Quora Question Pair Simillarity with TFIDFW2V

September 25, 2018

1 QUORA QUESTION PAIR SIMILLARITY WITH TFIDF WORD2VECTOR

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
from sklearn.metrics import precision_recall_curve, auc, roc_curve
```

```
C:\Users\Aravindh\Anaconda3\lib\site-packages\sklearn\cross_validation.py:41: DeprecationWarni
  "This module will be removed in 0.20.", DeprecationWarning)
```

2 4. Machine Learning Models

2.1 4.1 Reading data from file and storing into sql table

```
In [2]: #Creating db file from csv
        if not os.path.isfile('train.db'):
            disk_engine = create_engine('sqlite:///train.db')
            start = dt.datetime.now()
            chunksize = 180000
            j = 0
            index_start = 1
            for df in pd.read_csv('final_features.csv', names=['Unnamed: 0','id','is_duplicate
                df.index += index_start
                j+=1
                print('{} rows'.format(j*chunksize))
                df.to_sql('data', disk_engine, if_exists='append')
                index_start = df.index[-1] + 1
```

```
In [3]: #http://www.sqlitetutorial.net/sqlite-python/create-tables/
        def create_connection(db_file):
            """ create a database connection to the SQLite database
                specified by db_file
            :param db_file: database file
            :return: Connection object or None
            """
            try:
                conn = sqlite3.connect(db_file)
                return conn
            except Error as e:
                print(e)
```

```python
        return None


    def checkTableExists(dbcon):
        cursr = dbcon.cursor()
        str = "select name from sqlite_master where type='table'"
        table_names = cursr.execute(str)
        print("Tables in the databse:")
        tables =table_names.fetchall()
        print(tables[0][0])
        return(len(tables))
```

In [4]:
```python
read_db = 'train.db'
conn_r = create_connection(read_db)
checkTableExists(conn_r)
conn_r.close()
```

```
Tables in the databse:
data
```

In [5]:
```python
# try to sample data according to the computing power you have
if os.path.isfile(read_db):
    conn_r = create_connection(read_db)
    if conn_r is not None:
        # for selecting first 1M rows
        # data = pd.read_sql_query("""SELECT * FROM data LIMIT 100001;""", conn_r)

        # for selecting random points
        data = pd.read_sql_query("SELECT * From data ORDER BY RANDOM() LIMIT 100001;",
        conn_r.commit()
        conn_r.close()
```

In [6]:
```python
# remove the first row
data.drop(data.index[0], inplace=True)
y_true = data['is_duplicate']
data.drop(['Unnamed: 0', 'id','index','is_duplicate'], axis=1, inplace=True)
```

In [7]:
```python
data.head()
```

Out[7]:
```
              cwc_min             cwc_max             csc_min             csc_max  \
1  0.499987500312492   0.499987500312492   0.857130612419823   0.857130612419823
2                0.0                 0.0                 0.0                 0.0
3  0.999950002499875   0.499987500312492   0.499987500312492   0.333327777870369
4  0.749981250468738   0.749981250468738   0.666655555740738   0.499993750078124
5  0.454541322351615   0.333331111125926   0.374995312558593   0.199998666675556

              ctc_min             ctc_max last_word_eq first_word_eq  \
```

```
1  0.727266115762584  0.533329777801481          0.0           1.0
2                0.0                0.0          0.0           0.0
3  0.666655555740738       0.39999600004          0.0           0.0
4  0.699993000069999  0.583328472262731          1.0           1.0
5  0.380950566902062  0.210525761774311          0.0           0.0

  abs_len_diff mean_len          ...                     374_y  \
1          4.0     13.0          ...          15.2793600080768
2         17.0     15.5          ...         -9.91418486833573
3          4.0      8.0          ...          5.01649652945343
4          2.0     11.0          ...        -0.194829493761063
5         17.0     29.5          ...          26.1319680698216

               375_y              376_y              377_y              378_y  \
1   5.39919739216566   2.13777290284634  -3.06889878213406    3.3754405680229
2  -1.30738198757172  -3.15930802375078  -7.88982777297497  -2.88015016913414
3   3.73020273447037  -2.58491443842649  -3.04634397476912  -3.75857877545059
4    6.9257490132004   1.39664986729622  -4.04720836877823   5.88104222714901
5   35.5613279435784   13.3753447905183   -17.733818192035   36.5733992652968

               379_y               380_y              381_y  \
1    6.7361164689064   -7.80872184038162   13.0617727935314
2  -15.6520266830921   -8.98012767732143   1.77616566419601
3   2.87995103187859   -7.29886836372316    2.5331454873085
4    6.5948448330164  -0.014875266700983   13.9965298771858
5   3.44621949642897   -42.7041535656899  -12.3473131596111

               382_y               383_y
1  -3.52359987795353    -1.1000040769577
2  -11.1839511245489  -0.262479841709137
3   1.54575282335281    1.57878881692886
4  -4.40328110568225     3.5672604739666
5   10.4682890549302    5.42777295783162

[5 rows x 794 columns]
```

2.2 4.2 Converting strings to numerics

In [9]:
```
data = pd.DataFrame(np.array(data.values,dtype=np.float64))
```

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

In [10]:
```
data.to_csv('numeric_data.csv')
```

```
In [11]: # https://stackoverflow.com/questions/7368789/convert-all-strings-in-a-list-to-int
         y_true = list(map(int, y_true.values))
```

2.3 4.3 Random train test split(70:30)

```
In [17]: X_train,X_test, y_train, y_test = train_test_split(data, y_true, stratify=y_true, tes
```

```
In [18]: print("Number of data points in train data :",X_train.shape)
         print("Number of data points in test data :",X_test.shape)
```

```
Number of data points in train data : (70000, 794)
Number of data points in test data : (30000, 794)
```

```
In [19]: 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])/trai
         print("-"*10, "Distribution of output variable in train data", "-"*10)
         test_distr = Counter(y_test)
         test_len = len(y_test)
         print("Class 0: ",int(test_distr[1])/test_len, "Class 1: ",int(test_distr[1])/test_le
```

```
---------- Distribution of output variable in train data ----------
Class 0:  0.6322428571428571 Class 1:  0.36775714285714284
---------- Distribution of output variable in train data ----------
Class 0:  0.3677666666666667 Class 1:  0.3677666666666667
```

```
In [20]: # 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 pred

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

             # 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
```

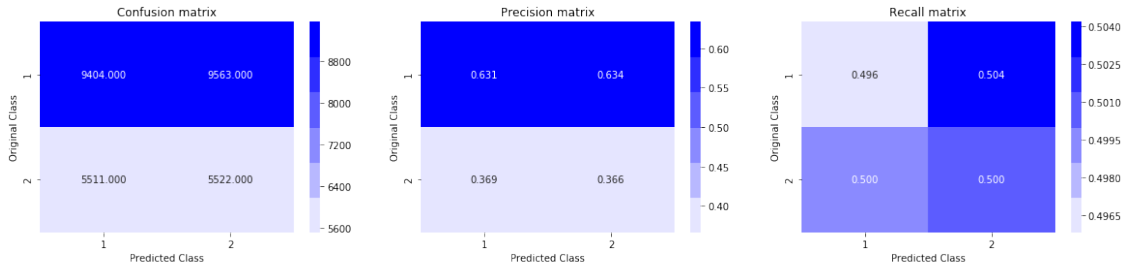
```
B =(C/C.sum(axis=0))
#divid each element of the confusion matrix with the sum of elements in that row
# C = [[1, 2],
#      [3, 4]]
# C.sum(axis = 0)  axis=0 corresonds to columns and axis=1 corresponds to rows in
# C.sum(axix =0) = [[4, 6]]
# (C/C.sum(axis=0)) = [[1/4, 2/6],
#                  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

2.4 4.4 Building a random model (Finding worst-case log-loss)

In [21]:
```
# 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 0.8877671212450066
```

