In [0]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?clien
t_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.co
m&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2
F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.co
m%2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.re
adonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&respon
se_type=code (https://accounts.google.com/o/oauth2/auth?client_id=9473189898
03-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=
urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2F%2Fwww.googleap
is.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive%
20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3
A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code)

Enter your authorization code:
··········
Mounted at /content/drive
```

In [0]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import StandardScaler
import random
import lightgbm as lgb
import seaborn as sns
```

In [0]:

```
train = pd.read_csv('/content/drive/My Drive/train.csv')
test = pd.read_csv('/content/drive/My Drive/test.csv')
train_copy = train.copy()
test_copy = test.copy()
```

In [0]:

```
train_y = train_copy['target'].values
train_X_column_name = train_copy.drop(['target', 'ID_code'], axis=1).columns
train_X = train_copy.drop(['target', 'ID_code'], axis=1).values
test_X = test_copy.drop(['ID_code'], axis=1).values
train_X_copy = train_X.copy()
test_X_copy = test_X.copy()
```

In [0]:

```
unique_samples = []
unique_count = np.zeros_like(test_X)
for feature in range(test_X.shape[1]):
    _, index_, count_ = np.unique(test_X[:, feature], return_index=True, return_counts=True
    unique_count[index_[count_ == 1], feature] += 1

real_sample_index = np.argwhere(np.sum(unique_count, axis=1) > 0)[:, 0]
synthetic_sample_index = np.argwhere(np.sum(unique_count, axis=1) == 0)[:, 0]

test_X_real = test_X[real_sample_index].copy()
```

In [0]:

```
print('There are ' + str(len(real_sample_index)) + ' real data samples in test set')
print('There are ' + str(len(synthetic_sample_index)) + ' synthetic data samples in test se
```

```
There are 100000 real data samples in test set
There are 100000 synthetic data samples in test set
```

In [0]:

```
generator_for_each_synthetic_sample = []
for cur_sample_index in synthetic_sample_index[:20000]:
    cur_synthetic_sample = test_X[cur_sample_index]
    potential_generators = test_X_real == cur_synthetic_sample

    features_mask = np.sum(potential_generators, axis=0) == 1
    verified_generators_mask = np.any(potential_generators[:, features_mask], axis=1)
    verified_generators_for_sample = real_sample_index[np.argwhere(verified_generators_mask
    generator_for_each_synthetic_sample.append(set(verified_generators_for_sample))
```

In [0]:

```
gen = generator_for_each_synthetic_sample[0]
for x in generator_for_each_synthetic_sample:
    if gen.intersection(x):
        gen = gen.union(x)

LB = generator_for_each_synthetic_sample[1]
for x in generator_for_each_synthetic_sample:
    if LB.intersection(x):
        LB = LB.union(x)

LB = list(LB)
gen = list(gen)
full = np.concatenate([train_X, np.concatenate([test_X[LB], test_X[gen]])])
```

In [0]:

```
print('There are ' + str(len(LB)) + ' data samples for public score in real data set')
print('There are ' + str(len(gen)) + ' data samples for private score in real data set')
```

```
There are 50000 data samples for public score in real data set
There are 50000 data samples for private score in real data set
```

In [0]:

```
full = pd.DataFrame(full)
full.columns = train_X_column_name
train_X = pd.DataFrame(train_X)
train_X.columns = train_X_column_name
test_X = pd.DataFrame(test_X)
test_X .columns = train_X_column_name

for feat in ['var_' + str(x) for x in range(200)]:
    count_values = full.groupby(feat)[feat].count()
    train_X['new_' + feat] = count_values.loc[train_X[feat]].values
    test_X['new_' + feat] = count_values.loc[test_X[feat]].values
```

In [0]:

```
seed = 0
param = {
    'num_leaves': 8,
    'min_data_in_leaf': 17,
    'learning_rate': 0.01,
    'min_sum_hessian_in_leaf': 9.67,
    'bagging_fraction': 0.8329,
    'bagging_freq': 2,
    'feature_fraction': 1,
    'lambda_l1': 0.6426,
    'lambda_l2': 0.3067,
    'min_gain_to_split': 0.02832,
    'max_depth': -1,
    'seed': seed,
    'feature_fraction_seed': seed,
    'bagging_seed': seed,
    'drop_seed': seed,
    'data_random_seed': seed,
    'objective': 'binary',
    'boosting_type': 'gbdt',
    'verbosity': -1,
    'metric': 'auc',
    'is_unbalance': True,
    'save_binary': True,
    'boost_from_average': 'false',
    'num_threads': 8
}
```

In [0]:

```
iterations = 110
test_hat = np.zeros([200000, 200])
i = 0
for feature in ['var_' + str(x) for x in range(200)]:  # loop over all features
    # print(feature)
    feat_choices = [feature, 'new_' + feature]
    lgb_train = lgb.Dataset(train_X[feat_choices], train_y)
    gbm = lgb.train(param, lgb_train, iterations, verbose_eval=-1)
    test_hat[:, i] = gbm.predict(test_X[feat_choices], num_iteration=gbm.best_iteration)
    i += 1

sub_preds = test_hat.sum(axis=1)
```

In [0]:

```
sub = pd.DataFrame()
sub['ID_code'] = test_copy['ID_code']
sub['target'] = sub_preds
sub.to_csv('/content/drive/My Drive/submission.csv', index=False)
```