Approaches Tried

In our first approach, we used a mapping function that set the numerical columns to 1.0 by default and added a new category for NAN values . After that we performed the **OneHot Encoding** which increased the column size of the data and then we got the accuracy[1] of **69.88** with **GradientBoostingClassifier.**
Then we tried the **FrequencyEncoding** for the categorical data which reduced the number of columns and also give it as numbers which could be compared within the column values. Tried taking the average of every numeric column for its fill **NAN** which gave it an accuracy of **70.388**.

The below are the accuracy we got by running Gradient Boosting Classifier

- **70.331** when max depth 10 .
- **70.274** anon_var_1 standardize like we standardize months_of_activity and customer score
- **69.847** max-depth 7, learning-rate 0.2, n_estimator 200
- **70.207** random state 12, subsample 0.75.
- **70.473** is final accuracy we got with alot of tuned parameters.
- **For the preprocessing steps :: Final submission**
  Finally, we found out the best parameter for XGBoost which gave us around **69.923** on public Leader board.
  After that, we ran Gradient boosting Classifier with fine tuned parameter which gave us **70.473** accuracy (For the above preprocessing steps)

Pre-Processing for the top submission

- In Preprocessing, we first removed the column "**sex**" from the train and the test set
- After that, we removed the "**id**" and the "**pricing_category**" column too
- Wrote a function called "**FillNANinNumericalColumns**" which takes as input the data frame and returns dataframe with the dataframe after.
  - For numerical column(**customer_score , months_of_activity**), we have a function called getUniformData which returns the random numbers in the NAN space which does not affect the mean and variance of that column. For the column **anon_var_1** we replace the NAN values with
  - For categorical column(**taxi_type, customer_score_confidence, drop_location_type**) we have a method called **FrequencyEncodingCategoricalColumn** which took the dataframe as input and then converted to the frequency values by taking the count of label(A or B etc) and dividing it by the length of the dataframe.
- We got the random numbers inserted for the training set (in case of the **customer_score, months_of_activity**) which we inserted in the NAN slots of the column in the test set.
- We even stored the mean of the **anon_var_1** column and then we used that value to insert in the test set
- We used the **GradientBoostingClassifier** and changed the parameters to obtain the result.

[1] All Accuracy are in percentage

Model Tried

These models were used with the previous submission approaches

- Adaboost **69.10**
- Gradient Boosting **69.88**
- XG Boost **70.06**
- KNN(k=3) **54.55**
- Voting Classifier(3 Classifier- Adaboost, Gradient Boosting, XG Boost) **70.0** With this we created an ensemble classifier but didn't we didn't figure out how to use that.
- Neural Network (4 Hidden Layer[40,30,20,10] hidden Unit per layer) **69.5**

Above all accuracy, we found on **Cross-validation with 80/20 data**.