In [1]:

```
#import necessary libraraies
import numpy as np  #linear algebra
import pandas as pd  #data processing
import matplotlib.pyplot as plt #plots
%matplotlib inline
import seaborn as sns
```

In [3]:

```
#read in the data
tr = pd.read_csv('Train (1).csv')
te = pd.read_csv('Test (1).csv')
ss = pd.read_csv('SampleSubmission.csv')

#make a copy of the data
train = tr.copy()
test = te.copy()
```

In [5]:

```
!pip install catboost
```

```
Collecting catboost
  Downloading https://files.pythonhosted.org/packages/90/86/c3dcb600b4f9e7584ed
90ea9d30a717fb5c0111574675f442c3e7bc19535/catboost-0.24.1-cp36-none-manylinux1_
x86_64.whl (https://files.pythonhosted.org/packages/90/86/c3dcb600b4f9e7584ed90
ea9d30a717fb5c0111574675f442c3e7bc19535/catboost-0.24.1-cp36-none-manylinux1_x8
6_64.whl) (66.1MB)
     |████████████████████████████████| 66.1MB 84kB/s
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages
 (from catboost) (1.4.1)
Requirement already satisfied: plotly in /usr/local/lib/python3.6/dist-packages
(from catboost) (4.4.1)
Requirement already satisfied: pandas>=0.24.0 in /usr/local/lib/python3.6/dist-
packages (from catboost) (1.1.2)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (f
rom catboost) (1.15.0)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.6/dist-p
ackages (from catboost) (1.18.5)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-pack
ages (from catboost) (3.2.2)
Requirement already satisfied: graphviz in /usr/local/lib/python3.6/dist-packag
es (from catboost) (0.10.1)
Requirement already satisfied: retrying>=1.3.3 in /usr/local/lib/python3.6/dist
-packages (from plotly->catboost) (1.3.3)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python
3.6/dist-packages (from pandas>=0.24.0->catboost) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-pa
ckages (from pandas>=0.24.0->catboost) (2018.9)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-pa
ckages (from matplotlib->catboost) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/di
st-packages (from matplotlib->catboost) (1.2.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /us
r/local/lib/python3.6/dist-packages (from matplotlib->catboost) (2.4.7)
Installing collected packages: catboost
Successfully installed catboost-0.24.1
```

In [6]:

```
#display all rows and columns
pd.set_option('display.max_columns',None)
pd.set_option('display.max_rows',None)
```

EDA

In [ ]:

```
train.head() #veiw the datframe
```

Out[6]:

	Applicant_ID	form_field1	form_field2	form_field3	form_field4	form_field5	form_field6	form_f
0	Apcnt_1000000	3436.0	0.28505	1.6560	0.0	0.000	0.0	10689
1	Apcnt_1000004	3456.0	0.67400	0.2342	0.0	0.000	0.0	898
2	Apcnt_1000008	3276.0	0.53845	3.1510	0.0	6.282	NaN	956
3	Apcnt_1000012	3372.0	0.17005	0.5050	0.0	0.000	192166.0	3044
4	Apcnt_1000016	3370.0	0.77270	1.1010	0.0	0.000	1556.0	214

In [ ]:

```
train.isnull().sum() #check for missing values
```

Out[7]:

```
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
```

In [ ]:
```
train.describe()  #data description of each column
```

Out[8]:
```
       form_field1  form_field2  form_field3  form_field4  form_field5   form_field6   forr
count  53471.000000 52156.000000 55645.000000 55645.000000 55645.000000  4.264000e+04  5.0837
mean    3491.795665     0.550737     1.052225     0.851979     1.956317  6.244479e+05  6.8652
std      188.462426     0.820979     2.147768     3.157692    10.512396  1.433422e+06  1.9127
min     2990.000000     0.000000     0.000000     0.000000     0.000000  0.000000e+00  0.0000
25%     3358.000000     0.070788     0.000000     0.000000     0.000000  1.400400e+04  6.8697
50%     3484.000000     0.267575     0.062000     0.000000     0.000000  1.155330e+05  2.7043
75%     3620.000000     0.719512     1.282000     0.000000     0.000000  5.259280e+05  6.9938
max     3900.000000    18.015050    57.371600    91.672200   407.748600  5.313546e+07  2.1587
```

In [7]:
```
#encode categorical columns
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
train['form_field47'] = le.fit_transform(train.form_field47)
test['form_field47'] = le.fit_transform(test.form_field47)
train['default_status'] = le.fit_transform(train.default_status)
```

In [8]:
```
features_with_integer_values = ['form_field16','form_field17','form_field18',
                                'form_field19','form_field20','form_field34','fo
                                'form_field36','form_field37','form_field38','fo

features_recorded_in_days = ['form_field26','form_field27','form_field28','form_

features_recorded_in_months = ['form_field32']

categorical_features = ['form_field47']

#All other features are of float type
```

In [9]:
```
mean_fill = features_with_integer_values + features_recorded_in_days + features_
#mean_fill
```

In [10]:
```
#fill the missing values for these columns with their mean
for i in mean_fill:
    train[i].fillna(train[i].mean(), inplace = True)
    test[i].fillna(test[i].mean(), inplace = True)
```

In [11]:
```
cont_feat = ['form_field1','form_field2','form_field3','form_field4','form_field
            'form_field9','form_field10','form_field11','form_field12','form_fie
             'form_field22','form_field23','form_field24','form_field25','form_f
            'form_field42','form_field43','form_field44','form_field48','form_fi
```

In [12]:
```
#fill the missing values for these columns with their -1
for i in cont_feat:
    train[i].fillna(-1, inplace = True)
    test[i].fillna(-1, inplace = True)
```

In [13]:
```
from sklearn.preprocessing import StandardScaler
```

In [14]:
```
#drop unwanted column
train.drop('Applicant_ID', axis=1, inplace= True)
test.drop('Applicant_ID', axis=1, inplace= True)
```

In [15]:
```
#instantiate Standard scaler
sc = StandardScaler()

#scale the data... scaling is done to center the data distribution around zero a
scaled_train = pd.DataFrame(sc.fit_transform(train), columns = train.columns)
scaled_test = pd.DataFrame(sc.fit_transform(test), columns = test.columns)
```

In [16]:
```
#divide the data to features and label
X = train.drop('default_status', axis = 1)
y = train['default_status']
```

MODELLING

In [17]:
```
#import models
from sklearn.model_selection import StratifiedKFold, RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier   # algorithm for training a
from xgboost import XGBClassifier                       # extreme boosting algor
from lightgbm import LGBMClassifier                               # extreme boos
from catboost import CatBoostClassifier
from sklearn.metrics import roc_curve, auc, roc_auc_score
```

In [22]:
```
# instantiate the models
rfc = RandomForestClassifier(n_estimators=400, random_state=42, n_jobs= -1)
xgb = XGBClassifier(n_estimators=300, random_state=42, scale_pos_weight=3.08, mi
lgb = LGBMClassifier(reg_lambda = 2, random_state = 42, n_estimators = 4000)
cb = CatBoostClassifier(n_estimators=4000,eval_metric='AUC',learning_rate=0.1, m
                       bootstrap_type='Bayesian', use_best_model=True,od_wait=50
```

In [19]:
```
#stratifiedKFold parameters
n=10
skf = StratifiedKFold(n_splits = n, shuffle = True, random_state=42)
```

CATBOOST MODEL

In [23]:

```
scores_cb = []
pred_test1 = 0
for train_index, test_index in skf.split(X,y):
    print('train index: ', train_index, '\n')
    print('test index: ', test_index, '\n')
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
    cb.fit(X_train, y_train,eval_set=[(X_train, y_train),(X_test, y_test)],verbos
    pred = cb.predict_proba(X_test)[:,1]
    print('score: ', roc_auc_score(y_test, pred))
    scores_cb.append(roc_auc_score(y_test, pred))
    test_pred1 = cb.predict_proba(test)[:,1]
    pred_test1 += test_pred1
```

```
train index:  [    0     1     2 ... 55996 55997 55999]

test index:  [   11    56    58 ... 55983 55984 55998]

0:      test: 0.7835423 test1: 0.7990652        best: 0.7990652 (0)     tota
l: 33.6ms       remaining: 2m 14s
100:    test: 0.8390957 test1: 0.8502515        best: 0.8502515 (100)   tota
l: 3.41s        remaining: 2m 11s
200:    test: 0.8461846 test1: 0.8529336        best: 0.8530022 (187)   tota
l: 6.76s        remaining: 2m 7s
300:    test: 0.8512596 test1: 0.8532851        best: 0.8533724 (241)   tota
l: 10.1s        remaining: 2m 4s
400:    test: 0.8559323 test1: 0.8538175        best: 0.8538424 (398)   tota
l: 13.5s        remaining: 2m 1s
500:    test: 0.8598518 test1: 0.8537553        best: 0.8538646 (402)   tota
l: 16.9s        remaining: 1m 57s
Stopped by overfitting detector  (120 iterations wait)

bestTest = 0.8538646099
```

In [24]:

```
np.mean(scores_cb)  #mean prediction
```

Out[24]:

```
0.8403533513267734
```

In [29]:

```
pred_test1
```

Out[29]:

```
array([3.16176296, 3.53000451, 3.5517105 , ..., 2.66059136, 4.92398845,
       2.1569985 ])
```

In [30]:

```
scores_cb
```

Out[30]:

```
[0.8538646099429127,
 0.8415649369027963,
 0.8243040352648234,
 0.8356135322792037,
 0.8379186537883418,
 0.8396292004435237,
 0.8398508071967806,
 0.8524512758205053,
 0.8346842926926143,
 0.8436521689362324]
```

In [25]:

```
#submission
final_pred1 = pred_test1/n
ss['default_status'] = final_pred1
ss.to_csv('cb2_blend_submission.csv', index = False)
```

XGBoost MODEL

In [26]:

```
scores_xgb = []
pred_test2 = 0
for train_index, test_index in skf.split(X,y):
    print('train index: ', train_index, '\n')
    print('test index: ', test_index, '\n')
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
    xgb.fit(X_train, y_train,eval_set=[(X_train, y_train),(X_test, y_test)],verb
    pred = xgb.predict_proba(X_test)[:,1]
    print('score: ', roc_auc_score(y_test, pred))
    scores_xgb.append(roc_auc_score(y_test, pred))
    test_pred2 = xgb.predict_proba(test)[:,1]
    pred_test2 += test_pred2
```

```
train index:  [    0     1     2 ... 55996 55997 55999]

test index:  [   11    56    58 ... 55983 55984 55998]

[0]     validation_0-error:0.317837     validation_1-error:0.300714
Multiple eval metrics have been passed: 'validation_1-error' will be used for
early stopping.

Will train until validation_1-error hasn't improved in 120 rounds.
[100]   validation_0-error:0.260099     validation_1-error:0.254286
[200]   validation_0-error:0.247619     validation_1-error:0.245179
[299]   validation_0-error:0.238651     validation_1-error:0.244107
score:  0.8507847330414031
train index:  [    1     2     4 ... 55997 55998 55999]

test index:  [    0     3    10 ... 55981 55986 55993]

[0]     validation_0-error:0.315218     validation_1-error:0.315357
Multiple eval metrics have been passed: 'validation_1-error' will be used for
```

In [28]:

```
np.mean(scores_xgb)
```

Out[28]:

```
0.8380533211359753
```

In [31]:

```
final_pred2 = pred_test2/n
ss['default_status'] = final_pred2
ss.to_csv('xgb2_blend_submission.csv', index = False)
```

LIGHT GRADIENT BOOSTING MODEL

In [32]:

```python
scores_lgb = []
pred_test3 = 0
for train_index, test_index in skf.split(X,y):
    print('train index: ', train_index, '\n')
    print('test index: ', test_index, '\n')
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
    lgb.fit(X_train, y_train,eval_set=[(X_train, y_train),(X_test, y_test)],verb
    pred = lgb.predict_proba(X_test)[:,1]
    print('score: ', roc_auc_score(y_test, pred))
    scores_lgb.append(roc_auc_score(y_test, pred))
    test_pred3 = lgb.predict_proba(test)[:,1]
    pred_test3 += test_pred3
```

```
train index:  [    0     1     2 ... 55996 55997 55999]

test index:  [   11    56    58 ... 55983 55984 55998]

Training until validation scores don't improve for 120 rounds.
[100]   training's binary_logloss: 0.365324     valid_1's binary_logloss: 0.394
553
[200]   training's binary_logloss: 0.332051     valid_1's binary_logloss: 0.396
52
Early stopping, best iteration is:
[106]   training's binary_logloss: 0.362855     valid_1's binary_logloss: 0.394
477
score:  0.8517341188511337
train index:  [    1     2     4 ... 55997 55998 55999]

test index:  [    0     3    10 ... 55981 55986 55993]

Training until validation scores don't improve for 120 rounds.
[100]   training's binary_logloss: 0.364357     valid_1's binary_logloss: 0.407
27
[200]   training's binary_logloss: 0.331725     valid_1's binary_logloss: 0.407
499
Early stopping, best iteration is:
[139]   training's binary_logloss: 0.350836     valid_1's binary_logloss: 0.406
752
score:  0.8386280413504132
train index:  [    0     1     2 ... 55997 55998 55999]

test index:  [    4    21    25 ... 55956 55965 55995]

Training until validation scores don't improve for 120 rounds.
[100]   training's binary_logloss: 0.361496     valid_1's binary_logloss: 0.425
859
[200]   training's binary_logloss: 0.328885     valid_1's binary_logloss: 0.426
724
Early stopping, best iteration is:
[117]   training's binary_logloss: 0.355089     valid_1's binary_logloss: 0.425
334
score:  0.820926122451021
train index:  [    0     1     2 ... 55996 55997 55998]

test index:  [    7    17    18 ... 55987 55992 55999]
```

```
Training until validation scores don't improve for 120 rounds.
[100]   training's binary_logloss: 0.363579     valid_1's binary_logloss: 0.411
121
[200]   training's binary_logloss: 0.330277     valid_1's binary_logloss: 0.411
887
Early stopping, best iteration is:
[110]   training's binary_logloss: 0.359771     valid_1's binary_logloss: 0.410
869
score:  0.8356321595237222
train index:  [    0     1     2 ... 55997 55998 55999]

test index:  [    9    27    28 ... 55969 55977 55978]

Training until validation scores don't improve for 120 rounds.
[100]   training's binary_logloss: 0.363432     valid_1's binary_logloss: 0.408
97
[200]   training's binary_logloss: 0.330487     valid_1's binary_logloss: 0.409
121
Early stopping, best iteration is:
[128]   training's binary_logloss: 0.353501     valid_1's binary_logloss: 0.408
32
score:  0.8385650881629207
train index:  [    0     1     2 ... 55997 55998 55999]

test index:  [    5    46    57 ... 55950 55970 55988]

Training until validation scores don't improve for 120 rounds.
[100]   training's binary_logloss: 0.363641     valid_1's binary_logloss: 0.408
585
[200]   training's binary_logloss: 0.331443     valid_1's binary_logloss: 0.409
906
Early stopping, best iteration is:
[113]   training's binary_logloss: 0.358962     valid_1's binary_logloss: 0.408
215
score:  0.8363912077197415
train index:  [    0     1     2 ... 55997 55998 55999]

test index:  [   14    24    32 ... 55973 55974 55976]

Training until validation scores don't improve for 120 rounds.
[100]   training's binary_logloss: 0.362926     valid_1's binary_logloss: 0.409
949
Early stopping, best iteration is:
[67]    training's binary_logloss: 0.376759     valid_1's binary_logloss: 0.409
462
score:  0.8375198420360171
train index:  [    0     1     3 ... 55997 55998 55999]

test index:  [    2     6    12 ... 55958 55989 55991]

Training until validation scores don't improve for 120 rounds.
[100]   training's binary_logloss: 0.364911     valid_1's binary_logloss: 0.397
628
[200]   training's binary_logloss: 0.331497     valid_1's binary_logloss: 0.398
31
Early stopping, best iteration is:
```

```
[133]	training's binary_logloss: 0.352612	valid_1's binary_logloss: 0.397
454
score:  0.8484394264531059
train index:  [    0     2     3 ... 55997 55998 55999]

test index:  [    1    13    23 ... 55962 55980 55990]

Training until validation scores don't improve for 120 rounds.
[100]	training's binary_logloss: 0.362725	valid_1's binary_logloss: 0.412
123
[200]	training's binary_logloss: 0.3308	valid_1's binary_logloss: 0.413
291
Early stopping, best iteration is:
[101]	training's binary_logloss: 0.362326	valid_1's binary_logloss: 0.412
004
score:  0.8344143306734776
train index:  [    0     1     2 ... 55995 55998 55999]

test index:  [    8    15    22 ... 55994 55996 55997]

Training until validation scores don't improve for 120 rounds.
[100]	training's binary_logloss: 0.363784	valid_1's binary_logloss: 0.406
999
Early stopping, best iteration is:
[75]	training's binary_logloss: 0.373766	valid_1's binary_logloss: 0.406
683
score:  0.8396772971250941
```

In [33]:

```
np.mean(scores_lgb)
```

Out[33]:

```
0.8381927634346648
```

In [34]:

```
final_pred3 = pred_test3/n
ss['default_status'] = final_pred1
ss.to_csv('lgb2_blend_submission.csv', index = False)
```

RANDOMFOREST MODEL

In [35]:
```
scores_rfc = []
pred_test4 = 0
for train_index, test_index in skf.split(X,y):
    print('train index: ', train_index, '\n')
    print('test index: ', test_index, '\n')
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
    rfc.fit(X_train, y_train)
    pred = rfc.predict_proba(X_test)[:,1]
    print('score: ', roc_auc_score(y_test, pred))
    scores_rfc.append(roc_auc_score(y_test, pred))
    test_pred4 = rfc.predict_proba(test)[:,1]
    pred_test4 += test_pred4
```

```
train index:  [    0     1     2 ... 55996 55997 55999]

test index:  [   11    56    58 ... 55983 55984 55998]

score:  0.8465616952448267
train index:  [    1     2     4 ... 55997 55998 55999]

test index:  [    0     3    10 ... 55981 55986 55993]

score:  0.8358065312293514
train index:  [    0     1     2 ... 55997 55998 55999]

test index:  [    4    21    25 ... 55956 55965 55995]

score:  0.8198869636711814
train index:  [    0     1     2 ... 55996 55997 55998]

test index:  [    7    17    18 ... 55987 55992 55999]

score:  0.8293548643583026
train index:  [    0     1     2 ... 55997 55998 55999]

test index:  [    9    27    28 ... 55969 55977 55978]

score:  0.8345217687810486
train index:  [    0     1     2 ... 55997 55998 55999]

test index:  [    5    46    57 ... 55950 55970 55988]

score:  0.8346735183463846
train index:  [    0     1     2 ... 55997 55998 55999]

test index:  [   14    24    32 ... 55973 55974 55976]

score:  0.8327792158896264
train index:  [    0     1     3 ... 55997 55998 55999]

test index:  [    2     6    12 ... 55958 55989 55991]

score:  0.8435358921917191
train index:  [    0     2     3 ... 55997 55998 55999]

test index:  [    1    13    23 ... 55962 55980 55990]
```

```
score:  0.8292353179276848
train index:  [    0     1     2 ... 55995 55998 55999]

test index:  [    8    15    22 ... 55994 55996 55997]

score:  0.8341715200068405
```

In [36]:

```
np.mean(scores_rfc)
```

Out[36]:

```
0.8340527287646966
```

In [37]:

```
final_pred4 = pred_test4/n
ss['default_status'] = final_pred1
ss.to_csv('rfc2_blend_submission.csv', index = False)
```

In [38]:

```
#read in the submission files
cb2 = pd.read_csv('/content/cb2_blend_submission.csv')
xbg2 = pd.read_csv('/content/xgb2_blend_submission.csv')
lgb2 = pd.read_csv('/content/lgb2_blend_submission.csv')
rfc2 = pd.read_csv('/content/rfc2_blend_submission.csv')
```

BLEND THE MODELS BASED ON LB PERFORMANCE ¶

In [41]:

```
final_blend2 = ((cb2.default_status * 0.7 + xbg2.default_status * 0.3) +
                (lgb2.default_status * 0.8 + rfc2.default_status * 0.2))
ss['default_status'] = final_blend2
ss.to_csv('final_submission.csv', index = False)
```

In [ ]: