**Customer Life Time Value Prediction**

**Problem Statement**

VahanBima is one of the leading insurance companies in India. It provides motor vehicle insurances at best prices with 24/7 claim settlement.  It offers different types of policies for both personal and commercial vehicles. It has established its brand across different regions in India.   
Around 90% of the businesses today use personalized services. The company wants to launch different personalized experience programs for customers of VahanBima. The personalized experience can be dedicated resources for claim settlement, different kinds of services at doorstep, etc. In order to do so, they would like to segment the customers into different tiers based on their customer lifetime value (CLTV).  
In order to do it, they would like to predict the customer lifetime value based on the activity and interaction of the customer with the platform. So, as a part of this challenge, your task at hand is to build a high performance and interpretable machine learning model to predict the CLTV based on the user and policy data.

**Feature Description**

|  |  |
| --- | --- |
| **Variable** | **Description** |
| id | Unique identifier of a customer |
| gender | Gender of the customer |
| area | Area of the customer |
| qualification | Highest qualification of the customer |
| Income | Income earned in a year (in rupees) |
| marital\_status | Marital status of the customer {0: single, 1: married} |
| vintage | No. of years since the first policy data |
| claim\_amount | Total amount claimed by the customer (in rupees) |
| num\_policies | Total no. of policies issued by the customer |
| policy | Active policy of the customer |
| type\_of\_policy | Type of active policy |
| cltv | Customer life time value (Target variable) |

**I have tried working on this problem with different approaches and following are the scores for each of those approaches. I will create a detailed explanation of top approach in later part of this document.**

Approach – 1 (score: -0.206343036133882)

Approach – 2 (score: 0.154707535777844)

Approach – 3 (score: 0.155677519469086)

Approach – 4 (score: 0.155096192531217)

Approach – 5 (score: 0.154984388208151)

Approach – 6 (score: 0.15816985489669)

**Following details are related to the highest scoring submission which was done during**

**approach 6.**

**Exploratory Data Analysis**

1. Finding the basic info about the data using Pandas info method.
2. Finding the number of rows and columns in the data.
3. Finding the names of all the features in the data.
4. Finding the names of numerical and categorical features.
5. Finding the unique values and their count in each of feature of data using Pandas unique and nunique methods.
6. Checking if there is any feature having zero variance. There was not such feature.
7. Checking if there are any missing values in any of the feature of the data. There were no missing values in the data.
8. Visualizing the missing values in the data (if any) with the python library named missingno.
9. Checking the presence of outliers using the boxplot.
10. Plotting the distribution of all the numerical features (here ‘claim\_amount’)
11. Here ‘claim\_amount’ feature had very different distribution than a normal distribution. Hence different transformations such log transformation, square-root transformation, power transformation and box cox transformation were performed on the ‘claim\_amount’ feature. But only log transformation was able to transform the data into the values which were kind of close to normal distribution.
12. Plotting the count plot for all the categorical features in the data.

**Data Preprocessing:**

1. The data was split into two groups i.e., group containing all the independent features (X) and the group which contains a target feature (y).
2. The feature named ‘id’ removed since it was not contributing to the model training.
3. Only the feature named ‘claim\_amount’ had more than 4 unique values. So, it was deduced that only ‘claim\_amount’ column is a numerical feature. All the other features were either purely categorical or a categorical feature with encoded numerical values.
4. The numerical feature named ‘claim\_amount’ had a lot of outliers. So, those outliers were replaced by np.nan (NumPy nan values).
5. The data had no missing values. But still implemented a missing value preprocessing step. This was done because it is not necessary that the test data will also have zero missing values. So we would need mean and standard deviation learned during train data transformation in testing data preprocessing too.
6. After the replacement of outliers in the ‘claim\_amount’ feature, all the missing values were dealt with the mean strategy using scikit-learn SimpleImputer transformer.
7. All the missing values from the categorical features were dealt with using the most\_frequent strategy using scikit-learn SimpleImputer transformer.
8. After all the missing value replacement is completed, ‘claim\_amount’ numerical feature was scaled using the scikit-learn MinMaxScaler transformer. All the values were placed between 0.01 and 1. The reason why 0 is not used instead of 0.01 is that in later stages, having zero might create some issues. One of the issue was related to the logarithmic transformation of ‘claim\_amount’ feature.
9. After scaling of the numerical column, a log transformation was performed on the numerical feature to make the data look a little similar to the normal distribution.
10. After log transformation, the encoding of the categorical features was done.
11. The categorical features namely ‘income’, ‘qualification’, ‘type\_of\_policy’ had the values which were showing some kind of order. So, ordinal encoding was performed on these features using scikit-learn OrdinalEncoder transformer.
12. All the other categorical features were not showing any kind of order. So, on these features, an one-hot encoding was performed using the scikit-learn OneHotEncoder transformer.
13. After all this preprocessing an importance of every feature of the data was found out for intuition purpose using random forest regressor estimator.
    1. Most contributing features were ‘claim\_amount’ (49.03%), ‘num\_policies’ (13.88%), ‘vintage’ (12.97%).
    2. Least contributing features were ‘area’ (0.79%), ‘policy’ (0.98%) and ‘marital\_status’ (1.72%)
14. After finding the feature importance, the data was split into 4 different groups namely X\_train, X\_val, y\_train, y\_val.

**Model Description for approach 6**

1. After preprocessing of the data, a cross-validation was performed on the data using many machine learning algorithms. The names of those algorithms were
   1. SVR
   2. KNeighborsRegressor
   3. DecisionTreeRegressor
   4. RandomForestRegressor
   5. XGBRegressor
   6. AdaBoostRegressor
   7. GradientBoostingRegressor
   8. LGBMRegressor
   9. CatBoostRegressor
   10. Voting regressor with combination of LGBMRegressor, CatBoostRegressor, RandomForestRegressor
2. Out of all the machine learning algorithms, following four gave the highest r2 score.
   1. XGBRegressor
   2. GradientBoostingRegressor
   3. LGBMRegressor
   4. CatBoostRegressor
3. Next, the coarse hyper-parameter tuning was performed on these high scoring algorithm models using scikit-learn RandomizedSearchCV.
4. After the coarse tuning, the parameters were adjusted near the ones found out, and then finer tuning was performed the models using scikit-learn GridSearchCV.
5. After both coarse and finer tuning, the discovered parameters were set to the models using scikit-learn set\_params method.
6. Now each of these models were trained on the original preprocessed data and then the models were saved into the python pickle file.
7. Out of above four highest scoring models, XGBRegressor gave the highest r2-score.
8. So, after loading and preprocessing the test data, the predictions were made using the tuned XGBRegressor model.
9. Then the predictions were saved as a csv file and then exported.