Project Summary-Group 6

Introduction

This project explores how machine learning can be used to predict and understand player engagement in online games. Using a synthetic dataset simulating player behavior, we analyzed a variety of factors such as playtime, achievements and others to determine how they influence player retention.

GitHub Repository

Here is the link to the GitHub Project : https://github.com/LogBlast/projectdata_group6

Dataset

This dataset captures comprehensive metrics and demographics related to player behavior in online gaming environments. It includes variables such as player demographics, game-specific details, engagement metrics, and a target variable reflecting player retention.

Features:

- **PlayerID:** Unique identifier for each player.
- **Age:** Age of the player.
- **Gender:** Gender of the player.
- **Location:** Geographic location of the player.
- **GameGenre:** Genre of the game the player is engaged in.
- **PlayTimeHours:** Average hours spent playing per session.
- **InGamePurchases:** Indicates whether the player makes in-game purchases (0 = No, 1 = Yes).
- **GameDifficulty:** Difficulty level of the game.
- **SessionsPerWeek:** Number of gaming sessions per week.
- **AvgSessionDurationMinutes:** Average duration of each gaming session in minutes.
- **PlayerLevel:** Current level of the player in the game.
- **AchievementsUnlocked:** Number of achievements unlocked by the player.
- **EngagementLevel:** Categorized engagement level reflecting player retention ('High', 'Medium', 'Low').

Usage:

This dataset is suitable for exploring patterns in online gaming behavior, developing machine learning models for player engagement prediction, and conducting research in gaming analytics.

Preprocessing

a) Data quality assessment

First, as we've seen in class, we checked for missing values using the method ".isnull().sum()" →

We also verified the absence of duplicate record with the method ".duplicated().sum()"

```
PlayerID                    0
Age                         0
Gender                      0
Location                    0
GameGenre                   0
PlayTimeHours               0
InGamePurchases             0
GameDifficulty              0
SessionsPerWeek             0
AvgSessionDurationMinutes   0
PlayerLevel                 0
AchievementsUnlocked        0
EngagementLevel             0
```

Then, we found out that some part of the data was incorrect : "How can a player have an average time per session of X minutes with 0 sessions per week ?"

So we dropped the corresponding rows and checked all the other "critical" numerical features.

```
In 'PlayTimeHours', there are : 0 invalid rows.

In 'SessionsPerWeek', there are : 1967 invalid rows.

In 'AvgSessionDurationMinutes', there are : 0 invalid rows.

In 'PlayerLevel', there are : 0 invalid rows.
```

Finally, we visualized quick data distributions to detect potential typos or outliers, but, in the end, the dataset demonstrated high quality with minimal cleaning requirements.

"This dataset is synthetic and was generated for educational purposes, making it ideal for data science and machine learning projects."
- Mr. Rabie El Kharoua (owner of the dataset)

b) Data type standardization

We clearly identified numerical and categorical data at first.

```
numerical_columns = df.select_dtypes(include=['number']).columns.to_list()
categorical_columns = df.select_dtypes(include=['object']).columns.to_list()
```

Then, we ensured consistent numerical formatting with ".to_numeric".

In addition, we chose to convert floats to integers, as they represented discrete counts (ex : playtime in hour) rather than continuous measurements. (ex : 'float' to 'int' in that particular case)

Finally, we checked again for missing values or "NaN" values that could have appeared during the previous operations.

c) Categorical Data Encoding

```
df_dummies = pd.get_dummies(data=df[[column for column in categorical_columns if column != "EngagementLevel"]], dtype=int)
```

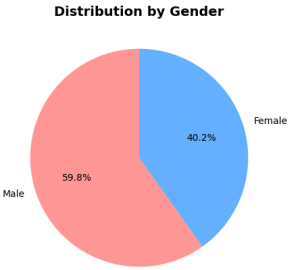
We one-hot encoded all the categorical columns using the method ".get_dummies()", all in order to get a faster and stronger model. We excluded 'EngagementLevel', our target variable for the supervised learning task.

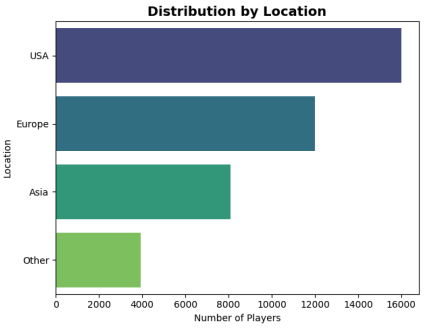
d) Ready for data visualization

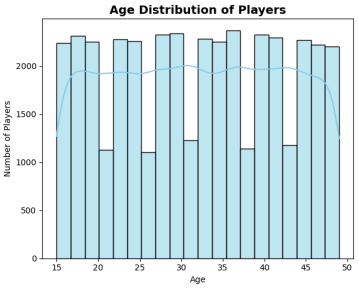
The preprocessing pipeline successfully transformed raw mixed-type data into a clean, readable, numerical dataset (without entering the curse of depth -> only 20+ columns) for future operations.

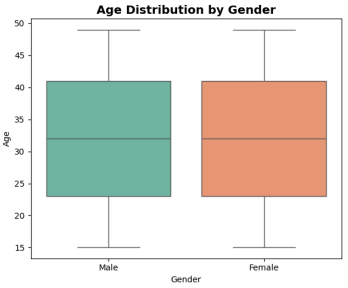
Data Visualization

First, we decided to make some demographic analysis on the players we have in our dataset:



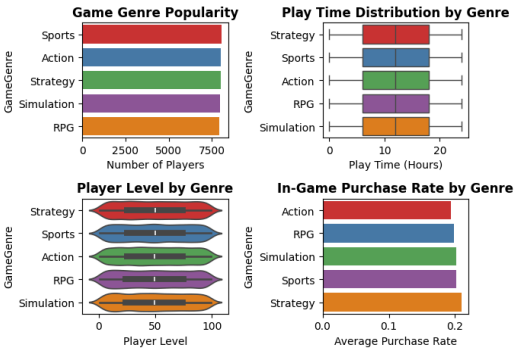


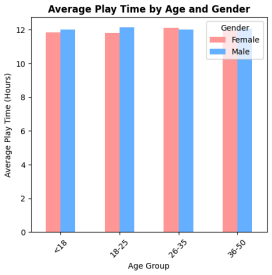


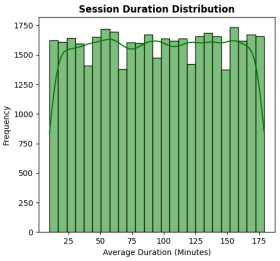


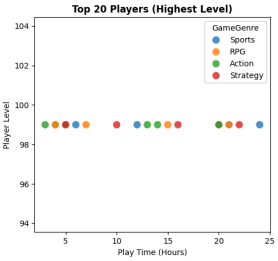
Then, we made some different visualizations on multiple criterias to have a better understanding of the data and identify patterns and trends.

There were some graphics that weren't very interesting where we found no highlights:

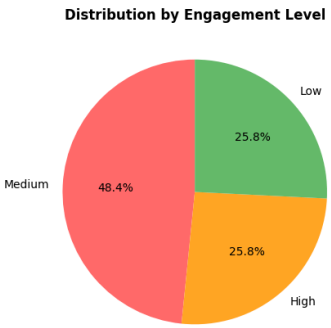


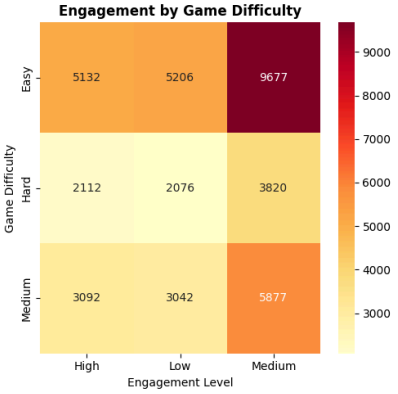


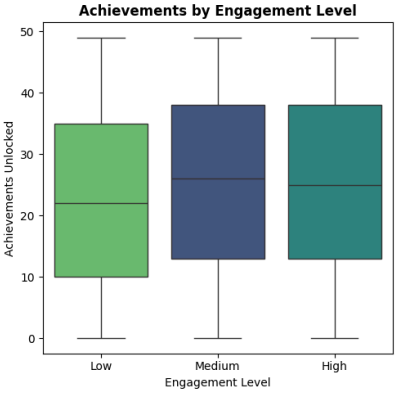




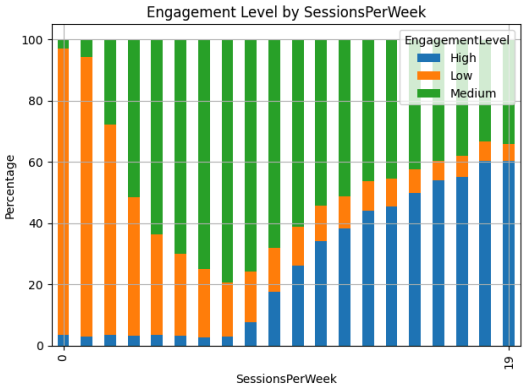
But on some graphics, we could see that the EngagementLevel causes some variation when coupled with other criterias:

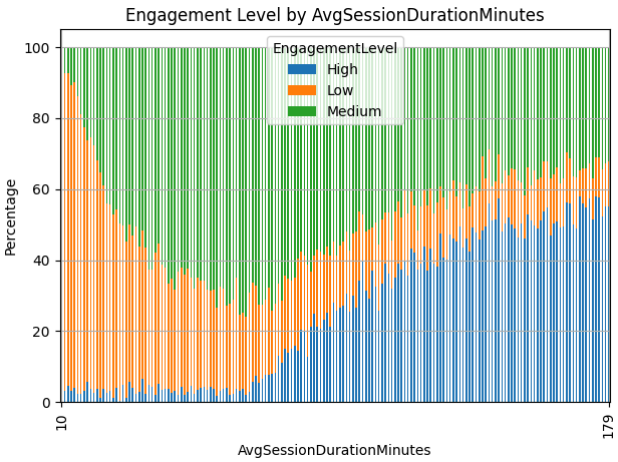




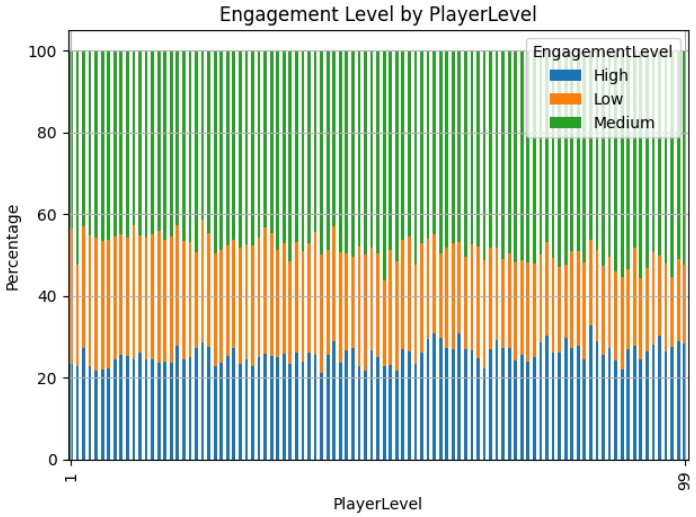


So we tried some other combinations with EngagementLevel, and found some other interesting patterns like those ones with SessionPerWeek and AvgSessionDurationMinutes where the higher they are, the higher the EngagementLevel was:

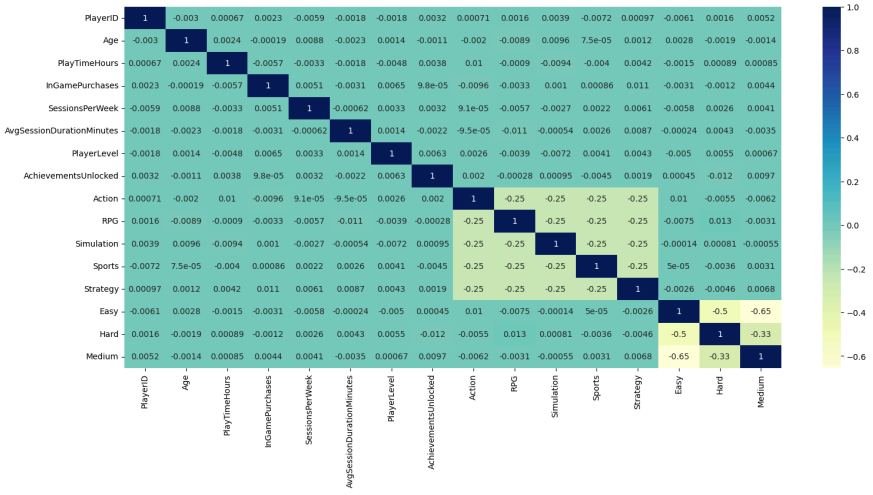




And we also had surprises like this one where we could think that the more the PlayerLevel is high the more the EngagementLevel should be, while it's not the case:



Finally, we built the heatmap of the dataset:



Feature Engineering

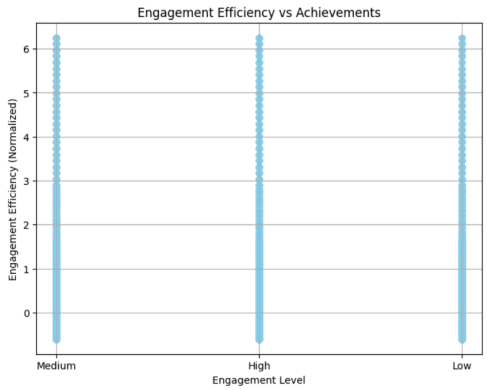
a) New feature : “Engagement Efficiency”

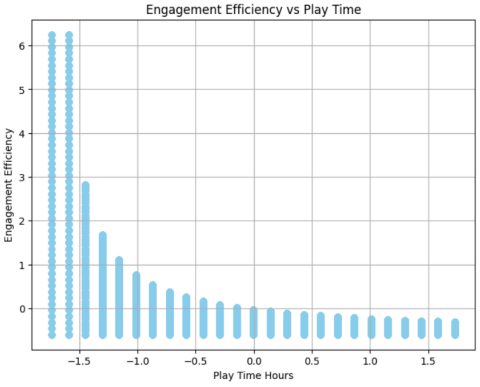
```
df['EngagementEfficiency'] = df['AchievementsUnlocked'] / df['PlayTimeHours'].replace(0, 1)
```

We’ve seen that data linked to “playtime” were probably the most important features. That’s why we created a feature to analyze how players use their time while playing.

We found out that playtime features were probably the most important in our data. So we created a new feature to see how well players use their gaming time. High scores mean players who rushed achievements, while low scores show players taking time to progress.

This feature was created to help us determine if duration of playtime is the only key or if the way you play matters.





.

b) New feature : “SessionIntensity”

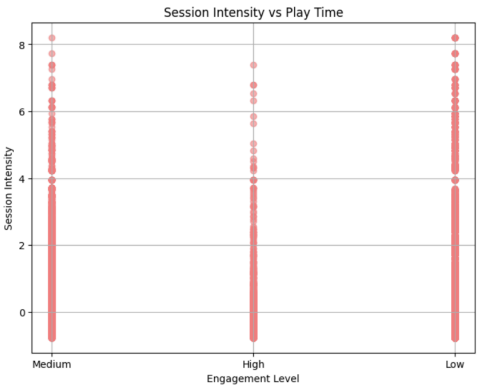
```
df['SessionIntensity'] = df['SessionsPerWeek'] / df['AvgSessionDurationMinutes'].
```

We wanted to divide the players into two groups:

- Marathon players, who play fewer times but for longer periods
- Casual players, who play more often but for shorter periods.

The closer it gets to 0, the more ‘intense’ the player is.

This feature aims at demonstrating the importance of ‘intense’ sessions in order to get players engaged in a game. (cf “Daily quests” & “Battle Pass” in many games)



We tried a validation with Pearsonr from Scipy, without great success.

c) Data normalization

Our data had different scales : some features, such as ‘InGamePurchases’ went from 0-1 while others, like ‘PlayerLevel’, went from 1-100. Since we didn't find many extreme values (outliers), we used standard scaling.

This makes all features equal in importance, helping the model to work better, and makes training faster.

```
scaler = StandardScaler()
numerical_features = ['PlayerLevel', 'PlayTimeHours', 'AvgSessionDurati
df[numerical_features] = scaler.fit_transform(df[numerical_features])
```

Machine Learning

In this section, we built a machine learning model to predict the players' Engagement Level (EngagementLevel) based on the other features in the dataset.

a) Data Splitting and Target Selection

We defined the EngagementLevel column as the target variable, which is a multiclass label (High, Medium, Low). Also, we have removed non-numerical or irrelevant columns from the dataset (e.g., Gender, Location, etc.) which had already been encoded using one-hot encoding during preprocessing.

```
# Split X and y
X = df.drop(columns=[target_column, 'Gender', 'Location', 'GameGenre',
'GameDifficulty'])
y = df[target_column]

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

b) Random Forest Model Training

At the beginning, we have tested linear regression (adapted to classification via logistic regression), but the results were misleading:

- It output probabilities or continuous values, which had to be manually thresholded.
- Despite showing a high accuracy (close to 0.9), the model failed to capture class imbalances and yielded biased predictions toward the majority class.

That's why we opted for Random Forest Classifier. It can handle multiclass classification natively, and it's more robust to overfitting when properly tuned.
A Random Forest allows non-linear decision boundaries and naturally provides feature importance scores, helping us interpret results.

We trained our Random Forest with the following parameters:

- n_estimators=200: limits the number of trees to avoid overfitting.
- max_depth=15: restricts tree depth to reduce complexity and improve generalization.
- min_samples_split=15 and min_samples_leaf=8: prevent the trees from learning noise by forcing minimal group sizes at splits and leaves.

c) Model Evaluation

We used several key metrics to evaluate our model:

- Accuracy: the overall proportion of correctly predicted labels.

```
Accuracy: 0.9003152088258471

Classification Report:
              precision    recall  f1-score   support

        High       0.92      0.86      0.89      2053
         Low       0.91      0.82      0.86      1697
      Medium       0.89      0.95      0.92      3864

    accuracy                           0.90      7614
   macro avg       0.91      0.88      0.89      7614
weighted avg       0.90      0.90      0.90      7614
```

- Macro F1-score: the unweighted average of F1-scores per class; it gives equal importance to all classes, which is helpful when classes are imbalanced.

```
Macro F1-score: 0.8912123820560104
```

- Micro F1-score: calculates F1 globally by counting the total true positives, false negatives, and false positives.

```
Micro F1-score: 0.9003152088258471
```

From our results, the accuracy was moderate, showing balanced performance. The macro F1-score was slightly lower, indicating the model struggled slightly with minority classes. Finally, the Micro F1-score was closer to accuracy, confirming that most predictions were correct in total count.

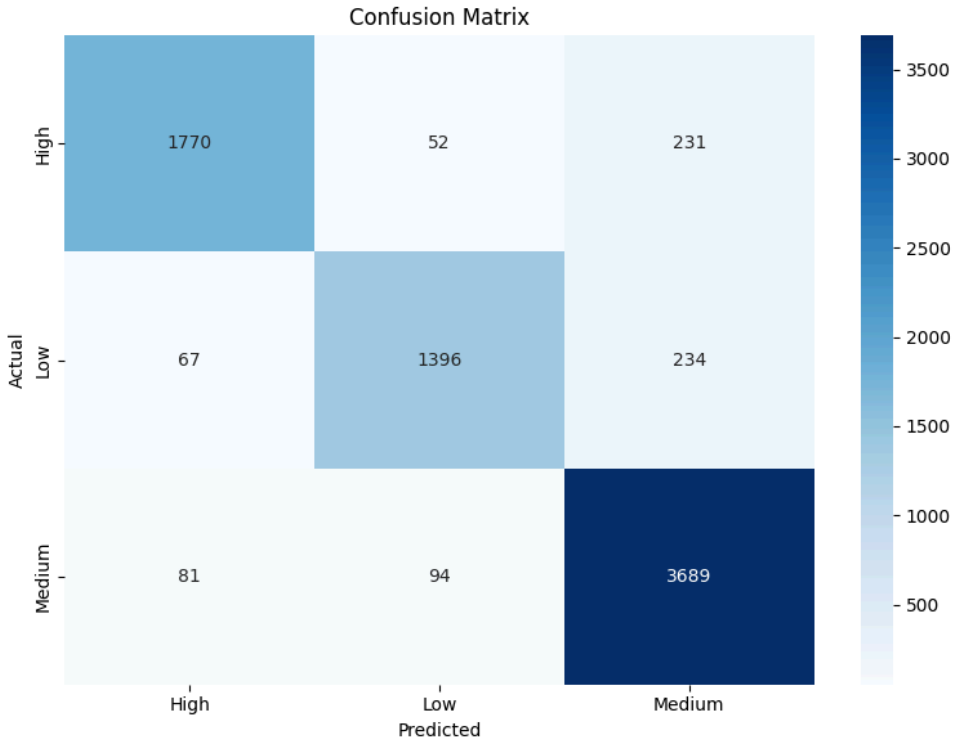
d) Confusion Matrix

A confusion matrix helps visualize the number of correct and incorrect predictions for each class. It is particularly useful for understanding:

- Which classes are often confused (e.g., Medium predicted instead of High).
- If the model is biased toward a particular class.

In our heatmap below:

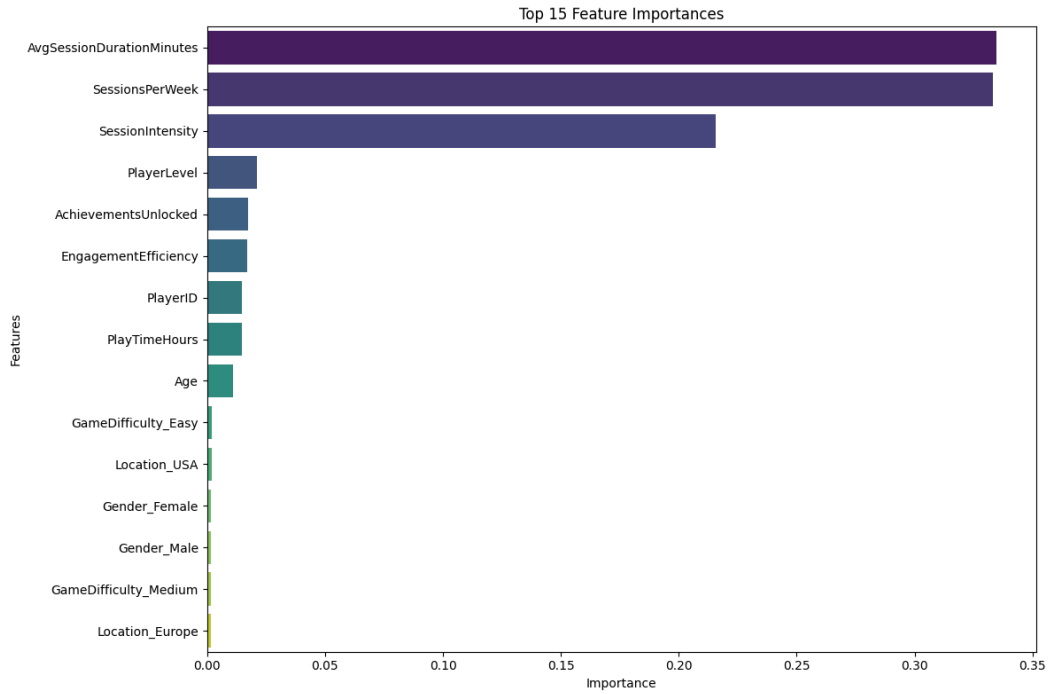
- Diagonal values indicate correct predictions.
- Off-diagonal values show misclassifications.



We noticed that the model frequently misclassifies both High and Low engagement as Medium, likely due to overlapping behavioral patterns that fall between extremes.

e) *Feature Importance (Top 15)*

The model provides a direct estimate of feature importance, which tells us how much each variable contributes to the prediction task.



In the graphic, we can see some features who are dominant :

- AvgSessionDurationMinutes : longer, more regular sessions typically imply higher engagement.
- SessionsPerWeek: highly correlated with player engagement.
- SessionIntensity : This metric measures if a player has many short sessions (high value) or few long sessions (low value).

This information can be used to optimize player retention strategies by focusing on session frequency and duration.

f) Cross-Validation

To ensure our model generalizes well, we used 5-fold cross-validation on the training set:

```
cv_scores = cross_val_score(model, X_train, y_train, cv=5,
scoring='accuracy')
print(f\"Cross-validation Accuracy: {cv_scores.mean():.3f} (+/-
{cv_scores.std() * 2:.3f})\")
```

```
Cross-validation Accuracy: 0.894 (+/- 0.006)
```

We can see that the average accuracy was consistent with our test set, proving the model stability. Also, the standard deviation was low, who indicate that the performance was robust across different data splits.

AI Responsibilities

As part of building a machine learning model that analyzes human behavior (player engagement), it is essential to address the ethical considerations and responsible AI practices that guided our work.

1. Fairness and Bias

To prevent discrimination, we excluded features like Gender and Location from model training. This ensures that engagement predictions are based on behavior rather than personal attributes.

2. Interpretability

We chose a Random Forest model not only for accuracy, but also for its transparency. Feature importance helped us understand which behaviors influence engagement, making the model explainable to non-experts.

3. Ethical Use of Data

The dataset is entirely synthetic and created for educational use. No real player data was used, ensuring privacy and avoiding ethical concerns around data consent or misuse.

4. Responsible Application

Our goal is to help improve player experience. This model can assist in identifying disengagement patterns.

5. Reproducibility

All steps (preprocessing, training, evaluation) were carefully documented and validated using cross-validation, ensuring the results are reliable and reproducible.