In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import re
import time
import warnings
import sqlite3
from sqlalchemy import create_engine # database connection
import csv
import os
warnings.filterwarnings("ignore")
import datetime as dt
import numpy as np
from nltk.corpus import stopwords
from sklearn.decomposition import TruncatedSVD
from sklearn.preprocessing import normalize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.manifold import TSNE
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics.classification import accuracy_score, log_loss
from sklearn.feature_extraction.text import TfidfVectorizer
from collections import Counter
from scipy.sparse import hstack
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
from sklearn.cross_validation import StratifiedKFold
from collections import Counter, defaultdict
from sklearn.calibration import CalibratedClassifierCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
import math
from sklearn.metrics import normalized_mutual_info_score
from sklearn.ensemble import RandomForestClassifier



from sklearn.model_selection import cross_val_score
from sklearn.linear_model import SGDClassifier
from mlxtend.classifier import StackingClassifier

from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
```

```
C:\Users\ankan\Anaconda3\lib\site-packages\sklearn\cross_validation.py:41: Depreca
tionWarning: This module was deprecated in version 0.18 in favor of the model_sele
ction module into which all the refactored classes and functions are moved. Also n
ote that the interface of the new CV iterators are different from that of this mod
ule. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
C:\Users\ankan\Anaconda3\lib\site-packages\sklearn\ensemble\weight_boosting.py:29:
DeprecationWarning: numpy.core.umath_tests is an internal NumPy module and should
not be imported. It will be removed in a future NumPy release.
  from numpy.core.umath_tests import inner1d
```

In [2]:

```
plt.style.use('fivethirtyeight')
plt.rcParams['figure.figsize'] = [10, 5]
warnings.filterwarnings("ignore", category=FutureWarning)
```

In [3]:

In [4]:

Out[4]:

	Unnamed: 0	id	is_duplicate	cwc_min	cwc_max	csc_min	csc_max	ctc_min	ctc_max	last_word_eq	...	
0	0	0	0	0.999980	0.833319	0.999983	0.999983	0.916659	0.785709	0.0	...	16.
1	1	1	0	0.799984	0.399996	0.749981	0.599988	0.699993	0.466664	0.0	...	-4.
2	2	2	0	0.399992	0.333328	0.399992	0.249997	0.399996	0.285712	0.0	...	8.
3	3	3	0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	...	3
4	4	4	0	0.399992	0.199998	0.999950	0.666644	0.571420	0.307690	0.0	...	-2.

5 rows × 797 columns

In [5]:

```
# remove the first row
data.drop(data.index[0], inplace=True)
y_true = data['is_duplicate']
```

In [6]:

Out[6]:

	cwc_min	cwc_max	csc_min	csc_max	ctc_min	ctc_max	last_word_eq	first_word_eq	abs_len_diff	mean_l
1	0.799984	0.399996	0.749981	0.599988	0.699993	0.466664	0.0	1.0	5.0	1.
2	0.399992	0.333328	0.399992	0.249997	0.399996	0.285712	0.0	1.0	4.0	1.
3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	2.0	1.
4	0.399992	0.199998	0.999950	0.666644	0.571420	0.307690	0.0	1.0	6.0	1
5	0.666656	0.571420	0.888879	0.799992	0.705878	0.705878	1.0	0.0	0.0	1

5 rows × 794 columns

In [7]:

```
# after we read from csv, each entry was read it as a string
# we convert all the features into numaric before we apply any model(why?)
cols = list(data.columns)
for i in cols:
    data[i] = data[i].apply(pd.to_numeric)
```

```
cwc_min
cwc_max
csc_min
csc_max
ctc_min
ctc_max
last_word_eq
first_word_eq
abs_len_diff
mean_len
token_set_ratio
token_sort_ratio
fuzz_ratio
fuzz_partial_ratio
longest_substr_ratio
freq_qid1
freq_qid2
q1len
q2len
```

In [8]:

```
# https://stackoverflow.com/questions/7368789/convert-all-strings-in-a-list-to-int
```

In [9]:

In [10]:

```
print("Number of data points in train data :",X_train.shape)
```

```
Number of data points in train data : (183900, 794)
Number of data points in test data : (45975, 794)
```

In [11]:

```
print("-"*10, "Distribution of output variable in train data", "-"*10)
train_distr = Counter(y_train)
train_len = len(y_train)
print("Class 0: ",int(train_distr[0])/train_len,"Class 1: ", int(train_distr[1])/trair
print("-"*10, "Distribution of output variable in train data", "-"*10)
test_distr = Counter(y_test)
test_len = len(y_test)
```

