Problem Formulation

The problem formulation is not a major aspect of our solution, but we feel it is worth mentioning.

This competition can be formulated in many ways. It was given in the problem statement that an active user is one that enters a competition, makes a submission, or engages through the discussion forums. The target variable is an outcome of four different activities, and this gave us an opportunity to try out different formulation techniques as discussed below:

1. We solved a binary classification problem. The outcome is 1 if user is active on a given month else 0. This formulation might be used by most and it worked as charm for us as well.
2. We solved a forecasting problem. The idea was to predict the number of different activities that a user might perform in the next three months. In order to avoid multiple models at each user level, we formulated the problem as a regression problem and used lag features, time features and other extraneous variables for model training. The performance of this approach was good enough but it didn't beat the performance of approach #1.
3. We decided to solve the problem as multi-class classification problem where each class is the four different activities performed by the user. This approach was competing well with approach #1. We improved the approach further by combining low frequency classes. We finally created three main classes as described below:
    a. A class to determine if a user participated in a competition
    b. A class to determine if a user submitted, discussed or commented
    c. A class if none of the above activities was performed by the user

    This approach gave us "slightly" better performance than approach #1 and we decided to stick to it. **Please note that we can solve the problem as binary classification problem and it should give almost similar results.**

Data Preparation

The steps for data preparation are given below:

1. Loading Train and Test data and merging them together
2. Reading the users data, competition participation, submissions, discussion and comments data and renaming the columns accordingly
3. Merging the complete data with each of the above data
4. Performing feature engineering as and when needed

Feature Engineering

Most of the time was spend in feature engineering. To understand the below features, consider an activity by a user to be a competition participation, submission, discussion, or comment. Separate features were created for each activity.

The list of some of the key features is given below:

1. The number of activities by a user in the previous month
2. The momentum in the number of activities was captured by taking a difference of the recent month activity from the previous month
3. The cumulative sum of the number of activities by the user was calculated and the number of activities per month was derived for all activities except submissions.
   Instead of just using number of submissions, we used number of submissions per competition which gave slightly better score.
4. The number of months since the last activity by the user
5. The number of months of the last activity since joining zindi
6. The number of active competitions of a user in the given month
7. The rank of the user interest in the competitions in the given month based on the user's competitions history.
   The steps to determine this feature is shared in the next page
8. The number of months since joining zindi
9. The number of total users, new users in each month
10. The mean, deviation and max experience of users in each month
11. Count based features for country and user_id
12. User based feature like featureX, featureY, country and points
13. Time based features like year and month

The rank of user interest for each month can be determined using below steps:

1. Take the competitions data (the best part is that this data has information about the future competitions as well). We convert this data into a format which looks like One Hot Encoding (OHE). So, we now have CompID and all of its features say FeatureA_1, FeatureA_2... etc. We selected only 5 competition features (FeatureA, FeatureB, FeatureC, FeatureD, and FeatureE).

2. Get all the competitions happening in each month. Merge this data with data in step #1. We then add the competition features for each month and generate a matrix [timestamp x competition_features]

3. Concatenate train and test data. Merge this data with data in step #1. We add the competition features for each user and generate a matrix [user_id x competition_features] across different timestamp

4. Perform matrix multiplication of [user_id x competition_features] * [competition_features x timestamp]. **We ensured that we are multiplying the matrix with a lag of one month to avoid data leakage**

5. We get a final matrix [user_id * timestamp] with respective user interest values

6. Rank the user interest in descending order across timestamp. The user with the highest interest is rank #1 and so on.

Cross Validation Strategy

We performed GroupKFold on timestamp column (which is unique combination of year and month). Our final submission was based on the average of 5 folds prediction. Our CV score improvement was consistent with the LB score. At regular intervals, we verified our score on a holdout dataset of last 3 months of training data.

Model Training

1. Model Used: LightGBM
2. Metric: auc_mu
3. We spend some time tuning the hyper parameters

Few things that didn't work

1. We tried different models like XGboost and Catboost but didn't see a lot of improvement in the final scores. We tried blending different type of boosting algorithms but didn't see much of improvement
2. User_ID as a feature
3. Count based features of various categorical features
4. Using only latest one year of data for training

********** END **********