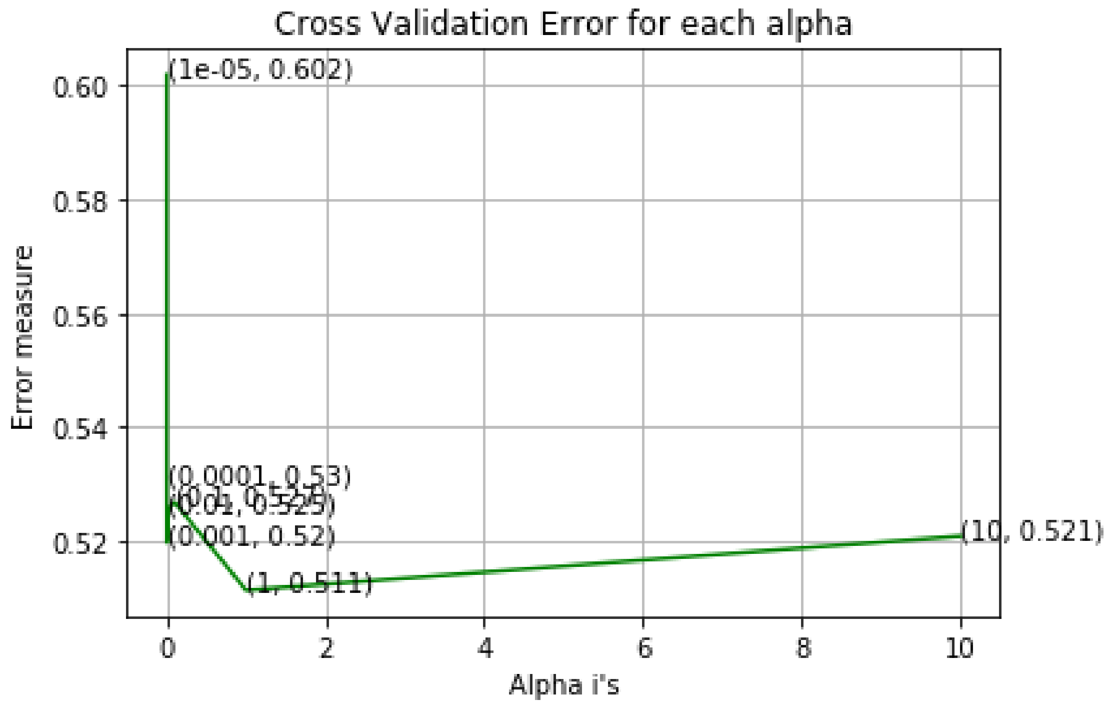


2.5 4.5 Logistic Regression with hyperparameter tuning

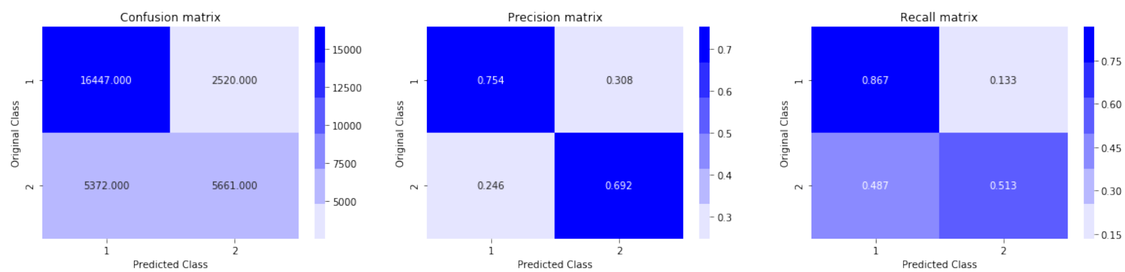
```
In [22]: alpha = [10 ** x for x in range(-5, 2)] # hyperparam for SGD classifier.

         # read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/
         # ------------------------------
         # default parameters
         # SGDClassifier(loss=hinge, penalty=l2, alpha=0.0001, l1_ratio=0.15, fit_intercept=Tr
         # shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate=op
         # class_weight=None, warm_start=False, average=False, n_iter=None)

         # some of methods
         # fit(X, y[, coef_init, intercept_init, ])        Fit linear model with Stochastic Gr
         # predict(X)        Predict class labels for samples in X.

         #-------------------------------
         # video link:
         #------------------------------


         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')
```

```python
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=4
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: 0.6018100038370673
For values of alpha =  0.0001 The log loss is: 0.530475304917484
For values of alpha =  0.001 The log loss is: 0.5198022981153775
For values of alpha =  0.01 The log loss is: 0.5253565762364582
For values of alpha =  0.1 The log loss is: 0.5268528680744162
For values of alpha =  1 The log loss is: 0.511444013388206
For values of alpha =  10 The log loss is: 0.5209142160855245
```




```
For values of best alpha =  1 The train log loss is: 0.5094050776341431
For values of best alpha =  1 The test log loss is: 0.511444013388206
Total number of data points : 30000
```



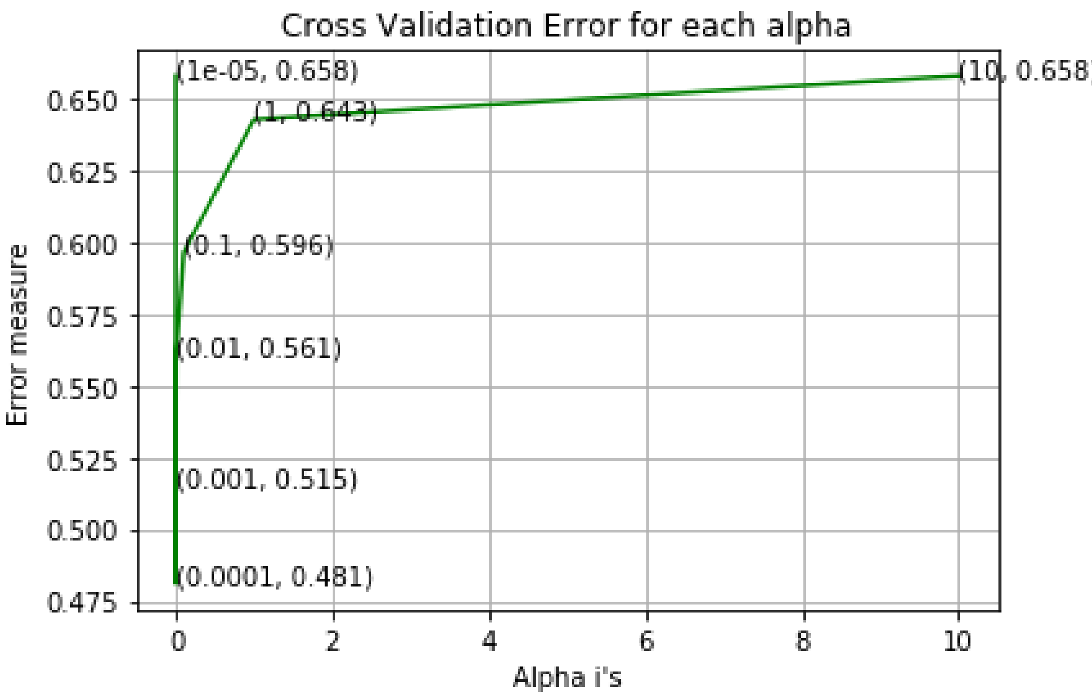
2.6 4.6 Linear SVM with hyperparameter tuning

```
In [23]: alpha = [10 ** x for x in range(-5, 2)] # hyperparam for SGD classifier.

         # read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated,
         # ------------------------------
         # default parameters
         # SGDClassifier(loss=hinge, penalty=l2, alpha=0.0001, l1_ratio=0.15, fit_intercept=Tr
```

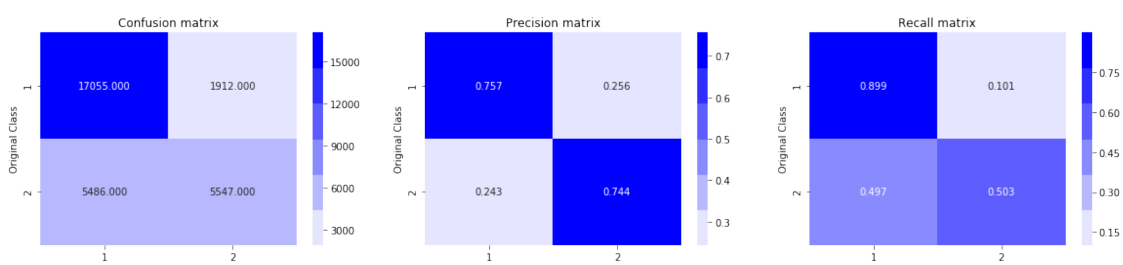
```python
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate=op
# class_weight=None, warm_start=False, average=False, n_iter=None)

# some of methods
# fit(X, y[, coef_init, intercept_init, ])        Fit linear model with Stochastic Gr
# predict(X)        Predict class labels for samples in X.

#-------------------------------
# video link:
#------------------------------


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: 0.6577563554122375
```

```
For values of alpha =  0.0001 The log loss is: 0.48095714968062203
For values of alpha =  0.001 The log loss is: 0.5153132694170238
For values of alpha =  0.01 The log loss is: 0.5605975343872299
For values of alpha =  0.1 The log loss is: 0.5963219033928199
For values of alpha =  1 The log loss is: 0.6426645798331788
For values of alpha =  10 The log loss is: 0.6577563554122375
```




```
For values of best alpha =  0.0001 The train log loss is: 0.47738141270571527
For values of best alpha =  0.0001 The test log loss is: 0.48095714968062203
Total number of data points : 30000
```



2.7 4.7 XGBoost with hyperparameter tuning

In [74]:
```
# use 20k datapoints for training and 5k points for testing
Xgb_X_train = X_train[:20000]
Xgb_y_train = y_train[:20000]
Xgb_X_test = X_test[:5000]
Xgb_y_test = y_test[:5000]

print(Xgb_X_train.shape)
print(Xgb_X_test.shape)
```

```
(20000, 794)
(5000, 794)
```

In [75]:
```
from xgboost import XGBClassifier
from sklearn.model_selection import RandomizedSearchCV
```

In [76]:
```
def XGB_best_params (X_train, y_train) :
    clf = XGBClassifier(n_jobs = -1)
    param_grid = {'learning_rate' : np.linspace(0,1,6),
                  'n_estimators' : [10, 30, 50, 100, 200, 500, 1000, 1200],
                  'max_depth' : list(range(1,7))}
    cv = 5
    rand_cv = RandomizedSearchCV(clf, param_grid, scoring='neg_log_loss', verbose=1, c
    rand_cv.fit(X_train, y_train)
    print('best Accuracy:', rand_cv.best_params_)
    print('best Score:', rand_cv.best_score_)
    #accessing cv_results
    cv_results = pd.DataFrame(rand_cv.cv_results_)
    plot_data_1 = cv_results[['param_n_estimators', 'mean_test_score']].sort_values('p
    #Function for cv_error vs alpha plot
    plt.figure(figsize=(10,6))
    plt.xlabel('Hyperparams')
    plt.ylabel('Best Score')
    plt.plot(plot_data_1['param_n_estimators'], -plot_data_1['mean_test_score'], marke
    plt.legend(loc='upper left')
```

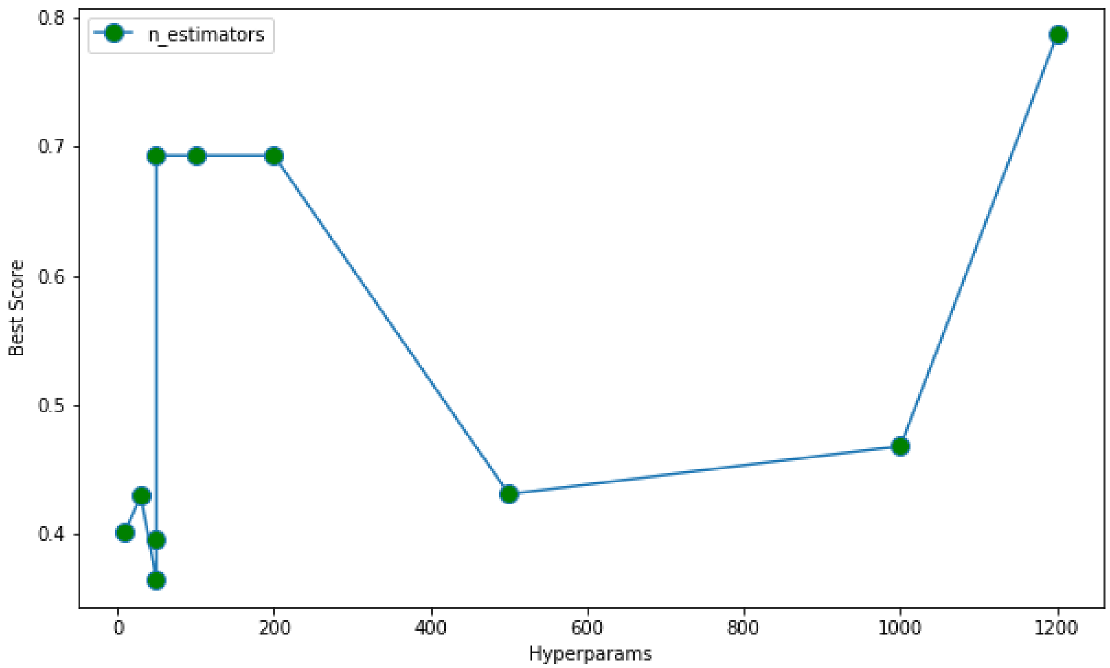
In [77]:
```
def XGB(learning_rate, n_estimators, max_depth, X_train, y_train, X_test, y_test) :
    clf = XGBClassifier(learning_rate=learning_rate, n_estimators=n_estimators, max_de
    clf.fit(X_train,y_train)
    y_pred = clf.predict(X_test)
    plot_confusion_matrix(y_test, y_pred)
```