```
---------- Distribution of output variable in train data ----------
Class 0:  0.6277650897226754 Class 1:  0.37223491027732464
---------- Distribution of output variable in train data ----------
Class 0:  0.37222403480152255 Class 1:  0.37222403480152255
```

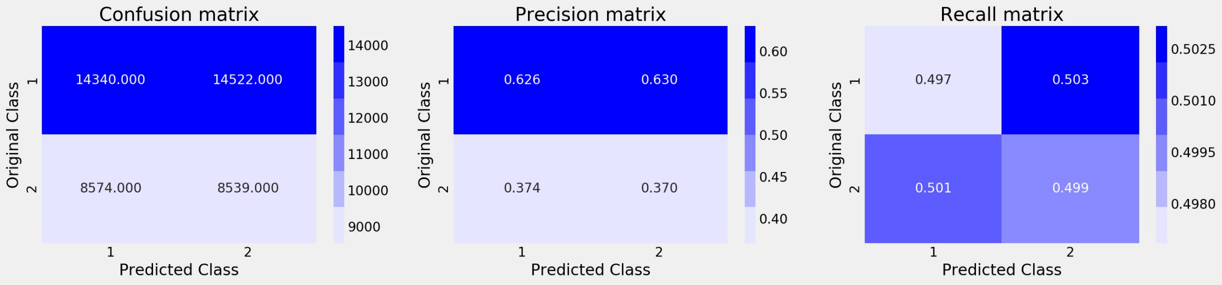
In [12]:

```
# This function plots the confusion matrices given y_i, y_i_hat.
def plot_confusion_matrix(test_y, predict_y):
    C = confusion_matrix(test_y, predict_y)
    # C = 9,9 matrix, each cell (i,j) represents number of points of class i are predi

    A =(((C.T)/(C.sum(axis=1))).T)
    #divid each element of the confusion matrix with the sum of elements in that colun

    # C = [[1, 2],
    #     [3, 4]]
    # C.T = [[1, 3],
    #        [2, 4]]
    # C.sum(axis = 1)  axis=0 corresonds to columns and axis=1 corresponds to rows in
    # C.sum(axix =1) = [[3, 7]]
    # ((C.T)/(C.sum(axis=1))) = [[1/3, 3/7]
    #                            [2/3, 4/7]]

    # ((C.T)/(C.sum(axis=1))).T = [[1/3, 2/3]
    #                            [3/7, 4/7]]
    # sum of row elements = 1

    B =(C/C.sum(axis=0))
    #divid each element of the confusion matrix with the sum of elements in that row
    # C = [[1, 2],
    #     [3, 4]]
    # C.sum(axis = 0)  axis=0 corresonds to columns and axis=1 corresponds to rows in
    # C.sum(axix =0) = [[4, 6]]
    # (C/C.sum(axis=0)) = [[1/4, 2/6],
    #                      [3/4, 4/6]]
    plt.figure(figsize=(20,4))

    labels = [1,2]
    # representing A in heatmap format
    cmap=sns.light_palette("blue")
    plt.subplot(1, 3, 1)
    sns.heatmap(C, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=l
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Confusion matrix")

    plt.subplot(1, 3, 2)
    sns.heatmap(B, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=l
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Precision matrix")

    plt.subplot(1, 3, 3)
    # representing B in heatmap format
    sns.heatmap(A, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=l
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Recall matrix")

    plt.show()
```

Building a random model (Finding worst-case log-loss)

```
In [13]: # we need to generate 9 numbers and the sum of numbers should be 1
         # one solution is to genarate 9 numbers and divide each of the numbers by their sum
         # ref: https://stackoverflow.com/a/18662466/4084039
         # we create a output array that has exactly same size as the CV data
         predicted_y = np.zeros((test_len,2))
         for i in range(test_len):
             rand_probs = np.random.rand(1,2)
             predicted_y[i] = ((rand_probs/sum(sum(rand_probs)))[0])
         print("Log loss on Test Data using Random Model",log_loss(y_test, predicted_y, eps=1e-

         predicted_y =np.argmax(predicted_y, axis=1)
         plot_confusion_matrix(y_test, predicted_y)
```

```
Log loss on Test Data using Random Model 0.8895063692021942
```



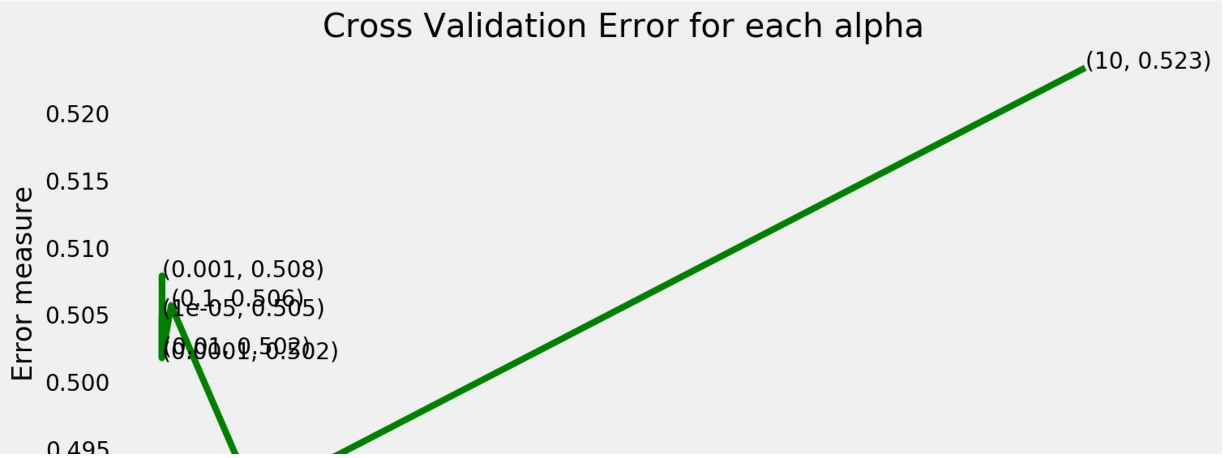
This Random Model gives us a log loss of 0.89

SGD classifier with logLoss

In [14]:

```
%%time
alpha = [10 ** x for x in range(-5, 2)] # hyperparam for SGD classifier.





log_error_array=[]
for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l2', loss='log', random_state=42)
    clf.fit(X_train, y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(X_train, y_train)
    predict_y = sig_clf.predict_proba(X_test)
    log_error_array.append(log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))
    print('For values of alpha = ', i, "The log loss is:",log_loss(y_test, predict_y,

fig, ax = plt.subplots()
ax.plot(alpha, log_error_array,c='g')
for i, txt in enumerate(np.round(log_error_array,3)):
    ax.annotate((alpha[i],np.round(txt,3)), (alpha[i],log_error_array[i]))
plt.grid()
plt.title("Cross Validation Error for each alpha")
plt.xlabel("Alpha i's")
plt.ylabel("Error measure")
plt.show()


best_alpha = np.argmin(log_error_array)
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l2', loss='log', random_state=42
clf.fit(X_train, y_train)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(X_train, y_train)

predict_y = sig_clf.predict_proba(X_train)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",log_l
predict_y = sig_clf.predict_proba(X_test)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",log_lo
predicted_y =np.argmax(predict_y,axis=1)
print("Total number of data points :", len(predicted_y))
plot_confusion_matrix(y_test, predicted_y)
```

```
For values of alpha =  1e-05 The log loss is: 0.5049291300170718
For values of alpha =  0.0001 The log loss is: 0.5017296695059896
For values of alpha =  0.001 The log loss is: 0.5078600577219928
For values of alpha =  0.01 The log loss is: 0.5021225961830948
For values of alpha =  0.1 The log loss is: 0.5056498273171316
For values of alpha =  1 The log loss is: 0.49129107667922123
For values of alpha =  10 The log loss is: 0.5233605636283891
```

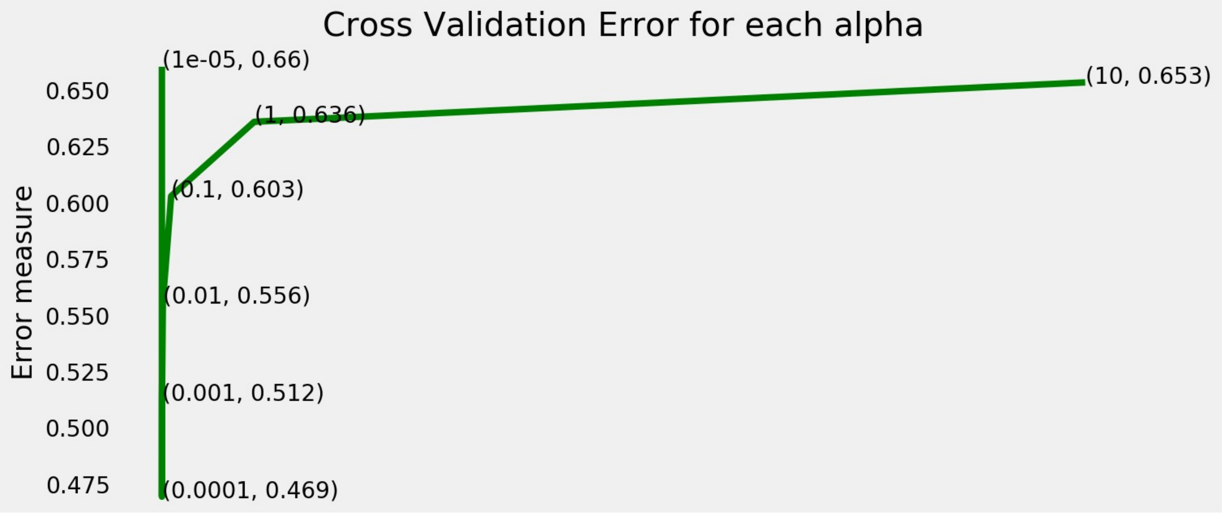


Linear SVM

In [15]:

```
alpha = [10 ** x for x in range(-5, 2)] # hyperparam for SGD classifier.



log_error_array=[]
for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l1', loss='hinge', random_state=42)
    clf.fit(X_train, y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(X_train, y_train)
    predict_y = sig_clf.predict_proba(X_test)
    log_error_array.append(log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15)
    print('For values of alpha = ', i, "The log loss is:",log_loss(y_test, predict_y,

fig, ax = plt.subplots()
ax.plot(alpha, log_error_array,c='g')
for i, txt in enumerate(np.round(log_error_array,3)):
    ax.annotate((alpha[i],np.round(txt,3)), (alpha[i],log_error_array[i]))
plt.grid()
plt.title("Cross Validation Error for each alpha")
plt.xlabel("Alpha i's")
plt.ylabel("Error measure")
plt.show()


best_alpha = np.argmin(log_error_array)
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l1', loss='hinge', random_state=
clf.fit(X_train, y_train)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(X_train, y_train)

predict_y = sig_clf.predict_proba(X_train)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",log_l
predict_y = sig_clf.predict_proba(X_test)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",log_lo
predicted_y =np.argmax(predict_y,axis=1)
print("Total number of data points :", len(predicted_y))
plot_confusion_matrix(y_test, predicted_y)
```

```
For values of alpha =  1e-05 The log loss is: 0.6601287487178833
For values of alpha =  0.0001 The log loss is: 0.46910800351575527
For values of alpha =  0.001 The log loss is: 0.5122461029613686
For values of alpha =  0.01 The log loss is: 0.5555002414784239
For values of alpha =  0.1 The log loss is: 0.6026416468477268
For values of alpha =  1 The log loss is: 0.635668470204849
For values of alpha =  10 The log loss is: 0.6531726339597976
```



In [16]:
```
import xgboost as xgb
from sklearn.model_selection import RandomizedSearchCV
import scipy.stats as st
from datetime import datetime
```

XGBOOST

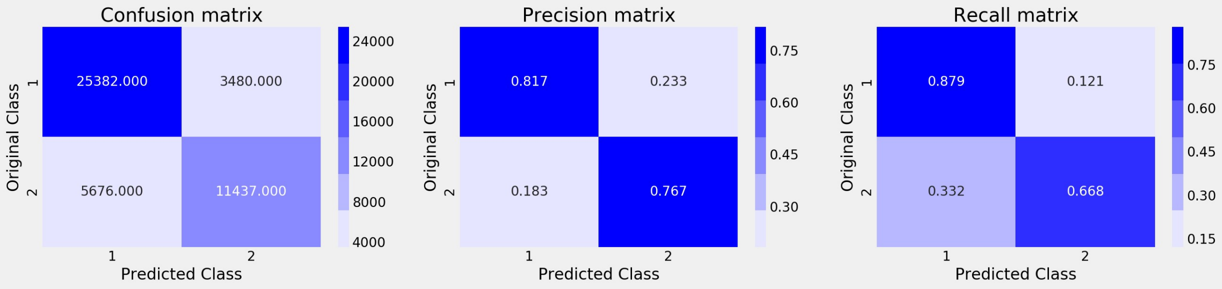
In [17]:
```
start = datetime.now()

params = {}
params['objective'] = 'binary:logistic'
params['eval_metric'] = 'logloss'
params['eta'] = 0.02
params['max_depth'] = 2

d_train = xgb.DMatrix(X_train, label=y_train)
d_test = xgb.DMatrix(X_test, label=y_test)

watchlist = [(d_train, 'train'), (d_test, 'valid')]

bst = xgb.train(params, d_train, 400, watchlist, early_stopping_rounds=20, verbose_eva

xgdmat = xgb.DMatrix(X_train,y_train)
predict_y = bst.predict(d_test)
print("The test log loss is:",log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-


print("Time taken to run this cell :", datetime.now() - start)
```

```
[01:49:35] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 6 extra nodes, 0 pruned nodes, max_depth=2
[0]	train-logloss:0.686771	valid-logloss:0.686806
Multiple eval metrics have been passed: 'valid-logloss' will be used for early sto
pping.

Will train until valid-logloss hasn't improved in 20 rounds.
[01:49:42] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 6 extra nodes, 0 pruned nodes, max_depth=2
[01:49:47] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 6 extra nodes, 0 pruned nodes, max_depth=2
[01:49:52] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 6 extra nodes, 0 pruned nodes, max_depth=2
[01:49:58] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 6 extra nodes, 0 pruned nodes, max_depth=2
[01:50:03] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 6 extra nodes, 0 pruned nodes, max_depth=2
[01:50:08] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 6 extra nodes, 0 pruned nodes, max_depth=2
[01:50:13] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
```

In [18]:

```
predicted_y =np.array(predict_y>0.5,dtype=int)
print("Total number of data points :", len(predicted_y))
```

```
Total number of data points : 45975
```



In [19]:

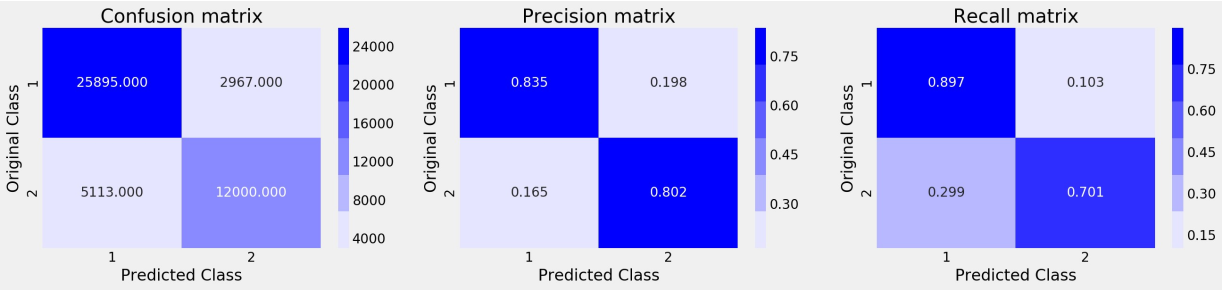
```
start = datetime.now()

params = {}
params['objective'] = 'binary:logistic'
params['eval_metric'] = 'logloss'
params['eta'] = 0.02
params['max_depth'] = 4

d_train = xgb.DMatrix(X_train, label=y_train)
d_test = xgb.DMatrix(X_test, label=y_test)

watchlist = [(d_train, 'train'), (d_test, 'valid')]

bst = xgb.train(params, d_train, 400, watchlist, early_stopping_rounds=20, verbose_eva

xgdmat = xgb.DMatrix(X_train,y_train)
predict_y = bst.predict(d_test)
print("The test log loss is:",log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-
```

```
[02:29:53] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 30 extra nodes, 0 pruned nodes, max_depth=4
[0]     train-logloss:0.684844  valid-logloss:0.684907
Multiple eval metrics have been passed: 'valid-logloss' will be used for early sto
pping.

Will train until valid-logloss hasn't improved in 20 rounds.
[02:30:02] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 30 extra nodes, 0 pruned nodes, max_depth=4
[02:30:11] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 30 extra nodes, 0 pruned nodes, max_depth=4
[02:30:20] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 30 extra nodes, 0 pruned nodes, max_depth=4
[02:30:29] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 30 extra nodes, 0 pruned nodes, max_depth=4
[02:30:38] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 30 extra nodes, 0 pruned nodes, max_depth=4
[02:30:46] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 30 extra nodes, 0 pruned nodes, max_depth=4
```

In [20]:

```
predicted_y =np.array(predict_y>0.5,dtype=int)
print("Total number of data points :", len(predicted_y))
```

```
Total number of data points : 45975
```



In [ ]: