Rick Brouwer (1672328)

**Course:** Data Science tools and techniques   
**Teacher(s):** Oliver Ntenje, Witek ten Hove et al.  
**Date:** 8-1-2025 |

Football analysis system

Table of content

[Introduction 2](#_Toc187054839)

[Context and Background Information 3](#_Toc187054840)

[The Value of Using AI/ML to Analyze Performance 4](#_Toc187054841)

[Project Breakdown 5](#_Toc187054842)

[Business Understanding 5](#_Toc187054843)

[Data Understanding 6](#_Toc187054844)

[Data Preparation 8](#_Toc187054845)

[Step 1: Acquiring the Data 10](#_Toc187054846)

[Step 2: Data Exploration and Cleaning 10](#_Toc187054847)

[Step 3: Feature Engineering 11](#_Toc187054848)

[Step 4: Training the Model 11](#_Toc187054849)

[Step 5: Model Evaluation 11](#_Toc187054850)

[Conclusion 12](#_Toc187054851)

[Reflection 12](#_Toc187054852)

# Introduction

This report presents the results of a football analytics project, which focuses on analyzing and measuring player performance and movement using computer vision techniques. The primary goal of this project is to build a machine learning model that can detect and track football players, referees, and the ball in broadcast footage, ultimately generating valuable insights such as player speed, distance covered, and team association based on uniform colors.

By leveraging YOLO (You Only Look Once) for object detection, integrating tracking algorithms, and applying perspective transformations to convert pixel data into real-world distances, the project aims to deliver an end-to-end analytics solution for coaches, analysts, and sports enthusiasts. Key metrics like speed, distance, and positional data can provide meaningful guidance in training, in-game decisions, and post-match evaluations.

This report follows a CRISP-DM structure to systematically cover every phase of the project, from business understanding and data gathering to model evaluation and maintenance considerations. The core objective is to demonstrate how real-time (or near real-time) computer vision and machine learning can illuminate the finer details of player performance and team dynamics over the course of a match.

# Context and Background Information

In this section, we explore the background of football, the importance of performance analytics in modern sports, and how AI/ML tools can provide a competitive edge.

**What is Football and why is AI relevant for this sport?**

Football is an internationally celebrated team sport, played by two squads of eleven players each. The objective is to score more goals than the opposing side by getting the ball into the opponent’s net, located at opposite ends of the rectangular pitch. Players use strategic passing, dribbling, tackling, and shooting maneuvers to outplay their opponents, with the entire team often transitioning between offensive and defensive roles throughout the game.

The **continuous movement** of all players on the field distinguishes football from turn-based or pause-heavy sports, demanding strong stamina, tactical awareness, and adaptability. Importantly, the **dynamic nature** of the game offers ample data points (e.g., position, speed, passes, shots) for advanced analytics.

Success in football depends on a combination of:

1. **Technical Skills** – Ball control, dribbling, passing accuracy, and finishing ability.
2. **Physical Attributes** – Speed, stamina, strength, and agility.
3. **Tactical Knowledge** – Positioning, formation discipline, and the ability to read the game.
4. **Team Cohesion** – Communication, coordinated plays, and shared understanding of strategy.

Analyzing these facets through data can help teams hone their strategies, monitor player fitness, and adapt tactics mid-game. Modern AI solutions can pinpoint movement patterns, coverage gaps, or possession details, helping coaches make data-driven improvements.

Afbeelding met stadion, gebouw, Sportlocatie, gras

Automatisch gegenereerde beschrijving*Figure 1: Football analysis example*

Football is highly dynamic, with situations shifting rapidly—such as counterattacks, overlaps, or quick transitions from defense to offense. Traditional statistics like shots on target or possession percentages often provide an incomplete picture. More granular metrics—like speed in sprints, distance covered, and heat maps—can highlight a player’s in-game behavior and expose areas of improvement that generic stats overlook.

**Why insights are valuable:**

* **Substitutions**: Coaches can track players’ fatigue by monitoring decline in sprint speed or distance covered over time, deciding when to bring in fresh legs.
* **Injury Prevention**: Knowing a player’s high-intensity running load can help prevent muscle strains or injuries associated with overexertion.
* **On-the-fly Tactics**: If the data reveals that the opponent’s left flank is under heavy pressure, coaches might instruct players to exploit that weakness.

### The Value of Using AI/ML to Analyze Performance

Applying AI and machine learning to football video footage offers significant advantages:

1. **Automated Data Collection** – Eliminates the need for manual tagging of every play, reducing costs and human error.
2. **Objectivity** – Algorithmic measurements avoid biases that might arise from subjective evaluation.
3. **Scalability** – Large volumes of matches can be processed quickly, beneficial for clubs that want to analyze multiple games per week or entire leagues.
4. **Depth of Insight** – Complex data (player speed, acceleration, micro-movements) can be aggregated into tactical heat maps, player stamina curves, or possession retention stats.

In an era where marginal gains can decide the outcome of high-stakes matches, an AI-powered analysis tool that accurately measures speed and distance can greatly influence both match day decisions and long-term planning. Even outside of professional leagues, academies and smaller clubs can adopt such methods to evaluate player development more thoroughly. This project aims to democratize advanced analytics by showing that accessible camera setups and open-source AI models (like YOLO) can yield powerful insights into how matches unfold.

# Project Breakdown

Below is a detailed breakdown of the project following the **CRISP-DM** process: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, Deployment, and Maintenance.

### Business Understanding

The overarching goal is to develop an analytics pipeline that can ingest standard football match footage and output:

* **Player Detection**: Identify each player and referee in the frame.
* **Tracking**: Maintain player identities across frames to measure their movement patterns.
* **Team Assignment**: Categorize players as Team A or Team B based on uniform colors.
* **Performance Metrics**: Speed, distance covered, average position, and possibly ball possession analyses.

**Success Criteria**

In developing this football analytics system, four success criteria stand out as essential benchmarks for evaluating both its accuracy and practical value. First and foremost is detection accuracy, which is typically measured by how well the model performs on standard metrics like mAP (mean Average Precision). A high mAP indicates that the model is effectively recognizing players, referees, and the ball in various scenes.

Closely related to detection accuracy is tracking stability. Even if the model can detect objects reliably, it also needs to preserve their identities over time, ensuring minimal ID switches. This means the system should continuously follow each player (and referee) with consistent bounding boxes, even in congested situations such as penalty boxes or corner kicks.

Beyond just identifying and following players, the system aims to measure distances in the real world with acceptable precision—somewhere in the range of ±0.25 to ±0.35 meters. This distance measurement accuracy is crucial because coaches and analysts rely on metrics like total distance covered or sprint speeds to inform training and tactics. If the model miscalculates these distances, the resulting data would be less valuable.

Finally, the entire pipeline must demonstrate robustness when exposed to diverse environments. Football matches are played under myriad conditions: varying camera angles, multiple resolutions (from low-quality amateur footage to 4K broadcasts), and drastically different lighting (e.g., midday sun vs. night floodlights). Team uniforms also change from match to match, requiring the model to handle different color schemes. Satisfying all these criteria ensures that the system remains effective across a broad spectrum of real-world scenarios.

### Data Understanding

Developing an accurate model starts with understanding the specific nature of the footage that will train and challenge it, shaping both the learning process and the subsequent evaluation. In this project, the input comes from a **single, open 30-second clip of a professional football match**, sourced from Kaggle. Despite its brevity, this snippet provides sufficient variety in player movements and interactions to test whether the model can detect players, referees, and the ball under realistic conditions.

**Source of Video Footage**

Because the project relies on this short Kaggle-based except, the model’s performance hinges on how effectively it adapts to the details of that particular recording. Professional broadcasts typically benefit from higher resolution cameras, more stable angles, and clear lighting setups, all of which can aid in identifying crucial elements like bounding boxes for players or the ball. However, even in a professional match, sudden camera pans and zooms can occur, making the clip more challenging to parse. By focusing on this single snippet, the project aims to refine its detection approach without having to reconcile multiple dissimilar data sources.

Afbeelding met stadion, gras, Voetbalstadion, Kunstgras

Automatisch gegenereerde beschrijving*Figure 2: Input video from kaggle*

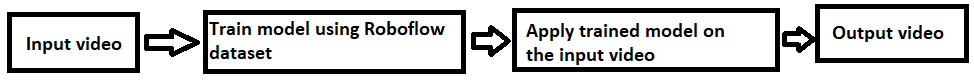
**Relevant Frame-by-Frame Details**

Within these 30 seconds of footage, each video frame contains several points of interest. **Player bounding boxes** define where individuals appear on the pitch, using coordinates such as (x, y) alongside width and height. A **ball bounding box** zeroes in on the football itself, which is especially important when analyzing possession or pass attempts. In parallel, bounding boxes for referees prevent the model from mixing them up with field players, preserving the accuracy of metrics tied to team performance.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving  
*Figure 3: Bounding box definition*

In the end we want to make a video out of the imput video, like this:



**Data Variations**

Although just one snippet is used, it still can exhibit notable variability. The **frame rate**—often around 25 to 30 frames per second in standard broadcasts—affects how smoothly player motion is captured and how granularly the model can measure sprints or quick turns. Resolution also matters; higher resolution makes it simpler to spot smaller details, like a ball partially obscured by a cluster of players, yet demands more computing power. Lighting conditions, camera angle, and partial obstructions further complicate the process. Even over a short clip, fleeting environmental changes—such as shifting shadows or a referee briefly entering the frame—can test the model’s resilience.

Despite relying on a single 30-second video, the project’s core practices—defining bounding boxes, tracking camera motion, and filtering referees—remain the same. By grappling with potential challenges within that snippet, the analytics pipeline proves its capacity to measure real-world football actions in a concise timeframe. The ultimate goal is to ensure that key elements—players, referees, and the ball—are accurately recognized and tracked, laying a solid foundation for more extensive match analysis in the future.

### Data Preparation

A solid data preparation strategy underpins the accuracy and reliability of any machine learning pipeline, and football analytics is no exception. Before the model can detect and track players, the raw footage must be systematically processed and enriched with meaningful labels. The first major step is labeling and annotation, where each frame is examined to locate and tag the players, the ball, and any referees that appear on-screen. In many cases, this process can be carried out manually or via semi-automated tools; either way, referees must be labeled as a separate class so that the model does not mistakenly consider them part of a team.

Once the footage is annotated, it is split into three distinct datasets: a training set, a validation set, and a test set. Typically, about 70 percent of the labeled frames form the training set, which feeds the YOLO model the bulk of information needed to learn how to identify players, referees, and the ball. Around 15 percent of the dataset is then set aside as the validation set, guiding decisions about hyperparameters like learning rate or confidence thresholds. The remaining 15 percent is held back as the test set, used only at the end of the project to gauge final performance and prevent overfitting issues.

This project is a concise demonstration of automated football analytics, focusing on detecting and tracking players, referees, and the ball in a short clip of professional match footage. At its core, the system uses object detection (via YOLO) and multi-object tracking (through ByteTrack) to locate these entities in each frame, then draws distinctive shapes—ellipses for players and referees, and a triangle for the ball—so that viewers can immediately see where everything is happening on the pitch.

The code is divided into logical parts. One module takes care of reading and writing videos, turning an MP4 file into a list of frames (and vice versa) so each frame can be analyzed individually. Another module deals with bounding boxes by providing helper functions that can determine their centers and widths, simplifying downstream calculations. The main logic resides in the Tracker class, where YOLO runs inference on each frame to detect objects, and ByteTrack assigns consistent IDs across frames. This way, even if a player moves quickly, the system recognizes it as the same player over time, rather than a new detection in every frame.

To avoid re-running the potentially expensive detection step for each test, the system can also read “stub” data from a pickle file that holds all previously computed bounding boxes. That makes it easy to experiment with different drawing methods, color schemes, or annotation ideas without waiting for the model to process the video again. Finally, there is a script called main.py that pulls everything together: it reads frames from the clip, initializes the tracker, retrieves or computes object tracks, draws the annotations, and writes out an annotated video file. In the end, you have a single short movie that shows each player, referee, and the ball marked with clear shapes and track IDs, making it simpler to analyze what’s going on in every moment of the match.

Although the clip itself is just 30 seconds, it includes enough variation in camera movement and player interactions to demonstrate whether YOLO can reliably detect objects and whether ByteTrack can maintain correct IDs over consecutive frames. This proof of concept suggests that if the system works well on a short snippet, it could be expanded to handle longer videos, potentially generating more advanced insights such as player speed, distance covered, or tactical heat maps. By automating tasks that once required painstaking human review, this project exemplifies how deep learning and computer vision tools can bring clarity to the fast-paced, ever-changing world of a football match.

# Step 1: Acquiring the Data

Acquiring suitable match data can follow one of two common approaches. One route is to rely on pre-made datasets, which are sometimes available on Kaggle, where you might find partially labeled football frames (highlighting players and the ball), or on Roboflow, which hosts specialized sports image collections. Such datasets can jump-start your training process by providing a basic level of annotation and variety in how the game is captured.

Afbeelding met tekst, schermopname, Multimediasoftware, Grafische software

Automatisch gegenereerde beschrijving  
*Figure: Roboflow dataset used*

The other approach is to gather custom footage, which might mean filming lower-division or amateur matches yourself under different conditions, or accessing higher-quality recordings from professional leagues—though this can involve licensing complexities. Regardless of the source, it’s crucial to recognize potential limitations and biases. Professional broadcasts may feature stable, high-resolution cameras, while amateur recordings tend to be shakier, with suboptimal lighting or vantage points. Certain matches might be played at night or in poor weather, reducing visibility and complicating detection. Additionally, if two teams wear similarly colored kits (like navy blue versus black), color-based methods for differentiating them can falter unless you’ve accounted for subtle differences in hue. Combining multiple angles, stadiums, and kit variations helps the model generalize better and withstand real-world usage.

Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijving  
*Figure 4: Roboflow classes used*

# Step 2: Data Exploration and Cleaning

Before proceeding with modeling, it’s important to explore the raw footage for anomalies or issues. You may discover frames with unexpectedly large bounding boxes due to extreme camera zooming, or others where the ball becomes invisible when too many players converge at once. In some cases, lighting shifts dramatically from one portion of the video to another, creating challenges for consistent detection. You might also find missing labels if, for example, a player runs offscreen or is only partially visible.

A descriptive statistics table can reveal trends, like how often the ball is actually visible or how many players typically appear in each frame. This helps you decide if you need additional filtering or special handling for certain frames. Cleaning efforts might include verifying that high optical flow values are indeed caused by camera movement (like a sudden pan) rather than an error in annotation, or ensuring that frames with no visible players are legitimately crowd shots or scene transitions rather than mislabeled data. Adjusting brightness or contrast for night matches can also improve accuracy and consistency in detection results.

# Step 3: Feature Engineering

Once you’ve prepared your data, the next step is to create additional features that can help the model or subsequent analyses. Player speed can be calculated by measuring changes in x and y coordinates over successive frames and then dividing distance by time to yield a velocity in meters per second or kilometers per hour. Distance covered expands on speed by accumulating these frame-by-frame movements to determine total distance traveled, which can reveal who works hardest on the pitch.

If you want to distinguish teams, you can label players via K-means clustering on their shirt colors, thus helping you track which side of the field has possession at any given time. Similarly, a ball-possession metric might come from matching the ball’s location to the nearest player bounding box, informing stats such as how long a team keeps control or how efficiently they move the ball around. For more realistic spatial analytics, perspective transforms map each bounding box coordinate onto a 2D representation of the pitch. This conversion is essential for converting pixel distances into real meters. You can also apply optical flow to subtract out camera movement, ensuring that any speed or distance calculations pertain to actual player motion rather than just a shifting viewpoint.

# Step 4: Training the Model

Training the YOLO model involves feeding it carefully prepared frames and bounding box labels while adjusting hyperparameters to balance accuracy and efficiency. You might start with around 50 epochs, watching the validation loss to see if it stops improving around epoch 40. A learning rate of 0.001 or 0.0005 often works well, and batch sizes of 16 or 32 typically strike a good compromise between performance and memory constraints. If your confidence threshold is set too high, you risk missing valid detections; set it too low, and you may introduce excessive false positives.

After detection, your tracking approach comes into play. Each inference pass produces bounding boxes that feed into a tracker, which updates player IDs over time. If you’re also classifying or clustering teams, this step might happen in parallel or after detection is complete. You can further refine your setup by running mini-grid searches or Bayesian optimization, tuning parameters like image size or data augmentation strength, and stopping early when the model no longer improves. This careful calibration ensures that the model generalizes beyond your initial training data, performing well in scenarios like real-time streams or large-scale batch analyses.

# Step 5: Model Evaluation

Once your model is trained and integrated with the tracker, you must evaluate how well it performs. Object detection can be assessed by measuring mAP (mean Average Precision) across players, ball, and referees, noting whether the system mislabels unrelated people or staff near the sidelines. Tracking performance can be gauged by looking at metrics such as IDF1 or counting how frequently ID switches occur, since each switch effectively breaks continuity and can undermine distance calculations.

Afbeelding met stadion, gras, Voetbalstadion, Kunstgras

Automatisch gegenereerde beschrijvingAfbeelding met gras, stadion, Kunstgras, Voetbalstadion

Automatisch gegenereerde beschrijving  
Figure: Input video vs output result

If you’ve implemented real-world distance or speed estimates, checking them against known measurements is key. For instance, you can measure how far a player travels between two points on the pitch that you’ve physically measured beforehand. If your error margin (perhaps ±0.2 or ±0.3 meters) remains consistent, you know the system is sufficiently accurate for practical usage. Visual inspection of the final annotated footage is also invaluable, letting you confirm that bounding boxes follow the right individuals and that track IDs remain stable, especially in crowded scenes where detection might falter.

# Conclusion

This streamlined football analytics pipeline demonstrates the power of combining machine learning and computer vision with domain-specific insights. By leveraging YOLO for detection, applying tracking algorithms to maintain consistent IDs, and integrating features like color clustering or perspective mapping, the system can produce data points that matter in real-world match analysis—ranging from player workload and speed to overall team strategies and formations. Once validated, such a framework is readily scalable, supporting additional enhancements like robust re-identification methods, specialized night-match training, or the integration of advanced optical flow techniques. Ultimately, the project highlights how AI-driven methods can transform raw, often chaotic football footage into structured, insightful information that helps teams, coaches, and analysts make better-informed decisions on and off the pitch.

# Reflection

Working on this football analytics project has been a really eye-opening experience. I faced a lot of challenges, especially with using Google Colab. At first, I found the interface confusing and struggled with managing the computer resources I needed. Problems like the session disconnecting, running out of memory, and setting up everything to work with tools like YOLO and ByteTrack were frustrating. But by sticking with it and using online tutorials and forums, I eventually got the hang of Colab and was able to use it more effectively.

This project was my first deep dive into artificial intelligence and computer vision, areas I hadn’t explored much in my previous group projects. Before, our projects were mostly about analyzing data and making visualizations. This time, I had to learn about how object detection works, how to track multiple objects at once, and how to convert what the computer sees into real-world measurements. I spent a lot of time understanding how YOLO can detect objects in real-time and how ByteTrack keeps track of players across different frames. This hands-on learning greatly expanded my knowledge and made me appreciate how complex it is to build a complete AI system for analyzing football matches.

Even though it was tough to learn all these new things, the experience was incredibly valuable. I learned practical skills like labeling video frames and adjusting the camera view to measure real distances on the pitch. I also worked on techniques to tell which team a player is on by looking at their uniform colors, which was really satisfying because it combined technical skills with specific insights about football. These skills have given me a solid foundation that I’m excited to use in future projects.

However, the project wasn’t without its difficulties. Integrating all the different parts of the system was more complicated than I expected, and I was only able to get up to the tracking part. Making sure the tracker worked well in busy and fast-moving scenes was particularly hard, especially when players overlapped or the camera moved quickly. Even though I couldn’t complete the entire project, the progress I made taught me a lot. It showed me the importance of taking things step by step and continuously improving the system when dealing with real-time data and complex algorithms.

Throughout this project, I gained a better understanding of how AI can be used in sports analytics and what its limitations are. I learned that collecting detailed and accurate data is crucial for getting useful insights and that precise tracking and distance measurements are key to evaluating player performance. The project also taught me to stay persistent and adaptable when facing technical problems, which has made me more resilient and determined to learn new technologies.

In the end, this football analytics project has been a transformative journey for me. Even though I ran into many challenges, especially with Google Colab and the complex parts of computer vision, I learned much more than I did in my previous group projects. Reaching the tracking stage was a big achievement, and the knowledge I gained has set a strong base for future work in AI and machine learning. This project not only improved my technical skills but also sparked a passion for using AI to find deeper insights in fast-paced, real-world situations like football matches. I’m confident that what I’ve learned will help me grow as a data scientist and an AI enthusiast.

# References

@codeinajiffy. (2024, April 19). Build an AI/ML Football Analysis system with YOLO, OpenCV, and Python.