In [0]:

```
def featurization(X):
    """Create 800 more new features from 200 original features .
    It will create below new features:
    a. Duplicate Count: Take minimum of 10 and value count for that particular value.
    b. Duplicate Value Count >2 : Multiply actual value of that feature with duplicate count (if o
nly duplicate count greater than 2)
    c. Duplicate Value Count >4 : Multiply actual value of that feature with duplicate count (if o
nly duplicate count greater than 4)
    d. Distance of mean : Calculate difference between current value and mean of that particular f
eature .Then mutiply it with duplicate count feature."""
    import pandas as pd
    from tqdm import tqdm
    import numpy as np
    from sklearn.model_selection import train_test_split
    import os
    import lightgbm as lgb
    from sklearn.metrics import roc_auc_score
    import pickle

    filename = '/content/drive/My Drive/proj_1/train_test_real.sav'
    target='target'
    features = [i for i in X.columns]
    train_test_real = pickle.load(open(filename, 'rb'))

    ##https://www.kaggle.com/super13579/lgbm-with-duplicate-flag-value-0-923?
scriptVersionId=12330297
    for f in tqdm(features):
        count=train_test_real[f].value_counts(dropna=True)
        X[f+'dup_count'] = X[f].map(count).map(lambda x:min(10,x)).astype(np.uint8)
        X[f + '_dup_value_2'] = X[f]* (X[f + 'dup_count'].map(lambda x:int(x>2))).astype(np.float32)
        X[f + '_dup_value_4'] = X[f]* (X[f + 'dup_count'].map(lambda x:int(x>4))).astype(np.float32)
    for f in tqdm(features):
        X[f+'distance_of_mean'] = X[f]-train_test_real[f].mean()
        X[f+'distance_of_mean'] = (X[f+'distance_of_mean']* X[f+'dup_count'].map(lambda x:int(x>1)))
.astype(np.float32)

    return X
```

In [0]:

```
def final_fun_score(X,Y):
    """Calculate auc score between actual target value and predicted target value"""
    import pandas as pd
    from tqdm import tqdm
    import numpy as np
    from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score
    import os
    import lightgbm as lgb
    from sklearn.metrics import roc_auc_score,roc_curve,auc
    import pickle
    import warnings
    import pickle

    warnings.filterwarnings("ignore")
    from google.colab import drive
    drive.mount('/content/drive')

    X.drop(['target','ID_code'],axis=1,inplace=True)
    print('Shape of input data before featurization'+str(X.shape))
    X=featurization(X)

    print('Shape of input data after featurization'+str(X.shape))

    pred=0
    for i in range(5):
        lm =lgb.Booster(model_file='/content/drive/My Drive/proj_1/model_1000_iteration_{}.sav'.for
mat(i))
        pred+=lm.predict(X)

    y_pred=pred/5
    y_pred=pd.DataFrame(y_pred)
```

```
    val_auc=roc_auc_score(Y, y_pred)
    print('auc:'+str(val_auc))

    return val_auc
```

In [0]:

```
def final_fun_predict(X):
    """Calculate predicted target value for input data"""
    import warnings
    import pandas as pd
    from tqdm import tqdm
    import numpy as np
    from sklearn.model_selection import train_test_split
    import os
    import lightgbm as lgb
    from sklearn.metrics import roc_auc_score

    import pickle
    warnings.filterwarnings("ignore")
    from google.colab import drive
    drive.mount('/content/drive')

    X.drop(['target','ID_code'],axis=1,inplace=True)
    print('Shape of input data before featurization'+str(X.shape))

    X=featurization(X)

    print('Shape of input data after featurization'+str(X.shape))

    #drop target and ID_code
    pred=0
    for i in range(5):
        lm =lgb.Booster(model_file='/content/drive/My Drive/proj_1/model_1000_iteration_{}.sav'.for
mat(i))
        pred+=lm.predict(X)

    y_pred=pred/5
    y_pred=pd.DataFrame(y_pred)

    return y_pred
```

Sample Test runs

In [9]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from google.colab import drive
drive.mount('/content/drive')
tr_data = pd.read_csv('/content/drive/My Drive/proj_1/train.csv')
y=tr_data['target']
X_train, X_test, y_train, y_test = train_test_split(tr_data, y, test_size = 0.20, stratify=y)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

In [5]:

```
acc=final_fun_score(X_test,y_test)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
Shape of input data before featurization(40000, 200)

100%|██████████| 200/200 [00:18<00:00, 10.61it/s]
100%|██████████| 200/200 [00:04<00:00, 31.72it/s]
```

```
Shape of input data after featurization(40000, 1000)
auc:0.873956281682195
```

In [10]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from google.colab import drive
drive.mount('/content/drive')
tr_data = pd.read_csv('/content/drive/My Drive/proj_1/train.csv')
y=tr_data['target']
X_train, X_test, y_train, y_test = train_test_split(tr_data, y, test_size = 0.20, stratify=y)
X_1=X_test.head(10)
print(X_1.shape)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
(10, 202)
```

In [11]:

```
pred=final_fun_predict(X_1)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
Shape of input data before featurization(10, 200)

100%|██████████| 200/200 [00:07<00:00, 28.00it/s]
100%|██████████| 200/200 [00:00<00:00, 403.95it/s]

Shape of input data after featurization(10, 1000)
```

In [12]:

```
print(pred)
```

```
          0
0  0.999977
1  0.999505
2  0.999985
3  0.999850
4  0.999906
5  0.999996
6  0.999975
7  0.999803
8  0.999876
9  0.999985
```

In [13]:

```
X_test.head(10)
```

Out[13]:

	ID_code	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	var_9	var_10	v
165242	train_165242	0	6.5401	6.9738	9.4903	4.2416	11.7262	-3.3781	5.7018	11.0580	4.7492	7.7976	-3.2499	-1
183855	train_183855	0	14.4793	6.5271	11.8568	7.7547	9.4347	-10.7928	6.1930	16.5088	-0.1384	8.0878	6.0408	-15
34281	train_34281	0	5.8153	2.5905	10.6170	7.2382	10.1710	-13.8336	5.1741	12.8028	3.4420	7.7383	-7.0680	1.
37995	train_37995	0	13.8082	-6.6342	10.0120	7.3657	12.7025	-19.7165	5.8690	17.4265	-1.8203	8.0585	-3.2560	-8
176993	train_176993	0	14.7594	-	12.4909	7.6346	10.5156	3.8747	4.7999	18.3569	4.2355	6.6202	3.0013	4.

	ID_code	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	var_9	var_10	v
				0.4673										
133070	train_133070	1	15.3286	1.8768	10.7884	7.1206	11.4307	-14.7345	6.9288	23.8428	1.8660	8.5872	10.0652	7.
121780	train_121780	0	10.4105	4.1128	10.8918	8.9215	13.0706	-2.9346	6.1087	13.9742	3.7364	5.9349	4.5088	-8
76790	train_76790	0	7.9070	2.5839	8.9171	5.1452	12.5271	10.8810	6.2358	14.8463	-2.7941	6.9351	-5.5187	-7
10702	train_10702	0	9.8550	4.2773	9.7551	8.1443	12.6756	-1.7684	5.9037	15.8464	2.6871	7.1397	6.4490	-6
55455	train_55455	0	8.2779	6.6452	13.4030	4.8533	8.9099	-4.7034	4.3028	17.4599	-5.3796	9.3989	8.5484	-5

10 rows × 202 columns