In [80]:
```
XGB_best_params(Xgb_X_train, Xgb_y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
```

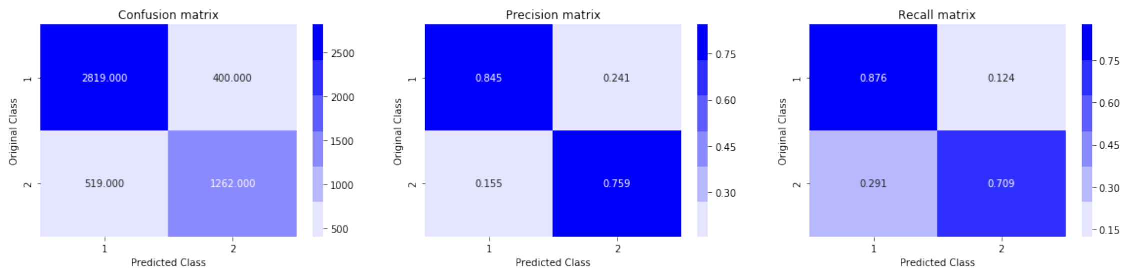
```
[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed: 12.7min
[Parallel(n_jobs=-1)]: Done  50 out of  50 | elapsed: 32.6min finished
```

```
best Accuracy: {'n_estimators': 50, 'max_depth': 6, 'learning_rate': 0.2}
best Score: -0.3646306380560294
```




```
In [83]: XGB(0.2, 50, 6, Xgb_X_train, Xgb_y_train, Xgb_X_test, Xgb_y_test)
```

```
C:\Users\Aravindh\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:151: DeprecationWa
  if diff:
```

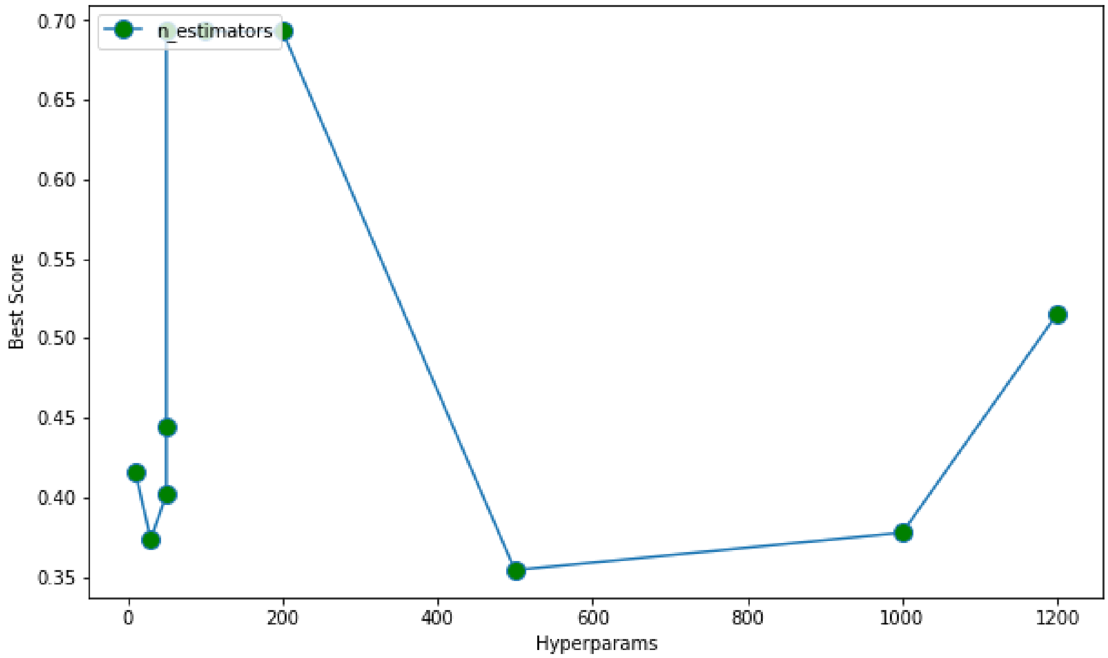


Lets see if the model improves by keeping learning rate much lower

In [89]:
```python
def XGB_best_params (X_train, y_train) :
    clf = XGBClassifier(n_jobs = -1)
    param_grid = {'learning_rate' : np.linspace(0,0.2,6),
                  'n_estimators' : [10, 30, 50, 100, 200, 500, 1000, 1200],
                  'max_depth' : list(range(1,7))}
    cv = 5
    rand_cv = RandomizedSearchCV(clf, param_grid, scoring='neg_log_loss', verbose=1, c
    rand_cv.fit(X_train, y_train)
    print('best Accuracy:', rand_cv.best_params_)
    print('best Score:', rand_cv.best_score_)
    #accessing cv_results
    cv_results = pd.DataFrame(rand_cv.cv_results_)
    plot_data_1 = cv_results[['param_n_estimators', 'mean_test_score']].sort_values('p
    #Function for cv_error vs alpha plot
    plt.figure(figsize=(10,6))
    plt.xlabel('Hyperparams')
    plt.ylabel('Best Score')
    plt.plot(plot_data_1['param_n_estimators'], -plot_data_1['mean_test_score'], marke
    plt.legend(loc='upper left')
```

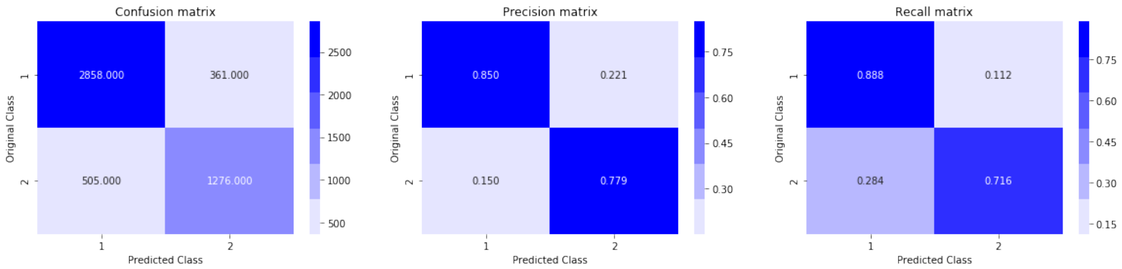
In [90]:
```python
XGB_best_params(Xgb_X_train, Xgb_y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits


[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed:  8.6min
[Parallel(n_jobs=-1)]: Done  50 out of  50 | elapsed: 34.0min finished


best Accuracy: {'n_estimators': 500, 'max_depth': 5, 'learning_rate': 0.04}
best Score: -0.35412609879250156
```




```
In [91]: XGB(0.04, 500, 5, Xgb_X_train, Xgb_y_train, Xgb_X_test, Xgb_y_test)
```

```
C:\Users\Aravindh\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:151: DeprecationWa
  if diff:
```



Yes, the model improves a little bit but not great improvement

```
In [96]: #lets viuallise the eval results & f
         clf = XGBClassifier(learning_rate=0.04, n_estimators=500, max_depth=5 , njobs=-1)
         clf.fit(Xgb_X_train, Xgb_y_train,
                 eval_set=[(Xgb_X_train, Xgb_y_train), (Xgb_X_test, Xgb_y_test)],
                 eval_metric='logloss',
                 verbose=True)
         y_pred = clf.predict(Xgb_X_test)
         fi = clf.feature_importances_
```

```
[0]	validation_0-logloss:0.675799	validation_1-logloss:0.675654
[1]	validation_0-logloss:0.659782	validation_1-logloss:0.65957
[2]	validation_0-logloss:0.644819	validation_1-logloss:0.644594
[3]	validation_0-logloss:0.63102	validation_1-logloss:0.630677
[4]	validation_0-logloss:0.618081	validation_1-logloss:0.617753
[5]	validation_0-logloss:0.60586	validation_1-logloss:0.605476
[6]	validation_0-logloss:0.593978	validation_1-logloss:0.593744
[7]	validation_0-logloss:0.583181	validation_1-logloss:0.582951
[8]	validation_0-logloss:0.573151	validation_1-logloss:0.57306
[9]	validation_0-logloss:0.563082	validation_1-logloss:0.563247
[10]	validation_0-logloss:0.553658	validation_1-logloss:0.553941
[11]	validation_0-logloss:0.544739	validation_1-logloss:0.545208
[12]	validation_0-logloss:0.535957	validation_1-logloss:0.53656
[13]	validation_0-logloss:0.527695	validation_1-logloss:0.528503
[14]	validation_0-logloss:0.519933	validation_1-logloss:0.520991
[15]	validation_0-logloss:0.512874	validation_1-logloss:0.514056
[16]	validation_0-logloss:0.505733	validation_1-logloss:0.507205
[17]	validation_0-logloss:0.499111	validation_1-logloss:0.500878
[18]	validation_0-logloss:0.492718	validation_1-logloss:0.494873
[19]	validation_0-logloss:0.486802	validation_1-logloss:0.489126
[20]	validation_0-logloss:0.481216	validation_1-logloss:0.483682
[21]	validation_0-logloss:0.476064	validation_1-logloss:0.478764
[22]	validation_0-logloss:0.471005	validation_1-logloss:0.47381
[23]	validation_0-logloss:0.465951	validation_1-logloss:0.469016
[24]	validation_0-logloss:0.461357	validation_1-logloss:0.464629
[25]	validation_0-logloss:0.457184	validation_1-logloss:0.460796
[26]	validation_0-logloss:0.452877	validation_1-logloss:0.456719
[27]	validation_0-logloss:0.448587	validation_1-logloss:0.452839
[28]	validation_0-logloss:0.44486	validation_1-logloss:0.449237
[29]	validation_0-logloss:0.440993	validation_1-logloss:0.445717
[30]	validation_0-logloss:0.437269	validation_1-logloss:0.442252
[31]	validation_0-logloss:0.433712	validation_1-logloss:0.439118
[32]	validation_0-logloss:0.430567	validation_1-logloss:0.436246
[33]	validation_0-logloss:0.427338	validation_1-logloss:0.433273
[34]	validation_0-logloss:0.424525	validation_1-logloss:0.430703
[35]	validation_0-logloss:0.421527	validation_1-logloss:0.427926
[36]	validation_0-logloss:0.418593	validation_1-logloss:0.425227
[37]	validation_0-logloss:0.41583	validation_1-logloss:0.422759
[38]	validation_0-logloss:0.413288	validation_1-logloss:0.420414
[39]	validation_0-logloss:0.410857	validation_1-logloss:0.418059
[40]	validation_0-logloss:0.408576	validation_1-logloss:0.415911
[41]	validation_0-logloss:0.406378	validation_1-logloss:0.413986
[42]	validation_0-logloss:0.404217	validation_1-logloss:0.412096
[43]	validation_0-logloss:0.402067	validation_1-logloss:0.41021
[44]	validation_0-logloss:0.40002	validation_1-logloss:0.408399
[45]	validation_0-logloss:0.398158	validation_1-logloss:0.406845
[46]	validation_0-logloss:0.396512	validation_1-logloss:0.405344
[47]	validation_0-logloss:0.39483	validation_1-logloss:0.403934
```

```
[48]	validation_0-logloss:0.393185	validation_1-logloss:0.402533
[49]	validation_0-logloss:0.391695	validation_1-logloss:0.401256
[50]	validation_0-logloss:0.3901	validation_1-logloss:0.399929
[51]	validation_0-logloss:0.388456	validation_1-logloss:0.398514
[52]	validation_0-logloss:0.386739	validation_1-logloss:0.397229
[53]	validation_0-logloss:0.385248	validation_1-logloss:0.396028
[54]	validation_0-logloss:0.383679	validation_1-logloss:0.394772
[55]	validation_0-logloss:0.382345	validation_1-logloss:0.39373
[56]	validation_0-logloss:0.380999	validation_1-logloss:0.392652
[57]	validation_0-logloss:0.379849	validation_1-logloss:0.391698
[58]	validation_0-logloss:0.37834	validation_1-logloss:0.390412
[59]	validation_0-logloss:0.37708	validation_1-logloss:0.389389
[60]	validation_0-logloss:0.375942	validation_1-logloss:0.388498
[61]	validation_0-logloss:0.374597	validation_1-logloss:0.387448
[62]	validation_0-logloss:0.373313	validation_1-logloss:0.386459
[63]	validation_0-logloss:0.372178	validation_1-logloss:0.38564
[64]	validation_0-logloss:0.371198	validation_1-logloss:0.385055
[65]	validation_0-logloss:0.37024	validation_1-logloss:0.384341
[66]	validation_0-logloss:0.369087	validation_1-logloss:0.383567
[67]	validation_0-logloss:0.368103	validation_1-logloss:0.382868
[68]	validation_0-logloss:0.367002	validation_1-logloss:0.382174
[69]	validation_0-logloss:0.366151	validation_1-logloss:0.381544
[70]	validation_0-logloss:0.365125	validation_1-logloss:0.380814
[71]	validation_0-logloss:0.364098	validation_1-logloss:0.380114
[72]	validation_0-logloss:0.363106	validation_1-logloss:0.379402
[73]	validation_0-logloss:0.362127	validation_1-logloss:0.378774
[74]	validation_0-logloss:0.3614	validation_1-logloss:0.378244
[75]	validation_0-logloss:0.360682	validation_1-logloss:0.377748
[76]	validation_0-logloss:0.35992	validation_1-logloss:0.377342
[77]	validation_0-logloss:0.359035	validation_1-logloss:0.376829
[78]	validation_0-logloss:0.358422	validation_1-logloss:0.376382
[79]	validation_0-logloss:0.3576	validation_1-logloss:0.37584
[80]	validation_0-logloss:0.356787	validation_1-logloss:0.375333
[81]	validation_0-logloss:0.356104	validation_1-logloss:0.375035
[82]	validation_0-logloss:0.355308	validation_1-logloss:0.374634
[83]	validation_0-logloss:0.354565	validation_1-logloss:0.374389
[84]	validation_0-logloss:0.353908	validation_1-logloss:0.374047
[85]	validation_0-logloss:0.353176	validation_1-logloss:0.373588
[86]	validation_0-logloss:0.352458	validation_1-logloss:0.373253
[87]	validation_0-logloss:0.351649	validation_1-logloss:0.372826
[88]	validation_0-logloss:0.350929	validation_1-logloss:0.372453
[89]	validation_0-logloss:0.350266	validation_1-logloss:0.372218
[90]	validation_0-logloss:0.349582	validation_1-logloss:0.371842
[91]	validation_0-logloss:0.348926	validation_1-logloss:0.371585
[92]	validation_0-logloss:0.348296	validation_1-logloss:0.371432
[93]	validation_0-logloss:0.347668	validation_1-logloss:0.370949
[94]	validation_0-logloss:0.347088	validation_1-logloss:0.370576
[95]	validation_0-logloss:0.34641	validation_1-logloss:0.370252
```

```
[96]          validation_0-logloss:0.34577        validation_1-logloss:0.369953
[97]          validation_0-logloss:0.345131        validation_1-logloss:0.369456
[98]          validation_0-logloss:0.344552        validation_1-logloss:0.3691
[99]          validation_0-logloss:0.343952        validation_1-logloss:0.368768
[100]          validation_0-logloss:0.343445        validation_1-logloss:0.368657
[101]          validation_0-logloss:0.342867        validation_1-logloss:0.368475
[102]          validation_0-logloss:0.342324        validation_1-logloss:0.368267
[103]          validation_0-logloss:0.341673        validation_1-logloss:0.367976
[104]          validation_0-logloss:0.340966        validation_1-logloss:0.367607
[105]          validation_0-logloss:0.340365        validation_1-logloss:0.367308
[106]          validation_0-logloss:0.339855        validation_1-logloss:0.367134
[107]          validation_0-logloss:0.339368        validation_1-logloss:0.366878
[108]          validation_0-logloss:0.338826        validation_1-logloss:0.3665
[109]          validation_0-logloss:0.338294        validation_1-logloss:0.366342
[110]          validation_0-logloss:0.337828        validation_1-logloss:0.366162
[111]          validation_0-logloss:0.337249        validation_1-logloss:0.365903
[112]          validation_0-logloss:0.336788        validation_1-logloss:0.365571
[113]          validation_0-logloss:0.336001        validation_1-logloss:0.365109
[114]          validation_0-logloss:0.335512        validation_1-logloss:0.36495
[115]          validation_0-logloss:0.33498        validation_1-logloss:0.364797
[116]          validation_0-logloss:0.334648        validation_1-logloss:0.364671
[117]          validation_0-logloss:0.333687        validation_1-logloss:0.364319
[118]          validation_0-logloss:0.332763        validation_1-logloss:0.36399
[119]          validation_0-logloss:0.332338        validation_1-logloss:0.363882
[120]          validation_0-logloss:0.331781        validation_1-logloss:0.363648
[121]          validation_0-logloss:0.331279        validation_1-logloss:0.363601
[122]          validation_0-logloss:0.330806        validation_1-logloss:0.36343
[123]          validation_0-logloss:0.330274        validation_1-logloss:0.363285
[124]          validation_0-logloss:0.329939        validation_1-logloss:0.363078
[125]          validation_0-logloss:0.329551        validation_1-logloss:0.362931
[126]          validation_0-logloss:0.328719        validation_1-logloss:0.362661
[127]          validation_0-logloss:0.328276        validation_1-logloss:0.362391
[128]          validation_0-logloss:0.327873        validation_1-logloss:0.362324
[129]          validation_0-logloss:0.327107        validation_1-logloss:0.362139
[130]          validation_0-logloss:0.326671        validation_1-logloss:0.361896
[131]          validation_0-logloss:0.326234        validation_1-logloss:0.361821
[132]          validation_0-logloss:0.3253        validation_1-logloss:0.361622
[133]          validation_0-logloss:0.325        validation_1-logloss:0.361492
[134]          validation_0-logloss:0.324518        validation_1-logloss:0.361386
[135]          validation_0-logloss:0.324084        validation_1-logloss:0.361222
[136]          validation_0-logloss:0.323541        validation_1-logloss:0.360917
[137]          validation_0-logloss:0.322585        validation_1-logloss:0.360756
[138]          validation_0-logloss:0.322207        validation_1-logloss:0.360788
[139]          validation_0-logloss:0.321577        validation_1-logloss:0.360605
[140]          validation_0-logloss:0.320828        validation_1-logloss:0.360346
[141]          validation_0-logloss:0.320133        validation_1-logloss:0.360063
[142]          validation_0-logloss:0.319344        validation_1-logloss:0.360004
[143]          validation_0-logloss:0.318631        validation_1-logloss:0.359839
```

```
[144]	validation_0-logloss:0.31827	validation_1-logloss:0.359698
[145]	validation_0-logloss:0.317529	validation_1-logloss:0.359657
[146]	validation_0-logloss:0.317079	validation_1-logloss:0.359623
[147]	validation_0-logloss:0.3167	validation_1-logloss:0.359536
[148]	validation_0-logloss:0.31631	validation_1-logloss:0.359339
[149]	validation_0-logloss:0.315706	validation_1-logloss:0.359146
[150]	validation_0-logloss:0.314899	validation_1-logloss:0.358902
[151]	validation_0-logloss:0.314288	validation_1-logloss:0.358871
[152]	validation_0-logloss:0.313813	validation_1-logloss:0.358817
[153]	validation_0-logloss:0.313258	validation_1-logloss:0.358785
[154]	validation_0-logloss:0.312909	validation_1-logloss:0.358657
[155]	validation_0-logloss:0.312613	validation_1-logloss:0.358591
[156]	validation_0-logloss:0.312256	validation_1-logloss:0.358399
[157]	validation_0-logloss:0.311699	validation_1-logloss:0.358215
[158]	validation_0-logloss:0.310945	validation_1-logloss:0.358029
[159]	validation_0-logloss:0.310088	validation_1-logloss:0.357859
[160]	validation_0-logloss:0.309616	validation_1-logloss:0.357684
[161]	validation_0-logloss:0.308916	validation_1-logloss:0.357337
[162]	validation_0-logloss:0.308324	validation_1-logloss:0.357263
[163]	validation_0-logloss:0.307955	validation_1-logloss:0.357145
[164]	validation_0-logloss:0.307444	validation_1-logloss:0.357117
[165]	validation_0-logloss:0.307046	validation_1-logloss:0.357144
[166]	validation_0-logloss:0.30655	validation_1-logloss:0.357145
[167]	validation_0-logloss:0.305868	validation_1-logloss:0.357147
[168]	validation_0-logloss:0.305587	validation_1-logloss:0.35701
[169]	validation_0-logloss:0.305155	validation_1-logloss:0.356925
[170]	validation_0-logloss:0.304596	validation_1-logloss:0.356926
[171]	validation_0-logloss:0.303925	validation_1-logloss:0.356647
[172]	validation_0-logloss:0.303313	validation_1-logloss:0.35664
[173]	validation_0-logloss:0.302673	validation_1-logloss:0.3565
[174]	validation_0-logloss:0.302465	validation_1-logloss:0.356468
[175]	validation_0-logloss:0.302181	validation_1-logloss:0.356352
[176]	validation_0-logloss:0.30173	validation_1-logloss:0.356346
[177]	validation_0-logloss:0.301357	validation_1-logloss:0.356296
[178]	validation_0-logloss:0.300717	validation_1-logloss:0.356235
[179]	validation_0-logloss:0.300075	validation_1-logloss:0.356178
[180]	validation_0-logloss:0.29985	validation_1-logloss:0.356068
[181]	validation_0-logloss:0.299187	validation_1-logloss:0.355929
[182]	validation_0-logloss:0.298542	validation_1-logloss:0.355835
[183]	validation_0-logloss:0.297977	validation_1-logloss:0.355794
[184]	validation_0-logloss:0.297374	validation_1-logloss:0.355722
[185]	validation_0-logloss:0.297145	validation_1-logloss:0.355748
[186]	validation_0-logloss:0.296848	validation_1-logloss:0.355652
[187]	validation_0-logloss:0.296295	validation_1-logloss:0.35556
[188]	validation_0-logloss:0.295965	validation_1-logloss:0.355517
[189]	validation_0-logloss:0.295457	validation_1-logloss:0.355452
[190]	validation_0-logloss:0.295042	validation_1-logloss:0.355344
[191]	validation_0-logloss:0.294612	validation_1-logloss:0.355245
```

```
[192]	validation_0-logloss:0.294244	validation_1-logloss:0.355124
[193]	validation_0-logloss:0.293933	validation_1-logloss:0.355124
[194]	validation_0-logloss:0.293437	validation_1-logloss:0.355065
[195]	validation_0-logloss:0.293234	validation_1-logloss:0.355071
[196]	validation_0-logloss:0.292649	validation_1-logloss:0.354908
[197]	validation_0-logloss:0.29197	validation_1-logloss:0.354854
[198]	validation_0-logloss:0.291617	validation_1-logloss:0.354834
[199]	validation_0-logloss:0.291038	validation_1-logloss:0.354833
[200]	validation_0-logloss:0.290674	validation_1-logloss:0.354681
[201]	validation_0-logloss:0.290411	validation_1-logloss:0.354586
[202]	validation_0-logloss:0.289824	validation_1-logloss:0.354572
[203]	validation_0-logloss:0.289472	validation_1-logloss:0.354487
[204]	validation_0-logloss:0.28922	validation_1-logloss:0.354512
[205]	validation_0-logloss:0.288669	validation_1-logloss:0.354451
[206]	validation_0-logloss:0.28847	validation_1-logloss:0.354478
[207]	validation_0-logloss:0.287852	validation_1-logloss:0.354495
[208]	validation_0-logloss:0.28733	validation_1-logloss:0.354399
[209]	validation_0-logloss:0.286705	validation_1-logloss:0.354314
[210]	validation_0-logloss:0.286206	validation_1-logloss:0.354183
[211]	validation_0-logloss:0.285523	validation_1-logloss:0.353979
[212]	validation_0-logloss:0.284899	validation_1-logloss:0.353886
[213]	validation_0-logloss:0.284656	validation_1-logloss:0.353834
[214]	validation_0-logloss:0.284256	validation_1-logloss:0.353792
[215]	validation_0-logloss:0.283998	validation_1-logloss:0.353836
[216]	validation_0-logloss:0.283649	validation_1-logloss:0.353785
[217]	validation_0-logloss:0.283249	validation_1-logloss:0.353782
[218]	validation_0-logloss:0.283075	validation_1-logloss:0.35383
[219]	validation_0-logloss:0.282595	validation_1-logloss:0.353736
[220]	validation_0-logloss:0.28217	validation_1-logloss:0.353668
[221]	validation_0-logloss:0.281613	validation_1-logloss:0.353502
[222]	validation_0-logloss:0.281245	validation_1-logloss:0.353507
[223]	validation_0-logloss:0.280709	validation_1-logloss:0.353505
[224]	validation_0-logloss:0.28012	validation_1-logloss:0.353501
[225]	validation_0-logloss:0.279945	validation_1-logloss:0.353487
[226]	validation_0-logloss:0.279718	validation_1-logloss:0.353395
[227]	validation_0-logloss:0.279508	validation_1-logloss:0.353351
[228]	validation_0-logloss:0.279082	validation_1-logloss:0.353354
[229]	validation_0-logloss:0.278539	validation_1-logloss:0.353163
[230]	validation_0-logloss:0.277981	validation_1-logloss:0.353212
[231]	validation_0-logloss:0.277349	validation_1-logloss:0.3531
[232]	validation_0-logloss:0.277057	validation_1-logloss:0.353001
[233]	validation_0-logloss:0.276713	validation_1-logloss:0.352907
[234]	validation_0-logloss:0.276318	validation_1-logloss:0.35283
[235]	validation_0-logloss:0.275813	validation_1-logloss:0.352796
[236]	validation_0-logloss:0.275371	validation_1-logloss:0.352748
[237]	validation_0-logloss:0.275064	validation_1-logloss:0.352751
[238]	validation_0-logloss:0.274649	validation_1-logloss:0.35282
[239]	validation_0-logloss:0.274258	validation_1-logloss:0.352826
```

```
[240]	validation_0-logloss:0.273876	validation_1-logloss:0.352813
[241]	validation_0-logloss:0.273646	validation_1-logloss:0.35285
[242]	validation_0-logloss:0.273433	validation_1-logloss:0.352814
[243]	validation_0-logloss:0.273148	validation_1-logloss:0.35285
[244]	validation_0-logloss:0.272615	validation_1-logloss:0.352657
[245]	validation_0-logloss:0.272196	validation_1-logloss:0.35248
[246]	validation_0-logloss:0.271628	validation_1-logloss:0.352556
[247]	validation_0-logloss:0.271066	validation_1-logloss:0.352538
[248]	validation_0-logloss:0.270699	validation_1-logloss:0.352478
[249]	validation_0-logloss:0.270296	validation_1-logloss:0.352501
[250]	validation_0-logloss:0.269992	validation_1-logloss:0.352431
[251]	validation_0-logloss:0.269642	validation_1-logloss:0.352369
[252]	validation_0-logloss:0.269296	validation_1-logloss:0.352384
[253]	validation_0-logloss:0.268647	validation_1-logloss:0.352279
[254]	validation_0-logloss:0.268138	validation_1-logloss:0.352129
[255]	validation_0-logloss:0.267579	validation_1-logloss:0.352118
[256]	validation_0-logloss:0.267149	validation_1-logloss:0.352061
[257]	validation_0-logloss:0.266619	validation_1-logloss:0.351948
[258]	validation_0-logloss:0.266057	validation_1-logloss:0.351999
[259]	validation_0-logloss:0.265459	validation_1-logloss:0.351868
[260]	validation_0-logloss:0.26518	validation_1-logloss:0.351834
[261]	validation_0-logloss:0.265059	validation_1-logloss:0.351888
[262]	validation_0-logloss:0.264753	validation_1-logloss:0.351798
[263]	validation_0-logloss:0.264443	validation_1-logloss:0.351829
[264]	validation_0-logloss:0.263896	validation_1-logloss:0.351747
[265]	validation_0-logloss:0.26364	validation_1-logloss:0.351741
[266]	validation_0-logloss:0.263436	validation_1-logloss:0.351709
[267]	validation_0-logloss:0.262812	validation_1-logloss:0.351504
[268]	validation_0-logloss:0.26248	validation_1-logloss:0.351417
[269]	validation_0-logloss:0.262041	validation_1-logloss:0.351483
[270]	validation_0-logloss:0.261669	validation_1-logloss:0.351496
[271]	validation_0-logloss:0.261517	validation_1-logloss:0.351508
[272]	validation_0-logloss:0.260993	validation_1-logloss:0.351513
[273]	validation_0-logloss:0.260622	validation_1-logloss:0.351452
[274]	validation_0-logloss:0.260167	validation_1-logloss:0.351489
[275]	validation_0-logloss:0.25954	validation_1-logloss:0.351442
[276]	validation_0-logloss:0.259152	validation_1-logloss:0.3514
[277]	validation_0-logloss:0.258598	validation_1-logloss:0.351317
[278]	validation_0-logloss:0.258153	validation_1-logloss:0.351236
[279]	validation_0-logloss:0.257811	validation_1-logloss:0.351363
[280]	validation_0-logloss:0.257343	validation_1-logloss:0.351279
[281]	validation_0-logloss:0.256787	validation_1-logloss:0.351328
[282]	validation_0-logloss:0.256522	validation_1-logloss:0.351266
[283]	validation_0-logloss:0.256338	validation_1-logloss:0.351197
[284]	validation_0-logloss:0.256203	validation_1-logloss:0.351188
[285]	validation_0-logloss:0.255643	validation_1-logloss:0.351084
[286]	validation_0-logloss:0.255159	validation_1-logloss:0.351168
[287]	validation_0-logloss:0.254507	validation_1-logloss:0.351097
```

```
[288]	validation_0-logloss:0.25417	validation_1-logloss:0.351112
[289]	validation_0-logloss:0.253894	validation_1-logloss:0.351096
[290]	validation_0-logloss:0.253367	validation_1-logloss:0.351134
[291]	validation_0-logloss:0.252776	validation_1-logloss:0.351022
[292]	validation_0-logloss:0.252607	validation_1-logloss:0.351018
[293]	validation_0-logloss:0.252277	validation_1-logloss:0.350991
[294]	validation_0-logloss:0.251839	validation_1-logloss:0.351033
[295]	validation_0-logloss:0.251372	validation_1-logloss:0.350834
[296]	validation_0-logloss:0.251008	validation_1-logloss:0.350875
[297]	validation_0-logloss:0.250765	validation_1-logloss:0.350919
[298]	validation_0-logloss:0.250298	validation_1-logloss:0.350847
[299]	validation_0-logloss:0.250039	validation_1-logloss:0.350827
[300]	validation_0-logloss:0.249592	validation_1-logloss:0.350713
[301]	validation_0-logloss:0.249387	validation_1-logloss:0.350725
[302]	validation_0-logloss:0.249247	validation_1-logloss:0.350694
[303]	validation_0-logloss:0.248901	validation_1-logloss:0.350718
[304]	validation_0-logloss:0.248306	validation_1-logloss:0.350704
[305]	validation_0-logloss:0.247914	validation_1-logloss:0.350723
[306]	validation_0-logloss:0.247739	validation_1-logloss:0.350733
[307]	validation_0-logloss:0.2474	validation_1-logloss:0.35066
[308]	validation_0-logloss:0.247218	validation_1-logloss:0.350683
[309]	validation_0-logloss:0.246851	validation_1-logloss:0.350653
[310]	validation_0-logloss:0.246608	validation_1-logloss:0.350634
[311]	validation_0-logloss:0.246083	validation_1-logloss:0.350612
[312]	validation_0-logloss:0.245889	validation_1-logloss:0.35064
[313]	validation_0-logloss:0.245393	validation_1-logloss:0.350519
[314]	validation_0-logloss:0.244845	validation_1-logloss:0.350366
[315]	validation_0-logloss:0.244489	validation_1-logloss:0.350326
[316]	validation_0-logloss:0.244172	validation_1-logloss:0.350414
[317]	validation_0-logloss:0.243622	validation_1-logloss:0.350459
[318]	validation_0-logloss:0.243173	validation_1-logloss:0.350451
[319]	validation_0-logloss:0.242839	validation_1-logloss:0.350444
[320]	validation_0-logloss:0.2423	validation_1-logloss:0.350338
[321]	validation_0-logloss:0.241883	validation_1-logloss:0.350378
[322]	validation_0-logloss:0.241691	validation_1-logloss:0.350381
[323]	validation_0-logloss:0.241153	validation_1-logloss:0.350346
[324]	validation_0-logloss:0.240859	validation_1-logloss:0.3504
[325]	validation_0-logloss:0.240461	validation_1-logloss:0.350393
[326]	validation_0-logloss:0.239911	validation_1-logloss:0.350407
[327]	validation_0-logloss:0.239603	validation_1-logloss:0.350363
[328]	validation_0-logloss:0.239101	validation_1-logloss:0.350347
[329]	validation_0-logloss:0.23866	validation_1-logloss:0.350398
[330]	validation_0-logloss:0.238073	validation_1-logloss:0.350448
