**About:**

Recommendation systems are not only useful for better source of driving traffic to retail or online businesses. It is also useful to reach the individual and give them a customized experience irrespective of their change in their interests over time.

Thus, not only does it capture the attention and increases the return on investment, it also increases the customer satisfaction which directly leads to loyal customers who generate profit.

**Problem Statement:**

Analytics Vidhya is a platform which also conducts a lot of online and live hackathons. Thus it becomes a place where a lot of programmers work on various topics and upload their codes.

Since the users are on different levels in their programming skills, the goal of this program is to automatically predict or identify the number of attempts the user is likely to solve a code with reasonable accuracy, given the problems details and the specialisation of the user. The data contains the description about the type of the problem, the difficulty level of the problem and also to identify the user is in beginner or an expert and their field of expertise in ML, dynamic programming or graph algorithms etc.

Doing so, the programming committee will be able to identify where and when the user will get stuck during coding and thus, will be able to the users individually by giving relevant hints or suggestions to the problems they face automatically.

**Approach:**

Any prediction is done by building a model and training it with the training dataset that we have got and then use the trained model to predict our test dataset.

Since we have three different files in our training dataset, we initially merge all the features into one final train dataset based on the common feature among the tables. This is as similar as using the primary key or a secondary key in a SQL database. I have repeated the same for out test data as well, since we are in need of the features like problem type and user details to predict the range of their attempts to solve a particular algorithm.

Now that the data is ready, I have checked for null values, duplicate values and the datatype of all the features in the data. I have chosen different methods to fill the null values of the features such as finding the mode or highest number of occurrence value.

This filling of null values depends on the feature as well, if we are filling for a feature like age, using median is a good method as well. I have also used a library function called the Iterative Imputer on certain features of the data, which automatically fills the null value based on basic strategies or by creating iterative rounds and predicting the missing value by trying to fit the known and unknown value on a regression line.

Now, the data has a feature called tags which talks about the expertise and specialization of each of the user. These tags for each user could either be just one domain or a series of domains. Thus, I have done one hot encoding and created dummy features for all of those tags in both the train and the test dataset.

Now, the data has some categorical features with the datatype as object. Machine understand only numbers and thus it becomes necessary for us to convert any object datatype into an int or float based on the requirements. Having said that, I have processed the data and done label encoding for those categorical features, thereby making it into integers.

**Model used - XG Boosting :**

I have used XG Boosting technique for this problem. Artificial neural networks tend to outperform other algorithms when the data is unstructured such as an image or a text. On the other hand, for a medium sized structured data, decision tree based algorithms work much better for good results.

XG boosting is one of the techniques under gradient boosting. Boosting technique is generally a sequential classifier, which constructs models after model. And in each of these models, it tries to use the previously misclassified data to construct a new model. Doing so, it keeps altering the evaluation criteria from the feedback of the previous model. Thus boosting techniques generally provides more accuracy.

XG boost applies a level wise growth, horizontally. That is it expands the tree level wise, it is efficient, flexible and portable. This extreme gradient boosting yields superior results using just less computation resources in the shortest amount of time as well.

XGboost does the following to increase the accuracy of the model, which is why I have chosen it to construct a recommendation engine and predict the attempts range of any programmer to solve a problem.

1. Parallelized tree building
2. Tree pruning using depth first approach
3. Cache awareness and out of core computing
4. Regularization in order to avoid overfitting of the model
5. Efficient handling of the missing data and
6. In built cross validation technique

Now that the model is constructed, I have fitted it to my train data and tried to predict the target for our test data using the same trained model. The evaluation metric I have used here is F1 score between the actual target value and the predicted value, with the average parameter set to ‘weighted’.

**Leader Board – public leaderboard**

****

**Leader Board – private leaderboard**

****