THE BRIDGE DIGITAL TALENT ACCELERATOR

Bootcamp Data Science
Blanca Novella Serrano
2024

Supervised Machine Learning

Predicting Engagement Levels in Online Games

ÍNDICE

01 Introduction

This project focuses on Supervised Machine Learning. We will train different models and compare their results to determine which one achieves the best performance in predicting the engagement level in online video games.

Source:
Rabie El Kharoua. (2024).
🎮 Predict Online Gaming Behavior Dataset [Data set].
Kaggle. https://doi.org/10.34740/KAGGLE/DSV/8742674.
https://www.kaggle.com/datasets/rabieelkharoua/predict-online-gaming-behavior-dataset

01.A Context

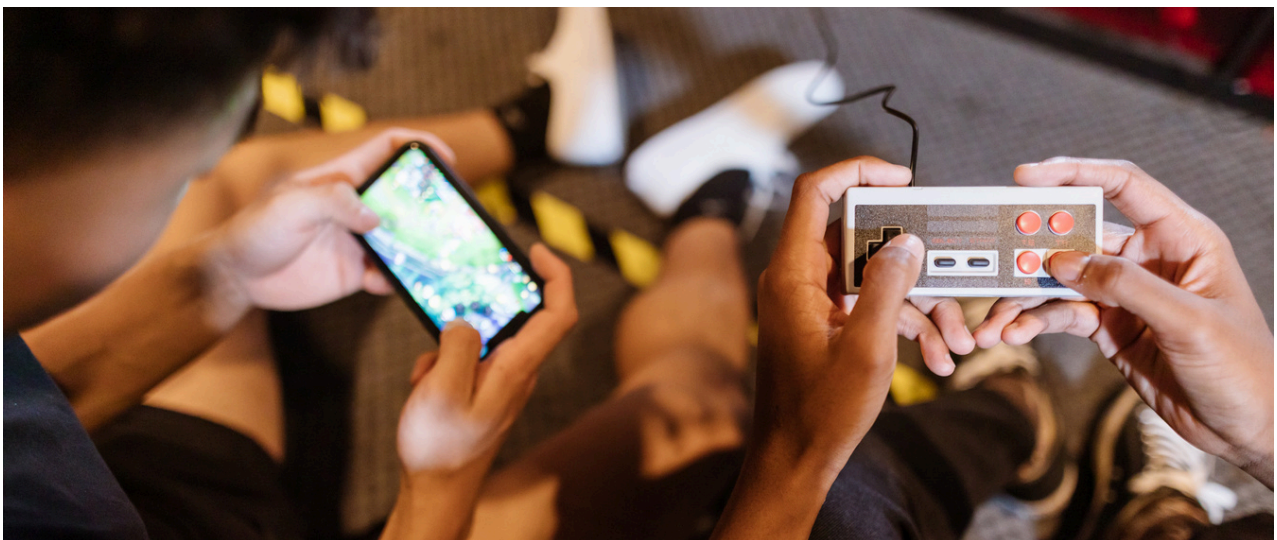
Objective:
Predict a player's engagement level based on given features such as playtime hours, number of sessions, game type, etc.

Methodology:
This is a classification problem, and we will employ supervised classification algorithms.

Metrics:
We will focus on balanced accuracy and macro recall, given the nature of the target variable.

_Business Impact:

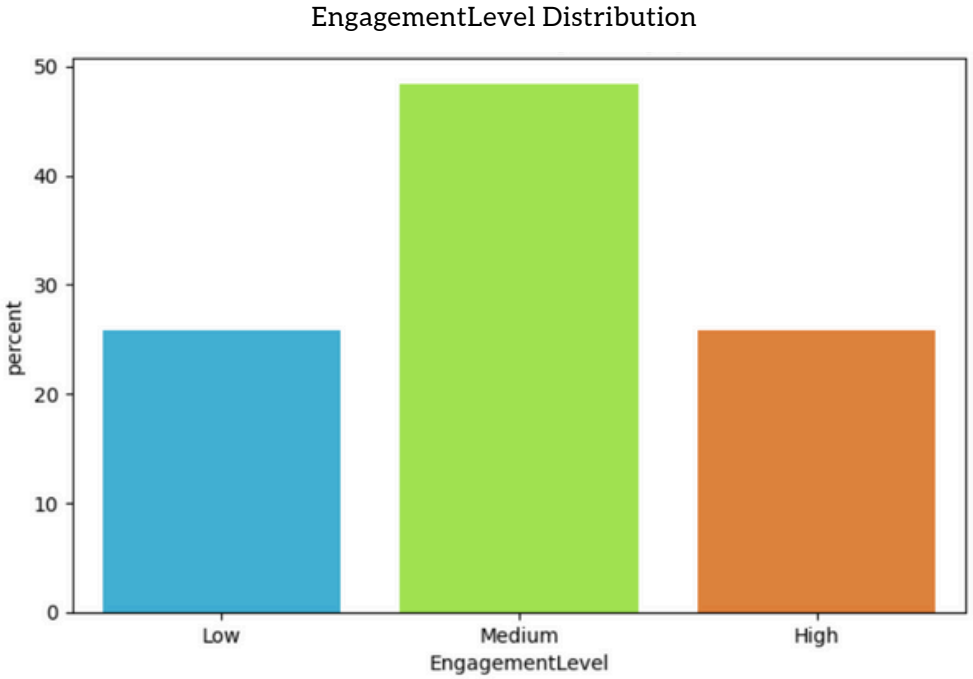
- Improve user retention strategies.
- Enhance monetisation strategies.
- Support decision-making in video game design.



01.B Compresión de los datos (EDA)

We use the online_gaming_behavior_dataset.csv dataset, which contains 40,034 entries and 13 columns. The dataset appears to have been preprocessed by the author. However, we will later apply transformations and remove the PlayerID column.
After splitting the dataset into train and test, we analyse the target variable EngagementLevel, classified into three categories: Medium, High, and Low. The data is imbalanced, with 50% of the samples belonging to the Medium category, while High and Low share the remaining 50% equally.

Fig.1 Traget Distribution



To understand how the other variables behave concerning the target, we classify them into numerical and categorical features and analyse their distribution. A heatmap correlation matrix is used for numerical variables.

