MSc Artificial Intelligence
Machine Learning & Pattern Recognition

Continuous Assessment One

by

Ugur ERCAN

Lecturer: Shahram Azizi Sazi

Date of Submission: 19/11/2025

Dublin Business School

13/14 Aungier St, Dublin 2,

D02 WC04, Ireland

Introduction

This project follows the CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology, and divides the machine learning workflow into six iterative phases:

1. Business Understanding
2. Data Understanding
3. Data Preparation
4. Modeling
5. Evaluation
6. Deployment

The objective of the study is to build a supervised machine learning model that can **predict whether a team will win a match in a competitive online multiplayer game**, League of Legends. Observations are player-level performance metrics for different matches. The target variable is the **'win'** feature which has Boolean values as 'True' and 'False'.

Although the interest is gaming, the methodology is transferrable to any prediction task involving team performance, decision-making, or event outcomes.

1. Business Understanding

League of Legends is a five-versus-five strategy game where two teams compete to destroy each other's base units. Every match produces structured data describing how players performed: kills, assists, gold earned, objectives captured, damage dealt, and many other numerical measures. These statistics are conceptually similar to performance metrics in sports analytics (e.g., football possession %, basketball rebounds, etc.).

The goal is to determine whether team-level gameplay metrics are reliable to predict match outcomes. This problem is important for esports analytics, coaching, competitive strategy, and performance optimisation across gaming environments. By exploring the key factors contributing to match success, this study aims to demonstrate the applicability of machine learning models, particularly **Support Vector Machines (SVM)** and **Random Forest** classifiers, to real competitive gaming data.

2. Data Understanding

Initial Data Overview

The dataset initially contained **40410 rows** and **94 columns** in an xlsx file. Each row represents a single player participating in a single match. A typical match consists of 10 players (5 per team). However, since each row is stored separately, the raw data is player-level, not team-level. There are 94 columns including IDs, gameplay stats, match details, mastery information, and ranking information.

Filtering Game Modes

League of Legends contains several different game modes. These modes differ in number of players, map layout, rules, match length, objectives. The goal of this project is to predict outcomes in standard competitive matches, which are played in a mode called 'CLASSIC'. The win conditions for any match changes for any game modes. Therefore, this project keeps the 'CLASSIC' matches as only source to ensure the dataset is consistent, and valid for modelling.

After filtering: *(29400 rows, 94 columns)*

Team Structure Observation

A check on team's number of players showed that there are 5864 teams with 5 players, 16 teams with 4 players.

The rare four-player teams likely come from incomplete data collection, remade matches, or players who disconnected before the system recorded them. They represent less than 1% of all observations and were retained because removing them could introduce bias and they do not materially affect model performance.

3. Data Preparation

Dropping Columns

The features that do not contribute to predicting team victory were removed.

After dropping columns*: (29400 rows, 75 columns)*

Handling Missing Values

- Rows missing ‘team_position’ or ‘win’ were removed because these are essential fields.
- All remaining missing numeric values were replaced with zero, which is appropriate for gameplay metrics (e.g., “0 barons killed” is valid).

After handling missing values: *(29384 rows, 75 columns)*

Aggregating Player-Level Data to Team-Level

The predictive goal concerns team outcomes, not individual performance. Thus, the player-level dataset was aggregated so **each row represents one team in one match**.

Aggregation method:

- ‘sum’ for kills, gold, damage, objectives
- ‘mean’ for vision score and crowd-control metrics
- ‘max’ for the ‘win’ label (if any player is marked as a winner, the whole team won)

Final team-level dataset shape: *(5880 rows, 19 columns)*

This corresponds to 2940 matches × 2 teams.

Feature and Target Definition

Final input features included numerical gameplay indicators such as:

- kills, deaths, assists
- damage dealt and damage taken
- objective control (dragons, barons, turrets)
- gold earned/spent
- vision-related statistics

Target variable ‘win’ is turned into integer values as 1 = win, 0 = loss.

Feature Scaling

After handling the columns and dataset, the features only include numerical values. StandardScaler was applied to all numerical features. SVM, in particular, requires feature scaling to perform properly because:

- it relies on distance-based calculations
- unscaled features distort decision boundaries

Scaling was implemented inside a scikit-learn Pipeline to avoid data leakage.

4. Modeling

The models used in this project are Support Vector Machine and Random Class classifiers. Both were embedded in Pipelines containing preprocessing and model steps.

After dividing the dataset into train and test sets, models are fitted on train sets and predictions made on test sets. The following accuracies are derived for base models:

- Random Forest: 0.9685
- SVM: 0.9583

Accuracy was selected as the primary metric because the dataset is perfectly balanced (equal number of wins and losses), and the cost of misclassifying either class is symmetric. In such conditions, accuracy provides an intuitive and reliable measure of model performance.

Additionally, a *classification report* was used to provide precision, recall, and F1-score for each class. These metrics confirm that the model performs consistently across both “win” and “loss” categories and is not biased toward one outcome. This deeper breakdown supplements accuracy by showing whether the model makes proportionally similar errors across classes.

5. Hyperparameter Tuning

To improve model performance, GridSearchCV was used together with a 5-fold StratifiedKFold cross-validation strategy. Hyperparameter tuning is an essential part of the modeling phase, even when baseline accuracy is already high, because default model settings are not always optimal for a specific dataset. Tuning also helps confirm the stability and generalisation of the model by evaluating multiple parameter configurations under systematic cross-validation.

StratifiedKFold was chosen because it preserves the 50-50 class balance (win vs. loss) in every fold, ensuring that each validation split reflects the true distribution of the target variable. This reduces variance and provides a more reliable estimate of performance.

6. Evaluation

After hyperparameter tuning, both models were evaluated on the held-out test set using the accuracy metric and the classification report. Accuracy was used as the primary metric because the dataset is balanced, and the cost of misclassification is equal for both classes. The classification report was also examined to ensure that precision, recall, and F1-scores were consistent across both the “win” and “loss” classes.

The final results were:

- Random Forest Test Accuracy: 0.967
- SVM Test Accuracy: 0.965

Both models achieved high and comparable performance. The results indicate that the features derived from aggregated team-level statistics are highly predictive of match outcomes. Precision and recall values were similar across both classes, showing that neither model was biased toward predicting wins or losses.

7. Overfitting Avoidance Mechanisms

Several measures were taken to avoid overfitting throughout the project:

- Train/Test Split: An 80/20 stratified split ensured that the distribution of the target variable was preserved between training and testing subsets.
- Cross-Validation: GridSearchCV with 5-fold StratifiedKFold ensured that model performance was validated on multiple subsets of the training data, providing a more stable estimate of generalisation.
- Pipeline Usage: Preprocessing steps, such as scaling, were embedded inside Pipelines to prevent data leakage and ensure that transformations were only fitted on the training data.
- Random Forest Regularisation: Hyperparameters such as min_samples_leaf, min_samples_split, and max_features controlled model complexity and prevented individual trees from growing too deep.
- SVM Regularisation (C parameter): The C parameter controlled the margin softness and prevented the model from fitting noise in the training data.

These mechanisms ensured that both models generalised well to unseen data, as confirmed by the similarity between cross-validation accuracy and test accuracy.

8. Results Analysis

The results show that both Random Forest and SVM models perform strongly on predicting match outcomes using team-level gameplay statistics. Several observations can be made:

- High Predictive Power: Gameplay indicators such as total damage, gold earned, kills, assists, and objective control (dragons, barons, turrets) are strong predictors of team success. This aligns with strategic elements of the game where resource management and objective control significantly influence outcomes.

- Model Comparison: The Random Forest model slightly outperformed the SVM on the final test accuracy, while the SVM performed marginally better in cross-validation. This suggests that both models are suitable for the dataset, with negligible performance differences.
- No Overfitting Detected: The close alignment between CV scores and test accuracies indicates that both models generalise well. Neither model exhibited significant performance drops between validation and test phases.
- Balanced Performance: Precision, recall, and F1-scores remained consistent across both classes, showing that the models were equally effective at identifying wins and losses.

Overall, the findings demonstrate that machine learning models can effectively model and predict competitive match outcomes using structured performance metrics.

Conclusion

This project applied the CRISP-DM methodology to build and evaluate machine learning models for predicting match outcomes in League of Legends using aggregated team-level gameplay statistics. The data underwent extensive preparation, including filtering, cleaning, aggregation, feature selection, and scaling. Two classification models, Support Vector Machine and Random Forest, were developed and tuned using GridSearchCV and StratifiedKFold cross-validation.

Both models achieved high predictive accuracy, confirming that gameplay metrics provide strong signals about team performance. The study shows that machine learning can be effectively used in esports analytics, providing insights that may support coaching, strategy, and performance optimisation.

The project also demonstrated the importance of systematic data preparation, hyperparameter tuning, and evaluation to ensure reliable and generalisable model performance. The methods and results can be applied to similar prediction tasks involving structured team performance data in other domains.