Quora Question Simillarity with TFIDF

September 25, 2018

1 QUORA QUESTION PAIR SIMILARITY WITH TFIDF

```
In [66]: import pandas as pd
         import matplotlib.pyplot as plt
         import re
         import time
         import warnings
         import numpy as np
         from nltk.corpus import stopwords
         from sklearn.preprocessing import normalize
         from sklearn.feature_extraction.text import CountVectorizer
         from sklearn.feature_extraction.text import TfidfVectorizer
         warnings.filterwarnings("ignore")
         import sys
         import os
         import pandas as pd
         import numpy as np
         from sklearn.model_selection import train_test_split
         from scipy.sparse import hstack
         from xgboost import XGBClassifier
         from sklearn.model_selection import RandomizedSearchCV
         from sklearn.linear_model import SGDClassifier
         from sklearn.calibration import CalibratedClassifierCV
         from sklearn.metrics.classification import accuracy_score, log_loss
         from sklearn.metrics import confusion_matrix
         import seaborn as sns
```

2 1. TRAIN-TEST SPLIT

1. seperate the text data
2. Split the data into train and test
3. Seperate the text data and vectorize it
4. Then join it with data that has advanced features

In [2]:
```
# avoid decoding problems
df = pd.read_csv("train.csv")

# encode questions to unicode
# https://stackoverflow.com/a/6812069
# ----------------- python 2 ---------------------
# df['question1'] = df['question1'].apply(lambda x: unicode(str(x),"utf-8"))
# df['question2'] = df['question2'].apply(lambda x: unicode(str(x),"utf-8"))
# ----------------- python 3 ---------------------
df['question1'] = df['question1'].apply(lambda x: str(x))
df['question2'] = df['question2'].apply(lambda x: str(x))
```

In [3]:
```
df.columns
```

Out[3]:
```
Index(['id', 'qid1', 'qid2', 'question1', 'question2', 'is_duplicate'], dtype='object'
```

In [4]:
```
#prepro_features_train.csv (Simple Preprocessing Feartures)
#nlp_features_train.csv (NLP Features)
if os.path.isfile('nlp_features_train.csv'):
    dfnlp = pd.read_csv("nlp_features_train.csv",encoding='latin-1')
else:
    print("download nlp_features_train.csv from drive or run previous notebook")

if os.path.isfile('df_fe_without_preprocessing_train.csv'):
    dfppro = pd.read_csv("df_fe_without_preprocessing_train.csv",encoding='latin-1')
else:
    print("download df_fe_without_preprocessing_train.csv from drive or run previous no
```

In [5]:
```
df1 = dfnlp.drop(['qid1','qid2','question1','question2', 'is_duplicate'],axis=1)
df2 = dfppro.drop(['qid1','qid2','question1','question2','is_duplicate'],axis=1)
df3 = dfnlp[['id', 'question1', 'question2']]
y_true = dfnlp.is_duplicate
```

In [6]:
```
print(df1.columns)
print(df1.shape)
df1.head()
```

```
Index(['id', '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'],
      dtype='object')
(404290, 16)
```

Out[6]:
```
   id   cwc_min   cwc_max   csc_min   csc_max   ctc_min   ctc_max  \
0   0  0.999980  0.833319  0.999983  0.999983  0.916659  0.785709
1   1  0.799984  0.399996  0.749981  0.599988  0.699993  0.466664
2   2  0.399992  0.333328  0.399992  0.249997  0.399996  0.285712
```

```
3   3  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
4   4  0.399992  0.199998  0.999950  0.666644  0.571420  0.307690

   last_word_eq  first_word_eq  abs_len_diff  mean_len  token_set_ratio  \
0           0.0            1.0           2.0      13.0              100
1           0.0            1.0           5.0      12.5               86
2           0.0            1.0           4.0      12.0               66
3           0.0            0.0           2.0      12.0               36
4           0.0            1.0           6.0      10.0               67

   token_sort_ratio  fuzz_ratio  fuzz_partial_ratio  longest_substr_ratio
0                93          93                 100              0.982759
1                63          66                  75              0.596154
2                66          54                  54              0.166667
3                36          35                  40              0.039216
4                47          46                  56              0.175000
```

```
In [7]: print(df2.columns)
        print(df2.shape)
        df2.head()
```

```
Index(['id', 'freq_qid1', 'freq_qid2', 'q1len', 'q2len', 'q1_n_words',
       'q2_n_words', 'word_Common', 'word_Total', 'word_share', 'freq_q1+q2',
       'freq_q1-q2'],
      dtype='object')
(404290, 12)
```

```
Out[7]:    id  freq_qid1  freq_qid2  q1len  q2len  q1_n_words  q2_n_words  \
        0   0          1          1     66     57          14          12
        1   1          4          1     51     88           8          13
        2   2          1          1     73     59          14          10
        3   3          1          1     50     65          11           9
        4   4          3          1     76     39          13           7

           word_Common  word_Total  word_share  freq_q1+q2  freq_q1-q2
        0         10.0        23.0    0.434783           2           0
        1          4.0        20.0    0.200000           5           3
        2          4.0        24.0    0.166667           2           0
        3          0.0        19.0    0.000000           2           0
        4          2.0        20.0    0.100000           4           2
```

```
In [27]: df3 = df3.fillna(' ')
         print(df3.columns)
         print(df3.shape)
         df3.head()
```

```
Index(['id', 'question1', 'question2'], dtype='object')
(404290, 3)
```