[331]	validation_0-logloss:0.237922	validation_1-logloss:0.350365
[332]	validation_0-logloss:0.237777	validation_1-logloss:0.350361
[333]	validation_0-logloss:0.237546	validation_1-logloss:0.350336
[334]	validation_0-logloss:0.237042	validation_1-logloss:0.35035
[335]	validation_0-logloss:0.236709	validation_1-logloss:0.350326
```

```
[336]	validation_0-logloss:0.236304	validation_1-logloss:0.350306
[337]	validation_0-logloss:0.236057	validation_1-logloss:0.350327
[338]	validation_0-logloss:0.235551	validation_1-logloss:0.350316
[339]	validation_0-logloss:0.235349	validation_1-logloss:0.350296
[340]	validation_0-logloss:0.235096	validation_1-logloss:0.350368
[341]	validation_0-logloss:0.234595	validation_1-logloss:0.350356
[342]	validation_0-logloss:0.234152	validation_1-logloss:0.350383
[343]	validation_0-logloss:0.233762	validation_1-logloss:0.350319
[344]	validation_0-logloss:0.233486	validation_1-logloss:0.350279
[345]	validation_0-logloss:0.232964	validation_1-logloss:0.350467
[346]	validation_0-logloss:0.232725	validation_1-logloss:0.350496
[347]	validation_0-logloss:0.232387	validation_1-logloss:0.350491
[348]	validation_0-logloss:0.231914	validation_1-logloss:0.350525
[349]	validation_0-logloss:0.231703	validation_1-logloss:0.350538
[350]	validation_0-logloss:0.231353	validation_1-logloss:0.350451
[351]	validation_0-logloss:0.230927	validation_1-logloss:0.350419
[352]	validation_0-logloss:0.230414	validation_1-logloss:0.350375
[353]	validation_0-logloss:0.229929	validation_1-logloss:0.350403
[354]	validation_0-logloss:0.229532	validation_1-logloss:0.350425
[355]	validation_0-logloss:0.229246	validation_1-logloss:0.350369
[356]	validation_0-logloss:0.229	validation_1-logloss:0.350374
[357]	validation_0-logloss:0.228759	validation_1-logloss:0.350266
[358]	validation_0-logloss:0.228583	validation_1-logloss:0.350206
[359]	validation_0-logloss:0.228104	validation_1-logloss:0.350131
[360]	validation_0-logloss:0.227861	validation_1-logloss:0.350126
[361]	validation_0-logloss:0.22751	validation_1-logloss:0.350146
[362]	validation_0-logloss:0.226967	validation_1-logloss:0.350114
[363]	validation_0-logloss:0.226461	validation_1-logloss:0.350197
[364]	validation_0-logloss:0.225984	validation_1-logloss:0.350091
[365]	validation_0-logloss:0.225702	validation_1-logloss:0.350064
[366]	validation_0-logloss:0.225281	validation_1-logloss:0.350022
[367]	validation_0-logloss:0.224891	validation_1-logloss:0.349969
[368]	validation_0-logloss:0.224428	validation_1-logloss:0.350032
[369]	validation_0-logloss:0.224164	validation_1-logloss:0.349999
[370]	validation_0-logloss:0.223929	validation_1-logloss:0.350003
[371]	validation_0-logloss:0.223733	validation_1-logloss:0.349908
[372]	validation_0-logloss:0.223232	validation_1-logloss:0.349922
[373]	validation_0-logloss:0.222704	validation_1-logloss:0.349805
[374]	validation_0-logloss:0.222452	validation_1-logloss:0.34988
[375]	validation_0-logloss:0.22223	validation_1-logloss:0.349862
[376]	validation_0-logloss:0.222046	validation_1-logloss:0.349921
[377]	validation_0-logloss:0.221656	validation_1-logloss:0.349903
[378]	validation_0-logloss:0.221203	validation_1-logloss:0.349797
[379]	validation_0-logloss:0.220795	validation_1-logloss:0.349756
[380]	validation_0-logloss:0.220472	validation_1-logloss:0.349656
[381]	validation_0-logloss:0.2201	validation_1-logloss:0.349539
[382]	validation_0-logloss:0.219644	validation_1-logloss:0.349501
[383]	validation_0-logloss:0.219232	validation_1-logloss:0.349468
```

```
[384]	validation_0-logloss:0.219069	validation_1-logloss:0.349399
[385]	validation_0-logloss:0.218723	validation_1-logloss:0.349359
[386]	validation_0-logloss:0.21856	validation_1-logloss:0.349341
[387]	validation_0-logloss:0.218158	validation_1-logloss:0.34936
[388]	validation_0-logloss:0.217734	validation_1-logloss:0.349312
[389]	validation_0-logloss:0.217291	validation_1-logloss:0.34925
[390]	validation_0-logloss:0.216967	validation_1-logloss:0.349251
[391]	validation_0-logloss:0.216679	validation_1-logloss:0.349296
[392]	validation_0-logloss:0.216526	validation_1-logloss:0.349267
[393]	validation_0-logloss:0.216107	validation_1-logloss:0.34926
[394]	validation_0-logloss:0.215832	validation_1-logloss:0.349284
[395]	validation_0-logloss:0.215622	validation_1-logloss:0.349273
[396]	validation_0-logloss:0.215193	validation_1-logloss:0.349211
[397]	validation_0-logloss:0.214887	validation_1-logloss:0.349152
[398]	validation_0-logloss:0.214523	validation_1-logloss:0.349153
[399]	validation_0-logloss:0.214083	validation_1-logloss:0.34909
[400]	validation_0-logloss:0.213844	validation_1-logloss:0.349072
[401]	validation_0-logloss:0.213452	validation_1-logloss:0.348958
[402]	validation_0-logloss:0.213023	validation_1-logloss:0.349021
[403]	validation_0-logloss:0.212741	validation_1-logloss:0.348969
[404]	validation_0-logloss:0.212433	validation_1-logloss:0.348924
[405]	validation_0-logloss:0.212113	validation_1-logloss:0.348982
[406]	validation_0-logloss:0.211774	validation_1-logloss:0.349023
[407]	validation_0-logloss:0.211441	validation_1-logloss:0.348955
[408]	validation_0-logloss:0.211011	validation_1-logloss:0.348989
[409]	validation_0-logloss:0.210542	validation_1-logloss:0.348955
[410]	validation_0-logloss:0.210426	validation_1-logloss:0.348927
[411]	validation_0-logloss:0.210004	validation_1-logloss:0.348851
[412]	validation_0-logloss:0.209558	validation_1-logloss:0.34875
[413]	validation_0-logloss:0.209345	validation_1-logloss:0.348707
[414]	validation_0-logloss:0.209165	validation_1-logloss:0.348658
[415]	validation_0-logloss:0.209043	validation_1-logloss:0.348635
[416]	validation_0-logloss:0.208617	validation_1-logloss:0.348722
[417]	validation_0-logloss:0.208351	validation_1-logloss:0.348771
[418]	validation_0-logloss:0.207852	validation_1-logloss:0.348763
[419]	validation_0-logloss:0.20761	validation_1-logloss:0.348784
[420]	validation_0-logloss:0.20744	validation_1-logloss:0.348795
[421]	validation_0-logloss:0.207108	validation_1-logloss:0.348836
[422]	validation_0-logloss:0.206683	validation_1-logloss:0.348772
[423]	validation_0-logloss:0.206568	validation_1-logloss:0.348825
[424]	validation_0-logloss:0.206195	validation_1-logloss:0.348801
[425]	validation_0-logloss:0.205933	validation_1-logloss:0.348729
[426]	validation_0-logloss:0.205846	validation_1-logloss:0.348725
[427]	validation_0-logloss:0.205646	validation_1-logloss:0.34877
[428]	validation_0-logloss:0.205183	validation_1-logloss:0.348667
[429]	validation_0-logloss:0.20506	validation_1-logloss:0.348705
[430]	validation_0-logloss:0.20481	validation_1-logloss:0.348671
[431]	validation_0-logloss:0.204522	validation_1-logloss:0.348784
```

```
[432]	validation_0-logloss:0.204408	validation_1-logloss:0.34869
[433]	validation_0-logloss:0.204066	validation_1-logloss:0.34861
[434]	validation_0-logloss:0.203729	validation_1-logloss:0.348599
[435]	validation_0-logloss:0.203347	validation_1-logloss:0.348534
[436]	validation_0-logloss:0.203066	validation_1-logloss:0.348512
[437]	validation_0-logloss:0.202806	validation_1-logloss:0.348527
[438]	validation_0-logloss:0.202568	validation_1-logloss:0.348547
[439]	validation_0-logloss:0.202165	validation_1-logloss:0.348535
[440]	validation_0-logloss:0.201757	validation_1-logloss:0.348465
[441]	validation_0-logloss:0.201406	validation_1-logloss:0.348484
[442]	validation_0-logloss:0.201146	validation_1-logloss:0.34842
[443]	validation_0-logloss:0.201031	validation_1-logloss:0.348411
[444]	validation_0-logloss:0.20084	validation_1-logloss:0.348389
[445]	validation_0-logloss:0.200404	validation_1-logloss:0.348266
[446]	validation_0-logloss:0.200054	validation_1-logloss:0.348241
[447]	validation_0-logloss:0.199589	validation_1-logloss:0.348232
[448]	validation_0-logloss:0.199241	validation_1-logloss:0.348132
[449]	validation_0-logloss:0.198913	validation_1-logloss:0.348145
[450]	validation_0-logloss:0.198463	validation_1-logloss:0.348187
[451]	validation_0-logloss:0.198271	validation_1-logloss:0.348174
[452]	validation_0-logloss:0.198138	validation_1-logloss:0.348189
[453]	validation_0-logloss:0.197964	validation_1-logloss:0.348242
[454]	validation_0-logloss:0.197869	validation_1-logloss:0.348222
[455]	validation_0-logloss:0.197674	validation_1-logloss:0.348234
[456]	validation_0-logloss:0.197489	validation_1-logloss:0.348304
[457]	validation_0-logloss:0.197374	validation_1-logloss:0.348336
[458]	validation_0-logloss:0.197205	validation_1-logloss:0.348361
[459]	validation_0-logloss:0.197097	validation_1-logloss:0.348346
[460]	validation_0-logloss:0.196715	validation_1-logloss:0.348278
[461]	validation_0-logloss:0.196359	validation_1-logloss:0.348289
[462]	validation_0-logloss:0.196131	validation_1-logloss:0.3483
[463]	validation_0-logloss:0.195808	validation_1-logloss:0.34839
[464]	validation_0-logloss:0.195373	validation_1-logloss:0.348423
[465]	validation_0-logloss:0.19502	validation_1-logloss:0.348316
[466]	validation_0-logloss:0.194655	validation_1-logloss:0.348217
[467]	validation_0-logloss:0.194546	validation_1-logloss:0.348213
[468]	validation_0-logloss:0.194351	validation_1-logloss:0.348201
[469]	validation_0-logloss:0.194037	validation_1-logloss:0.348189
[470]	validation_0-logloss:0.193651	validation_1-logloss:0.348136
[471]	validation_0-logloss:0.193414	validation_1-logloss:0.348134
[472]	validation_0-logloss:0.193254	validation_1-logloss:0.348104
[473]	validation_0-logloss:0.193129	validation_1-logloss:0.348095
[474]	validation_0-logloss:0.192776	validation_1-logloss:0.348114
[475]	validation_0-logloss:0.192613	validation_1-logloss:0.348108
[476]	validation_0-logloss:0.192407	validation_1-logloss:0.348168
[477]	validation_0-logloss:0.192023	validation_1-logloss:0.348244
[478]	validation_0-logloss:0.191664	validation_1-logloss:0.348128
[479]	validation_0-logloss:0.191458	validation_1-logloss:0.348184
```

```
[480]       validation_0-logloss:0.191247       validation_1-logloss:0.348173
[481]       validation_0-logloss:0.190856       validation_1-logloss:0.348184
[482]       validation_0-logloss:0.190652       validation_1-logloss:0.348173
[483]       validation_0-logloss:0.190437       validation_1-logloss:0.348206
[484]       validation_0-logloss:0.190017       validation_1-logloss:0.34824
[485]       validation_0-logloss:0.189675       validation_1-logloss:0.348258
[486]       validation_0-logloss:0.18946       validation_1-logloss:0.348243
[487]       validation_0-logloss:0.189228       validation_1-logloss:0.348324
[488]       validation_0-logloss:0.189148       validation_1-logloss:0.348322
[489]       validation_0-logloss:0.188906       validation_1-logloss:0.34831
[490]       validation_0-logloss:0.188721       validation_1-logloss:0.348294
[491]       validation_0-logloss:0.188603       validation_1-logloss:0.348321
[492]       validation_0-logloss:0.188476       validation_1-logloss:0.348361
[493]       validation_0-logloss:0.188103       validation_1-logloss:0.348375
[494]       validation_0-logloss:0.187998       validation_1-logloss:0.348325
[495]       validation_0-logloss:0.18782       validation_1-logloss:0.348373
[496]       validation_0-logloss:0.187638       validation_1-logloss:0.348387
[497]       validation_0-logloss:0.187202       validation_1-logloss:0.348391
[498]       validation_0-logloss:0.186832       validation_1-logloss:0.34845
[499]       validation_0-logloss:0.186444       validation_1-logloss:0.348466
```

```
C:\Users\Aravindh\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:151: DeprecationWa
  if diff:
```

In [150]:
```
evals_result = clf.evals_result()
evals_result #to find the minimum of train and test log loss
```

Out[150]:
```
{'validation_0': {'logloss': [0.675799,
   0.659782,
   0.644819,
   0.63102,
   0.618081,
   0.60586,
   0.593978,
   0.583181,
   0.573151,
   0.563082,
   0.553658,
   0.544739,
   0.535957,
   0.527695,
   0.519933,
   0.512874,
   0.505733,
   0.499111,
   0.492718,
```

0.486802,
0.481216,
0.476064,
0.471005,
0.465951,
0.461357,
0.457184,
0.452877,
0.448587,
0.44486,
0.440993,
0.437269,
0.433712,
0.430567,
0.427338,
0.424525,
0.421527,
0.418593,
0.41583,
0.413288,
0.410857,
0.408576,
0.406378,
0.404217,
0.402067,
0.40002,
0.398158,
0.396512,
0.39483,
0.393185,
0.391695,
0.3901,
0.388456,
0.386739,
0.385248,
0.383679,
0.382345,
0.380999,
0.379849,
0.37834,
0.37708,
0.375942,
0.374597,
0.373313,
0.372178,
0.371198,
0.37024,
0.369087,

```
0.368103,
0.367002,
0.366151,
0.365125,
0.364098,
0.363106,
0.362127,
0.3614,
0.360682,
0.35992,
0.359035,
0.358422,
0.3576,
0.356787,
0.356104,
0.355308,
0.354565,
0.353908,
0.353176,
0.352458,
0.351649,
0.350929,
0.350266,
0.349582,
0.348926,
0.348296,
0.347668,
0.347088,
0.34641,
0.34577,
0.345131,
0.344552,
0.343952,
0.343445,
0.342867,
0.342324,
0.341673,
0.340966,
0.340365,
0.339855,
0.339368,
0.338826,
0.338294,
0.337828,
0.337249,
0.336788,
0.336001,
0.335512,
```

```
0.33498,
0.334648,
0.333687,
0.332763,
0.332338,
0.331781,
0.331279,
0.330806,
0.330274,
0.329939,
0.329551,
0.328719,
0.328276,
0.327873,
0.327107,
0.326671,
0.326234,
0.3253,
0.325,
0.324518,
0.324084,
0.323541,
0.322585,
0.322207,
0.321577,
0.320828,
0.320133,
0.319344,
0.318631,
0.31827,
0.317529,
0.317079,
0.3167,
0.31631,
0.315706,
0.314899,
0.314288,
0.313813,
0.313258,
0.312909,
0.312613,
0.312256,
0.311699,
0.310945,
0.310088,
0.309616,
0.308916,
0.308324,
```

0.307955,
0.307444,
0.307046,
0.30655,
0.305868,
0.305587,
0.305155,
0.304596,
0.303925,
0.303313,
0.302673,
0.302465,
0.302181,
0.30173,
0.301357,
0.300717,
0.300075,
0.29985,
0.299187,
0.298542,
0.297977,
0.297374,
0.297145,
0.296848,
0.296295,
0.295965,
0.295457,
0.295042,
0.294612,
0.294244,
0.293933,
0.293437,
0.293234,
0.292649,
0.29197,
0.291617,
0.291038,
0.290674,
0.290411,
0.289824,
0.289472,
0.28922,
0.288669,
0.28847,
0.287852,
0.28733,
0.286705,
0.286206,

```
0.285523,
0.284899,
0.284656,
0.284256,
0.283998,
0.283649,
0.283249,
0.283075,
0.282595,
0.28217,
0.281613,
0.281245,
0.280709,
0.28012,
0.279945,
0.279718,
0.279508,
0.279082,
0.278539,
0.277981,
0.277349,
0.277057,
0.276713,
0.276318,
0.275813,
0.275371,
0.275064,
0.274649,
0.274258,
0.273876,
0.273646,
0.273433,
0.273148,
0.272615,
0.272196,
0.271628,
0.271066,
0.270699,
0.270296,
0.269992,
0.269642,
0.269296,
0.268647,
0.268138,
0.267579,
0.267149,
0.266619,
0.266057,
```

```
0.265459,
0.26518,
0.265059,
0.264753,
0.264443,
0.263896,
0.26364,
0.263436,
0.262812,
0.26248,
0.262041,
0.261669,
0.261517,
0.260993,
0.260622,
0.260167,
0.25954,
0.259152,
0.258598,
0.258153,
0.257811,
0.257343,
0.256787,
0.256522,
0.256338,
0.256203,
0.255643,
0.255159,
0.254507,
0.25417,
0.253894,
0.253367,
0.252776,
0.252607,
0.252277,
0.251839,
0.251372,
0.251008,
0.250765,
0.250298,
0.250039,
0.249592,
0.249387,
0.249247,
0.248901,
0.248306,
0.247914,
0.247739,
```

0.2474,
0.247218,
0.246851,
0.246608,
0.246083,
0.245889,
0.245393,
0.244845,
0.244489,
0.244172,
0.243622,
0.243173,
0.242839,
0.2423,
0.241883,
0.241691,
0.241153,
0.240859,
0.240461,
0.239911,
0.239603,
0.239101,
0.23866,
0.238073,
0.237922,
0.237777,
0.237546,
0.237042,
0.236709,
0.236304,
0.236057,
0.235551,
0.235349,
0.235096,
0.234595,
0.234152,
0.233762,
0.233486,
0.232964,
0.232725,
0.232387,
0.231914,
0.231703,
0.231353,
0.230927,
0.230414,
0.229929,
0.229532,

0.229246,
0.229,
0.228759,
0.228583,
0.228104,
0.227861,
0.22751,
0.226967,
0.226461,
0.225984,
0.225702,
0.225281,
0.224891,
0.224428,
0.224164,
0.223929,
0.223733,
0.223232,
0.222704,
0.222452,
0.22223,
0.222046,
0.221656,
0.221203,
0.220795,
0.220472,
0.2201,
0.219644,
0.219232,
0.219069,
0.218723,
0.21856,
0.218158,
0.217734,
0.217291,
0.216967,
0.216679,
0.216526,
0.216107,
0.215832,
0.215622,
0.215193,
0.214887,
0.214523,
0.214083,
0.213844,
0.213452,
0.213023,

```
0.212741,
0.212433,
0.212113,
0.211774,
0.211441,
0.211011,
0.210542,
0.210426,
0.210004,
0.209558,
0.209345,
0.209165,
0.209043,
0.208617,
0.208351,
0.207852,
0.20761,
0.20744,
0.207108,
0.206683,
0.206568,
0.206195,
0.205933,
0.205846,
0.205646,
0.205183,
0.20506,
0.20481,
0.204522,
0.204408,
0.204066,
0.203729,
0.203347,
0.203066,
0.202806,
0.202568,
0.202165,
0.201757,
0.201406,
0.201146,
0.201031,
0.20084,
0.200404,
0.200054,
0.199589,
0.199241,
0.198913,
0.198463,
```

0.198271,
0.198138,
0.197964,
0.197869,
0.197674,
0.197489,
0.197374,
0.197205,
0.197097,
0.196715,
0.196359,
0.196131,
0.195808,
0.195373,
0.19502,
0.194655,
0.194546,
0.194351,
0.194037,
0.193651,
0.193414,
0.193254,
0.193129,
0.192776,
0.192613,
0.192407,
0.192023,
0.191664,
0.191458,
0.191247,
0.190856,
0.190652,
0.190437,
0.190017,
0.189675,
0.18946,
0.189228,
0.189148,
0.188906,
0.188721,
0.188603,
0.188476,
0.188103,
0.187998,
0.18782,
0.187638,
0.187202,
0.186832,

```
  0.186444]},
 'validation_1': {'logloss': [0.675654,
  0.65957,
  0.644594,
  0.630677,
  0.617753,
  0.605476,
  0.593744,
  0.582951,
  0.57306,
  0.563247,
  0.553941,
  0.545208,
  0.53656,
  0.528503,
  0.520991,
  0.514056,
  0.507205,
  0.500878,
  0.494873,
  0.489126,
  0.483682,
  0.478764,
  0.47381,
  0.469016,
  0.464629,
  0.460796,
  0.456719,
  0.452839,
  0.449237,
  0.445717,
  0.442252,
  0.439118,
  0.436246,
  0.433273,
  0.430703,
  0.427926,
  0.425227,
  0.422759,
  0.420414,
  0.418059,
  0.415911,
  0.413986,
  0.412096,
  0.41021,
  0.408399,
  0.406845,
  0.405344,
```

0.403934,
0.402533,
0.401256,
0.399929,
0.398514,
0.397229,
0.396028,
0.394772,
0.39373,
0.392652,
0.391698,
0.390412,
0.389389,
0.388498,
0.387448,
0.386459,
0.38564,
0.385055,
0.384341,
0.383567,
0.382868,
0.382174,
0.381544,
0.380814,
0.380114,
0.379402,
0.378774,
0.378244,
0.377748,
0.377342,
0.376829,
0.376382,
0.37584,
0.375333,
0.375035,
0.374634,
0.374389,
0.374047,
0.373588,
0.373253,
0.372826,
0.372453,
0.372218,
0.371842,
0.371585,
0.371432,
0.370949,
0.370576,

0.370252,
0.369953,
0.369456,
0.3691,
0.368768,
0.368657,
0.368475,
0.368267,
0.367976,
0.367607,
0.367308,
0.367134,
0.366878,
0.3665,
0.366342,
0.366162,
0.365903,
0.365571,
0.365109,
0.36495,
0.364797,
0.364671,
0.364319,
0.36399,
0.363882,
0.363648,
0.363601,
0.36343,
0.363285,
0.363078,
0.362931,
0.362661,
0.362391,
0.362324,
0.362139,
0.361896,
0.361821,
0.361622,
0.361492,
0.361386,
0.361222,
0.360917,
0.360756,
0.360788,
0.360605,
0.360346,
0.360063,
0.360004,

0.359839,
0.359698,
0.359657,
0.359623,
0.359536,
0.359339,
0.359146,
0.358902,
0.358871,
0.358817,
0.358785,
0.358657,
0.358591,
0.358399,
0.358215,
0.358029,
0.357859,
0.357684,
0.357337,
0.357263,
0.357145,
0.357117,
0.357144,
0.357145,
0.357147,
0.35701,
0.356925,
0.356926,
0.356647,
0.35664,
0.3565,
0.356468,
0.356352,
0.356346,
0.356296,
0.356235,
0.356178,
0.356068,
0.355929,
0.355835,
0.355794,
0.355722,
0.355748,
0.355652,
0.35556,
0.355517,
0.355452,
0.355344,

0.355245,
0.355124,
0.355124,
0.355065,
0.355071,
0.354908,
0.354854,
0.354834,
0.354833,
0.354681,
0.354586,
0.354572,
0.354487,
0.354512,
0.354451,
0.354478,
0.354495,
0.354399,
0.354314,
0.354183,
0.353979,
0.353886,
0.353834,
0.353792,
0.353836,
0.353785,
0.353782,
0.35383,
0.353736,
0.353668,
0.353502,
0.353507,
0.353505,
0.353501,
0.353487,
0.353395,
0.353351,
0.353354,
0.353163,
0.353212,
0.3531,
0.353001,
0.352907,
0.35283,
0.352796,
0.352748,
0.352751,
0.35282,

0.352826,
0.352813,
0.35285,
0.352814,
0.35285,
0.352657,
0.35248,
0.352556,
0.352538,
0.352478,
0.352501,
0.352431,
0.352369,
0.352384,
0.352279,
0.352129,
0.352118,
0.352061,
0.351948,
0.351999,
0.351868,
0.351834,
0.351888,
0.351798,
0.351829,
0.351747,
0.351741,
0.351709,
0.351504,
0.351417,
0.351483,
0.351496,
0.351508,
0.351513,
0.351452,
0.351489,
0.351442,
0.3514,
0.351317,
0.351236,
0.351363,
0.351279,
0.351328,
0.351266,
0.351197,
0.351188,
0.351084,
0.351168,

```
0.351097,
0.351112,
0.351096,
0.351134,
0.351022,
0.351018,
0.350991,
0.351033,
0.350834,
0.350875,
0.350919,
0.350847,
0.350827,
0.350713,
0.350725,
0.350694,
0.350718,
0.350704,
0.350723,
0.350733,
0.35066,
0.350683,
0.350653,
0.350634,
0.350612,
0.35064,
0.350519,
0.350366,
0.350326,
0.350414,
0.350459,
0.350451,
0.350444,
0.350338,
0.350378,
0.350381,
0.350346,
0.3504,
0.350393,
0.350407,
0.350363,
0.350347,
0.350398,
0.350448,
0.350365,
0.350361,
0.350336,
0.35035,
```

0.350326,
0.350306,
0.350327,
0.350316,
0.350296,
0.350368,
0.350356,
0.350383,
0.350319,
0.350279,
0.350467,
0.350496,
0.350491,
0.350525,
0.350538,
0.350451,
0.350419,
0.350375,
0.350403,
0.350425,
0.350369,
0.350374,
0.350266,
0.350206,
0.350131,
0.350126,
0.350146,
0.350114,
0.350197,
0.350091,
0.350064,
0.350022,
0.349969,
0.350032,
0.349999,
0.350003,
0.349908,
0.349922,
0.349805,
0.34988,
0.349862,
0.349921,
0.349903,
0.349797,
0.349756,
0.349656,
0.349539,
0.349501,

```
0.349468,
0.349399,
0.349359,
0.349341,
0.34936,
0.349312,
0.34925,
0.349251,
0.349296,
0.349267,
0.34926,
0.349284,
0.349273,
0.349211,
0.349152,
0.349153,
0.34909,
0.349072,
0.348958,
0.349021,
0.348969,
0.348924,
0.348982,
0.349023,
0.348955,
0.348989,
0.348955,
0.348927,
0.348851,
0.34875,
0.348707,
0.348658,
0.348635,
0.348722,
0.348771,
0.348763,
0.348784,
0.348795,
0.348836,
0.348772,
0.348825,
0.348801,
0.348729,
0.348725,
0.34877,
0.348667,
0.348705,
0.348671,
```

0.348784,
0.34869,
0.34861,
0.348599,
0.348534,
0.348512,
0.348527,
0.348547,
0.348535,
0.348465,
0.348484,
0.34842,
0.348411,
0.348389,
0.348266,
0.348241,
0.348232,
0.348132,
0.348145,
0.348187,
0.348174,
0.348189,
0.348242,
0.348222,
0.348234,
0.348304,
0.348336,
0.348361,
0.348346,
0.348278,
0.348289,
0.3483,
0.34839,
0.348423,
0.348316,
0.348217,
0.348213,
0.348201,
0.348189,
0.348136,
0.348134,
0.348104,
0.348095,
0.348114,
0.348108,
0.348168,
0.348244,
0.348128,

```
0.348184,
0.348173,
0.348184,
0.348173,
0.348206,
0.34824,
0.348258,
0.348243,
0.348324,
0.348322,
0.34831,
0.348294,
0.348321,
0.348361,
0.348375,
0.348325,
0.348373,
0.348387,
0.348391,
0.34845,
0.348466]}}
```

3 Results :

```
In [147]: from prettytable import PrettyTable
          x = PrettyTable()
          x.field_names = ["MODEL", "Hyperparameters", "Test-LOG-LOSS", "Train-log-loss"]

          #TFIDFW2V
          x.add_row(['TFIDFW2V with Random Model', 'Random values', 0.88, '-'])
          x.add_row(['--'*5,'--'*5,'--'*5,'--'*5])
          x.add_row(['TFIDFW2V with Logistic Regression', 'Alpha=1', 0.51, 0.50])
          x.add_row(['--'*5,'--'*5,'--'*5,'--'*5])
          x.add_row(['TFIDFW2V with Linear SVM', 'Alpha=0.0001', 0.48, 0.49])
          x.add_row(['--'*5,'--'*5,'--'*5,'--'*5])
          x.add_row(['TFIDFW2V with XGBOOST', 'n_estimators = 500\n Tree-max_depth = 5\n Learni
          x.add_row(['--'*5,'-'*8,'-'*8,'-'*5])
          print(x)
```

```
+-----------------------------------+-----------------------+---------------+----------------+
|               MODEL               |    Hyperparameters    | Test-LOG-LOSS | Train-log-loss |
+-----------------------------------+-----------------------+---------------+----------------+
|     TFIDFW2V with Random Model    |     Random values     |      0.88     |       -        |
|             ----------            |       ----------      |   ----------  |   ----------   |
| TFIDFW2V with Logistic Regression |        Alpha=1        |      0.51     |      0.5       |
|             ----------            |       ----------      |   ----------  |   ----------   |
|      TFIDFW2V with Linear SVM     |      Alpha=0.0001     |      0.48     |      0.49      |
```

----------	----------	----------	----------
TFIDFW2V with XGBOOST	n_estimators = 500 Tree-max_depth = 5 Learning Rate = 0.04	0.35	0.2
----------	--------	--------	-----

OBSERVATION

Quora Question pair simillarity was trained with 100k points & 20k points with XGboost coz of computation constraints

1. Quora Question pair simmilarity is trained and tested with TFIDFW2V and the results were good.

2. we get a minimal test log loss of 0.2 with GBDT. even when trained with only 20000 points

3. there are chances that XGBoost may perform very well given that we can take whole data into account.