|  |  |
| --- | --- |
|  | Categorical Feature Encoding Challenge II |
|  |  |
|  | Pamu, Asish  4/30/20 |

Contents

[Project Statement 1](#_Toc39166627)

[Data 1](#_Toc39166628)

[Exploratory Data Analysis 1](#_Toc39166629)

[Target Variable 1](#_Toc39166630)

[Binary features distribution 2](#_Toc39166631)

[Nominal Feature Distribution 3](#_Toc39166632)

[Ordinal feature distribution 5](#_Toc39166633)

[Cyclic features 6](#_Toc39166634)

[Data Preprocessing 6](#_Toc39166635)

[Data Cleaning 6](#_Toc39166636)

[Splitting Data 8](#_Toc39166637)

[Data Imbalance 8](#_Toc39166638)

[Modeling 8](#_Toc39166639)

[Limitations 9](#_Toc39166640)

[Conclusions 9](#_Toc39166641)

[References 9](#_Toc39166642)

[Appendix 10](#_Toc39166643)

# Table of Figures

[Figure 1: Distribution of Target variable 4](#_Toc39166983)

[Figure 2: Correlation of numerical variables 5](#_Toc39166984)

[Figure 3: Binary feature distribution on train data 5](#_Toc39166985)

[Figure 4: Binary feature distribution on test data 6](#_Toc39166986)

[Figure 5: Binary feature distribution of train data along with target variable 6](#_Toc39166987)

[Figure 6: Nominal feature distributions on train data 7](#_Toc39166988)

[Figure 7: Nominal feature distribution on test data 7](#_Toc39166989)

[Figure 8: Nominal feature distribution on train data along with target level 8](#_Toc39166990)

[Figure 9: Ordinal distribution on train data along with target variable 8](#_Toc39166991)

[Figure 10: Distribution of cyclic feature – Day 9](#_Toc39166992)

[Figure 11: Distribution of cyclic feature – Month 9](#_Toc39166993)

[Figure 12: Number of unique values for features in dataset 10](#_Toc39166994)

[Figure 13: Variable Data Type and Non-null values in Dataset 10](#_Toc39166995)

[Figure 14: Confusion matrix 12](#_Toc39166996)

# Execute Summary

The project objective is to improve the classification accuracy in predicting target variable which is binary variable i.e., two levels 0’s and 1’s where 0’s and 1’s are in the ratio of 81% and 19% respectfully. which clearly states there is imbalance in target variable levels. All the input variables to predict the target variable are categorical variables. So, here is another challenge, which is addressed by performing various methods of label encoding of these categorical variables and using the encoded values to predict the target variable. To fine hyperparameters the dataset is split into train and validation. Models are trained on training dataset and based on the performance of the model on validation dataset hyper parameters are tuned to improve accuracy. Few classification models were developed to classify our target variable. Extreme Gradient Boosting is the champion model with an accuracy of 57% and area under receiver operating characteristic curve is 78% on validation dataset. This model is used to predict the values for the test data set given and achieved a pubic score of 76.29%.

# Project Statement

Encoding the categorical data (binary, nominal, ordinal) for binary classification for a synthetically created dataset.

# Data

We are provided with 3 datasets, i.e., training dataset, test dataset and sample submission dataset

Training dataset consist of 600,000 records and test datasets consist of 400,000 records.

Both training and test dataset contains 25 features that include

* 1 id variable
* 5 binary features
* 5 low- and 5 high-cardinality nominal features
* 3 low- and 3 high-cardinality ordinal features
* 2 cyclical features namely day, month
* 1 target variable

# Exploratory Data Analysis

## Target Variable

Distribution of target variable clearly shows that target ratio is unbalanced. 81.28% belongs to class 0 and 18.72% belongs to class 1(Figure 1).

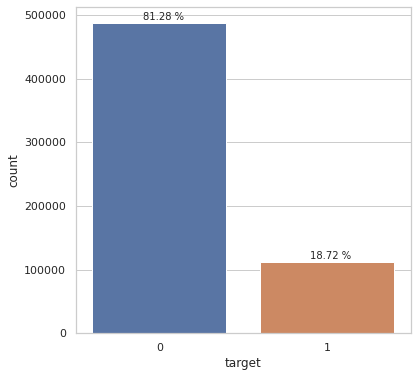


Figure 1: Distribution of Target variable

Next, we looked at correlations between numerical variables (figure 2), no special correlation between these features, except from the pairs of ( ord\_0 - target) and ( montdh - target)

