JOBATHON JANUARY 2023 Approach Document

Problem Statement:

We have to predict Customer Life Time Value of customers of VahanBima company. The target variable is a continuous numerical variable. So the task is **‘Regression’.**

Approach:

Data Loading and Inspection:

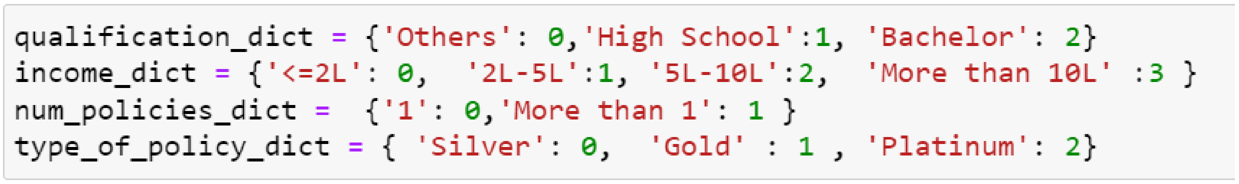
The train dataset has 89392 records and the test dataset has 59595 records. There are no missing values in any column in both the dataset and 26 duplicate rows in training if the ‘id’ column is dropped.

Features:

- **Id** is just noise that has no impact on cltv **(target variable)**
- **Categorical columns**: gender, area, qualification, income, marital_status, num_policies, policy, type_of_policy
- **Numerical columns**: vintage, claim_amount, cltv

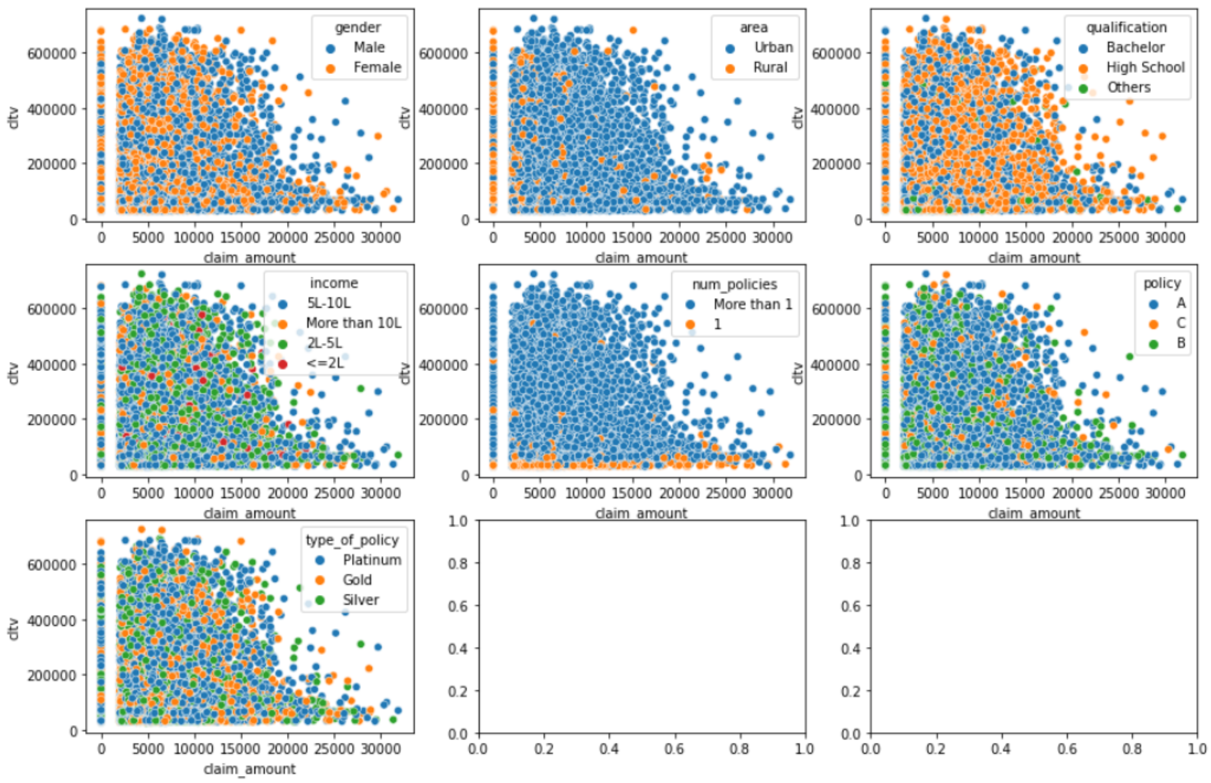
Ordinal Encoding:

The following columns are ordinal categorical columns and I did this ordinal encoding.



```
qualification_dict = {'Others': 0,'High School':1, 'Bachelor': 2}
income_dict = {'<=2L': 0,  '2L-5L':1, '5L-10L':2,  'More than 10L' :3 }
num_policies_dict =  {'1': 0,'More than 1': 1 }
type_of_policy_dict = { 'Silver': 0,  'Gold' : 1 , 'Platinum': 2}
```

Bivariate Analysis

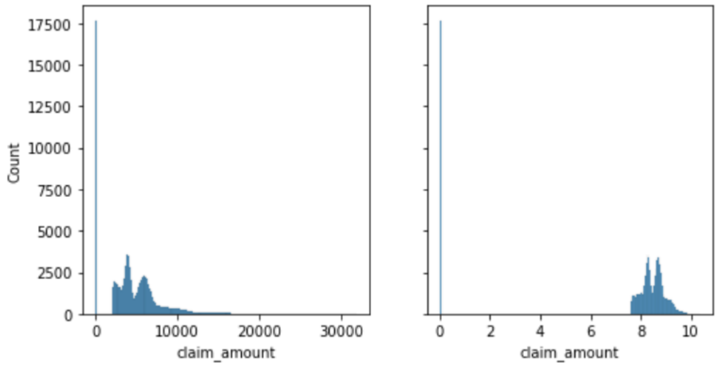


One Hot Encoding:

Then I one-hot encoded the remaining nominal categorical variables and trained the models.

But doing the EDA on the dataset (using **sweetviz** and **seaborn**) I observed the different ordinal categories of the particular columns are not showing any effect on cltv except in **'area'** and **'num_policies'** features. But they got only 2 categories in each of them. So whether I one-hot or label-encode they will be no difference. I tried various approaches like one-hot encoding some variables and ordinal encoding other variables , one-hot encoding entire dataset with **pd.get_dummies().** But I observed there is no considerable difference in the performance of the models.

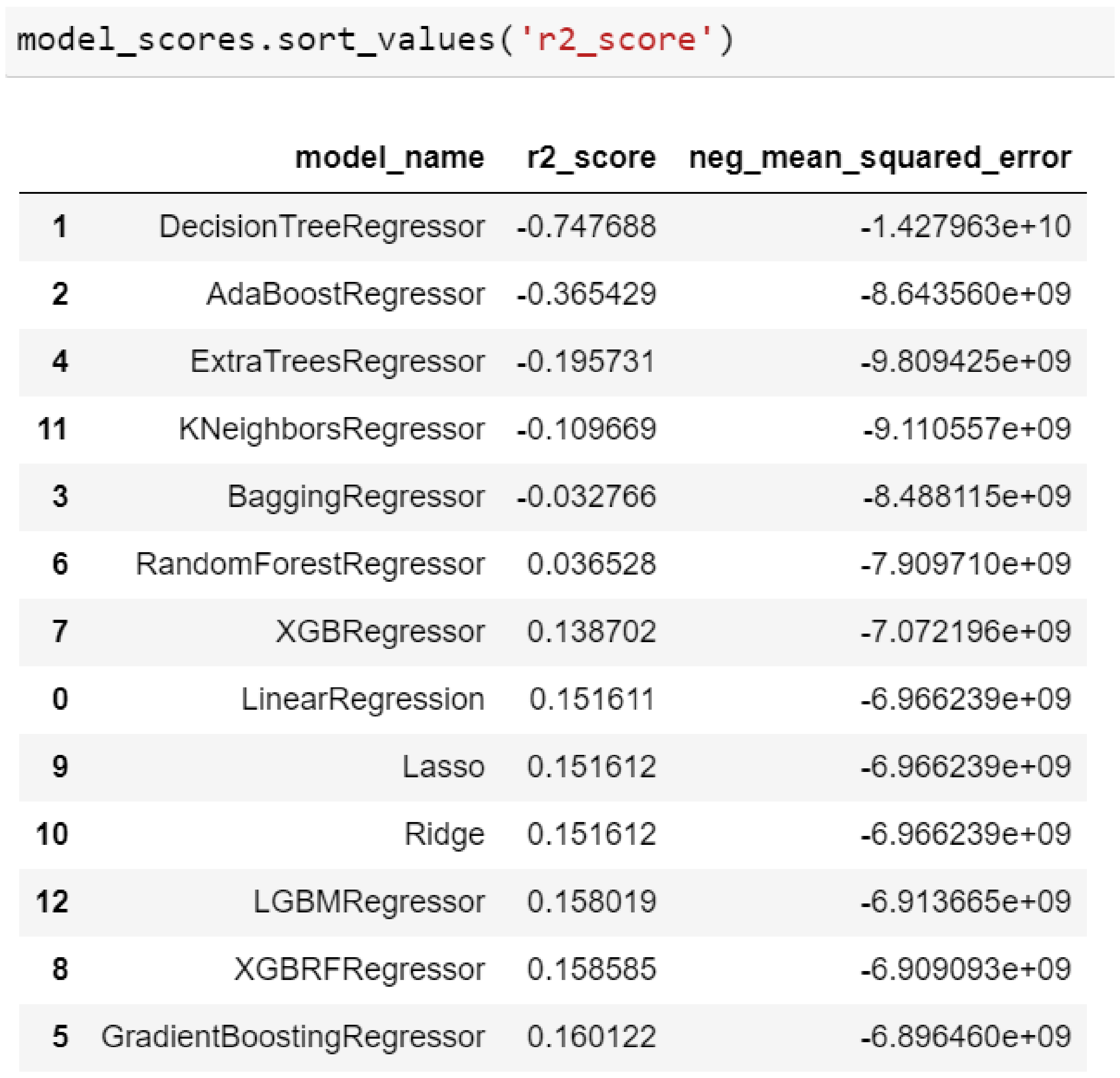
Feature Scaling:



Almost 20 percent of the **‘claim_amount’** column has zero values. Also ‘claim_amount’ and ‘cltv’ column are heavily right skewed. So I applied the **log transformation** on these columns and trained the models. But there was no considerable improvement.

Model Selection:

I selected the below models and trained them on the training dataset and assessed their performance using **cross validation** with 5 folds.



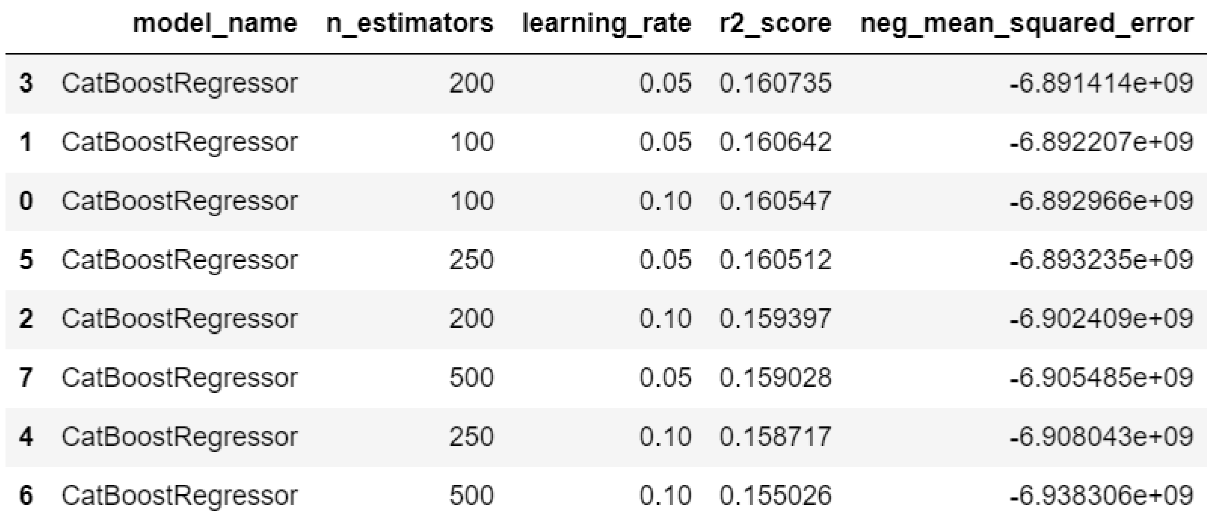
```
model_scores.sort_values('r2_score')
```

	model_name	r2_score	neg_mean_squared_error
1	DecisionTreeRegressor	-0.747688	-1.427963e+10
2	AdaBoostRegressor	-0.365429	-8.643560e+09
4	ExtraTreesRegressor	-0.195731	-9.809425e+09
11	KNeighborsRegressor	-0.109669	-9.110557e+09
3	BaggingRegressor	-0.032766	-8.488115e+09
6	RandomForestRegressor	0.036528	-7.909710e+09
7	XGBRegressor	0.138702	-7.072196e+09
0	LinearRegression	0.151611	-6.966239e+09
9	Lasso	0.151612	-6.966239e+09
10	Ridge	0.151612	-6.966239e+09
12	LGBMRegressor	0.158019	-6.913665e+09
8	XGBRFRegressor	0.158585	-6.909093e+09
5	GradientBoostingRegressor	0.160122	-6.896460e+09

I selected the LGBMRegressor, XGBRFRegressor, GradientBoosingRegressor based on this test and trained those models.

Hyperparameter Tuning:

Then I tried various hyperparameter values for these models. I even tried dividing the trainin dataset into two parts with one part having claim_amount has zero (20% of the dataset) and other part where claim_amount >0. This approach has provided the meagre improvement. Then I tried the CatBoostRegressor and it have the highest r2_score.



	model_name	n_estimators	learning_rate	r2_score	neg_mean_squared_error
3	CatBoostRegressor	200	0.05	0.160735	-6.891414e+09
1	CatBoostRegressor	100	0.05	0.160642	-6.892207e+09
0	CatBoostRegressor	100	0.10	0.160547	-6.892966e+09
5	CatBoostRegressor	250	0.05	0.160512	-6.893235e+09
2	CatBoostRegressor	200	0.10	0.159397	-6.902409e+09
7	CatBoostRegressor	500	0.05	0.159028	-6.905485e+09
4	CatBoostRegressor	250	0.10	0.158717	-6.908043e+09
6	CatBoostRegressor	500	0.10	0.155026	-6.938306e+09

Finally the submission with the **average of the top 3 models of CatBoostRegressor** has given the best score.