```
# instal catboost library
!pip install catboost
```

```
Collecting catboost
  Downloading https://files.pythonhosted.org/packages/90/86/c3dcb600b4f9e7584ed90ea9c
     |████████████████████████████████| 66.1MB 57kB/s
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: graphviz in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: pandas>=0.24.0 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: plotly in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from ca
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: retrying>=1.3.3 in /usr/local/lib/python3.6/dist-packa
Installing collected packages: catboost
Successfully installed catboost-0.24.1
```

```
# Loading files from drive to colab
from google.colab import files
uploaded = files.upload()
```

```
Choose Files No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Test.csv to Test.csv
Saving Train.csv to Train.csv
```

```
# impoting libraries
import pandas as pd
import numpy as np
from sklearn.metrics import roc_auc_score, log_loss
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import StratifiedKFold
import xgboost as xgb
import catboost as cat_
import seaborn as sns
import lightgbm as lgb
```

```
# for easy training and stacking throughout our workflow. In this class, a variable "val_p
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_round
                               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
                x_train_predict = self.model.predict_proba(X_train, ntree_limit = self.model.g
                self.val_p[test_index] = x_test_predict
                self.test_p += self.model.predict_proba(self.test, ntree_limit = self.model.ge
                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.b
                x_train_predict = self.model.predict_proba(X_train, num_iteration = self.model
                self.val_p[test_index] = x_test_predict
                self.test_p += self.model.predict_proba(self.test, num_iteration = self.model.
                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.l
                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_f
                elif self.model_type == 'xgboost': x_test_predict, x_train_predict = xgboost_f
                elif self.model_type == 'lgbm': x_test_predict, x_train_predict = lgbm_fit(X_t

                print('\nValidation scores', roc_auc_score(y_test, x_test_predict), log_loss(y
                print('Training scores', roc_auc_score(y_train, x_train_predict), log_loss(y_t
                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.
                plt.figure(figsize=(30,25))
                sns.barplot(x="Value", y="Feature", data=feat_imp.sort_values(by="Value", asce
                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


# importing datasets
train = pd.read_csv('Train.csv')
test = pd.read_csv('Test.csv')


train.drop(['Applicant_ID'], 1, inplace = True)
test.drop(['Applicant_ID',], 1, inplace = True)


# replacing the target subset with a logic output
train.default_status.replace({"yes":1,"no":0},inplace=True)


train.head(3)
```

```
# shows the number of rows
ntrain = train.shape[0]
ntest = test.shape[0]


# fixing up the nan values and encoding the categorical values
from sklearn.preprocessing import LabelEncoder
for col in train.columns:
  train[col] = train[col].fillna(-999)
  if train[col].dtype == np.number:
    continue
  train[col] = LabelEncoder().fit_transform(train[col])

from sklearn.preprocessing import LabelEncoder
for col in test.columns:
  test[col] = test[col].fillna(-999)
  if test[col].dtype == np.number:
    continue
  test[col] = LabelEncoder().fit_transform(test[col])



train.head(3)
```

```
target = train["default_status"]
train = train.drop(["default_status"],1)



# I'm using the stacking ensemble technique
# Base model 1 Xgboostclassifier
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, gpu_id=0)

func_= func(train, target, test, xgboost, 'xgboost', 1000)
val_p2, test_p2, model2 = func_()
```

```
================================Fold1====================================
[0]     validation_0-auc:0.759679
Will train until validation_0-auc hasn't improved in 30 rounds.
[1]     validation_0-auc:0.78366
[2]     validation_0-auc:0.785419
[3]     validation_0-auc:0.79478
[4]     validation_0-auc:0.796595
[5]     validation_0-auc:0.799415
[6]     validation_0-auc:0.800883
[7]     validation_0-auc:0.803309
[8]     validation_0-auc:0.804341
[9]     validation_0-auc:0.80512
[10]    validation_0-auc:0.806688
[11]    validation_0-auc:0.808588
[12]    validation_0-auc:0.810452
[13]    validation_0-auc:0.811706
[14]    validation_0-auc:0.812292
[15]    validation_0-auc:0.812807
[16]    validation_0-auc:0.81319
[17]    validation_0-auc:0.81721
[18]    validation_0-auc:0.818944
[19]    validation_0-auc:0.819679
[20]    validation_0-auc:0.820208
[21]    validation_0-auc:0.820556
[22]    validation_0-auc:0.821641
[23]    validation_0-auc:0.821978
[24]    validation_0-auc:0.822185
[25]    validation_0-auc:0.823409
[26]    validation_0-auc:0.824158
[27]    validation_0-auc:0.824847
[28]    validation_0-auc:0.825587
[29]    validation_0-auc:0.825827
[30]    validation_0-auc:0.826344
[31]    validation_0-auc:0.826643
[32]    validation_0-auc:0.826994
[33]    validation_0-auc:0.8272
[34]    validation_0-auc:0.827604
[35]    validation_0-auc:0.827769
[36]    validation_0-auc:0.828041
[37]    validation_0-auc:0.828345
[38]    validation_0-auc:0.828547
[39]    validation_0-auc:0.828717
[40]    validation_0-auc:0.828984
[41]    validation_0-auc:0.829289
[42]    validation_0-auc:0.829659
[43]    validation_0-auc:0.829946
[44]    validation_0-auc:0.830175
[45]    validation_0-auc:0.830271
[46]    validation_0-auc:0.830361
[47]    validation_0-auc:0.830586
[48]    validation_0-auc:0.830678
[49]    validation_0-auc:0.830751
[50]    validation_0-auc:0.830971
[51]    validation_0-auc:0.831083
[52]    validation_0-auc:0.831219
[53]    validation_0-auc:0.831341
[54]    validation_0-auc:0.831495
[55]    validation_0-auc:0.831589
[56]    validation_0-auc:0.8318
[57]    validation_0-auc:0.831929
[58]    validation_0-auc:0.832112
```

```
[59]    validation_0-auc:0.832234
[60]    validation_0-auc:0.832366
[61]    validation_0-auc:0.832432
[62]    validation_0-auc:0.832684
[63]    validation_0-auc:0.832831
[64]    validation_0-auc:0.832898
[65]    validation_0-auc:0.832947
[66]    validation_0-auc:0.833059
[67]    validation_0-auc:0.833156
[68]    validation_0-auc:0.833344
[69]    validation_0-auc:0.833415
[70]    validation_0-auc:0.833535
[71]    validation_0-auc:0.833674
[72]    validation_0-auc:0.833779
[73]    validation_0-auc:0.83381
[74]    validation_0-auc:0.833901
[75]    validation_0-auc:0.83402
[76]    validation_0-auc:0.83417
[77]    validation_0-auc:0.834321
[78]    validation_0-auc:0.834382
[79]    validation_0-auc:0.834469
[80]    validation_0-auc:0.834551
[81]    validation_0-auc:0.834532
[82]    validation_0-auc:0.834585
[83]    validation_0-auc:0.8347
[84]    validation_0-auc:0.834761
[85]    validation_0-auc:0.834811
[86]    validation_0-auc:0.834944
[87]    validation_0-auc:0.834961
[88]    validation_0-auc:0.835009
[89]    validation_0-auc:0.835133
[90]    validation_0-auc:0.835213
[91]    validation_0-auc:0.835332
[92]    validation_0-auc:0.835348
[93]    validation_0-auc:0.835345
[94]    validation_0-auc:0.835421
[95]    validation_0-auc:0.835489
[96]    validation_0-auc:0.83552
[97]    validation_0-auc:0.835583
[98]    validation_0-auc:0.835636
[99]    validation_0-auc:0.835758
[100]   validation_0-auc:0.83583
[101]   validation_0-auc:0.835878
[102]   validation_0-auc:0.83586
[103]   validation_0-auc:0.835914
[104]   validation_0-auc:0.835972
[105]   validation_0-auc:0.836036
[106]   validation_0-auc:0.836036
[107]   validation_0-auc:0.836118
[108]   validation_0-auc:0.836161
[109]   validation_0-auc:0.83622
[110]   validation_0-auc:0.836292
[111]   validation_0-auc:0.836317
[112]   validation_0-auc:0.836373
[113]   validation_0-auc:0.836348
[114]   validation_0-auc:0.836367
[115]   validation_0-auc:0.836441
[116]   validation_0-auc:0.836501
[117]   validation_0-auc:0.836531
[118]   validation_0-auc:0.836552
[119]   validation_0-auc:0.836595
[120]   validation_0-auc:0.836637
```

```
[121]   validation_0-auc:0.836661
[122]   validation_0-auc:0.83669
[123]   validation_0-auc:0.836755
[124]   validation_0-auc:0.836721
[125]   validation_0-auc:0.836747
[126]   validation_0-auc:0.836799
[127]   validation_0-auc:0.836794
[128]   validation_0-auc:0.83685
[129]   validation_0-auc:0.836833
[130]   validation_0-auc:0.8369
[131]   validation_0-auc:0.836977
[132]   validation_0-auc:0.837028
[133]   validation_0-auc:0.837051
[134]   validation_0-auc:0.837057
[135]   validation_0-auc:0.837094
[136]   validation_0-auc:0.837099
[137]   validation_0-auc:0.837114
[138]   validation_0-auc:0.8372
[139]   validation_0-auc:0.837198
[140]   validation_0-auc:0.837156
[141]   validation_0-auc:0.83719
[142]   validation_0-auc:0.837259
[143]   validation_0-auc:0.837291
[144]   validation_0-auc:0.837302
[145]   validation_0-auc:0.837278
[146]   validation_0-auc:0.837265
[147]   validation_0-auc:0.837326
[148]   validation_0-auc:0.837427
[149]   validation_0-auc:0.837508
[150]   validation_0-auc:0.83751
[151]   validation_0-auc:0.837523
[152]   validation_0-auc:0.837539
[153]   validation_0-auc:0.83758
[154]   validation_0-auc:0.837568
[155]   validation_0-auc:0.83758
[156]   validation_0-auc:0.837622
[157]   validation_0-auc:0.837671
[158]   validation_0-auc:0.837698
[159]   validation_0-auc:0.837662
[160]   validation_0-auc:0.837684
[161]   validation_0-auc:0.83767
[162]   validation_0-auc:0.837746
[163]   validation_0-auc:0.837746
[164]   validation_0-auc:0.837818
[165]   validation_0-auc:0.837849
[166]   validation_0-auc:0.837885
[167]   validation_0-auc:0.83788
[168]   validation_0-auc:0.837907
[169]   validation_0-auc:0.83788
[170]   validation_0-auc:0.83789
[171]   validation_0-auc:0.83787
[172]   validation_0-auc:0.837939
[173]   validation_0-auc:0.837936
[174]   validation_0-auc:0.83795
[175]   validation_0-auc:0.837963
[176]   validation_0-auc:0.837959
[177]   validation_0-auc:0.837904
[178]   validation_0-auc:0.837965
[179]   validation_0-auc:0.837977
[180]   validation_0-auc:0.837973
[181]   validation_0-auc:0.837959
```

```
[182]   validation_0-auc:0.837948
[183]   validation_0-auc:0.837954
[184]   validation_0-auc:0.837962
[185]   validation_0-auc:0.837997
[186]   validation_0-auc:0.838001
[187]   validation_0-auc:0.837977
[188]   validation_0-auc:0.837984
[189]   validation_0-auc:0.837994
[190]   validation_0-auc:0.838054
[191]   validation_0-auc:0.838046
[192]   validation_0-auc:0.838025
[193]   validation_0-auc:0.838016
[194]   validation_0-auc:0.83805
[195]   validation_0-auc:0.838074
[196]   validation_0-auc:0.83807
[197]   validation_0-auc:0.838107
[198]   validation_0-auc:0.838143
[199]   validation_0-auc:0.838169
[200]   validation_0-auc:0.838189
[201]   validation_0-auc:0.83819
[202]   validation_0-auc:0.838181
[203]   validation_0-auc:0.83819
[204]   validation_0-auc:0.838225
[205]   validation_0-auc:0.838246
[206]   validation_0-auc:0.838266
[207]   validation_0-auc:0.83828
[208]   validation_0-auc:0.838284
[209]   validation_0-auc:0.838335
[210]   validation_0-auc:0.838339
[211]   validation_0-auc:0.838349
[212]   validation_0-auc:0.838312
[213]   validation_0-auc:0.838257
[214]   validation_0-auc:0.838262
[215]   validation_0-auc:0.838275
[216]   validation_0-auc:0.838311
[217]   validation_0-auc:0.838314
[218]   validation_0-auc:0.838285
[219]   validation_0-auc:0.838275
[220]   validation_0-auc:0.838264
[221]   validation_0-auc:0.838267
[222]   validation_0-auc:0.838252
[223]   validation_0-auc:0.838287
[224]   validation_0-auc:0.83831
[225]   validation_0-auc:0.838315
[226]   validation_0-auc:0.838331
[227]   validation_0-auc:0.838302
[228]   validation_0-auc:0.838286
[229]   validation_0-auc:0.838311
[230]   validation_0-auc:0.838279
[231]   validation_0-auc:0.838312
[232]   validation_0-auc:0.838308
[233]   validation_0-auc:0.838331
[234]   validation_0-auc:0.838343
[235]   validation_0-auc:0.838354
[236]   validation_0-auc:0.83836
[237]   validation_0-auc:0.838351
[238]   validation_0-auc:0.838362
[239]   validation_0-auc:0.838366
[240]   validation_0-auc:0.838371
[241]   validation_0-auc:0.83835
[242]   validation_0-auc:0.838372
[243]   validation_0-auc:0.838413
```