharpe ratio is a measure of the risk-adjusted return of an investment. In sports betting, it is calculated by dividing the expected return on a bet by the standard deviation of the expected return. A higher Sharpe ratio indicates a better risk-adjusted return [29].

  To use the Max Sharpe betting strategy, a bettor first estimates the probability of each possible outcome of a sporting event. They then convert these probabilities into odds using an appropriate formula, such as the implied probability formula. Next, they calculate the expected return for each available bet by multiplying the odds by the estimated probability and subtracting 1. Finally, they calculate the standard deviation of the expected return using historical data or statistical models [10].

  Once the expected returns and standard deviations have been calculated for each available bet, the Sharpe ratio can be calculated. The bet with the highest Sharpe ratio is selected as the optimal bet to place.

5.1.2 Profit Evaluation Block

The Profit Evaluation block in the Framework for analyzing efficiency of predictive sports markets is used to evaluate the profitability of a betting strategy. This block takes as input the predicted probabilities of the outcomes of sporting events, along with the actual outcomes of those events and the betting odds offered by bookmakers.

Using this information, the Profit Evaluation block calculates the profit or loss that would result from using a given betting strategy.

The Profit Evaluation block can be used to evaluate the profitability of a range of different betting strategies, including the Fraction Betting Strategy and the Max Sharpe Betting Strategy. By comparing the profitability of different strategies, it can help bettors to identify the most profitable approach to use when betting on sporting events.

5.2 Framework on a cluster

Some workflows may require more time and resources to complete their computations due to several reasons. One reason could be the quantity of data to be processed, which may be large and complex, requiring more time and computational resources to process.

In high-performance computing environments, it is common to use a batch scheduler to allocate and manage resources for computational tasks. One popular batch scheduler is the Simple Linux Utility for Resource Management (SLURM).

Therefore, it was decided to run more complex workflows on a cluster, for which the Research Center for Informatics cluster[1] (RCI cluster) was chosen. To execute the framework on a cluster, the functionalities of the Ray Tune library can be utilized.

Ray Tune is a distributed hyper-parameter tuning library that allows for efficient experimentation and optimization of machine learning models. It is a powerful tool for running complex workflows on a cluster, as it can handle the parallelization of tasks across multiple nodes, reducing the time required to run the workflows.

[1] http://rci.cvut.cz/

By using Ray Tune in conjunction with the RCI cluster, the Framework for analyzing efficiency of predictive sports markets is able to take advantage of the additional computing resources provided by the cluster. Ray Tune is able to distribute the hyper-parameter tuning jobs across multiple nodes, allowing for faster processing of large datasets and more efficient optimization of machine learning models.

To run the framework on a cluster using Ray Tune, a virtual environment is created from the *environment-cluster.yaml* file using the Anaconda[2] distribution platform. The workflow is then configured to use Ray Tune as the hyper-parameter tuning library, and the parameters for the tuning process are specified in a configuration file.

Ray Tune provides the ability to monitor the resource usage and performance of each node in the cluster during the execution of a workflow. It offers a dashboard 5.2 that allows users to visualize the progress of their training jobs. This can be useful for identifying any issues or bottlenecks in the workflow and optimizing resource allocation to improve overall performance.

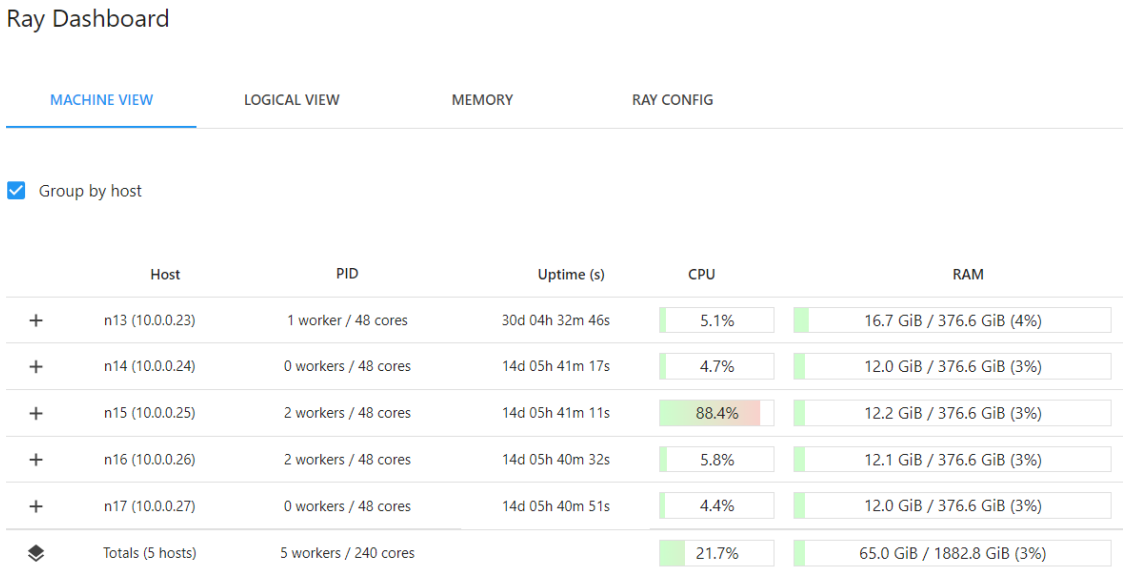

Figure 5.2. Ray Tune dashboard for cluster

Overall, by using Ray Tune for cluster running, the Framework for analyzing efficiency of predictive sports markets is able to take advantage of the power and efficiency of distributed computing, resulting in faster and more accurate predictions and more efficient optimization of betting strategies.

5.2.1 SLURM submission script

The Ray framework does not provide pre-written scripts for running programs on the cluster. Therefore, I have personally authored all the scripts provided below to accommodate the specific requirements of each case.

To run a workflow using the Framework for analyzing efficiency of predictive sports markets on a cluster, one can use the script 5.3. I have specifically designed this script for running complex jobs on external computing resources such as clusters.

The script requires the user to provide several key parameters. These include the partition of the cluster to be used, the number of nodes to be allocated from the

[2] https://www.anaconda.com/products/distribution

partition, the number of CPU[3] cores to be allocated for each job, and the amount of RAM[4] to be allocated for each CPU. The values of these parameters will depend on the specific workflow being executed and the available resources on the cluster.

In addition to these parameters, the user must also provide some additional information to the script. This includes the name of the file where the desired workflow is implemented, the path to this file, the name of the cluster user who will execute the workflow, and the name of the virtual environment that should be used for the workflow.

Once these inputs have been provided, the script can initiate the execution of the workflow on the cluster.

The parameters related to the allocation of computing resources are specified using the #SBATCH. On the other hand, the parameters that relate to the start of a particular workflow are specified as bash script arguments. These arguments are passed to the script when it is executed.

```
#!/bin/bash

...
#SBATCH --partition=cpu
#SBATCH --nodes=2
#SBATCH --exclusive
#SBATCH --cpus-per-task=2
#SBATCH --mem-per-cpu=2GB
...

. ray_cluster.sh \
    --python_runfile="XGBoost.py" \
    --runfile_dir="BT/sports-pipeline-framework/workflows/advanced/" \
    --username="oganetig" \
    --conda_env="sports-pipeline-framework"
```

Figure 5.3. SLURM job starting script

By separating the parameters related to cluster resources and workflow start, it becomes easier to manage and modify the execution of workflows on a cluster. It allows for greater flexibility and control over the allocation of resources, while also making it easier to customize the workflow execution based on specific needs and requirements.

5.2.2 Ray cluster launch script

The script mentioned in the text is called *ray_cluster.sh*. It is used to create a Ray cluster and run the workflow on it. The script I have developed is intended to run on a cluster of nodes that are under the management of SLURM, a widely used job scheduling system in high-performance computing setups.

By leveraging Ray, the Framework for analyzing efficiency of predictive sports markets can distribute computations across multiple nodes and take advantage of parallelism to speed up the training of machine learning models.

[3] Central processing unit
[4] Random-access memory

The script contains several code blocks that perform different functions such as parsing input parameters, getting the allocated nodes from SLURM, starting the head node and worker nodes, launching the workflow, and waiting for the job to end. In the following sections, we will describe the main code blocks of the *ray_cluster.sh* file in more detail.

- **Input parameters parser**

  The *ray_cluster.sh* file receives several input parameters that are used to define the execution environment for the workflow. These parameters include the name and path of the workflow file, the name of the user, and the name of the virtual environment. These parameters are crucial for setting up the execution environment and ensuring that the workflow is executed correctly. By parsing the input parameters, the script can extract the necessary information to launch the Ray cluster and run the workflow on it.

```
#!/bin/bash
...
for argument in "$@"
do
  key=$(echo ${argument} | cut -f1 -d=)

  key_length=${#key}
  value="${argument:${key_length}+1}"

  declare ${key#"--"}=${value}
done
...
```

Figure 5.4. Input parameters parser

- **Allocated nodes information obtaining**

  The script obtains the names of the nodes that are allocated for the job from SLURM. By obtaining this information, the script can ensure that the Ray cluster is launched on available nodes and avoid potential conflicts with other jobs running on the same cluster.

```
...
nodes=$(scontrol show hostnames "$SLURM_JOB_NODELIST")
nodes_array=($nodes)
node_1=${nodes_array[0]}
...
```

Figure 5.5. Allocated nodes information obtainment

IP address obtainment and conversion

After obtaining the names of the nodes allocated for the job from SLURM, the script retrieves the IP address of the first node that is allocated for the job. Since Ray requires an IPv4 address to function correctly, the script converts the IPv6 address of the first node into an IPv4 address. This conversion is necessary for Ray to properly communicate between nodes and function correctly in the cluster environment.

```
...
ip=$(srun --nodes=1 --ntasks=1 -w "$node_1" hostname --ip-address)

if [[ "$ip" == *" "* ]]; then
  echo "$ip" | IFS=' ' read -ra ADDR
  if [[ ${#ADDR[0]} -gt 16 ]]; then
    ip=${ADDR[1]}
  else
    ip=${ADDR[0]}
  fi
fi
...
```

Figure 5.6. IP address obtainment and conversion

Ray cluster head node initialization

The next step in the *ray_cluster.sh* script is to start the head node for the Ray cluster on the first allocated node. This head node acts as a central point of communication for all the worker nodes and is started on port 6379, which is the default port for Ray. The head node is responsible for coordinating the execution of tasks across the worker nodes and ensuring fault-tolerance in case of any worker node failures.

```
...
port=6379
ip_head=$ip:$port
export ip_head

srun --job-name="ray-head" --nodes=1 --ntasks=1
     --cpus-per-task=$(SLURM_CPUS_PER_TASK-1) -w "$node_1" --pty bash \
     conda-run.sh "$conda_env" \
     "ray start --head --num-cpus=$(SLURM_CPUS_PER_TASK-1)
     --node-ip-address=$ip --port=$port --block" &
sleep 10
...
```

Figure 5.7. Ray cluster head node starting

The *conda-run.sh* script is responsible for setting up and activating the specified virtual environment using the *conda_env* parameter. It ensures that the necessary dependencies and packages are installed within the virtual environment before executing the desired tasks.

