Home Insurance

CEBD1260 Machine Learning Project

REETA SHARMA

Date: 2020.11.27

Introduction

Home Insurance company does not currently have a dynamic conversion rate model that can give them confidence a quoted price will lead to a purchase. Using an anonymized database of information on customer and sales activity, including property and coverage information,this dataset is challenging to predict which customers will purchase a given quote. Accurately predicting conversion would help Home Insurance company to better understand the impact proposed pricing changes and maintain an ideal portfolio of customer segments . In this project, I used lightGBM machine learning model, after doing data prepossessing. feature engineering, training and validation and building model, we could reach AUC score 0.96.

Main findings

The problem I need to solve is to classify 'Quote_conversionflag'. The goal is to determine which customer will buy insurance quote or not. The target variable: Quote_conversionFlag (0 or 1), the type of learning is supervised learning, it is a classification problem.

I used lightGBM machine learning algorithm. Since my dataset is highly imbalanced and with many missing values(NaN), I spent 80% of time to do the dataset preprocessing and feature engineer. Also my dataset has the categorical values and data is sparse. So I saved my data as sparse matrix for training the model. And 20% of time to select algorithm and tuning the parameters.

Other findings

Most of features are complexed to handle because they are hidden

categorical values making it impossible to determine what exactly they are.

Data Preprocessing

- This part basically we need handle the NAN values in our dataset
- Transfer 'object' values into digital data

Handle NAN Value

- NAN Value has crucial part in data preprocessing. what I did is fill in the float and integer columns with -1, fill in 'unknown' into the 'object' columns.
- I filled missing values in my columns are having float or int values with -1 because my dataset has hidden categorical features, which means column’s values are look like int or float but actually that are categorical.

Transfer Object type or hidden categorical values

- Since computer can only read the digital values that we have to transfer all the 'object' values into the digital values.
- As I know, my data set has hidden categorical features. So I set the threshold = 70 for unique values of columns.
- I used label encoding for convert my object and categorical features to digital type. And then I applied one-hot-encoding to my all label encoded features and save them as sparse matrix because my data is very sparse.

Feature Engineering

- This part I am going to explain the Feature Engineering that I did for my project and the purpose of doing it.
- Basically my dataset has not clearly defined features. So I can’t do much of feature engineering. But in this dataset there is given ‘Original_Quote_Date’ .
- From this date feature, I can create or add year,month weekday and quarter new features to my data frame.

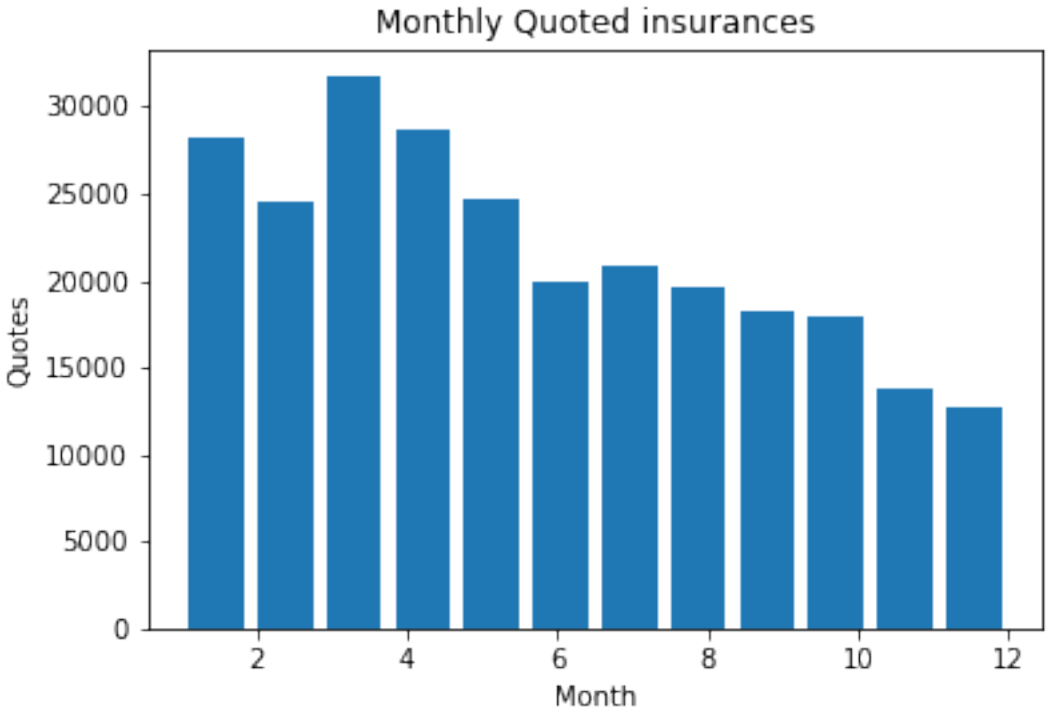
['Original_Quote_Date'] Feature:

- This feature contains all the insurance quoted dates which is an object type in the oringal data frame. Therefore we think we need to transfer string type to the date type in order to let our computer read it and run.
- More specificaly we need to add some columns like: 'df['weekday']', ' df['year']' etc, which are some important features that are gotten from the ['Original_Quote_Date'].

  We formatted and extracted some data from ['Original_Quote_Date'], the date and time column. New features such as the year, month, weekday at which the insurance is quoted occur were obtained and added.

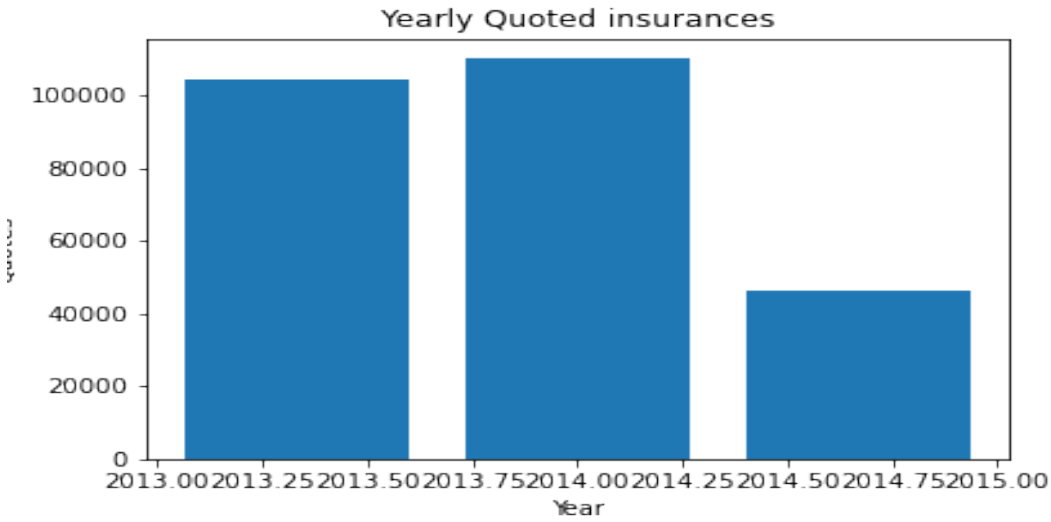
- Here is the Monthly distribution of quoted insurance.

  From the below plot, I can clearly see that March month has maximum quote conversions and December has minimum. Also, overall quote conversion has decending order from January to December.



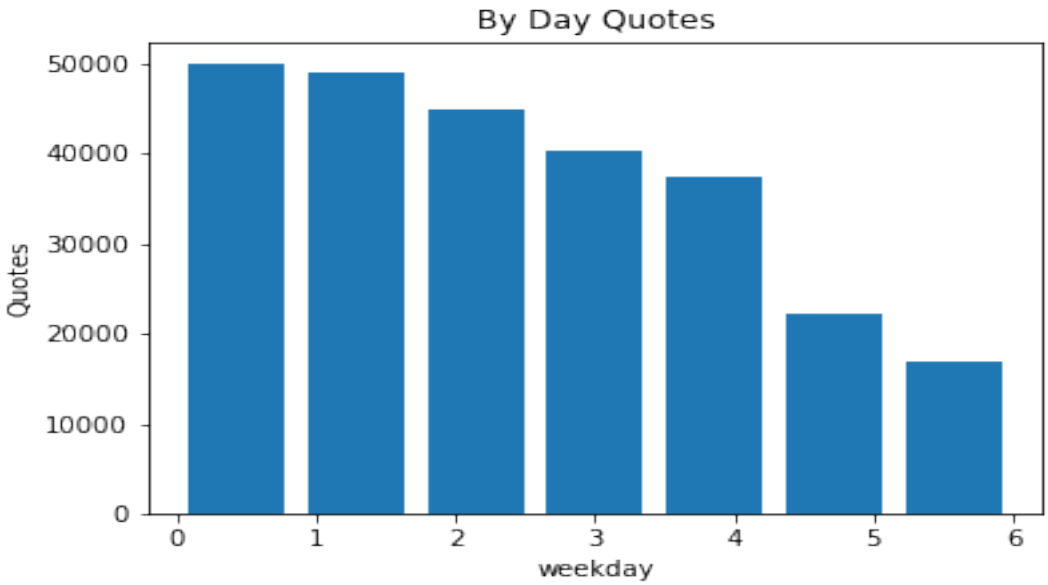
- Here is Yearly distribution of quote conversion.

  From the below plot,I can see that maximum quote conversions in the year 2014.



- Here is weekday distribution that shows that at which day has more quote conversion.

  From the below plot, I can say that on Monday's there are maximum quote conversions. And Weekend has minimum.



Correlation :

I found the correlation between the features and then I dropped out one of the two highly correlated features. Before Correlation, I have 299 columns and after the correlation, I have 180 columns and adding the 4 date columns: [“Month”],[“Year”],[“weekday”] and [“Quarter”]which are generated from ‘Original_Quote_Date’.

Prediction Algorithm

The lightGBM python library was used in order to produce models for the prediction of the attribute Quote_conversionFlag. Following the preprocessing and the feature engineering task, as dataset was fed to this algorithm along with the following parameters, which are explained below:

Since the predicted value is a binary variable, the following estimator was selected (i.e. LGBMClassifier):

model = lgb.LGBMClassifier(**params)

Following the selection of the specific estimator, the following parameters were passed to it:

LGB_default = {'n_jobs':4 , 'n_estimators':10000, 'learning_rate': 0.01, 'boost_from_average':'False', 'num_leaves':64, 'num_threads' :4, 'max_depth': -1, 'tree_learner':'serial' , 'feature_fraction': 0.7, 'bagging_freq': 5, 'bagging_fraction': 0.7, 'min_data_in_leaf': 100, 'silent':-1,'verbose': -1, 'max_bin': 255, ' bagging_seed': 11 }

LGB_PARAM = {'n_jobs':4 , 'n_estimators':10000, 'learning_rate': 0.02, 'boost_from_average':'False', 'num_leaves':64, 'num_threads' :4, 'max_depth': -1, 'tree_learner':'serial' , 'feature_fraction': 0.7, 'bagging_freq': 5, 'bagging_fraction': 0.9, 'min_data_in_leaf': 100, 'silent':-1,'verbose': -1, 'max_bin': 512, ' bagging_seed': 11 }

The boosting type selected was the Gradient Boosting Decision TREE (gbdt), which is also the default value. The objective selected was binary as this is a binary classification application The learning rate selected was 2% (0.02). Default is (0.01) which could lead to overfitting. During my testing of the algorithm to generate the models, I played with the feature_fraction and bagging_fraction parameter. They were quite useful as the could speed up training and deal with overfitting. Verbose was used to better understand the

logic behind the training of the model and identify the exact Epoch were the error was minimized for the evaluation of the test data. max_depth limits the max depth of the tree model. The default is -1.The max_bin parameter which is the max number of bins that feature values are be bucketed, the default value is 255. At 512 the accuracy of the training is reduced but may improve the overall power and avoid over-fitting. The metric used for the performance evaluation was AUC (area under curve). This allows us to generate these metrics in the model so that they could easily be visualized with the built-in plot function provided by the library.

Methodologies:

Train/test/valid split

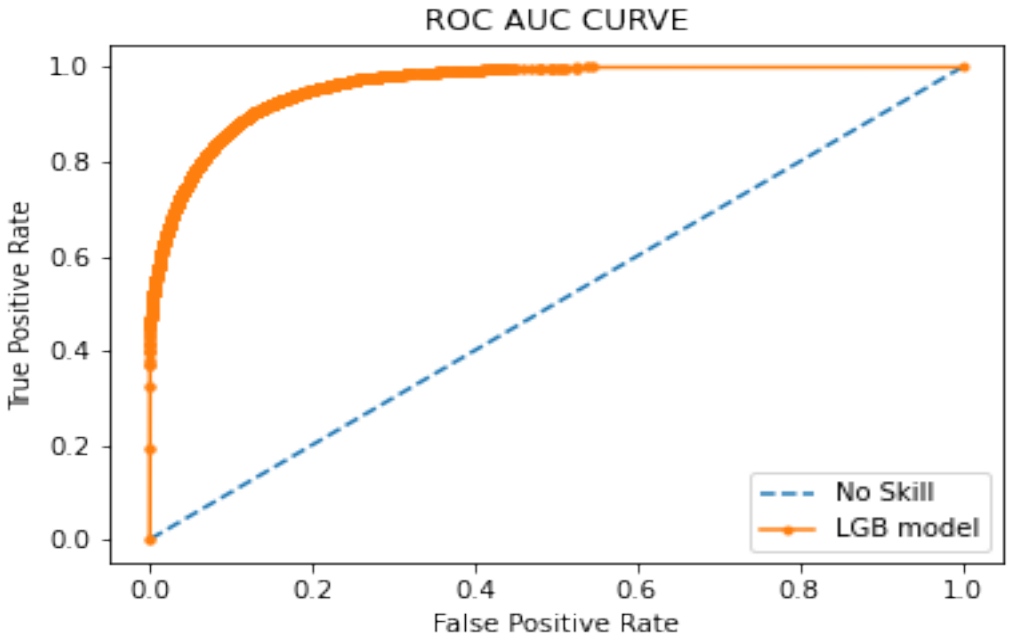
For training data, I have 2532 feature after adding and encoding new features like year,month,weekday,etc.

Use Cross-Validation to allow us to utilize our data better.

Simple K-Folds — We split our data into K parts, let's use K= 5 for example, then we will have 5 models. Use the means value of all 5 output to predict for new input.

Final Test Result:

Roc Auc Curve for Lgb model



F1-Score: For LGB model, My f1-score is 0.77

Confusion Matrix:

confusion matrix

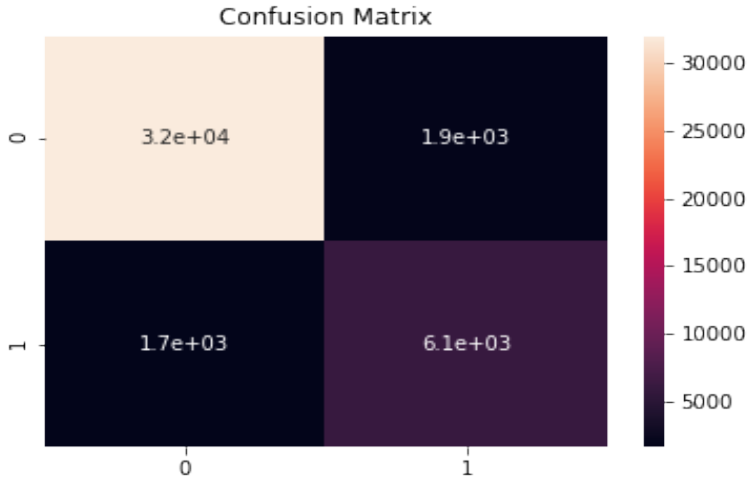
LGB Model

```
array([[31983,  1914],
       [ 1697,  6126]])
```

confusion matrix

random forest

```
array([[33897,     0],
       [ 7513,   310]])
```



References:

https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html

https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html