In this challenge, the objective is to construct a model that helps game designers to identify and give different prices to new users who are likely to spend money. I treated this problem as a binary classification on whether the users will spend or not. Then, I inferred the probabilities of spending money by the classification of the user groups given the features of a user's gaming profile, so that game designers can give higher prices to users who have high probability to spend money.

I spent 70% of my time doing data exploration, feature selection, and feature engineering on three given data files respectively. Firstly, I identified that each feature belongs to categorical variables or continuous variables.

For each categorical variable, I checked the number of their levels. If the feature has a large number of levels, and it does not have good business value, I will drop these features. For example, in ka_devices tables, I dropped device model and device operating system features as they have over 10k levels and do not have great values to the model based on domain knowledge. If the feature has an appropriate number of levels, to reduce model complexity, I grouped the levels of features whose frequency are less than 1% of length of observations as 'Other'.

For each numerical variable, I checked their missing values. In ka_actions, almost half of the observations in the features 'game_stats_tutorial_complete' and 'game_stats_tutorial_complete_time' were missing. Instead of removing or imputing these missing values, I dropped these two features because there was no intuitive way to replace these missing values. I plotted the box plot and density distribution of the rest of the numerical variables. I removed the outliers and since all of their distributions were right-skewed, I imputed the missing values with their median. For other features with a small number of missing values, I remove the observations as it did not affect the data significantly.

In addition, I used one hot encoding to handle categorical variables and standardized numerical variables in training data and test data respectively, so that mean and variance of training data would not affect test data.

After finishing with feature selection and engineering, I joined three tables and converted total_spend to a binary class as 'whether_spend'. I checked the proportion of the target's two classes, the proportion of users who had spent money was around 1.14% while 98.86% of users did not spend money. Therefore, this was a highly imbalanced data set. I used three strategies to address this issue.
1. I used stratified sampling technique when splitting data into training data and test data, so that binary target labels can have the same proportion in training and test dataset.
2. I combined random oversampling from the minority class using SMOTE and random undersampling from the majority class on training data only.
3. I used confusion matrix, specificity score, and AUC-ROC score as my evaluation metric to evaluate models' success.

Finally, the data is ready for the modeling. I trained Random Forest as my baseline model. Then, I compared RF with Logistic Regression, Gaussian Naive Bayes, and LigthtGBM model. I found that Logistic Regression did the best job. Then, I used Randomized Search Cross Validation to tune the hyperparameters of the Logistic Regression classifier and found the set of hyperparameters that had the best recall score. In comparison to the baseline model, the tuned Logistic Regression classifier performed better on test data.