- **Ray cluster worker nodes starting** The next step in the script is to start the worker nodes on the remaining allocated nodes. These worker nodes are responsible for performing distributed computing tasks and connecting to the head node that was started earlier.

  This process allows for parallelization of the workflow and increased efficiency in executing the tasks. Each worker node is assigned a subset of the tasks and communicates with the head node to receive instructions and share results.

```
...
worker_num=$(SLURM_JOB_NUM_NODES - 1)
for i in $(seq $worker_num); do
 node_i=${nodes_array[$i]}
 echo "STARTING WORKER $i at $node_i"
  srun --nodes=1 --ntasks=1 -w "$node_i" --pty bash \
   conda-run.sh "$conda_env" \
   "ray start --address=$ip_head --block
    --num-cpus=$SLURM_CPUS_PER_TASK" &
  sleep 5
done
...
```

Figure 5.8. Ray cluster worker nodes starting

- **Ray cluster worker nodes starting** The export commands set environment variables RAY_ADDRESS and RAY_ADDRESS_IP, which enables Ray to be initialized in Python code using *ray.init()* without specifying an address. The script launches the specified workflow using the Ray cluster as the backend for distributed computing. Then script waits for the workflow to finish before terminating the Ray cluster and ending the job.

```
...
RAY_ADDRESS=$ip_head
RAY_ADDRESS_IP=$ip
export RAY_ADDRESS
export RAY_ADDRESS_IP

python -u ${runfile_dir}${python_runfile}

srun --nodes=1 --ntasks=1 --cpus-per-task=1 -w "$node_1" \
    conda-run.sh "$conda_env" \
    "ray stop" &

wait
```

Figure 5.9. Ray cluster job starting

5.3 MLfLow integration

In the context of MLOps, keeping track of experiments is crucial for understanding and improving the performance of machine learning models. To accomplish this, the Framework for analyzing efficiency of predictive sports markets incorporates the use of MLflow. By using MLflow, our framework enables the logging of experiment parameters, hyperparameters, and metrics, which can be used to analyze and compare different models.

By default, MLflow comes with a local server that can be used to store logs and track experiments. However, in the context of the Framework for analyzing efficiency of predictive sports markets, a separate server[5] is used instead.

The use of a dedicated server for logging and monitoring the progress of workflows provides several advantages. Firstly, it allows for better management of experiments as multiple users can access the same logs simultaneously. This facilitates collaboration and enables teams to work more efficiently. Additionally, using a centralized server enables real-time monitoring of the workflow progress, making it easier to detect and address issues quickly. Finally, the use of a separate server also helps to ensure the security and privacy of the logs and experimental data, as access can be restricted to authorized personnel only.

The decision to use a separate server for MLflow in the Framework for analyzing efficiency of predictive sports markets reflects the framework's commitment to providing a robust and scalable platform for machine learning development and deployment.

5.3.1 MLflow configuration file

Users can specify the configuration of their MLflow runs by editing the *mlflow_config.yaml* file 5.10 called in their project directory. This file can be used to specify the experiment name and the tracking URI to be used by MLflow.

The experiment name is used to group together related runs in MLflow. By default, if an experiment with the specified name does not exist, MLflow will create a new one. The tracking URI specifies the location where the logs and artifacts should be stored. In the Framework for analyzing efficiency of predictive sports markets, this URI is set to the Potato server used for experiment tracking.

```
...

EXPERIMENT_NAME: Betting framework - oganetig
TRACKING_URI: http://localhost:1111
```

Figure 5.10. MLflow server configuration file

By using the *mlflow_config.yaml* file, users can easily manage their MLflow experiments and track their workflow runs in a centralized manner.

5.3.2 Filter for MLflow

The *mlflow_config.yaml* file in the Framework for analyzing efficiency of predictive sports markets allows the user to specify a blacklist or a whitelist for prefixes, suffixes,

[5] https://potato-server.readthedocs.io/en/latest/ida-server.html

and postfixes to filter the parameters and metrics logged using MLflow. This feature is useful to avoid logging irrelevant information.

A prefix blacklist or whitelist can be specified in the PREFIX_BLACKLIST or PREFIX_WHITELIST fields of the configuration file, respectively. These fields accept a list of strings representing the prefixes to be blacklisted or whitelisted.

Similarly, a suffix blacklist or whitelist can be specified in the SUFFIX_BLACKLIST or SUFFIX_WHITELIST fields of the configuration file, respectively. These fields accept a list of strings representing the suffixes to be blacklisted or whitelisted.

Finally, postfixes blacklist or whitelist can be specified in the POSTFIX_BLACKLIST or POSTFIX_WHITELIST fields of the configuration file, respectively. These fields accept a list of strings representing the postfixes to be blacklisted or whitelisted.

For example, to blacklist parameters or metrics with the prefix *load_block* and the suffix *wrapper*, the following configuration file 5.11 can be used:

```
PREFIX_BLACKLIST: ["load_block"]
SUFFIX_BLACKLIST: ["wrapper"]
POSTFIX_BLACKLIST: []

PREFIX_WHITELIST: []
SUFFIX_WHITELIST: []
POSTFIX_WHITELIST: []

...
```

Figure 5.11. MLflow filter configuration file

5.3.3 Logging process

The *run_mlflow* function 5.12 provides an easy way to enable logging for a particular workflow by connecting to MLflow using the information specified in the configuration file. This configuration file, called *mlflow_config* 5.10, allows the user to specify the experiment name and tracking URI for the MLflow server.

```
def run_mlflow():
    if not is_mlflow_running():
        os.environ['MLFLOW_TRACKING_URI'] = mlflow_config['TRACKING_URI']
        mlflow.set_experiment(mlflow_config['EXPERIMENT_NAME'])
        mlflow.start_run()
```

Figure 5.12. Function to start the MLflow run

The *is_mlflow_running* function 5.13 allows users to check whether MLflow is connected to the workflow. If MLflow is connected, the function returns True, indicating that the logging of parameters and metrics is active. On the other hand, if MLflow is not connected, the function returns False. It is recommended to use this function in the workflow code to ensure that all important parameters and metrics are being logged.

```
def is_mlflow_running():
    if mlflow.active_run() is not None:
        return True
    return False
```

Figure 5.13. Function to check if the MLflow run is active

To simplify the process of logging parameters, a dedicated function called *log_parameters* has been implemented within the Block class. This class serves as the base for various blocks such as LoadBlock, TransformBlock, RatingBlock, StatisticalBlock, and others, ensuring consistency and reusability across different parts of the framework. This function is called at the end of each block if MLflow is connected.

By incorporating the *log_parameters* function into the workflow blocks, the framework enables seamless and automatic logging of the relevant parameters. This logging process enhances the transparency and reproducibility of the workflows, as all the essential information related to the parameter values is captured and recorded.

5.3.4 Usage example

The main focus of our efforts is to ensure the proper monitoring of workflow execution. The primary goal is to log the parameters outlined in the JSON files associated with each workflow. Within the Rating workflow, the parameters are stored in a file called *ratingParams*, which is located in the same directory.

Moreover, our attention will be directed towards tracking the initial parameters found within the blocks of each workflow. As a demonstration, the initial parameters in the Rating workflow are defined within the RatingBlock *init_parameters* section as illustrated below.

```
class RatingBlock(Block):

    init_parameters = {name: {'group_training': False,
    'group_testing': False, 'group_training_extraction': False,
    'group_testing_extraction': False}}

    ...
```

Figure 5.14. Rating block initial parameters

Additionally, it is essential to note that we will also be capturing and monitoring the hyper-parameters that become accessible upon the completion of the hyper-parameter optimization process. These recorded hyper-parameters play a crucial role in evaluating the workflow's performance and making necessary adjustments to enhance its overall parameterization.

These parameters are intended to enable the choice of various scopes. This can be helpful if we want to train using data from a certain group of sports and then just utilize data from one of those sports for testing. Therefore, we do not require the enumeration scope to choose the sport during testing.

To illustrate the application of MLflow, it is ideal to show how it works on two workflows, that is Rating and Optimisation. Here we can demonstrate the difference between logging the parameters for a workflow that uses hyper-parameter optimization and one that does not.

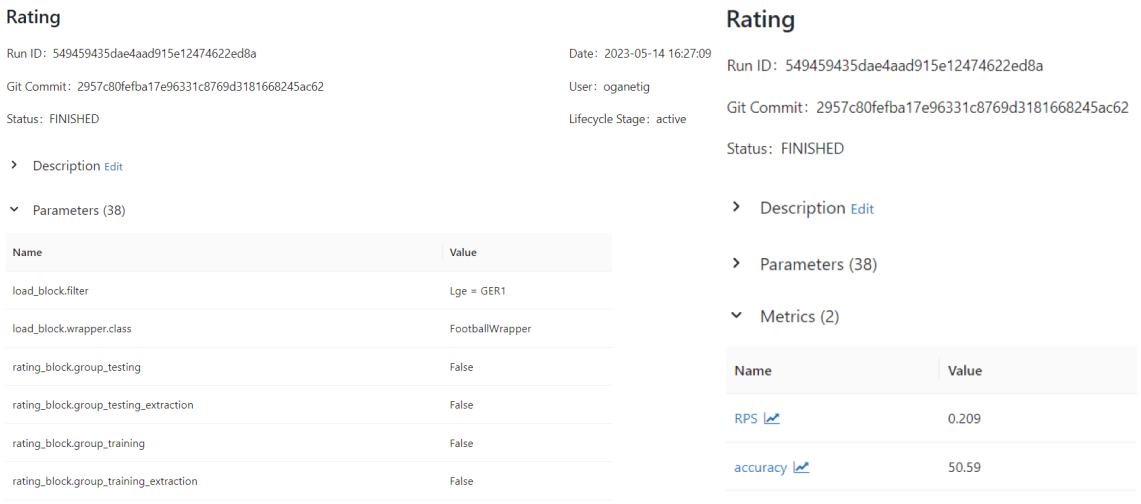

Figure 5.15. Rating workflow run log in MLflow

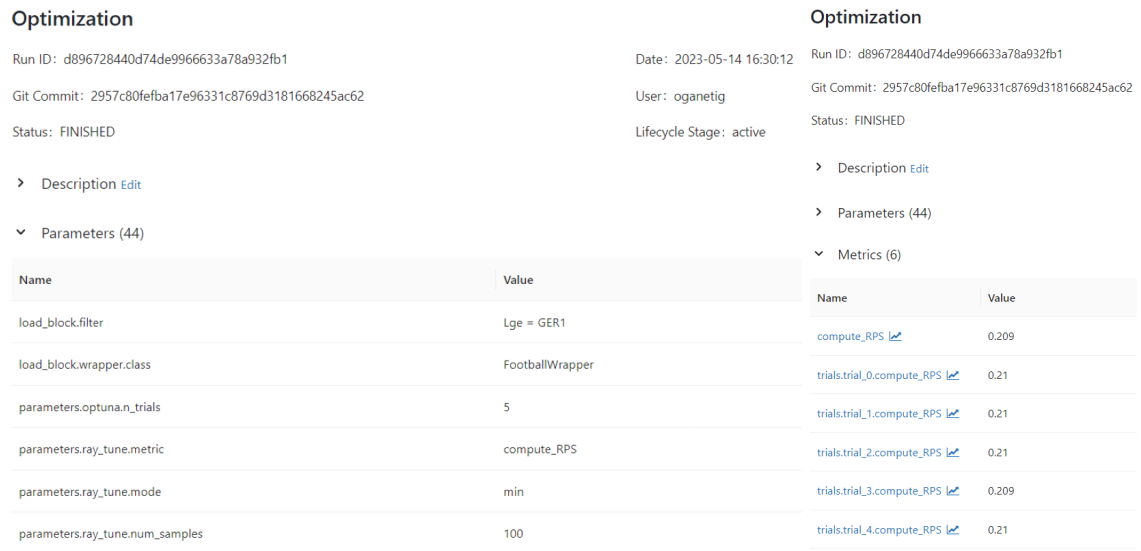

Figure 5.16. Optimization workflow run log in MLflow

In the case of the Optimization Workflow, it is worth noting that the framework goes beyond just storing the best result obtained. In fact, it also saves the results of the trials that were deemed least successful.

By saving the results of the less successful trials, the framework provides a comprehensive view of the optimization landscape. This approach allows users to analyze and understand the variations in performance across different parameter combinations. While the focus is often on identifying the best-performing parameters, understanding the less successful trials can offer valuable insights as well.

5.4 Advanced workflow configuration file

The Workflow Parameters File in the Framework for analyzing efficiency of predictive sports markets is a crucial configuration file that determines the parameters and settings for the entire workflow. With this file, users can tailor the workflow to meet specific needs and modify its behavior. The following sections will highlight the major modifications that have been implemented, which can impact the utilization of the configuration file. These changes may require users to modify existing workflows to take advantage of the new features.

5.4.1 Separate the parameters for the Optuna and Ray Tune

The latest update to the Framework for analyzing efficiency of predictive sports markets has brought significant changes to the Workflow Parameters File. One noteworthy modification is the separation of parameters for hyper-parameter optimization into distinct sections for Optuna and Ray Tune optimization workflows.

```
'rating_block': {
        'optimization': {
            'score_function': compute_RPS,
            'parameters': {
                'ray_tune': {
                    'num_samples': 100,
                    'metric': 'compute_RPS',
                    'mode': 'min',
                    'search_alg': OptunaSearch(),
                    'scheduler': ASHAScheduler(),
                },
                'optuna': {
                    'n_trials': 5,
                    'direction': "maximize"
                }
            }
        },
    ...
```

Figure 5.17. Parameters for optimization workflows

This change enhances the efficiency and flexibility of the framework, allowing for more precise control over the optimization process. With this improvement, users can fine-tune the parameters for each optimization workflow according to their specific needs.

5.4.2 Multiple instances of the same block

With the advanced configuration file 5.19 in the Framework for analyzing efficiency of predictive sports markets, it is now possible to use multiple instances of the same type of block in a workflow by initializing each block with a unique name. This is a significant improvement from the previous version, which only allowed for one instance of each block. For example, the BettingOptimization workflow 5.18 can now incorporate two load blocks and two transform blocks to obtain data about matches and odds from two different tables in the database. This means that the workflow can retrieve and process data from multiple sources simultaneously, which can save time and improve efficiency.

```
params = Parameters(params=bettingOptimizationParams.params)

load_matches = LoadBlock("load_block_matches", params)
matches_wrap = load_matches.compute()
transform_matches = TransformBlock("transform_block_matches", params)
matches_wrap = transform_matches.compute(matches_wrap)

load_odds = LoadBlock("load_block_odds", params)
odds_wrap = load_odds.compute()
transform_odds = TransformBlock("transform_block_odds", params)
odds_wrap = transform_odds.compute(odds_wrap)
...
```

Figure 5.18. Betting Optimization workflow phases

The ability to use multiple instances of the same block type in a workflow is a crucial feature that provides greater flexibility and enables the framework to handle more complex workflows. It allows users to customize their workflows to meet their specific needs and requirements. This enhancement is particularly useful for workflows that require processing of large amounts of data from multiple sources, such as in the case of the BettingOptimization workflow.

```
{
    'load_block_matches': {
        ...
    },
    'load_block_odds': {
        ...
    },
    'transform_block_matches':{
        ...
    },
    'transform_block_odds':{
        ...
    }
...
```

Figure 5.19. Betting Optimization parameters file

5.5 Betting Strategy

The Framework for analyzing efficiency of predictive sports markets has recently introduced Betting Optimization Blocks, allowing users to optimize their betting strategies for enhanced performance. These blocks are specifically designed to collaborate with the Load, Transform, and Predictor Blocks, resulting in a more holistic approach to the betting workflow. This chapter will delve into the functionalities offered by the Betting Optimization Blocks, offering insights into their seamless integration into workflows. Furthermore, the advantages of leveraging these blocks will be explored, accompanied by illustrative examples showcasing their customization potential to align with individual requirements.

The recent updates to the Load Block and Transform Block in the Framework for analyzing efficiency of predictive sports markets have brought about new functionality that allows users to select a specific bookmaker to retrieve data from the database. The database table containing the bookmaker data may have more than one set of odds for a match, from the same bookmaker, due to differences in the date and time the data was recorded. These odds can be affected by several external factors, making it crucial to choose the most recent and relevant data for each match.

```
{
    'transform_block_odds':{
        'only_latest_odds': False
        'only_first_odds': True
    },
...
```

Figure 5.20. Parameters for odds in the TransformBlock

In this context, the Load Block now includes a parameter that enables the selection of a specific bookmaker when retrieving data from the database. On the other hand, the Transform Block includes a parameter that allows users to sort the odds data according to their date and time, ensuring that the most needed data is selected for each match. These new functionalities enhance the precision and accuracy of the betting workflow, leading to improved performance and better outcomes. In the following sections, we will explore these updates in more detail and provide examples of how they can be implemented in practice.

5.6 Experiments

In this section, we will present the outcomes of utilizing the Framework for analyzing efficiency of predictive sports markets with various bookmakers, sports, and leagues taking into account the selection of the latest and earliest odds available. The core configurations for a workflow include the following parameters:

5.6.1 Football experiments

For this series of tests, a specific set of configuration parameters were employed to ensure consistency and comparability in the analysis. The Bundesliga was chosen as the target league for these experiments. The Double Poisson model was utilized as the predictive model in this context.

To establish a baseline for the betting experiments, a starting budget of 1,000 currency units was allocated. This starting budget serves as the foundation for placing bets and tracking the overall profitability of the betting strategies employed. In addition to the starting budget, a fraction ratio of 16 was applied in these tests.

The training window spanned from August 11, 2000, to August 10, 2014, encompassing a significant period of historical data. Following the completion of the training phase, the testing window was set to commence from August 11, 2014.

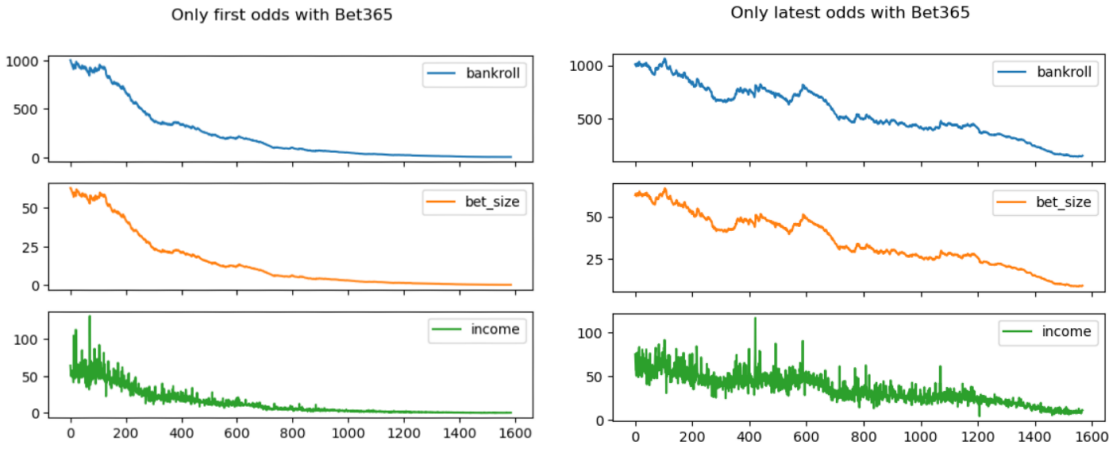

Figure 5.21. Results of the Bundesliga workflow run with bet365

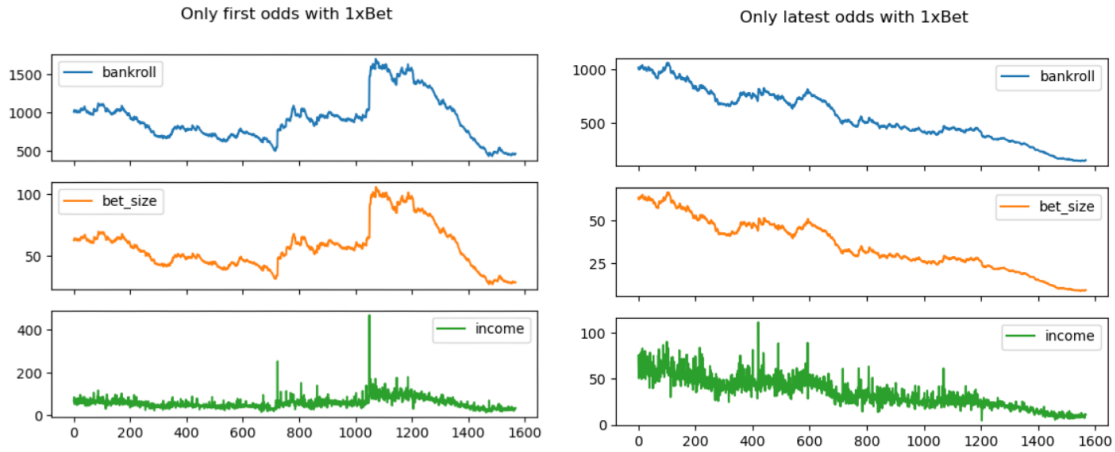

Figure 5.22. Results of the Bundesliga workflow run with 1xBet

The experimentation involved testing the workflow with two different bookmakers and examining the impact of utilizing both the first and the latest betting odds. By analyzing the outcomes outlined below, a clear and direct relationship between the choice of betting odds and the resulting income can be observed.

5.6.2 Basketball experiments

The tests in this series employed specific configuration parameters for consistent analysis. The NBA league was chosen as the target league, and the Bivariate Poisson model was used for predictions. A starting budget of 1,000 currency units and a fraction ratio of 16 were allocated. The training window ranged from August 11, 2000, to August 10, 2014, with testing starting from August 11, 2014.

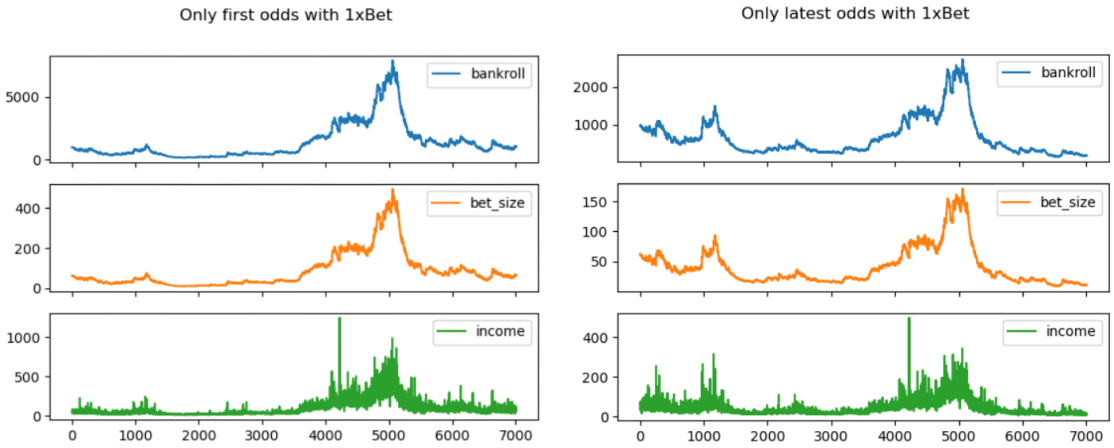

Figure 5.23. Results of the NBA workflow run with 1xBet

Analyzing the following outcomes, as in the case of the soccer league, you can also see a correlation between the choice of betting odds and the income received.

Through these experiments below, we aim to analyze the impact of expanding the training window on the overall outcomes of the betting strategies employed.

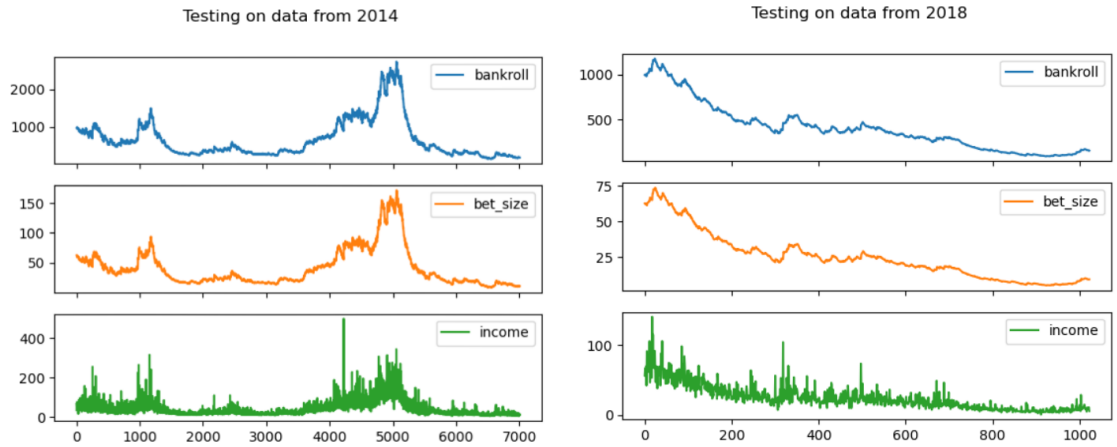

Figure 5.24. Results of the NBA workflow run with different testing window size

The first experiment in 5.24 utilized a relatively smaller training window, allowing the predictive models to capture shorter-term patterns and trends in the data. On the other hand, the second experiment extended the training window, enabling the models to capture longer-term patterns and derive more comprehensive insights from the historical data. Upon analyzing the results, it becomes evident that the larger training window had a significantly positive effect on the outcome of the bets.

Chapter 6
Conclusion

Throughout the investigation of the Framework for analyzing efficiency of predictive sports markets, an extensive analysis of the framework's overall structure was conducted, resulting in a deep understanding of the fundamental mechanisms governing its operation. The various workflows utilized within the framework were explored, and an efficient logging function was implemented for various blocks based on MLOps principles. This implementation streamlined the process of monitoring and tracking the performance of the workflows, which is crucial for ensuring their optimal functioning.

In the context of the project, a major refactoring was undertaken with the aim of improving the quality of the codebase, making it more modular, and easier to maintain. The refactoring involved restructuring the existing code, improving the naming conventions, and reducing the amount of code duplication. By doing so, the project became more organized and easier to work with, even for future developers who might not have been involved in the project's initial development.

In addition, the RCI cluster was used to transfer intricate and time-consuming workflows to computing resources outside of the system. These resources were specifically designed to manage large-scale data processing and complex computations, and their utilization allowed for the maximization of workflow efficiency and effectiveness. This was crucial in achieving precise and dependable outcomes in a timely manner.

The implementation of several new blocks for the Framework for analyzing efficiency of predictive sports markets has allowed for the optimization of betting strategies. These blocks have been designed to work in tandem with existing workflows to enhance the capabilities of the framework. By implementing these new blocks, users of the framework can optimize their strategies more effectively, leading to better results and increased profitability.

6.1 Further extensibility

In the upcoming years, advanced techniques will be incorporated into the framework for betting, including Markowitz strategy and Kelly Growth Rate strategies. Complex calculations and profound knowledge of statistical modeling and probability theory are required for these sophisticated methods. The inclusion of these advanced techniques in the Framework for analyzing efficiency of predictive sports markets will provide users with robust tools for enhancing their betting strategies and attaining greater success in their betting endeavors.

References

[1] Prince Canuma. *MLOps: What It Is, Why It Matters, and How to Implement It.* 2023.
`https://neptune.ai/blog/mlops`.

[2] Ondřej Hubáček, Gustav Šír, and Filip železný. Forty years of score-based soccer match outcome prediction: an experimental review. *IMA Journal of Management Mathematics*. 2021, 33 DOI 10.1093/imaman/dpab029.