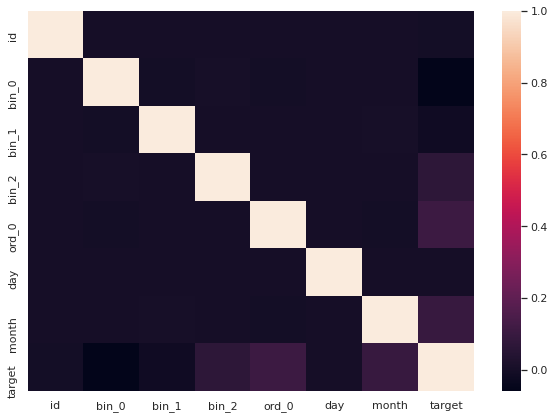


Figure 2: Correlation of numerical variables

## Binary features distribution

From binary features distribution plot on training and test data(Figure 3 & figure 4), we can say that both training and test data has similar patterns. In bin\_0, bin\_1 and bin\_2, level 0 is more frequent than level 1. In variable bin\_3, number of “False” are more than number of “True”. Similarly, in bin\_4, Number of “No” are more than number of ”Yes” but the difference is much less when compared to all other binary features

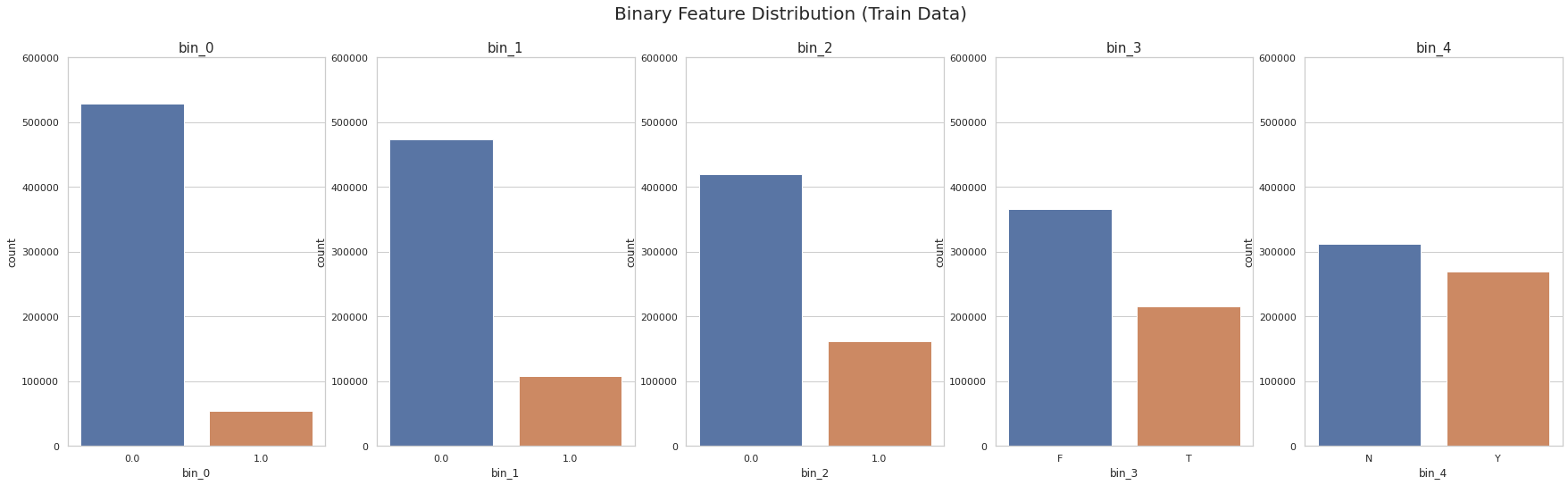


Figure 3: Binary feature distribution on train data

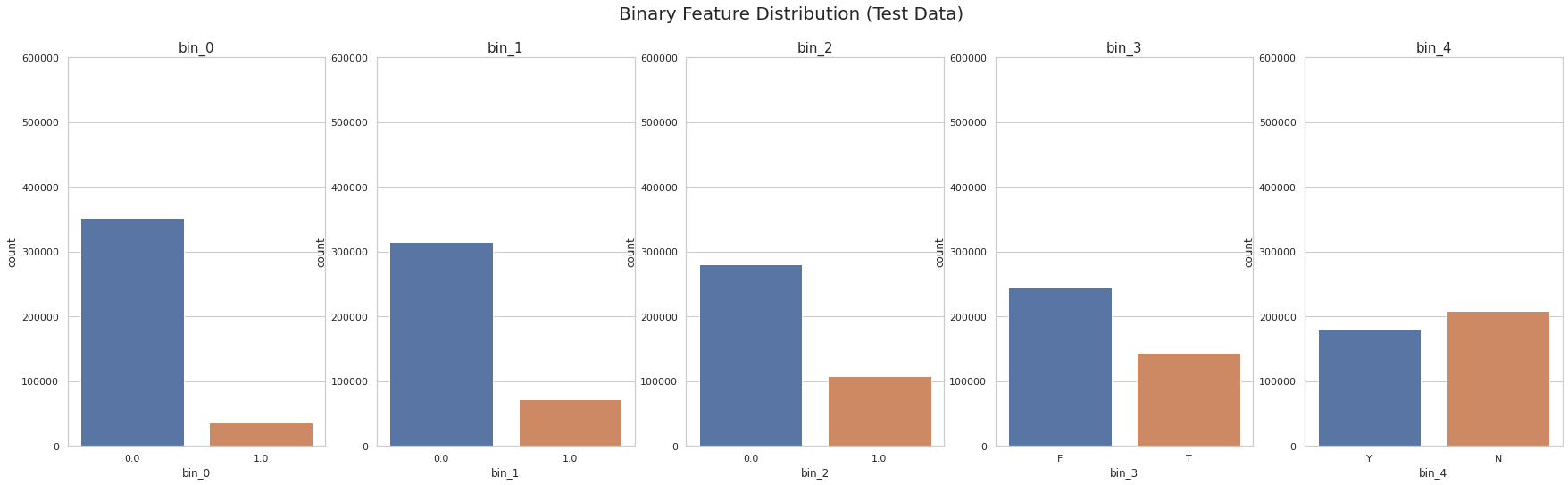


Figure 4: Binary feature distribution on test data

If we look at target class distribution in each of binary feature (figure 5), we can clearly say that target class 0 has high number of records than target class 1 because we have imbalance in the dataset. As we move from bin\_0 to bin\_4 the distribution of target class is approaching 50%.

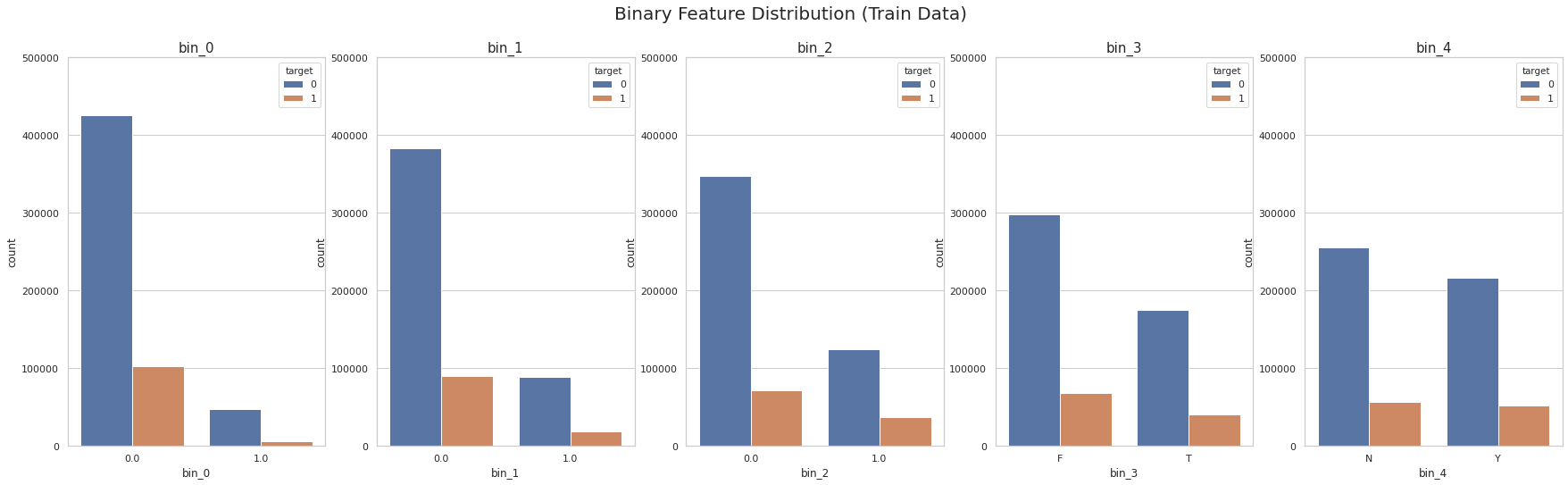


Figure 5: Binary feature distribution of train data along with target variable

## Nominal Feature Distribution

From nominal features distribution plot on training and test data(figure 6 & figure 7), we can say that both training and test data has similar patterns. The plots are only for low cardinality features i.e., features nom\_0 to nom\_5. In all the plots we can see that the distribution is not equal among levels in feature.

* In nom\_0, Red is most frequent level followed by Blue and Green
* In nom\_1,Traingle is the most frequent level followed by Polygon, Trapezoid, Circle, Square and Star
* In nom\_2, Hamster is the most frequent level followed by Axolotl, Lion, Dog, Cat and Snake
* In nom\_3, India is the most frequent level followed by Costa Rica, Russia, Finland, Canada and China
* In nom\_4, Theremin is the most frequent level followed by Bassoon, Oboe and Piano

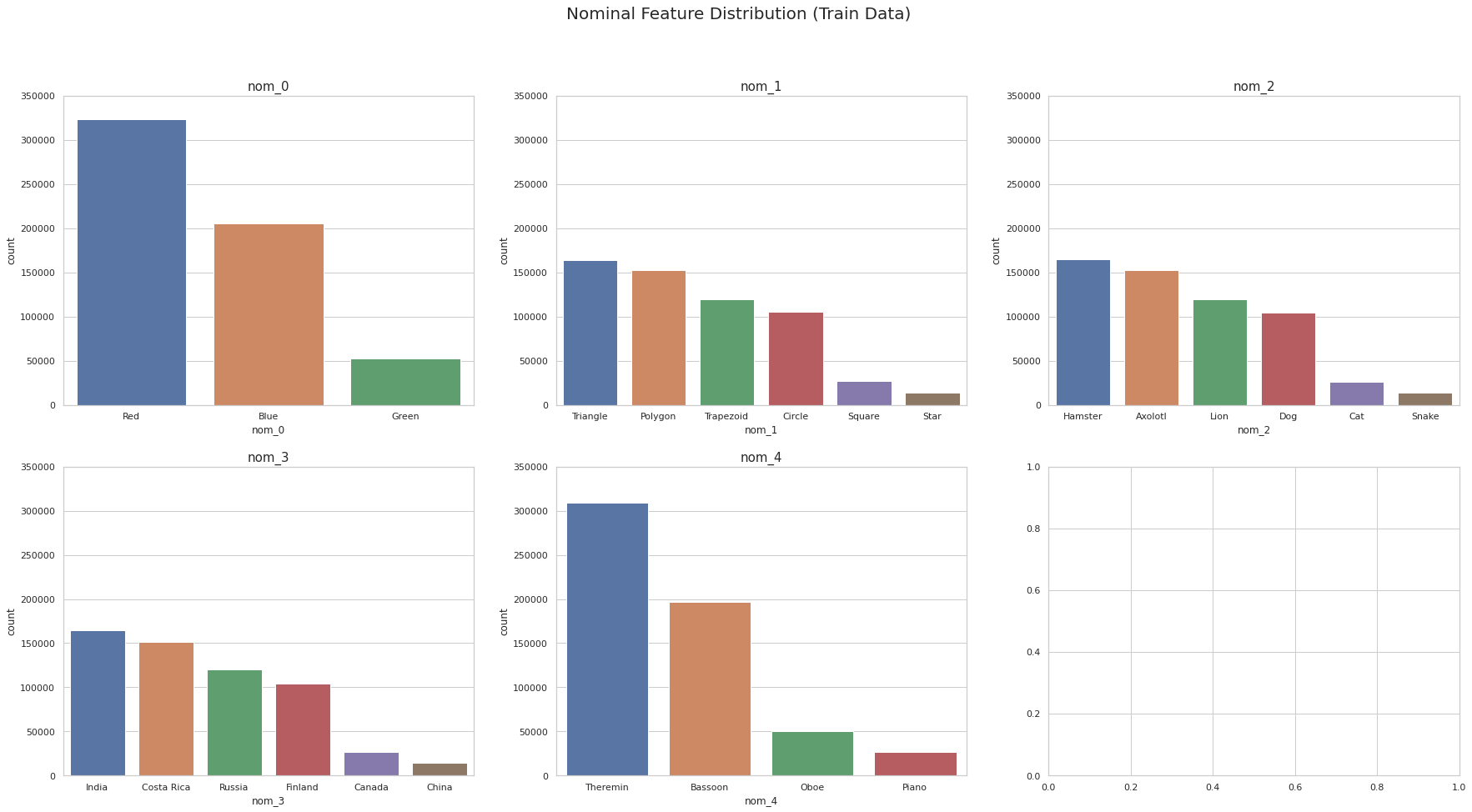
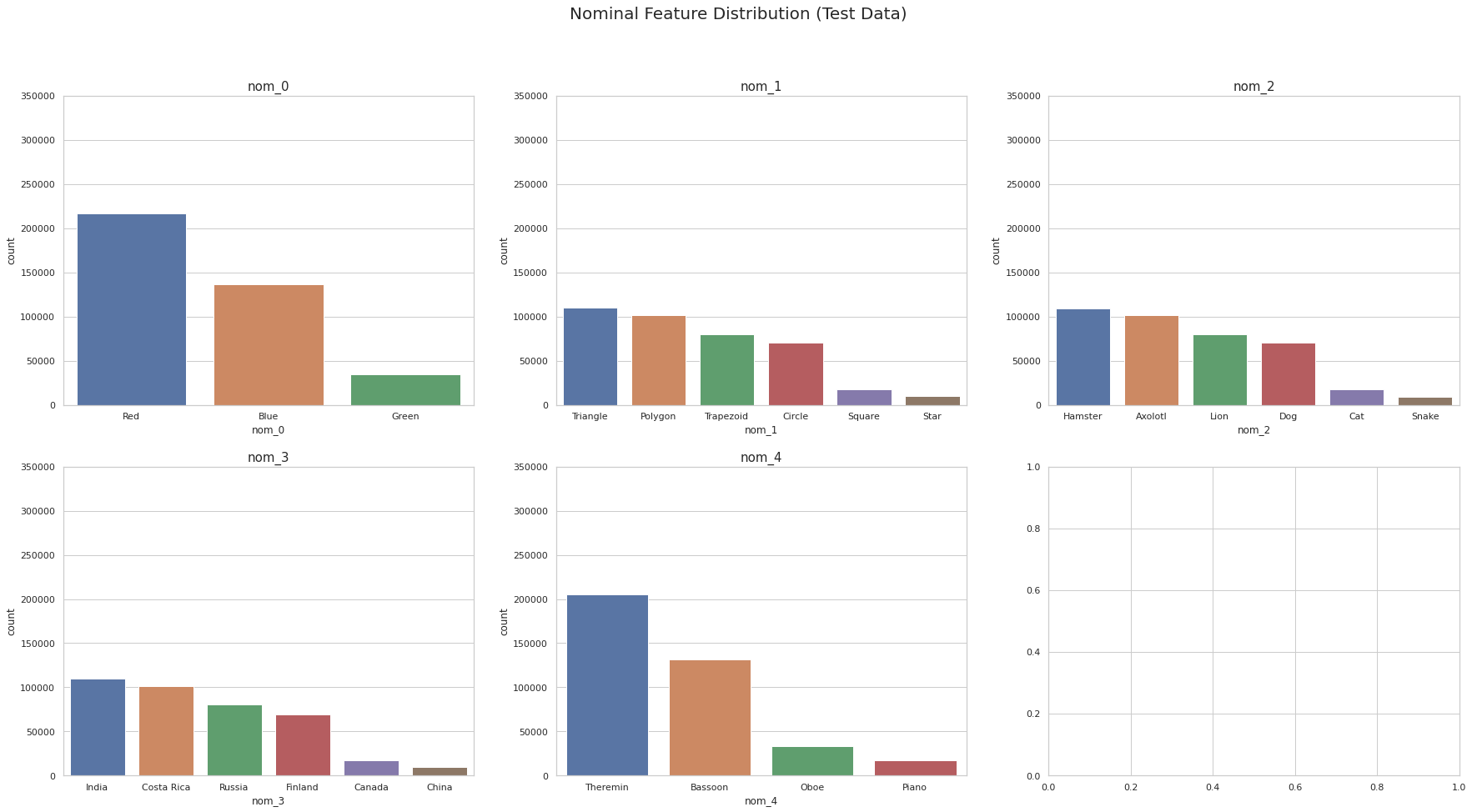


Figure 6: Nominal feature distributions on train data

Figure 7: Nominal feature distribution on test data

If we look at target class distribution in each of binary feature (figure 8), we can clearly say that target class 0 has high number of records than target class 1 because we have imbalance in the dataset.

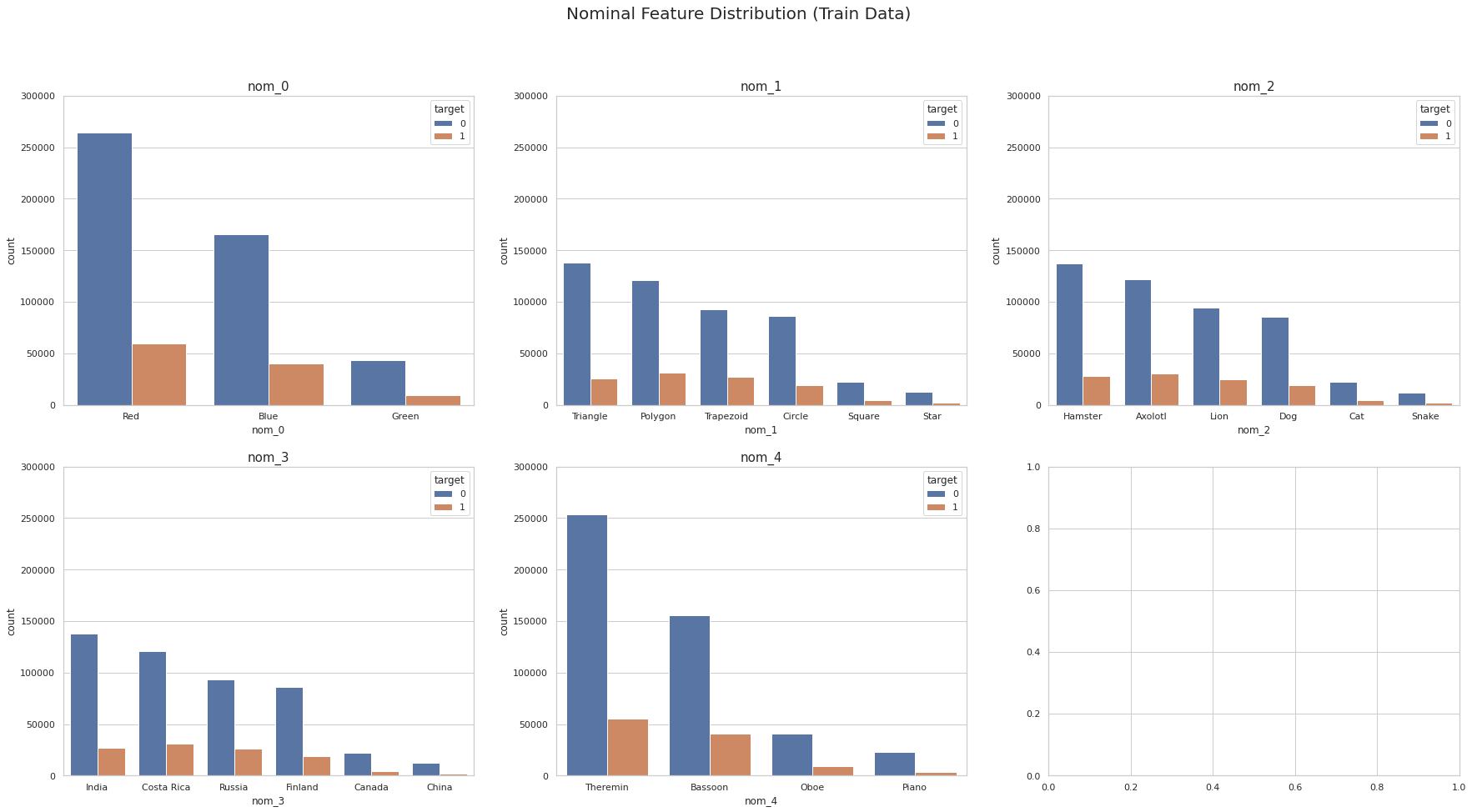


Figure 8: Nominal feature distribution on train data along with target level

## Ordinal feature distribution

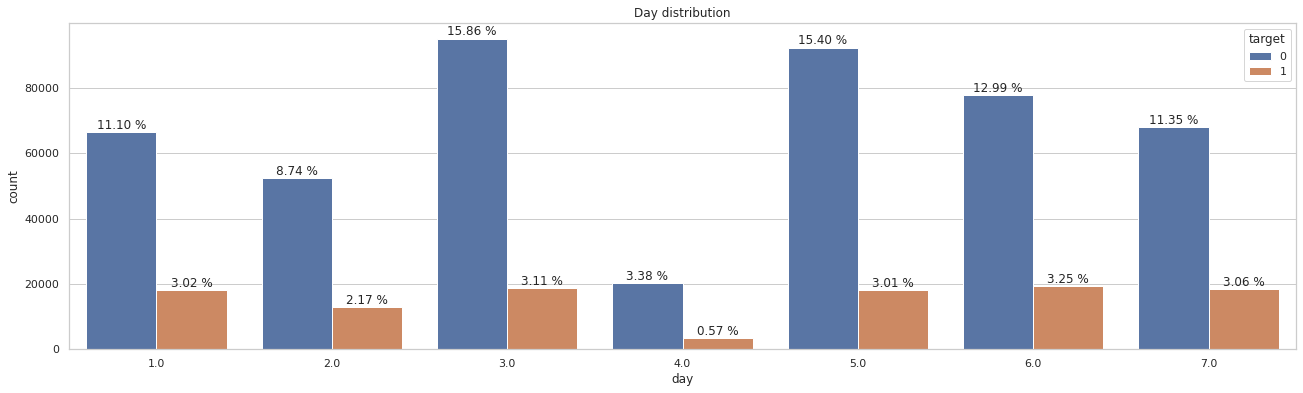


Figure 9: Ordinal distribution on train data along with target variable

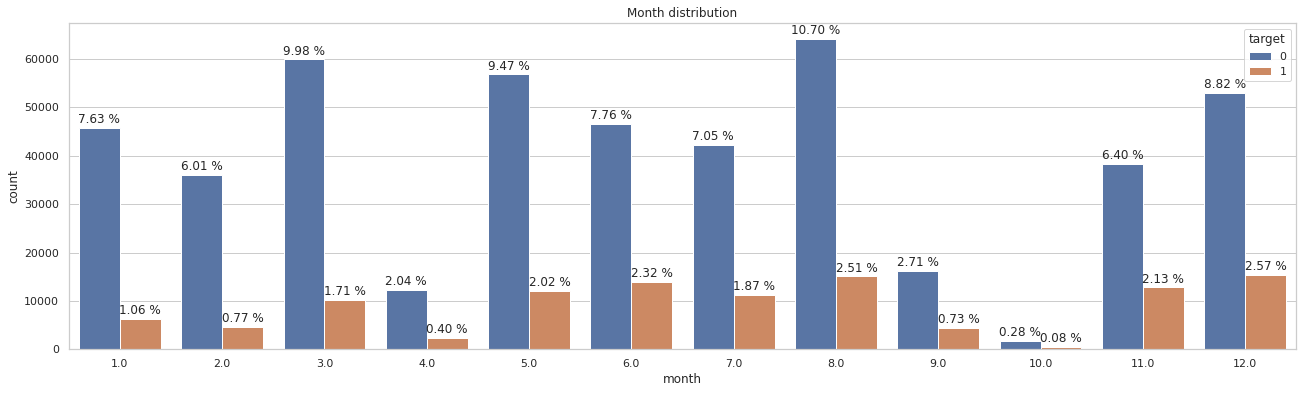
From figure 9, number of levels in ordinal levels is increasing as we move form ord\_0 to ord\_5. We can clearly say that target class 0 has high number of records than target class 1 because we have imbalance in the dataset.

## Cyclic features

If we look at distribution of “Day” variable we cannot find a fixed pattern which suggest number of records are high on that day(Figure 10). This may be because it is time series variable, needs to be visualized with line chart instead of histogram.

Figure 10: Distribution of cyclic feature – Day

From figure 11, It is more likely a sample will has the True target if it happens in August (16.54 %), March (15.43 %) and get the False target in December (3.98 %).But we only see the affections of time series data when it is plotted via line charts

Figure 11: Distribution of cyclic feature – Month

# Data Preprocessing

## Data Cleaning

As there are many categorical variables in the dataset i.e. about 23 variables, different approaches have been taken to handle the variable like encoding them into numerical values. These approaches taken to handle them, were detailed further in this report. We used Label encoder, hand-on encoding and category encoder package for encoding nominal features, binary features and ordinal variables. Null values are masked before using label encoder, this helps in while imputing for missing values

* Binary: The two binary variables were encoded using the label encoder function from scikit learn package. No alternative necessary
* Nominal: The nominal variables were encoded using the label encoder function as well. Because there were no orders and no observable connections between the variable values.
* Ordinal: The first ordinal variable in the dataset is distinct integer type, it is left manipulated. Let us consider the variable with 190 levels, each level is assigned with numeric values in the increasing order from 1,2,3 … to 190. The variable Ord\_5 has 183 levels, “category\_encoder” package of Python was used to encode this variable.



