DSN AI Bootcamp Qualification Hackathon

Kowope Mart is a Nigerian-based retail company with a vision to provide quality goods, education and automobile services to its customers at affordable price and reduce if not eradicate charges on card payments and increase customer satisfaction with credit rewards that can be used within the Mall. To achieve this, the company has partnered with DSBank on co-branded credit card with additional functionality such that customers can request for loan, pay for goods even with zero-balance and then pay back within an agreed period of time. This innovative strategy has increased sales for the company. However, there has been recent cases of credit defaults and Kowope Mart will like to have a system that profiles customers who are worthy of the card with minimum if not zero risk of defaulting.

Importing Libraries

In [1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

# To allow maximum display of rows and columns
pd.set_option("display.max_rows", 999)
pd.set_option("display.max_columns", 999)
```

In [2]:

```python
train = pd.read_csv('Train.csv')
test = pd.read_csv('Test.csv')
sample = pd.read_csv('SampleSubmission.csv')
```

In [3]:

```python
train.head()
```

	Applicant_ID	form_field1	form_field2	form_field3	form_field4	form_field5	form_field6	form_field7	form_field8	form_field
0	Apcnt_1000000	3436.0	0.28505	1.6560	0.0	0.000	0.0	10689720.0	252072.0	4272776.0
1	Apcnt_1000004	3456.0	0.67400	0.2342	0.0	0.000	0.0	898979.0	497531.0	9073814.0
2	Apcnt_1000008	3276.0	0.53845	3.1510	0.0	6.282	NaN	956940.0	NaN	192944.0
3	Apcnt_1000012	3372.0	0.17005	0.5050	0.0	0.000	192166.0	3044703.0	385499.0	3986472.0
4	Apcnt_1000016	3370.0	0.77270	1.1010	0.0	0.000	1556.0	214728.0	214728.0	1284089.0

Basic EDA

In [4]:

```python
train['default_status'].value_counts()
```

```
no     42285
yes    13715
Name: default_status, dtype: int64
```

In [5]:
```
# Check for missing data
train.isnull().sum()
```

```
Applicant_ID        0
form_field1      2529
form_field2      3844
form_field3       355
form_field4       355
form_field5       355
form_field6     13360
form_field7      5163
form_field8     13360
form_field9      8008
form_field10      355
form_field11    31421
form_field12     9895
form_field13     5889
form_field14        0
form_field15    22475
form_field16    13036
form_field17    11151
form_field18    10402
form_field19        4
form_field20      355
form_field21    15854
form_field22    20400
form_field23    28123
form_field24    13297
form_field25     5450
form_field26     7438
form_field27     9299
form_field28      355
form_field29      355
form_field30    25509
form_field31    39408
form_field32     5450
form_field33     1256
form_field34      355
form_field35    23148
form_field36     1995
form_field37     5450
form_field38      355
form_field39     4211
form_field40    43729
form_field41    38229
form_field42     1323
form_field43      568
form_field44     5383
form_field45    31317
form_field46    15904
form_field47        0
form_field48    20889
form_field49      355
form_field50    11056
default_status      0
dtype: int64
```

In [6]:
```
train['default_status'] = train['default_status'].replace({"yes":1, "no":0})
```

In [7]:
```
train.head()
```

	Applicant_ID	form_field1	form_field2	form_field3	form_field4	form_field5	form_field6	form_field7	form_field8	form_field
0	Apcnt_1000000	3436.0	0.28505	1.6560	0.0	0.000	0.0	10689720.0	252072.0	4272776.0
1	Apcnt_1000004	3456.0	0.67400	0.2342	0.0	0.000	0.0	898979.0	497531.0	9073814.0
2	Apcnt_1000008	3276.0	0.53845	3.1510	0.0	6.282	NaN	956940.0	NaN	192944.0
3	Apcnt_1000012	3372.0	0.17005	0.5050	0.0	0.000	192166.0	3044703.0	385499.0	3986472.0
4	Apcnt_1000016	3370.0	0.77270	1.1010	0.0	0.000	1556.0	214728.0	214728.0	1284089.0

In [8]:
```
train.fillna(-9999999, inplace=True)
test.fillna(-9999999, inplace=True)
```

In [9]:
```
train.head(3)
```

	Applicant_ID	form_field1	form_field2	form_field3	form_field4	form_field5	form_field6	form_field7	form_field8	form_field
0	Apcnt_1000000	3436.0	0.28505	1.6560	0.0	0.000	0.0	10689720.0	252072.0	4272776.0
1	Apcnt_1000004	3456.0	0.67400	0.2342	0.0	0.000	0.0	898979.0	497531.0	9073814.0
2	Apcnt_1000008	3276.0	0.53845	3.1510	0.0	6.282	-9999999.0	956940.0	-9999999.0	192944.0

In [10]:
```
corr_train = train.corr()
```

In [11]:
```
pd.options.display.max_rows = 999
corr_train['default_status'].sort_values(ascending=False)
```

```
default_status    1.000000
form_field11      0.248940
form_field40      0.114511
form_field31      0.106481
form_field2       0.073154
form_field36      0.031275
form_field14      0.008272
form_field38      0.004157
form_field5       0.004157
form_field4       0.004157
form_field3       0.004156
form_field49      0.004156
form_field20      0.004156
form_field34      0.004156
form_field29      0.001627
form_field19     -0.000097
form_field43     -0.002028
form_field28     -0.003263
form_field42     -0.010385
form_field33     -0.019181
form_field41     -0.022153
form_field35     -0.031690
form_field23     -0.045615
form_field30     -0.055623
form_field1      -0.064350
form_field46     -0.073663
form_field22     -0.089150
form_field48     -0.099327
form_field13     -0.104310
form_field44     -0.112228
form_field15     -0.117681
form_field50     -0.121536
form_field39     -0.129072
form_field27     -0.138463
form_field7      -0.139719
form_field26     -0.141957
form_field21     -0.160028
form_field37     -0.162129
form_field32     -0.162196
form_field25     -0.162253
form_field10     -0.169186
form_field12     -0.186554
form_field24     -0.207701
form_field16     -0.208048
form_field17     -0.213790
form_field45     -0.216719
form_field18     -0.218506
form_field6      -0.234609
form_field9      -0.254291
form_field8      -0.272644
Name: default_status, dtype: float64
```

In [12]:
```
import datasist as ds
```

In [13]:
```
ylabel = train.default_status
df_train = train.drop(columns=['default_status'])

all_data, ntrain, ntest = ds.structdata.join_train_and_test(df_train, test)
```

In [14]:
```
all_data.shape
```

