In [ ]:
```
import pandas as pd
import numpy as np
import warnings
import time
warnings.filterwarnings("ignore")
import lightgbm as lgb
from bayes_opt import BayesianOptimization
from sklearn.metrics import roc_auc_score
```

In [ ]:
```
application_train = pd.read_csv('../input/application_train.csv')
from sklearn.preprocessing import LabelEncoder
def label_encoder(input_df, encoder_dict=None):
    """ Process a dataframe into a form useable by LightGBM """
    # Label encode categoricals
    categorical_feats = input_df.columns[input_df.dtypes == 'object']
    for feat in categorical_feats:
        encoder = LabelEncoder()
        input_df[feat] = encoder.fit_transform(input_df[feat].fillna('N
ULL'))
    return input_df, categorical_feats.tolist(), encoder_dict
application_train, categorical_feats, encoder_dict = label_encoder(appl
ication_train)
X = application_train.drop('TARGET', axis=1)
y = application_train.TARGET
```

In [ ]:
```
def lgb_eval(num_leaves, feature_fraction, bagging_fraction, max_depth,
 lambda_l1, lambda_l2, min_split_gain, min_child_weight):
    params = {'application':'binary','num_iterations':4000, 'learning_r
ate':0.05, 'early_stopping_round':100, 'metric':'auc'}
    params["num_leaves"] = round(num_leaves)
    params['feature_fraction'] = max(min(feature_fraction, 1), 0)
    params['bagging_fraction'] = max(min(bagging_fraction, 1), 0)
    params['max_depth'] = round(max_depth)
    params['lambda_l1'] = max(lambda_l1, 0)
    params['lambda_l2'] = max(lambda_l2, 0)
```

```
    params['min_split_gain'] = min_split_gain
    params['min_child_weight'] = min_child_weight
    cv_result = lgb.cv(params, train_data, nfold=n_folds, seed=random_s
eed, stratified=True, verbose_eval =200, metrics=['auc'])
    return max(cv_result['auc-mean'])
```

In [ ]:
```
lgbBO = BayesianOptimization(lgb_eval, {'num_leaves': (24, 45),
                                        'feature_fraction': (0.1, 0.9),
                                        'bagging_fraction': (0.8, 1),
                                        'max_depth': (5, 8.99),
                                        'lambda_l1': (0, 5),
                                        'lambda_l2': (0, 3),
                                        'min_split_gain': (0.001, 0.1),
                                        'min_child_weight': (5, 50)}, r
andom_state=0)
```

In [ ]:
```
 lgbBO.maximize(init_points=init_round, n_iter=opt_round)
```

In [ ]:
```
 lgbBO.res['max']['max_params']
```

In [ ]:
```
X = application_train.drop('TARGET', axis=1)
y = application_train.TARGET
def bayes_parameter_opt_lgb(X, y, init_round=15, opt_round=25, n_folds=
5, random_seed=6, n_estimators=10000, learning_rate=0.05, output_proces
s=False):
    # prepare data
    train_data = lgb.Dataset(data=X, label=y, categorical_feature = cat
egorical_feats, free_raw_data=False)
    # parameters
    def lgb_eval(num_leaves, feature_fraction, bagging_fraction, max_de
pth, lambda_l1, lambda_l2, min_split_gain, min_child_weight):
        params = {'application':'binary','num_iterations': n_estimators
, 'learning_rate':learning_rate, 'early_stopping_round':100, 'metric':
'auc'}
        params["num_leaves"] = int(round(num_leaves))
        params['feature_fraction'] = max(min(feature_fraction, 1), 0)
        params['bagging_fraction'] = max(min(bagging_fraction, 1), 0)
        params['max_depth'] = int(round(max_depth))
```

```
        params['lambda_l1'] = max(lambda_l1, 0)
        params['lambda_l2'] = max(lambda_l2, 0)
        params['min_split_gain'] = min_split_gain
        params['min_child_weight'] = min_child_weight
        cv_result = lgb.cv(params, train_data, nfold=n_folds, seed=rand
om_seed, stratified=True, verbose_eval =200, metrics=['auc'])
        return max(cv_result['auc-mean'])
    # range
    lgbBO = BayesianOptimization(lgb_eval, {'num_leaves': (24, 45),
                                            'feature_fraction': (0.1,
0.9),
                                            'bagging_fraction': (0.8, 1
),
                                            'max_depth': (5, 8.99),
                                            'lambda_l1': (0, 5),
                                            'lambda_l2': (0, 3),
                                            'min_split_gain': (0.001,
0.1),
                                            'min_child_weight': (5, 50
)}, random_state=0)
    # optimize
    lgbBO.maximize(init_points=init_round, n_iter=opt_round)

    # output optimization process
    if output_process==True: lgbBO.points_to_csv("bayes_opt_result.csv"
)

    # return best parameters
    return lgbBO.res['max']['max_params']

opt_params = bayes_parameter_opt_lgb(X, y, init_round=5, opt_round=10,
n_folds=3, random_seed=6, n_estimators=100, learning_rate=0.05)
```

In [ ]:
```
print(opt_params)
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]: