Final Notebook

This notebook contain whole pipeline from data processing, featurization, model building to predicting the target values for Elo Merchant Category Recommendation.

Blog

https://gouravrathore99.medium.com/a-case-study-on-elo-merchant-category-recommendation-part-ii-c61641a8b0c5

In [1]:
```
import pandas as pd
import numpy as np
from tqdm import tqdm
import os
import datetime
import pickle
import warnings
warnings.simplefilter("ignore")
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LogisticRegression
from joblib import dump, load
import xgboost as xgb
import func
```

Function 1

In [2]:
```
def function_1(card_id):
    """This function include entire pipeline, from data preprocessing to making final predictions.
    It take in card_id from test as string for input and return loyalty score for the card_id."""

    if not os.path.exists("data/final_train.csv") or not os.path.exists("data/final_test.csv"):

        if not os.path.exists("data/train_featurized.csv") or not os.path.exists("data/test_featurized.csv") or
        not os.path.exists("data/all_transaction_features.csv"):

            if not os.path.exists("data/train_processed.csv") or not os.path.exists("data/test_processed.csv") or
            not os.path.exists("data/historical_transactions_processed.csv") or\
            not os.path.exists("data/new_transactions_processed.csv"):

                train, test, historical_transactions, new_transactions = func.load_data()
                train, test = func.process_train_test(train, test)
                historical_transactions, new_transactions = func.process_transactions(historical_transactions,
                                                                                      new_transactions)
                train, test = func.feature_train_test(train, test)
                all_transaction_features = func.feature_transactions(historical_transactions, new_transactions)
                final_train, final_test = func.data_prepare(train, test, all_transaction_features)

            else:
                train = pd.read_csv("data/train_processed.csv")
                test = pd.read_csv("data/test_processed.csv")
                train, test = func.feature_train_test(train, test)
                historical_transactions = pd.read_csv("data/historical_transactions_processed.csv")
                new_transactions = pd.read_csv("data/new_transactions_processed.csv")
                all_transaction_features = func.feature_transactions(historical_transactions, new_transactions)
                final_train, final_test = func.data_prepare(train, test, all_transaction_features)

        else:
            train = pd.read_csv('data/train_featurized.csv')
            test = pd.read_csv('data/test_featurized.csv')
            all_transaction_features = pd.read_csv('data/all_transaction_features.csv')
            final_train, final_test = func.data_prepare(train, test, all_transaction_features)

    else:
        final_test = pd.read_csv('data/final_test.csv')
        final_test = func.reduce_mem_usage(final_test)

    if not os.path.exists("data/Model1.sav") or not os.path.exists("data/Model2.sav") or\
    not os.path.exists("data/Model3.sav") or not os.path.exists("data/Model4.sav"):
        final_train = pd.read_csv('data/final_train.csv')
        final_train = func.reduce_mem_usage(final_train)
        func.build_model(final_train)

    if str(card_id) in final_test['card_id'].astype('string').values:
        X_test = final_test[final_test['card_id'] == str(card_id)].drop(columns = ['card_id'])
        y = 0
        for i in range(4):
            model = load("".join(("data/Model", str(i + 1), ".sav")))
            y += (func.predict(X_test, model) / 4)
        return y
    else:
        print("Sorry, no data available for Card Id", card_id)
        return
```

Function 2

In [3]:
```
def function_2(card_id, target):
    """This function include entire pipeline, from data preprocessing to making final predictions.
    It takes in card_id from test as string and loyalty score for input and returns the evaluation
    mertric for the model."""

    Y_test_pred = function_1(card_id)
    rmse = np.sqrt(mean_squared_error([target], [Y_test_pred]))

    return rmse
```

In [4]:
```
card_id = "C_ID_0ab67a22ab"
test_pred = function_1(card_id)
print("The Loyalty Score for Card ID {} is {}".format(card_id, test_pred))
```

```
The Loyalty Score for Card ID C_ID_0ab67a22ab is -3.8906837105751038
```

In [5]:
```
target = -3.8906837105751038
rsme = function_2(card_id, target)
print("The RSME Score for Model is", rsme)
```

```
The RSME Score for Model is 0.0
```