```
(80000, 51)
```

In [15]:
```
ds.structdata.display_missing(all_data)
```

	features	missing_counts	missing_percent
0	Applicant_ID	0	0.0
1	form_field1	0	0.0
2	form_field2	0	0.0
3	form_field3	0	0.0
4	form_field4	0	0.0
5	form_field5	0	0.0
6	form_field6	0	0.0
7	form_field7	0	0.0
8	form_field8	0	0.0
9	form_field9	0	0.0
10	form_field10	0	0.0
11	form_field11	0	0.0
12	form_field12	0	0.0
13	form_field13	0	0.0
14	form_field14	0	0.0
15	form_field15	0	0.0
16	form_field16	0	0.0
17	form_field17	0	0.0
18	form_field18	0	0.0
19	form_field19	0	0.0
20	form_field20	0	0.0
21	form_field21	0	0.0
22	form_field22	0	0.0
23	form_field23	0	0.0
24	form_field24	0	0.0
25	form_field25	0	0.0
26	form_field26	0	0.0
27	form_field27	0	0.0
28	form_field28	0	0.0
29	form_field29	0	0.0
30	form_field30	0	0.0
31	form_field31	0	0.0
32	form_field32	0	0.0
33	form_field33	0	0.0
34	form_field34	0	0.0
35	form_field35	0	0.0
36	form_field36	0	0.0
37	form_field37	0	0.0
38	form_field38	0	0.0
39	form_field39	0	0.0
40	form_field40	0	0.0
41	form_field41	0	0.0
42	form_field42	0	0.0
43	form_field43	0	0.0
44	form_field44	0	0.0
45	form_field45	0	0.0
46	form_field46	0	0.0
47	form_field47	0	0.0
48	form_field48	0	0.0
49	form_field49	0	0.0
50	form_field50	0	0.0

Feature engineering

In [16]:
```
all_data['form_field51'] = all_data['form_field3'] + all_data['form_field4'] + all_data['form_field5']
all_data['form_field52'] = all_data['form_field32'] + all_data['form_field33']
all_data['form_field53'] = all_data['form_field14'] + all_data['form_field15']
all_data['form_field54'] = all_data['form_field29'] / 4
all_data['form_field55'] = all_data['form_field30'] / 4
all_data['form_field56'] = all_data['form_field31'] / 4
```

In [17]:
```python
corr_all_data = all_data.corr()
```

In [18]:
```python
pd.options.display.max_rows = 999
corr_all_data
```

	form_field1	form_field2	form_field3	form_field4	form_field5	form_field6	form_field7	form_field8	form_field9	for
form_field1	1.000000	0.090485	0.363649	0.363650	0.363650	0.333756	0.163851	0.297202	0.186195	0.10
form_field2	0.090485	1.000000	0.074331	0.074331	0.074331	0.006698	-0.023951	-0.020457	-0.018356	-0.0
form_field3	0.363649	0.074331	1.000000	1.000000	1.000000	0.137196	0.066988	0.119931	0.075849	0.06
form_field4	0.363650	0.074331	1.000000	1.000000	1.000000	0.137196	0.066988	0.119931	0.075849	0.06
form_field5	0.363650	0.074331	1.000000	1.000000	1.000000	0.137195	0.066987	0.119930	0.075848	0.06
form_field6	0.333756	0.006698	0.137196	0.137196	0.137195	1.000000	0.264588	0.882272	0.409258	0.25
form_field7	0.163851	-0.023951	0.066988	0.066988	0.066987	0.264588	1.000000	0.364294	0.361523	0.88
form_field8	0.297202	-0.020457	0.119931	0.119931	0.119930	0.882272	0.364294	1.000000	0.580688	0.40
form_field9	0.186195	-0.018356	0.075849	0.075849	0.075848	0.409258	0.361523	0.580688	1.000000	0.51
form_field10	0.108814	-0.028845	0.066058	0.066058	0.066057	0.254101	0.888935	0.406593	0.519760	1.00
form_field11	-0.094502	0.056548	0.061044	0.061044	0.061046	-0.386758	-0.176264	-0.419993	-0.333220	-0.2
form_field12	0.309596	0.040044	0.130079	0.130080	0.130078	0.648271	0.245916	0.637386	0.468784	0.25
form_field13	0.136002	-0.003594	0.052416	0.052416	0.052416	0.228996	0.333995	0.326657	0.289508	0.41
form_field14	-0.019352	-0.004436	-0.018748	-0.018748	-0.018748	-0.010659	-0.000717	-0.008310	-0.001642	0.00
form_field15	0.056355	-0.006425	0.015046	0.015046	0.015046	0.148205	0.185529	0.206292	0.190935	0.19
form_field16	0.349273	0.048420	0.144689	0.144689	0.144687	0.948075	0.223413	0.828788	0.366949	0.20
form_field17	0.378476	0.049003	0.159731	0.159731	0.159729	0.858840	0.226386	0.750775	0.360668	0.19
form_field18	0.381633	0.049526	0.166821	0.166821	0.166820	0.822330	0.230536	0.718859	0.349602	0.18
form_field19	0.032395	0.005027	0.089076	0.089076	0.089076	0.012226	0.005967	0.010686	0.006757	0.00
form_field20	0.364015	0.074837	0.998997	0.998997	0.998997	0.137335	0.067056	0.120053	0.075926	0.06
form_field21	0.312894	0.055087	0.126666	0.126666	0.126665	0.804854	0.200804	0.722218	0.344354	0.18
form_field22	0.232314	0.197880	0.104922	0.104922	0.104921	0.527609	0.100819	0.465687	0.228921	0.08
form_field23	0.174605	0.178689	0.078894	0.078894	0.078893	0.418854	0.048178	0.361023	0.171249	0.03
form_field24	0.361732	0.049899	0.142723	0.142723	0.142722	0.755948	0.217341	0.679252	0.399768	0.20
form_field25	0.453969	0.058736	0.243150	0.243150	0.243149	0.564324	0.173605	0.493384	0.291101	0.14
form_field26	0.453662	0.078763	0.203014	0.203014	0.203013	0.586396	0.178419	0.519457	0.313733	0.15
form_field27	0.409027	0.077051	0.178509	0.178509	0.178508	0.649131	0.176105	0.575226	0.326902	0.16
form_field28	0.364144	0.070542	0.998419	0.998419	0.998419	0.147910	0.072781	0.133764	0.087658	0.07
form_field29	0.366114	0.073506	0.998923	0.998923	0.998923	0.140503	0.070252	0.124335	0.081592	0.07
form_field30	0.199427	0.100145	0.086845	0.086846	0.086844	0.119993	0.071502	0.096088	0.101844	0.08
form_field31	0.124381	0.053985	0.051633	0.051633	0.051637	-0.077316	-0.069715	-0.108466	-0.121242	-0.0
form_field32	0.453962	0.058838	0.243150	0.243150	0.243149	0.564317	0.173550	0.493369	0.291148	0.14
form_field33	0.082967	0.012590	0.136489	0.136489	0.136489	0.052363	0.018023	0.048059	0.033551	0.01
form_field34	0.364014	0.074837	0.998997	0.998997	0.998997	0.137335	0.067056	0.120052	0.075926	0.06
form_field35	0.217429	0.205138	0.094705	0.094705	0.094704	0.487321	0.061305	0.428719	0.196248	0.05
form_field36	0.305378	0.084716	0.409470	0.409470	0.409470	0.089830	0.067134	0.076947	0.073945	0.07
form_field37	0.453945	0.058863	0.243150	0.243150	0.243149	0.564249	0.173462	0.493244	0.290922	0.14
form_field38	0.364015	0.074839	0.998997	0.998997	0.998997	0.137335	0.067055	0.120053	0.075926	0.06
form_field39	0.585674	0.075534	0.277791	0.277791	0.277790	0.493886	0.241147	0.431735	0.226516	0.13
form_field40	0.112209	0.053691	0.041919	0.041919	0.041921	-0.056559	-0.054575	-0.094329	-0.121439	-0.0
form_field41	0.141392	0.111504	0.054179	0.054179	0.054179	0.120532	0.060081	0.095531	0.057957	0.07
form_field42	0.429914	0.074286	0.483858	0.483858	0.483859	0.118835	0.075988	0.114781	0.126617	0.08
form_field43	0.374359	0.071975	0.781042	0.781042	0.781043	0.103973	0.061064	0.094758	0.097116	0.06
form_field44	0.542233	0.085558	0.242471	0.242472	0.242471	0.485401	0.244274	0.430281	0.224412	0.14
form_field45	0.175617	-0.023040	0.070269	0.070269	0.070268	0.253300	0.271173	0.298810	0.367056	0.32
form_field46	0.273455	0.099287	0.125781	0.125782	0.125781	0.147299	0.099878	0.125891	0.124004	0.10
form_field48	0.231807	0.181026	0.099368	0.099368	0.099367	0.541163	0.116934	0.492191	0.266457	0.11
form_field49	0.363650	0.074331	1.000000	1.000000	1.000000	0.137197	0.066988	0.119932	0.075849	0.06
form_field50	0.215494	0.337752	0.098173	0.098173	0.098173	0.316908	0.136030	0.316853	0.344681	0.15
form_field51	0.363650	0.074331	1.000000	1.000000	1.000000	0.137196	0.066988	0.119931	0.075849	0.06
form_field52	0.432663	0.056873	0.271631	0.271631	0.271630	0.515837	0.159472	0.451972	0.268968	0.13
form_field53	-0.017342	-0.004662	-0.018204	-0.018204	-0.018204	-0.005396	0.005865	-0.000987	0.005132	0.00
form_field54	0.366114	0.073506	0.998923	0.998923	0.998923	0.140503	0.070252	0.124335	0.081592	0.07
form_field55	0.199427	0.100145	0.086845	0.086846	0.086844	0.119993	0.071502	0.096088	0.101844	0.08
form_field56	0.124381	0.053985	0.051633	0.051633	0.051637	-0.077316	-0.069715	-0.108466	-0.121242	-0.0

In [19]:
```
all_data.head(2)
```

	Applicant_ID	form_field1	form_field2	form_field3	form_field4	form_field5	form_field6	form_field7	form_field8	form_field
0	Apcnt_1000000	3436.0	0.28505	1.6560	0.0	0.0	0.0	10689720.0	252072.0	4272776.0
1	Apcnt_1000004	3456.0	0.67400	0.2342	0.0	0.0	0.0	898979.0	497531.0	9073814.0

In [20]:
```
all_data.drop(columns=['Applicant_ID'], inplace=True)
```

In [21]:
```
import category_encoders as ce

enc = ce.OrdinalEncoder()
all_data = enc.fit_transform(all_data)
```

In [22]:
```
all_data.head(3)
```

	form_field1	form_field2	form_field3	form_field4	form_field5	form_field6	form_field7	form_field8	form_field9	form_field1
0	3436.0	0.28505	1.6560	0.0	0.000	0.0	10689720.0	252072.0	4272776.0	11333126.0
1	3456.0	0.67400	0.2342	0.0	0.000	0.0	898979.0	497531.0	9073814.0	2533168.0
2	3276.0	0.53845	3.1510	0.0	6.282	-9999999.0	956940.0	-9999999.0	192944.0	1079864.0

In [23]:
```
#split new train and test
train = all_data[:ntrain]
test = all_data[ntrain:]

print(f"Shape of train {train.shape}")
print(f"Shape of test {test.shape}")
```

```
Shape of train (56000, 56)
Shape of test (24000, 56)
```

Model

In [24]:
```
from sklearn.metrics import roc_auc_score, log_loss
from sklearn.model_selection import StratifiedKFold
import xgboost as xgb
import catboost as cat_
import lightgbm as lgb
```

In [25]:
```python
class func() :
    def __init__(self, train, label, test, model, model_type, random_state):
        self.train, self.label, self.test = train, label, test
        self.model, self.model_type = model, model_type
        self.random_state = random_state

        assert self.model_type in ('catboost', 'xgboost', 'lgbm'), 'Incorrect model_type'
    def __call__(self, plot = True):
        return self.fit(plot)

    def fit(self, plot):
        def catboost_fit(X_train, X_test, y_train, y_test):
            self.model.fit(X_train,y_train,eval_set=[(X_test,y_test)],early_stopping_rounds=500,
                           verbose=50,use_best_model=True)
            x_test_predict = self.model.predict_proba(X_test)[:,1]
            x_train_predict = self.model.predict_proba(X_train)[:,1]
            self.val_p[test_index] = x_test_predict
            self.test_p += self.model.predict_proba(self.test)[:,1]
            return x_test_predict, x_train_predict

        def xgboost_fit(X_train, X_test, y_train, y_test):
            self.model.fit(X_train, y_train, early_stopping_rounds = 30, eval_metric="auc",
                           eval_set=[(X_test, y_test)], verbose = True)
            x_test_predict = self.model.predict_proba(X_test, ntree_limit = self.model.get_booster().be
            x_train_predict = self.model.predict_proba(X_train, ntree_limit = self.model.get_booster().
            self.val_p[test_index] = x_test_predict
            self.test_p += self.model.predict_proba(self.test, ntree_limit = self.model.get_booster().b
            return x_test_predict, x_train_predict

        def lgbm_fit(X_train, X_test, y_train, y_test):
            self.model.fit(X_train, y_train, early_stopping_rounds = 30, eval_metric="auc",
                           eval_set=[(X_test, y_test)], verbose = True)
            x_test_predict = self.model.predict_proba(X_test, num_iteration = self.model.best_iteratior
            x_train_predict = self.model.predict_proba(X_train, num_iteration = self.model.best_iterati
            self.val_p[test_index] = x_test_predict
            self.test_p += self.model.predict_proba(self.test, num_iteration = self.model.best_iteratic
            return x_test_predict, x_train_predict


        self.val_p = np.zeros(self.train.shape[0])
        mean_val = []
        mean_train = []
        self.test_p = np.zeros(self.test.shape[0])
        splits = 5
        kf = StratifiedKFold(n_splits = splits)
        for fold_count, (train_index, test_index) in enumerate(kf.split(self.train, self.label)):
            X_train,X_test = self.train.iloc[train_index],self.train.iloc[test_index]
            y_train,y_test = self.label.iloc[train_index],self.label.iloc[test_index]

            print(f"==============================Fold{fold_count+1}===============================
            if self.model_type == 'catboost': x_test_predict, x_train_predict = catboost_fit(X_train, X
            elif self.model_type == 'xgboost': x_test_predict, x_train_predict = xgboost_fit(X_train, X
            elif self.model_type == 'lgbm': x_test_predict, x_train_predict = lgbm_fit(X_train, X_test,

            print('\nValidation scores', roc_auc_score(y_test, x_test_predict), log_loss(y_test, x_test
            print('Training scores', roc_auc_score(y_train, x_train_predict), log_loss(y_train, x_trair
            mean_val.append(roc_auc_score(y_test, x_test_predict))
            mean_train.append(roc_auc_score(y_train, x_train_predict))

        if plot:
            feat_imp = pd.DataFrame(sorted(zip(self.model.feature_importances_,self.train.columns)), cc
            plt.figure(figsize=(30,25))
            sns.barplot(x="Value", y="Feature", data=feat_imp.sort_values(by="Value", ascending=False))
            plt.ylabel('Feature Importance Score')
            plt.show()
        print(np.mean(mean_val), np.mean(mean_train), np.std(mean_val))
        return self.val_p, self.test_p/splits, self.model
```

In [26]:

