Machine Learning - Group Assignment (Group 2)

(*Codalab Group Name: emreguler*) (*Members: Kexin Wang, Konstantina Lampropoulou, Emre Guler*)

1. Feature Engineering (and Experiments)

In our machine learning project, the initial step in feature engineering involved handling missing values and irrelevant features. Initially, we removed all rows containing null values and excluded the *UserID* and *TimeUtc* features from the dataset. We applied one-hot encoding to all columns related to game modes and tasks. This approach yielded a validation Mean Absolute Error (MAE) score of approximately 162 and a test MAE score of around 175, surpassing the competition baseline. However, this method resulted in significant information loss due to the exclusion of rows with missing values.

To address this and also improve further in the competition, we implemented a second approach **focused on retaining more information**. We imputed null values in categorical columns with a blank space (' ') and set missing values in the *LevelProgressionAmount* to 0. Additionally, we included integer conversion of *UserIDs*, which, despite the potential risk, was deemed beneficial as our chosen model did not assume linearity in feature relationships. After removing *TimeUtc* again and applying one-hot encoding to the game-related columns, this method resulted in a validation MAE score of around 132 and a test MAE score of 159.

Further feature engineering experiments included several more sophisticated techniques:

- ➢ Datetime Feature Extraction: Extracting features such as day, month, and hour from the *TimeUtc* column.
- ➢ User Statistics: Extracting statistics (minimum, mean, maximum) of *ResponseValue* per user to unique columns.
- ➢ Total Session Length: Deriving total session length by sorting entries by *TimeStamp* and grouping by *UserID*.
- ➢ Imputation: Identifying and imputing different types of null categories (e.g., *New_Session*, *Task_Completed*, *First_Task_Selected*, *First_Task_Started*).
- ➢ Outlier Removal: Implementing various methods such as (i) thresholding *CurrentSessionLength*, (ii) frequency thresholding for *CurrentTask* and *LastTaskCompleted*, (iii) session entry thresholding via manually created *SessionID*, and (iv) light user thresholding based on the number of sessions per user.

These experimental approaches led to validation MAE scores as low as 78. However, they introduced overfitting issues - which we could not solve - resulting in test MAE scores exceeding 200, and thus, were not included in the final model.

2. Learning Algorithm(s)

Given the regression nature of the task, we compared multiple regression models: *LinearRegression*, *DecisionTreeRegressor*, *RandomForestRegressor*, *SGDRegressor*, and *GradientBoostingRegressor*. During the initial phase, *RandomForestRegressor* emerged as the best-performing model, closely followed by *GradientBoostingRegressor*. This observation directed our focus towards *RandomForestRegressor* for further tuning. The superior performance of tree-based models was expected due to their robustness in handling datasets with mixed numerical and categorical features. **Had we performed matrix transformations or vectorization for categorical features to make them have numerical representations**, neural networks or linear regression models might have been more suitable and efficient.

3. Hyperparameter Tuning

After our decision to move forward with *RandomForestRegressor,* we applied *RandomizedSearchCV* with the parameter list of *n_estimators, max_depth, min_samples_split, min_samples_leaf,* and *max_features.* The search for the best test MAE score of 159 resulted with the parameter selection of *{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 4, 'n_estimators': 144}*.

4. Discussion (of the Solution Performance)

While we saw the initial MAE improvements as we fed the models with more information, we could not escape from the overfitting pitfall. The feature engineering solutions above (that led to the overfitting problem) were more focused on the user information which had the narrowest *ResponseValue* intervals among all the features - and therefore led us to believe that a better pattern recognition can be extracted. While this is not a bad idea since over 5,000 users also occur in the test dataset, it lacked the generalization through other features like *CurrentTask* - besides our outlier detection attempts.

5. Codalab Submission

Codalab submissions have been done under the username “emreguler”.

6. Specific Contributions of Group Members

Konstantina Lampropoulou	- 1st & 2nd Approach Imputation & Encoding - Hyperparameter Tuning
Kexin Wang	- 2nd Approach Imputation & Encoding - Predictor Model Selection - Hyperparameter Tuning
Emre Guler	- 1st Approach Imputation Encoding - 3rd Approach Feature Engineering - Model Selection

7. References

- For preprocessing, imputation, encoding, model & feature selection: https://scikit-learn.org/
- For Python syntax and compiler error type problems: https://stackoverflow.com/, https://chatgpt.com/