Figure 12: Number of unique values for features in dataset



Figure 13: Variable Data Type and Non-null values in Dataset

From figure 13, The dataset has different data types. Data types of all the variables need to be changed to numeric i.e., float64 or int64. The drawback for encoding the variables from string to numerical was to preserve the missing values as nominal encoders available picked up the missing values as a separate category. The dataset has missing values. There are several methods for handling missing values.

* Drop missing values
* Fill missing values with test statistic
* Predict missing value with a machine learning algorithm

Instead of simply dropping the records with missing values and losing significant amount of data, we imputed missing values with advanced methods. Since the values are categorical in nature, one possibility is using the most occurring value for imputation. But this creates bias in the data that could skew the results.

Unfortunately, using advanced methods such as using KNN or other predictive models to impute the missing values are met with memory error due to the large size of the dataset. We even tried splitting 600,000 rows into ten 60,000 row datasets, even that did not work. Finally, we used the Simple Imputer function from the Scikit Learn package to implement imputation with most frequent value(mode).

## Splitting Data

The dataset is split into training and validation sets using sklearn “train\_test\_split” function. The model is trained on training dataset while the validation dataset is used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. We used 75 -25 split for training and validation datasets respectfully.

Training dataset consist of 450,000 records and validation dataset consists of 150,000 records with 24 features in both datasets. We removed the “id” variable in both training, validation and test datasets. All the variables in the training and validation datasets are standardized using “standard scaler” function in sklearn package.

## Data Imbalance

Imbalance in dataset can be handled in several ways like using adjusting prior probabilities, unequal case weights, sampling methods. We used unequal case weights to deal with data imbalance. In unequal case weights, rebalancing the training set would be to increase the weights for the samples in the minority classes. So, the predictive models for classification use case weights where each individual data point can be given more emphasis in the model training phase. For many models, this can be interpreted as having identical duplicate data points with the exact same predictor values

# Modeling

We used extreme gradient boosting model to classify the target variable because which drives fast learning through parallel and distributed computing and offers efficient memory usage. XGBoost is an ensemble learning method, Bagging and boosting are two widely used ensemble learners. Though these two techniques can be used with several statistical models, the most predominant usage has been with decision trees. In contrast to bagging techniques like Random Forest, in which trees are grown to their maximum extent, boosting makes use of trees with fewer splits. Having many trees might lead to overfitting. So, it is necessary to carefully choose the stopping criteria for boosting.

We used binary logistic as objective, learning rate of 0.2, max depth of 15, number of estimators = 400 and subsample = 0.8. we used unequal case weights to adjust imbalance in target class so that model will not give more weight to majority level in target variable. The Booster parameters (tree depth, minimum weight for branching) and the regularization parameter (gamma) was selected using Grid Search with 5-fold (Scikit Learn function). A set of values was selected as candidates and the best combination from those values were used in tuning. Other parameters were tuned with trial and error basis. The reason for the two-pronged step is that 3 parameters with 3 possible candidate values in 5-fold already had 135 possible steps. Adding more parameters could crash the system for want of memory. Area under ROC (Receiver operating curve) is 78%.

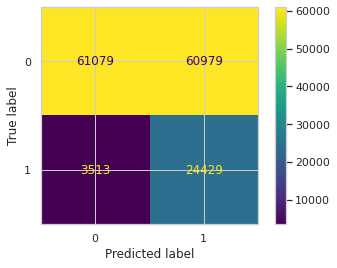


Figure 14: Confusion matrix

We predicted the class for test dataset and the results are submitted to Kaggle. We got a public score of 0.76296 and a private score of 0.76460

# Limitations

* Imputation using machine learning algorithm like KNN cannot be done on this dataset because of large size of the dataset
* Categorical variable with many levels were not grouped further into few meaningful levels, as we lack the data description of those levels in the problem statement

# Conclusion

Extreme gradient boosting model is the champion model with accuracy of 57% and area under receiver operating characteristic curve is 78% on validation dataset.

# References

1. <https://numpy.org/doc/1.18/reference/index.html>
2. <https://xgboost.readthedocs.io/en/latest/parameter.html>
3. <https://scikit-learn.org/stable/modules/classes.html>
4. <https://seaborn.pydata.org/>
5. <http://contrib.scikit-learn.org/category_encoders/>

# Appendix