Numerical Variables:

- **Age, PlayerLevel, and AchievementsUnlocked:** Continuous variables that could be categorised into intervals (e.g., age groups or player levels). However, we will first train the model with their raw values as they are well distributed.

- **SessionsPerWeek and AvgSessionDurationMinutes:** These show significant variability in relation to the target, making them potentially useful.

- **PlaytimeHours**: Might be relevant, though it does not exhibit strong variation across target classes.

Categorical Variables:

- **InGamePurchases:** A binary feature already encoded, highly imbalanced, with 0 (no purchases) as the dominant category.

- **Gender, GameDifficulty:** Irregular category distributions.

- **Location, GameGenre: More balanced distributions, with GameGenre being evenly spread among categories.**

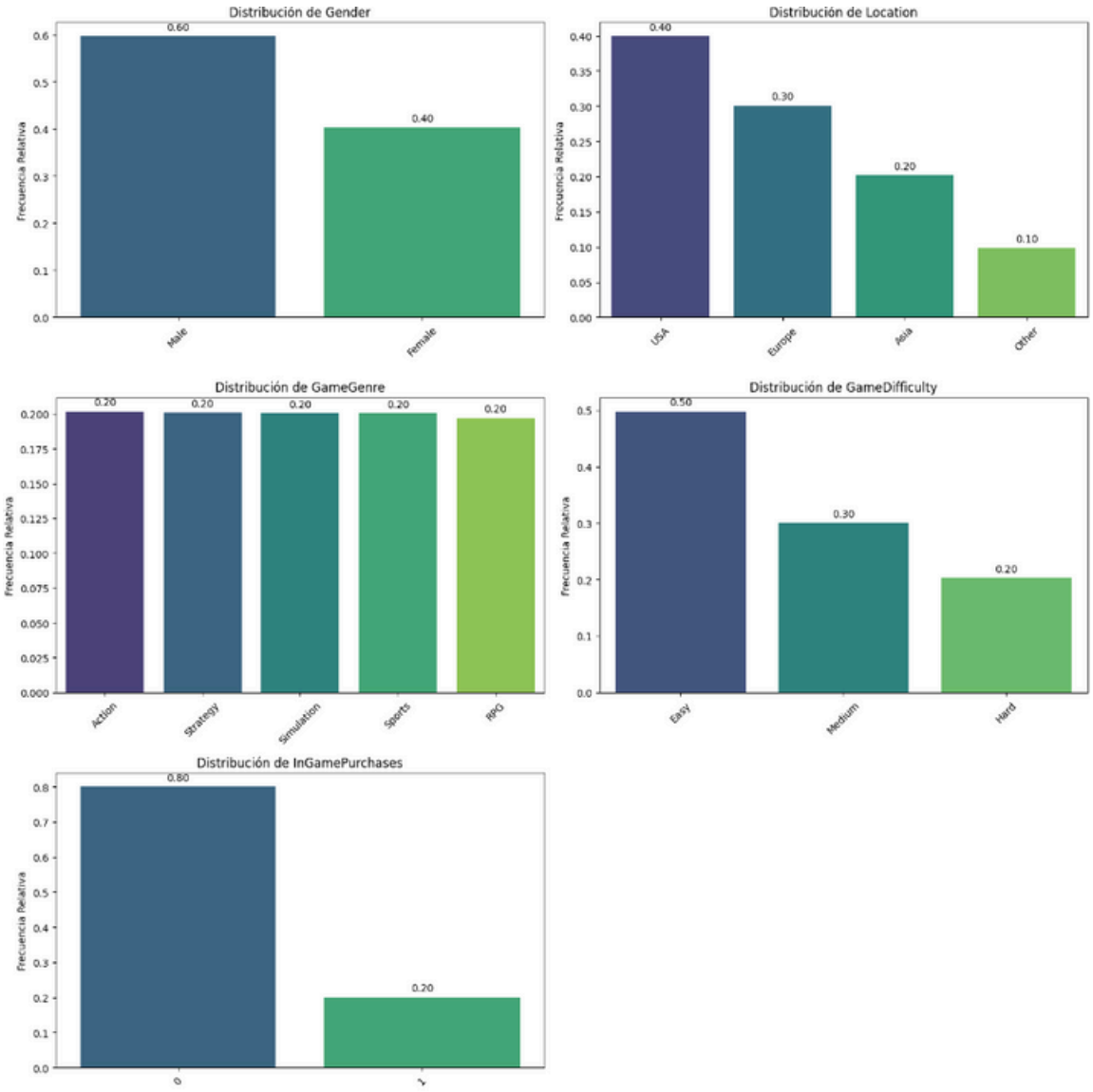


Fig.2 Categorical variables distribution.