```
catboost = cat_.CatBoostClassifier(iterations=5000, learning_rate=0.02, l2_leaf_reg=3.5, depth=8, rsm=0

func_= func(train, ylabel, test, catboost, 'catboost', 600)
val_p1, test_p1, model1 = func_()
```

```
================================Fold1===================================
0:      test: 0.8015353 best: 0.8015353 (0)     total: 200ms    remaining: 16m 39s
50:     test: 0.8281125 best: 0.8281125 (50)    total: 3.45s    remaining: 5m 34s
100:    test: 0.8316572 best: 0.8316572 (100)   total: 6.54s    remaining: 5m 17s
150:    test: 0.8335836 best: 0.8335836 (150)   total: 9.77s    remaining: 5m 13s
200:    test: 0.8351943 best: 0.8351943 (200)   total: 13.2s    remaining: 5m 14s
250:    test: 0.8360911 best: 0.8360911 (250)   total: 16.4s    remaining: 5m 11s
300:    test: 0.8368408 best: 0.8368408 (300)   total: 19.5s    remaining: 5m 3s
350:    test: 0.8373601 best: 0.8373601 (350)   total: 22.6s    remaining: 4m 58s
400:    test: 0.8377086 best: 0.8377154 (399)   total: 26.2s    remaining: 5m
450:    test: 0.8379165 best: 0.8379165 (450)   total: 29.6s    remaining: 4m 58s
500:    test: 0.8382834 best: 0.8382894 (499)   total: 33.1s    remaining: 4m 57s
550:    test: 0.8386169 best: 0.8386169 (550)   total: 36.8s    remaining: 4m 57s
600:    test: 0.8390101 best: 0.8390142 (598)   total: 41s      remaining: 4m 59s
650:    test: 0.8391172 best: 0.8391574 (639)   total: 45.4s    remaining: 5m 3s
700:    test: 0.8390479 best: 0.8391574 (639)   total: 49.1s    remaining: 5m 1s
750:    test: 0.8391650 best: 0.8391966 (747)   total: 53.3s    remaining: 5m 1s
800:    test: 0.8390124 best: 0.8391966 (747)   total: 57.7s    remaining: 5m 2s
850:    test: 0.8391910 best: 0.8391966 (747)   total: 1m 2s    remaining: 5m 3s
900:    test: 0.8391540 best: 0.8392702 (857)   total: 1m 6s    remaining: 5m 1s
950:    test: 0.8391052 best: 0.8392702 (857)   total: 1m 9s    remaining: 4m 56s
1000:   test: 0.8393361 best: 0.8393361 (1000)  total: 1m 13s   remaining: 4m 54s
1050:   test: 0.8392769 best: 0.8393826 (1025)  total: 1m 17s   remaining: 4m 51s
1100:   test: 0.8393224 best: 0.8394163 (1087)  total: 1m 21s   remaining: 4m 47s
```

In [27]:

```
boost = cat_.CatBoostClassifier(n_estimators=10000, max_depth=6, eval_metric='Logloss', reg_lambda = 370

c_= func(train, ylabel, test, catboost, 'catboost', 1000)
_p2, test_p2, model2 = func_()
```

```
100:    learn: 0.4145674        test: 0.4266873 best: 0.4266873 (100)   total: 3.35s    remaining: 5m 26s
150:    learn: 0.4084435        test: 0.4222000 best: 0.4222000 (150)   total: 4.88s    remaining: 5m 18s
200:    learn: 0.4052562        test: 0.4201575 best: 0.4201575 (200)   total: 6.57s    remaining: 5m 20s
250:    learn: 0.4028537        test: 0.4187157 best: 0.4187157 (250)   total: 8s       remaining: 5m 10s
300:    learn: 0.4007668        test: 0.4176123 best: 0.4176123 (300)   total: 9.62s    remaining: 5m 10s
350:    learn: 0.3989869        test: 0.4167682 best: 0.4167682 (350)   total: 11.2s    remaining: 5m 7s
400:    learn: 0.3974400        test: 0.4161748 best: 0.4161748 (400)   total: 12.9s    remaining: 5m 10s
450:    learn: 0.3960274        test: 0.4157666 best: 0.4157665 (449)   total: 14.5s    remaining: 5m 6s
500:    learn: 0.3945813        test: 0.4153171 best: 0.4153171 (500)   total: 15.8s    remaining: 4m 59s
550:    learn: 0.3938023        test: 0.4150819 best: 0.4150819 (550)   total: 17.5s    remaining: 5m
600:    learn: 0.3927903        test: 0.4148587 best: 0.4148587 (600)   total: 18.9s    remaining: 4m 55s
650:    learn: 0.3917615        test: 0.4147261 best: 0.4147201 (646)   total: 20.5s    remaining: 4m 54s
700:    learn: 0.3906029        test: 0.4145424 best: 0.4145421 (697)   total: 21.8s    remaining: 4m 49s
750:    learn: 0.3895537        test: 0.4143875 best: 0.4143875 (750)   total: 23.6s    remaining: 4m 50s
800:    learn: 0.3885455        test: 0.4141987 best: 0.4141737 (797)   total: 25.2s    remaining: 4m 49s
850:    learn: 0.3877537        test: 0.4141036 best: 0.4141036 (850)   total: 26.8s    remaining: 4m 48s
900:    learn: 0.3868301        test: 0.4140542 best: 0.4140358 (882)   total: 28.3s    remaining: 4m 45s
950:    learn: 0.3860724        test: 0.4140095 best: 0.4140095 (950)   total: 29.9s    remaining: 4m 44s
1000:   learn: 0.3852650        test: 0.4139585 best: 0.4139479 (993)   total: 31.5s    remaining: 4m 43s
1050:   learn: 0.3845052        test: 0.4138916 best: 0.4138916 (1050)  total: 33.2s    remaining: 4m 42s
1100:   learn: 0.3837575        test: 0.4138273 best: 0.4138273 (1099)  total: 34.7s    remaining: 4m 40s
1150:   learn: 0.3831756        test: 0.4137638 best: 0.4137612 (1148)  total: 36.3s    remaining: 4m 39s
1200:   learn: 0.3826108        test: 0.4137437 best: 0.4137138 (1189)  total: 37.8s    remaining: 4m 36s
1250:   learn: 0.3820732        test: 0.4137220 best: 0.4137138 (1189)  total: 39.4s    remaining: 4m 35s
1300:   learn: 0.3816694        test: 0.4136823 best: 0.4136757 (1294)  total: 40.9s    remaining: 4m 33s
```

In [28]:
```python
xgboost = xgb.XGBClassifier(objective ='binary:logistic',
                            eta = 0.99,
                            max_depth = 6,
                            n_estimators = 5000,
                            reg_lambda = 500,
                            sub_sample = 0.8,
                            colsample_bytree = 0.8)

func_= func(train, ylabel, test, xgboost, 'xgboost', 1000)
val_p3, test_p3, model3 = func_()
```

```
================================Fold1===================================
[10:10:42] WARNING: C:\Users\Administrator\workspace\xgboost-win64_release_1.1.0\src\learner.cc:480:
Parameters: { sub_sample } might not be used.

  This may not be accurate due to some parameters are only used in language bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through this
  verification. Please open an issue if you find above cases.


[0]	validation_0-auc:0.73545
Will train until validation_0-auc hasn't improved in 30 rounds.
[1]	validation_0-auc:0.79772
[2]	validation_0-auc:0.80689
[3]	validation_0-auc:0.81549
[4]	validation_0-auc:0.81723
[5]	validation_0-auc:0.82027
[6]	validation_0-auc:0.82079
[7]	validation_0-auc:0.82206
[8]	validation_0-auc:0.82277
[9]	validation_0-auc:0.82371
[10]	validation_0-auc:0.82439
[11]	validation_0-auc:0.82508
[12]	validation_0-auc:0.82569
[13]	validation_0-auc:0.82650
```

In [29]:
```python
xgboost = xgb.XGBClassifier(objective="binary:logistic",
    learning_rate=0.05,
    seed=9616,
    max_depth=25,
    gamma=10,
    n_estimators=500)

func_= func(train, ylabel, test, xgboost, 'xgboost', 600)
val_p4, test_p4, model4 = func_()
```

```
[76]	validation_0-auc:0.82702
[77]	validation_0-auc:0.82705
[78]	validation_0-auc:0.82717
[79]	validation_0-auc:0.82724
[80]	validation_0-auc:0.82735
[81]	validation_0-auc:0.82738
[82]	validation_0-auc:0.82743
[83]	validation_0-auc:0.82750
[84]	validation_0-auc:0.82753
[85]	validation_0-auc:0.82756
[86]	validation_0-auc:0.82762
[87]	validation_0-auc:0.82772
[88]	validation_0-auc:0.82774
[89]	validation_0-auc:0.82774
[90]	validation_0-auc:0.82776
[91]	validation_0-auc:0.82776
[92]	validation_0-auc:0.82783
[93]	validation_0-auc:0.82786
[94]	validation_0-auc:0.82790
[95]	validation_0-auc:0.82796
[96]	validation_0-auc:0.82802
[97]	validation_0-auc:0.82807
[98]	validation_0-auc:0.82804
[99]	validation_0-auc:0.82814
[100]	validation_0-auc:0.82820
```

In [30]:
```python
import lightgbm as lgb
lgb_model = lgb.LGBMClassifier(objective =  'binary',
                               metric= 'auc',
                               boosting_type= 'gbdt',
                               lambda_l1= 0.0004912993970392775,
                               lambda_l2= 9.424350138808432,
                               num_leaves= 25,
                               feature_fraction= 1.0,
                               bagging_fraction= 0.9540416539312312,
                               bagging_freq= 7,
                               min_child_samples= 100, n_estimators = 300)

func_= func(train, ylabel, test, lgb_model, 'lgbm', 1000)
val_p5, test_p5, model5 = func_()
```

```
================================Fold1====================================
[1]	valid_0's auc: 0.798176
Training until validation scores don't improve for 30 rounds
[2]	valid_0's auc: 0.803149
[3]	valid_0's auc: 0.806508
[4]	valid_0's auc: 0.811159
[5]	valid_0's auc: 0.813934
[6]	valid_0's auc: 0.816593
[7]	valid_0's auc: 0.817332
[8]	valid_0's auc: 0.818491
[9]	valid_0's auc: 0.820709
[10]	valid_0's auc: 0.821381
[11]	valid_0's auc: 0.821702
[12]	valid_0's auc: 0.822839
[13]	valid_0's auc: 0.823591
[14]	valid_0's auc: 0.824699
[15]	valid_0's auc: 0.825143
[16]	valid_0's auc: 0.825585
[17]	valid_0's auc: 0.826202
[18]	valid_0's auc: 0.826797
[19]	valid_0's auc: 0.82737
[20]	valid_0's auc: 0.827781
[21]	valid_0's auc: 0.828295
[22]	valid_0's auc: 0.828649
```

In [31]:
```
import lightgbm as lgb
lgb_model = lgb.LGBMClassifier(num_leaves= 100,
         min_data_in_leaf= 60,
        objective =  'binary',
         max_depth= -1,
         learning_rate= 0.05,
         boosting= "gbdt",
         feature_fraction= 0.35,
         lambda_l1= 1,
         lambda_l2= 1,
         verbosity= -1,
         metric= 'auc',
        num_iterations= 2200,
        min_child_samples= 100,
        n_estimators = 300)

func_= func(train, ylabel, test, lgb_model, 'lgbm', 600)
val_p6, test_p6, model6 = func_()
```

```
===============================Fold1===================================
[1]	valid_0's auc: 0.790597
Training until validation scores don't improve for 30 rounds
[2]	valid_0's auc: 0.809651
[3]	valid_0's auc: 0.811826
[4]	valid_0's auc: 0.814203
[5]	valid_0's auc: 0.813813
[6]	valid_0's auc: 0.815074
[7]	valid_0's auc: 0.815722
[8]	valid_0's auc: 0.817021
[9]	valid_0's auc: 0.816736
[10]	valid_0's auc: 0.819553
[11]	valid_0's auc: 0.819464
[12]	valid_0's auc: 0.819206
[13]	valid_0's auc: 0.818977
[14]	valid_0's auc: 0.819096
[15]	valid_0's auc: 0.821431
[16]	valid_0's auc: 0.821957
[17]	valid_0's auc: 0.82353
[18]	valid_0's auc: 0.823499
[19]	valid_0's auc: 0.823584
[20]	valid_0's auc: 0.823795
[21]	valid_0's auc: 0.825268
[22]	valid_0's auc: 0.825016
```

In [32]:
```
from sklearn.linear_model import  LinearRegression, Ridge, Lasso
stack = np.column_stack((val_p1, val_p2, val_p3, val_p4, val_p5, val_p6))
stack_p = np.column_stack((test_p1, test_p2, test_p3, test_p4, test_p5, test_p6))
predict = LinearRegression().fit(stack, ylabel).predict(stack_p)
```

In [33]:
```
sample['default_status'] = predict

sample.to_csv('solution_stack4.csv',index=False)
```

In [ ]:
```

```