ML Hackathon Approach

The problem statement consisted of building a classifier, capable of predicting the target variable which indicated if the user will buy the product in next 3 months.

The initial approach was to build a Logistic regression model. Firstly, the train data was imported, and initial inspection was done. It was clearly evident that two features had a huge percentage of null values. The 'products_purchased' had more than 50% null values and hence was dropped, since imputing more than 50% of the values in a feature might result in incorrect information. And the 'signup_date' had 38% null values and hence was imputed using the mode. The date columns were converted into datetime datatype in order to extract features. Year, month, week, day and day of week were extracted from the 'created_at' feature.

Now, the data had to be prepared to be fed to the model. The date columns were converted into int datatype. The data was split into X and y, and the model was built. Since there were a lot of features, Recursive feature elimination was used to eliminate statistically insignificant features. Features were then removed manually as well since they had high p- value or high VIF. The final model obtained was not able to predict well. And the results were quite poor. The reason for this might be due do underfitting problem in logistic regression. The model showed signs of high bias and low variance.

Next approach was to use Decision Trees since it is known to work better with missing values and also can be used if the data is not linearly separable. From the EDA that was done, it was evident that the distribution of the features is mostly skewed, hence Decision trees work better with skewed data. The missing values for 'products_purchased' and 'signup_date' was imputed using mode. The model provided very good precision, recall and F1 score for the train data but was unable to produce the same for the test data. The model was very sensitive to (small) changes in the training data. This is because Decision trees are prone to overfitting and hence our model had low bias and high variance.

Finally Random Forest technique was implemented to solve all the above problems of underfitting and overfitting. Random forest unlike Decision trees is not prone to overfitting since its result is based on average or majority ranking and hence there is no problem of overfitting. The random forest chooses features randomly during the training process. Therefore, it does not depend highly on any specific set of features. This randomized feature selection makes random forest much more accurate than a decision tree.

The same data that was prepared for decision trees was used for Random Forest as well. The optimal hyperparameters were first chosen using grid search. Later trial and error were also used to obtain the best hyperparameters for this particular dataset. The trick was to find the right balance between the number of trees and the tree depth of the Random Forest to get the best possible F1 score.

Random Forest is our final model for which the highest F1 score of 0.76 is obtained.