Fig.3 Numerical variables distribution in relation to the target

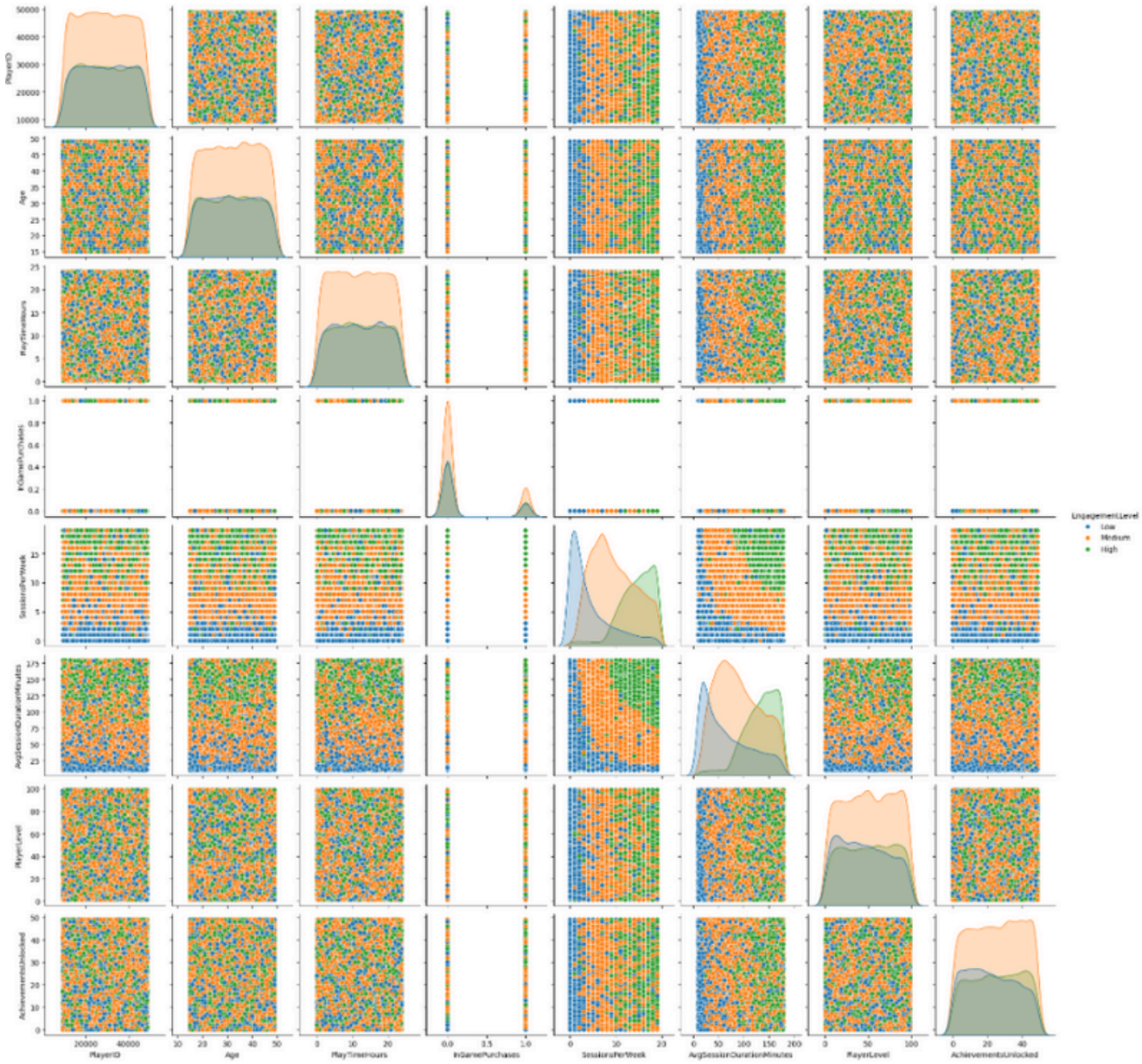


Fig.4 Numerical variables distribution

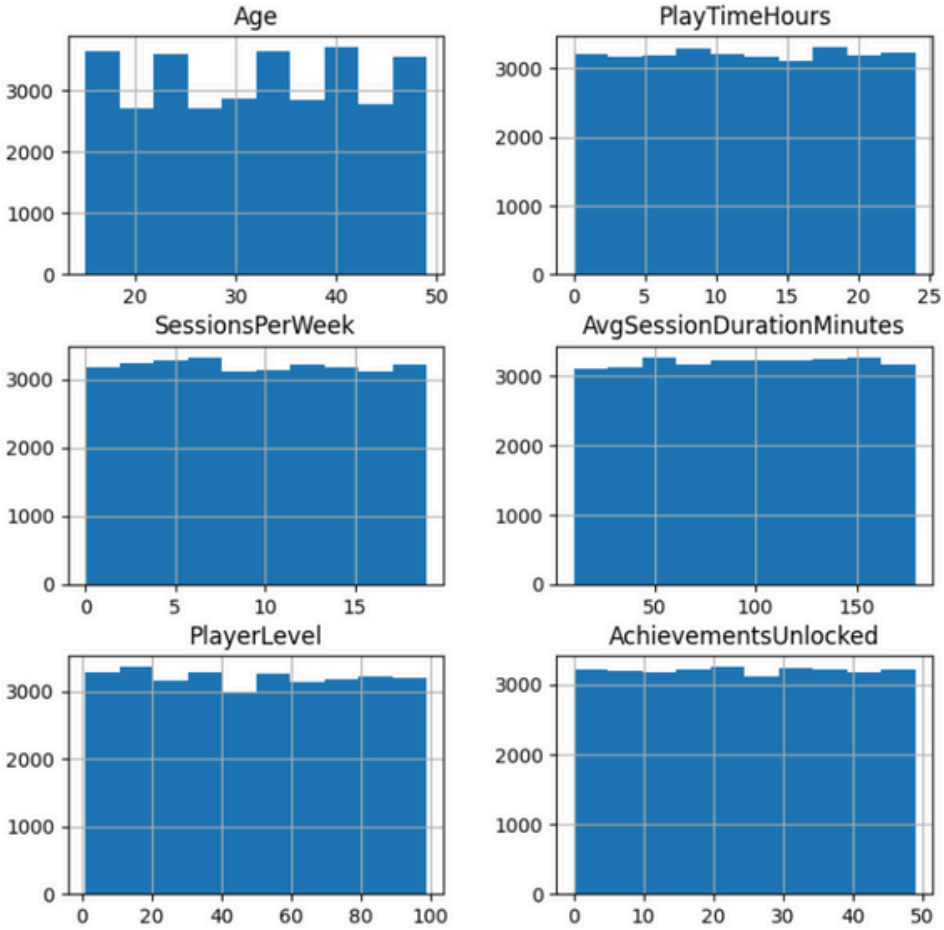
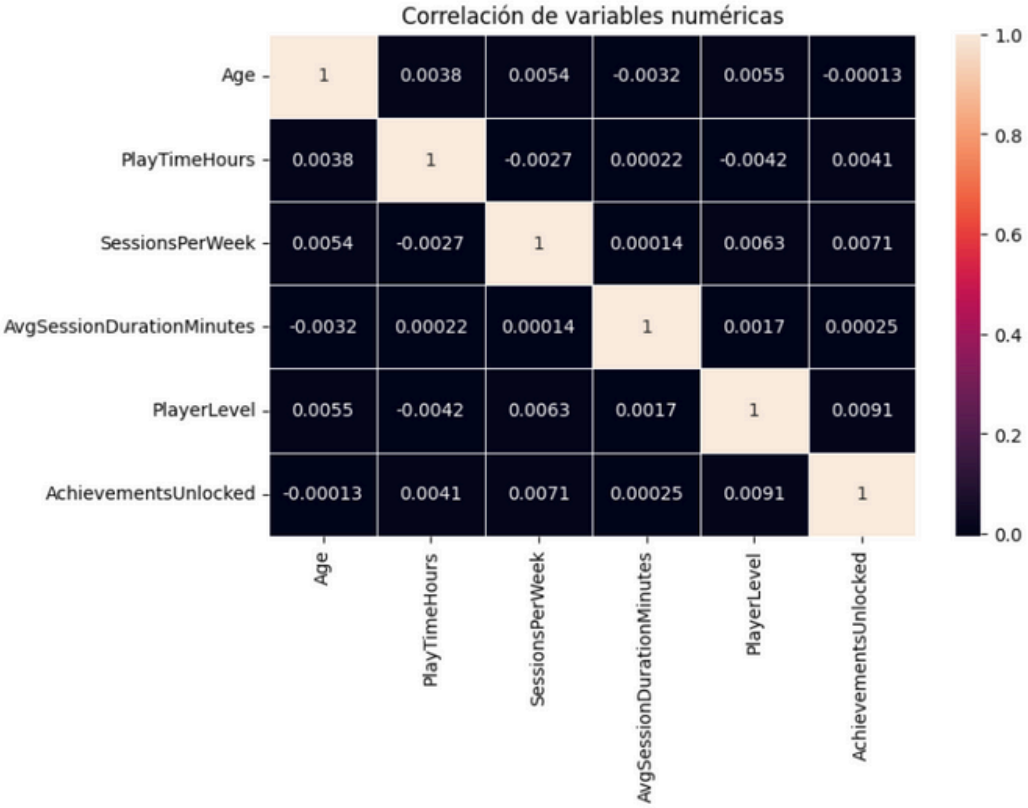


Fig.5 Heat Map and Correlation Matrix



02 Data Processing

- **Numerical variables:** Scaled using MinMaxScaler() to standardise different scales (e.g., age vs. play frequency).
- **Categorical variables**: Encoded with OneHotEncoder(), as none have an excessive number of unique values.
- Preprocessing is implemented using a pipeline.

03 Baseline

We create a pipeline for each model to preprocess data and evaluate metrics using cross-validation (cross_validate) to assess training and validation performance, as well as execution times.

Since our target is imbalanced, we apply StratifiedKFold() to maintain class proportions across folds.

Tested models:

- DecisionTreeClassifier
- RandomForestClassifier
- LGBMClassifier
- XGBClassifier
- LogisticRegressor
- KNN

Initial results indicate high validation and training metrics, especially for tree-based models, suggesting potential overfitting.

Learning Curves of each bease line are shown bellow:

Fig.6 Baseline DecisionTreeClassifier

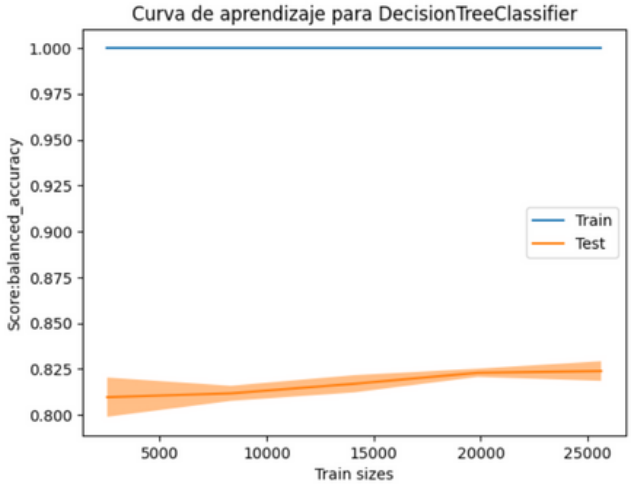


Fig.7 Baseline RandomForestClassifier

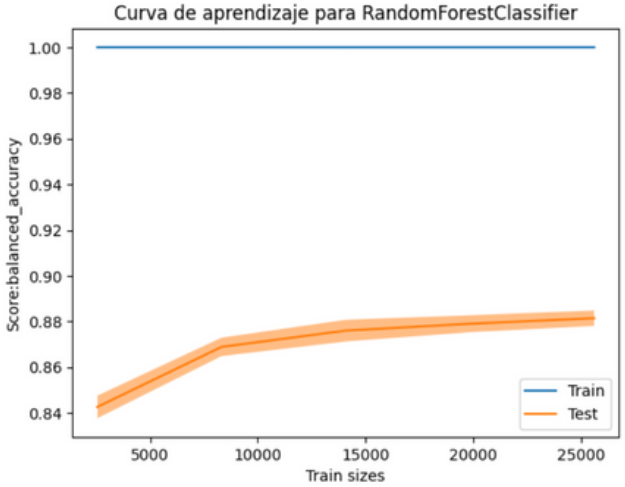


Fig.8 Baseline DecisionTreeClassifier

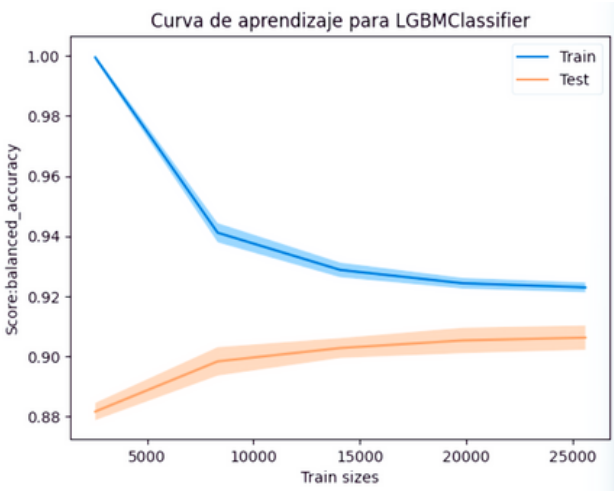


Fig.9 Baseline RandomForestClassifier

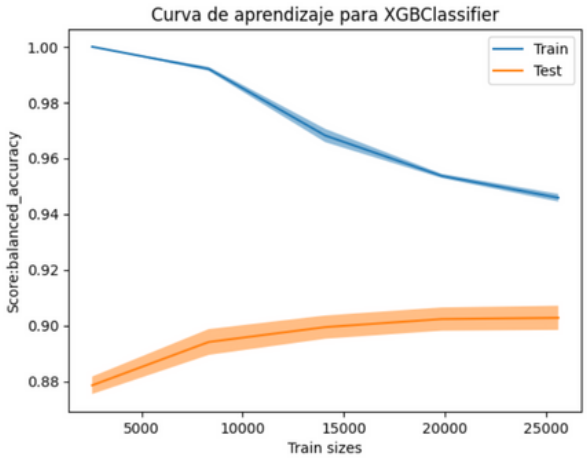


Fig.10 Baseline CatBoostClassifier

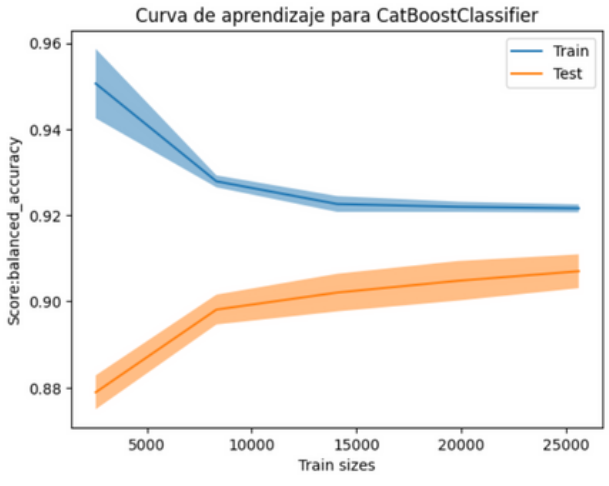
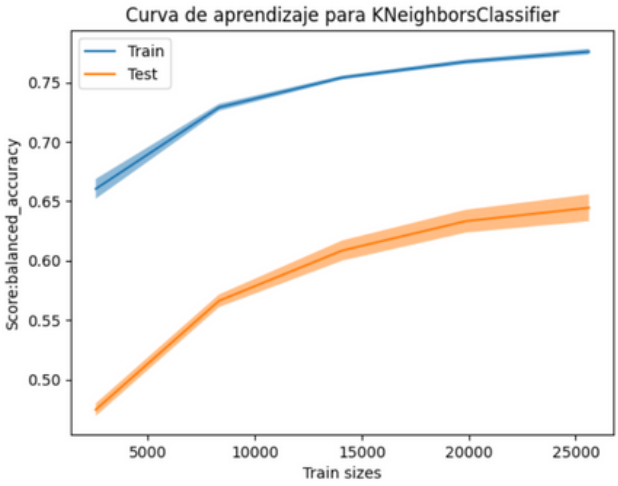


Fig.11 Baseline KNearestNeighbors



04 Model Comparison: Optimisation, Learning Curves and Evaluation

04.A LGBMClassifier

First model:

- Feature selection using model-based methods.
- RandomOverSampler() applied.
- GridSearchCV() used for hyperparameter tuning.
- Result: Overfitting tendency in learning curves.

Fig.12 Learning Curve first model LGBM

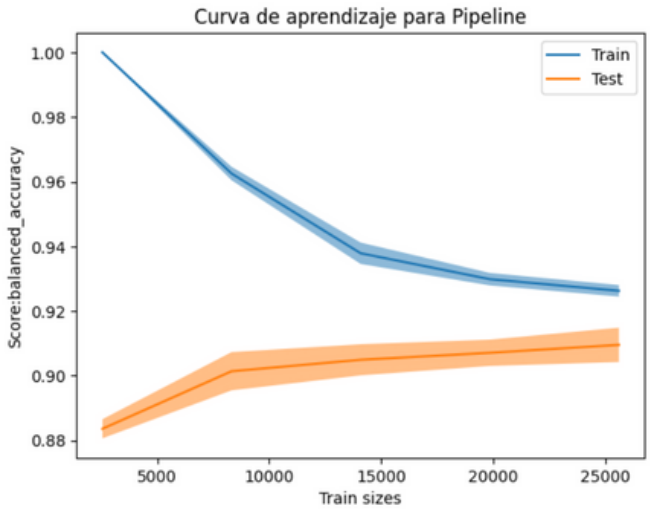
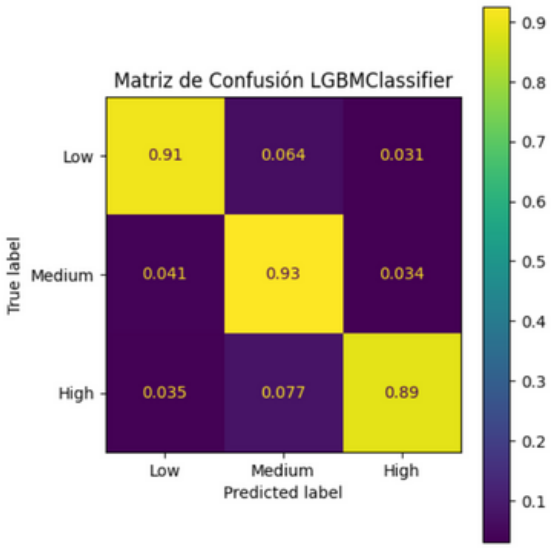


Fig.13 Confusion Matrix first model LGBM



Second model:

- Trained with all features.
- Used class_weight='balanced' instead of oversampling.
- Limited tree depth and leaf nodes.
- Better generalisation without sacrificing accuracy.

Fig.14 Learning Curve second model LGBM

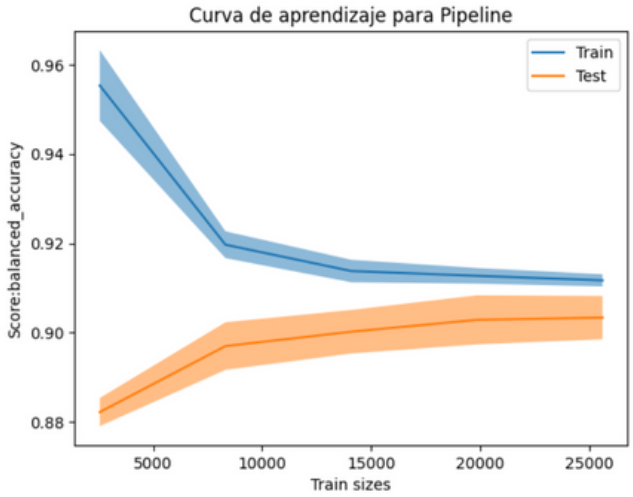


Fig.15 Confusion Matrix second model LGBM

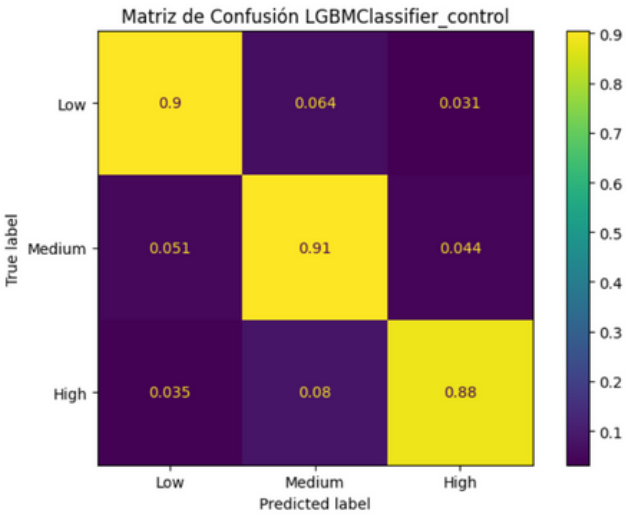
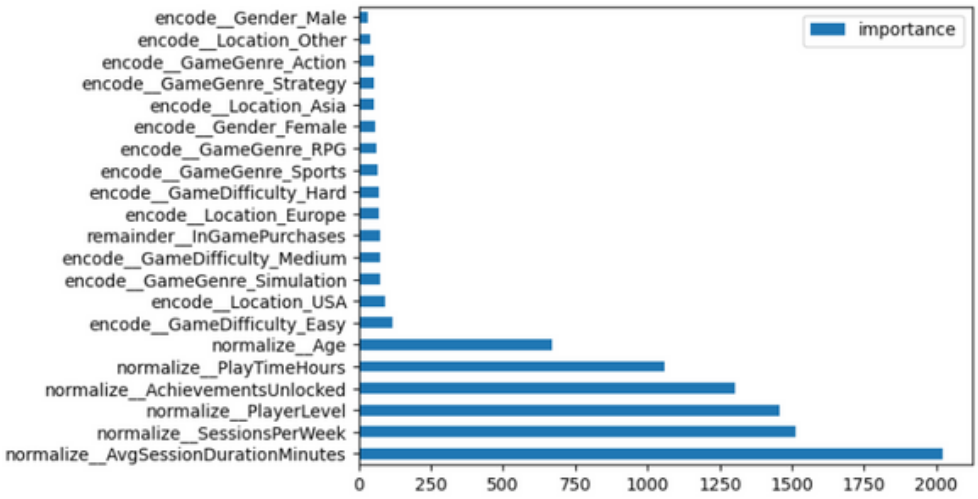


Fig.16 Feature Importance LGBMClassifier



04.B LogisticRegressor

- Applied L1 and L2 regularisation, but cross-validation showed no preference for regularisation.
- Performs well despite limitations, with errors concentrated in the majority class.

Fig.17 Coef Logisticregressor

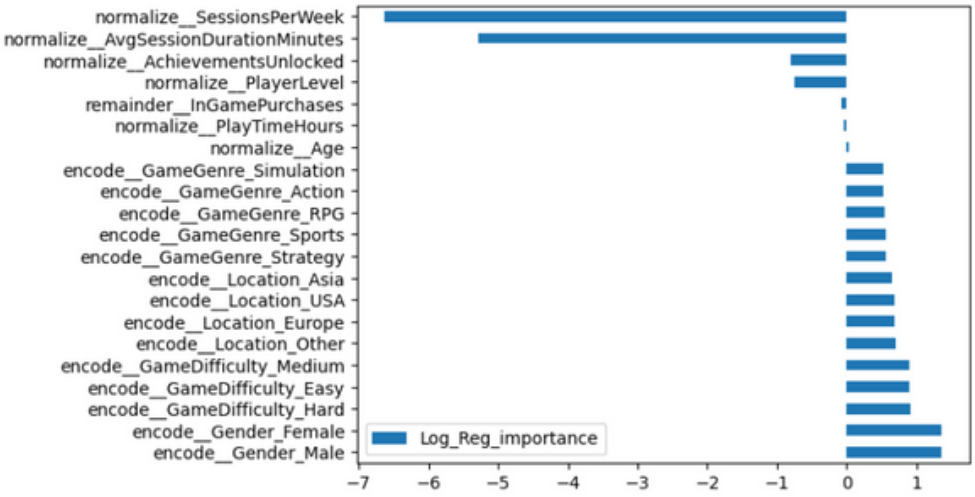


Fig.18 Learning Curve LogisticregRegressor Optimized

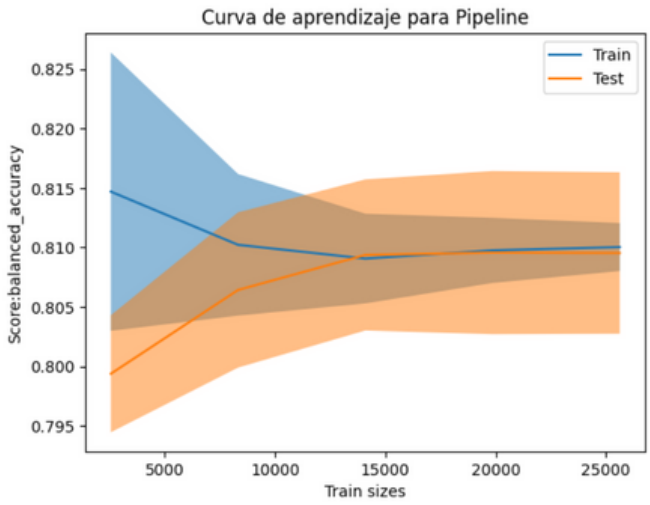
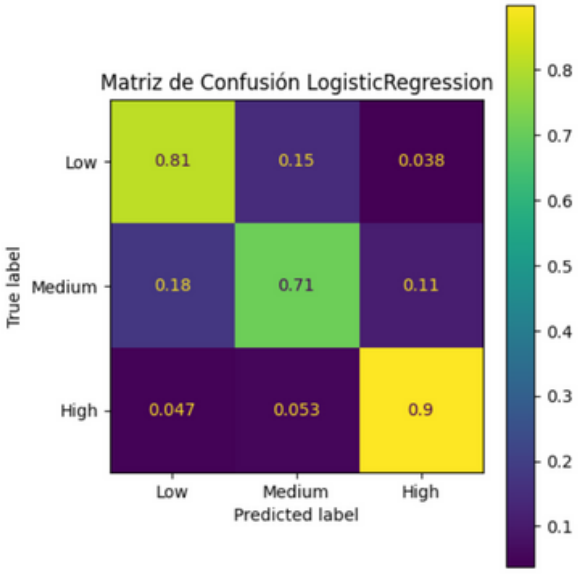


Fig.19 Confusion Matrix LogisticregRegressor Opt.



04.C RandomForestClassifier

- **First optimisation:** Used oversampling, feature selection, and tuned hyperparameters. However, generalisation did not improve, and execution time increased.
- **Second optimisation:** Removed oversampling and applied class_weight balancing, resulting in better generalisation.
- **Third optimisation**: Increased depth and nodes beyond 50 trees, leading to overfitting.
- **Best version:** Second optimisation.

.

Fig.18 Learning Curve RandomForest _1

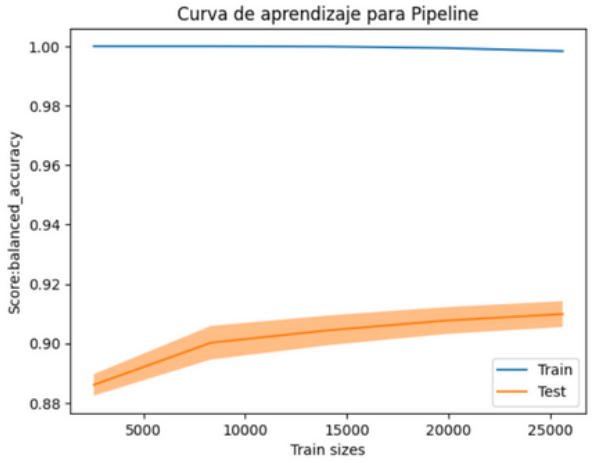


Fig.19 Learning Curve RandomForest _3

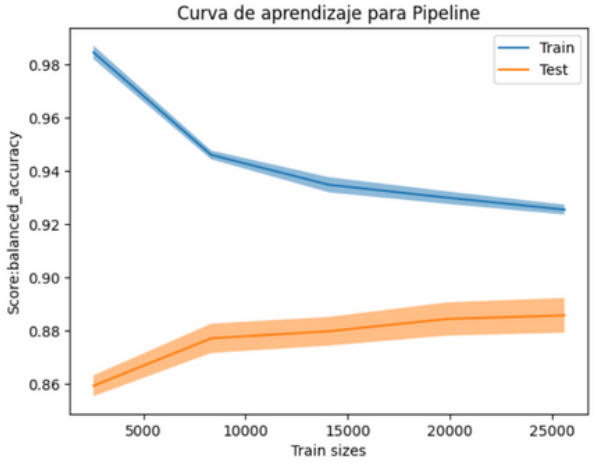


Fig.20 Learning Curve RandomForest _2

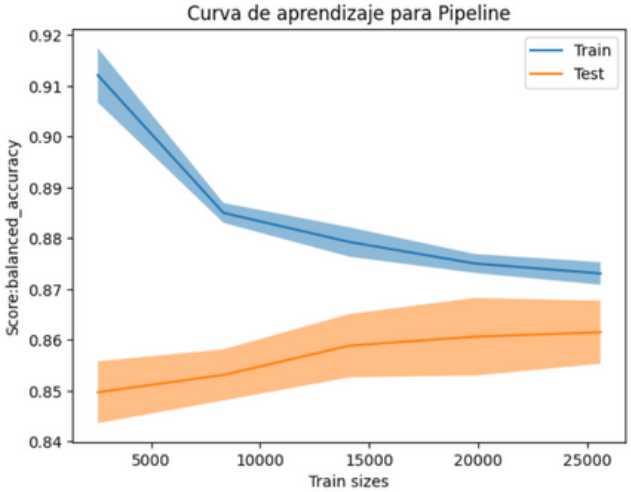
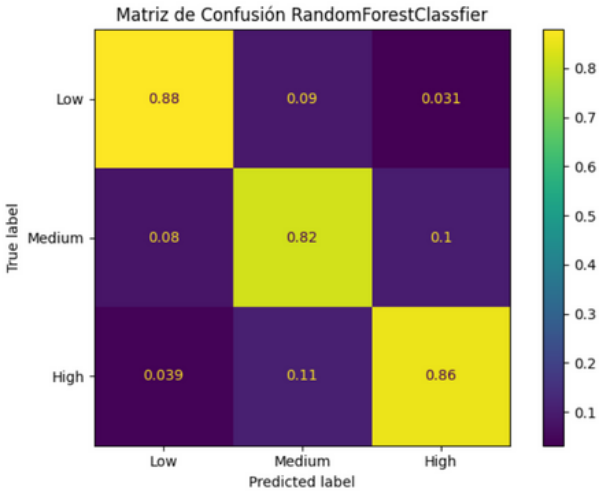


Fig.21 Confusion Matrix RandomForest _2



04.C Red Neuronal Simple (MLP)

- Designed a one-hidden-layer neural network.
- Used compile_sample_weight and compile_class_weight to handle class imbalance.
- Similar predictive performance to LGBMClassifier, with faster execution time and no overfitting.

Fig.22 Arquitrcture MLP

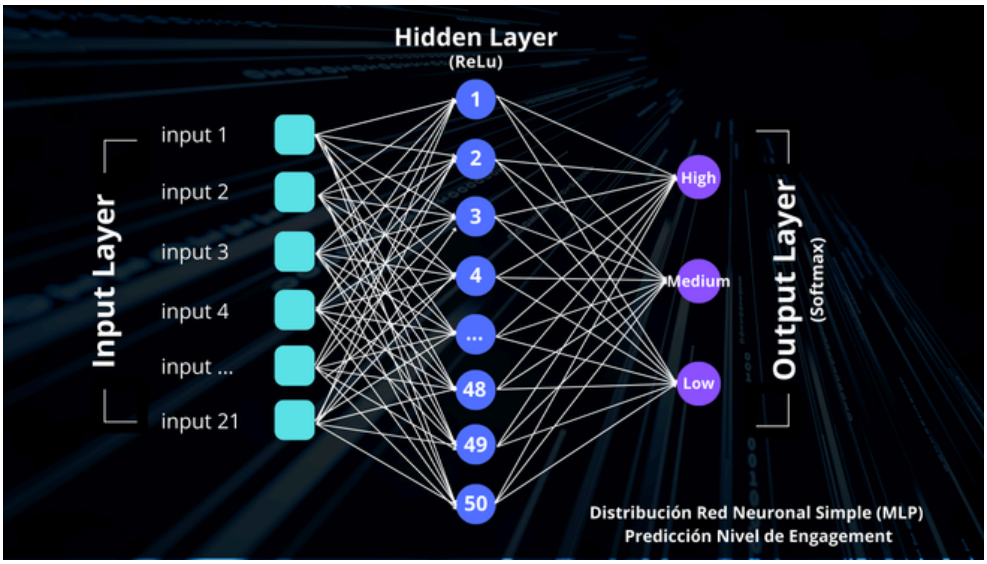


Fig.23 Curve Accuracy / Val_Accuracy

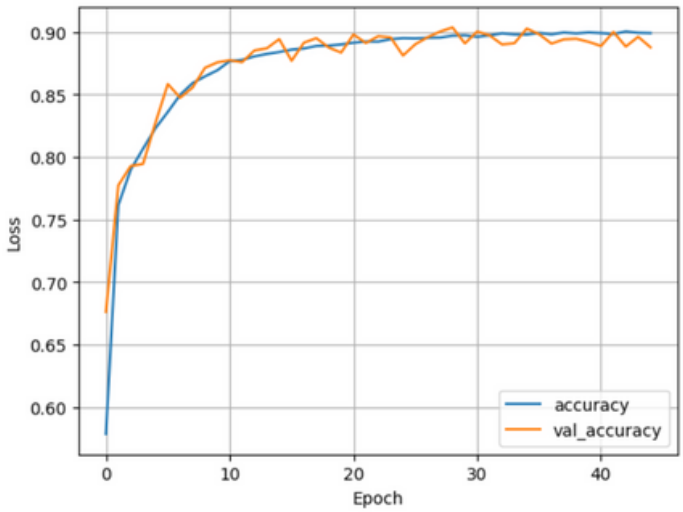


Fig.24 Curve Loss/ Val_loss

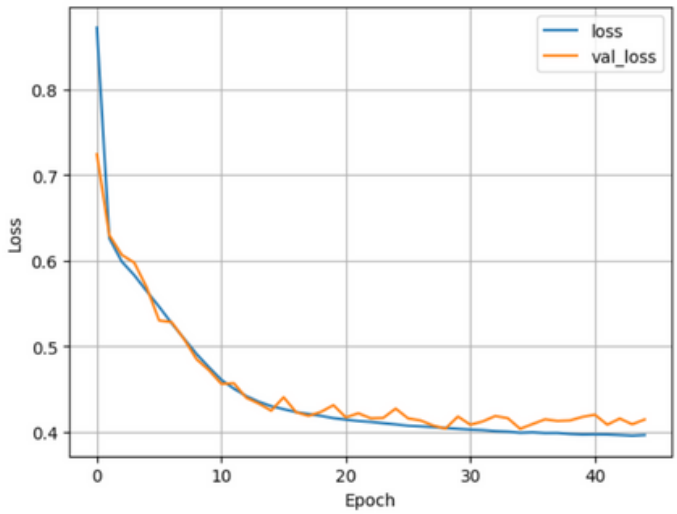
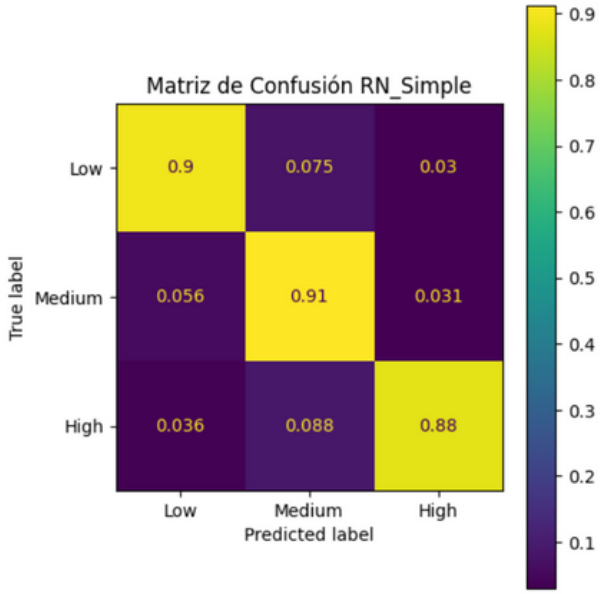


Fig.25 Confusion Matrix MLP



05 Conclusiones

- Tree-based models require careful tuning due to their high adaptability and tendency to overfit.
- Logistic Regression delivers good results at lower computational cost, despite its limitations.
- Neural Networks (MLP) match the best-performing model while reducing execution time.
- Feature selection was not beneficial, as models performed better with all features, even those with lower importance.

Bootcamp Data Science
Blanca Novella Serrano
2024

Supervised Machine Learning
Predicting Engagement Levels in Online Games