[3] Matej Uhrín, Gustav Šír, Ondřej Hubáček, and Filip Železný. *Optimal sports betting strategies in practice: an experimental review*. 2021.

[4] Njeru Mwendi Edwin. Software Frameworks, Architectural and Design Patterns. *Journal of Software Engineering and Applications*. 2014, 2014 670-678.

[5] Prof Dr, Jörg Becker, and Michael zur Muehlen. Workflow Application Architectures: Classification and Characteristics of Workflow-based Information Systems. 2002,

[6] Mariette Awad, and Rahul Khanna. *Machine Learning.* In: *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*. Berkeley, CA: Apress, 2015. 1–18. ISBN 978-1-4302-5990-9.
`https://doi.org/10.1007/978-1-4302-5990-9_1`.

[7] Matthew Stewart. *The Actual Difference Between Statistics and Machine Learning.* 2019.
`https://towardsdatascience.com/the-actual-difference-between-statistics-and-machine-learning-64b49f07ea3`.

[8] M. J. Maher. Modelling association football scores. *Statistica Neerlandica*. 1982, 36 (3), 109-118. DOI https://doi.org/10.1111/j.1467-9574.1982.tb00782.x.

[9] Robert Ankomah, Emmanuel Amoah, and Elvis Obeng. Predictive Modeling of Association Football Scores Using Bivariate Poisson. 2020, 63-69. DOI 10.5923/j.ajms.20201003.01.

[10] Ondřej Hubáček, Gustav Šourek, and Filip Železný. Score-based soccer match outcome modeling – an experimental review. *MathSport International 2019 Conference Proceedings*. 2019, 165-168.

[11] Kyle Hoekstra. *Who Was Chess Master Arpad Elo, and What is the Elo Rating System?* 2021.
`https://www.historyhit.com/gaming/arpad-elo-rating-system/`.

[12] Daniel Berrar, Philippe Lopes, and Werner Dubitzky. Incorporating domain knowledge in machine learning for soccer outcome prediction. *Machine Learning*. 2019, 108 DOI 10.1007/s10994-018-5747-8.

[13] Werner Dubitzky, Philippe Lopes, Jesse Davis, and Daniel Berrar. The Open International Soccer Database for machine learning. *Machine Learning*. 2019, 108 DOI 10.1007/s10994-018-5726-0.

[14] Andrew Chen, Andy Chow, Aaron Davidson, Arjun DCunha, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Clemens Mewald, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, Avesh Singh, Fen Xie, Matei Zaharia, Richard Zang, Juntai Zheng, and Corey Zumar. *Developments in MLflow: A System to Accelerate the Machine Learning Lifecycle*. In: *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*. New York, NY, USA: Association for Computing Machinery, 2020. ISBN 9781450380232. https://doi.org/10.1145/3399579.3399867.

[15] Sayak Paul. *Hyperparameter Optimization & Tuning for Machine Learning (ML)*. 2018. https://www.datacamp.com/tutorial/parameter-optimization-machine-learning-models.

[16] Kizito Nyuytiymbiy. *Parameters and Hyperparameters in Machine Learning and Deep Learning*. 2020. https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac.

[17] Davis David. *Hyperparameter Optimization Techniques to Improve Your Machine Learning Model's Performance*. 2020. https://www.freecodecamp.org/news/hyperparameter-optimization-techniques-machine-learning/.

[18] Aayush Bajaj. *Performance metrics in machine learning [complete guide*. 2023. https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide.

[19] Dileep Kumar, Liaquat Ali, and Sheeraz Memon. Design and Implementation of High Performance Computing (HPC) Cluster. 2018,

[20] Tadeáš Kyral. *A framework for large scale assessment of machine learning in sports*. Czech Technical University in Prague, Faculty of Electrical Engineering. 2022. http://hdl.handle.net/10467/101426.

[21] Matthias Feurer, and Frank Hutter. *Hyperparameter Optimization.* In: 2019. 3-33. ISBN 978-3-030-05317-8.

[22] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019,

[23] *Optuna.trial.Trial*. https://optuna.readthedocs.io/en/stable/reference/generated/optuna.trial.Trial.html.

[24] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E. Gonzalez, and Ion Stoica. Tune: A Research Platform for Distributed Model Selection and Training. *arXiv e-prints*. 2018, arXiv:1807.05118.

[25] Richard Liaw. *Cutting edge hyperparameter tuning with Ray Tune*. 2019. https://medium.com/riselab/cutting-edge-hyperparameter-tuning-with-ray-tune-be6c0447afdf.

[26] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine.. *The Annals of Statistics*. 2001, 29 (5), 1189 – 1232. DOI 10.1214/aos/1013203451.

[27] Tianqi Chen, and Carlos Guestrin. *XGBoost: A Scalable Tree Boosting System.* In: 2016. 785-794.

[28] Oddspedia Experts. *Bankroll management strategies + systems sports betting guide - oddspedia*. 2022. https://oddspedia.com/betting/strategies-systems/bankroll-management.

[29] Ryan Thaxton. *How to use the Sharpe ratio to calculate risk-vs-reward*. 2023. https://www.cityindex.com/en-sg/news-and-analysis/how-to-use-the-sharpe-ratio-to-calculate-risk-vs-reward/.

Appendix A
Project structure

```
.
|-- blocks
|   `-- betting
|-- dataloaders
|   |-- filters
|   `-- sports
|       |-- match
|       `-- odds
|-- datawrappers
|   `-- sport
|       |-- match
|       `-- odds
|-- learners
|   |-- testers
|   `-- trainers
|-- mlruns
|-- models
|-- transformers
|   |-- dataset
|   |-- features
|   `-- labels
|-- utils
`-- workflows
    `-- advanced
```