Out[27]:
```
   id                                          question1  \
0   0  what is the step by step guide to invest in sh...
1   1  what is the story of kohinoor  koh i noor  dia...
2   2  how can i increase the speed of my internet co...
3   3  why am i mentally very lonely  how can i solve...
4   4  which one dissolve in water quikly sugar  salt...

                                           question2
0  what is the step by step guide to invest in sh...
1  what would happen if the indian government sto...
2  how can internet speed be increased by hacking...
3  find the remainder when  math 23  24   math  i...
4            which fish would survive in salt water
```

In [28]:
```
df_text = pd.DataFrame()
df_text['Text'] = df3.question1 + ' ' + df3.question2
```

In [29]:
```
print(df_text.columns)
print(df_text.shape)
df_text.head()
```

```
Index(['Text'], dtype='object')
(404290, 1)
```

Out[29]:
```
                                                Text
0  what is the step by step guide to invest in sh...
1  what is the story of kohinoor  koh i noor  dia...
2  how can i increase the speed of my internet co...
3  why am i mentally very lonely  how can i solve...
4  which one dissolve in water quikly sugar  salt...
```

In [30]:
```
df2['id']=df1['id']
df_text['id']=df1['id']
df_temp  = df1.merge(df2, on='id',how='left')
```

In [31]:
```
df_temp = df_temp.merge(df_text, on='id', how='left')
print(df_temp.columns)
df_temp.head()
```

```
Index(['id', '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', 'q1_n_words', 'q2_n_words', 'word_Common',
       'word_Total', 'word_share', 'freq_q1+q2', 'freq_q1-q2', 'Text'],
      dtype='object')
```

```
Out[31]:    id   cwc_min   cwc_max   csc_min   csc_max   ctc_min   ctc_max  \
         0   0  0.999980  0.833319  0.999983  0.999983  0.916659  0.785709
         1   1  0.799984  0.399996  0.749981  0.599988  0.699993  0.466664
         2   2  0.399992  0.333328  0.399992  0.249997  0.399996  0.285712
         3   3  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
         4   4  0.399992  0.199998  0.999950  0.666644  0.571420  0.307690

            last_word_eq  first_word_eq  abs_len_diff  \
         0           0.0            1.0           2.0
         1           0.0            1.0           5.0
         2           0.0            1.0           4.0
         3           0.0            0.0           2.0
         4           0.0            1.0           6.0

                                ...                           q1len  q2len  \
         0                      ...                              66     57
         1                      ...                              51     88
         2                      ...                              73     59
         3                      ...                              50     65
         4                      ...                              76     39

            q1_n_words  q2_n_words  word_Common  word_Total  word_share  freq_q1+q2  \
         0          14          12         10.0        23.0    0.434783           2
         1           8          13          4.0        20.0    0.200000           5
         2          14          10          4.0        24.0    0.166667           2
         3          11           9          0.0        19.0    0.000000           2
         4          13           7          2.0        20.0    0.100000           4

            freq_q1-q2                                               Text
         0           0  what is the step by step guide to invest in sh...
         1           3  what is the story of kohinoor  koh i noor  dia...
         2           0  how can i increase the speed of my internet co...
         3           0  why am i mentally very lonely  how can i solve...
         4           2  which one dissolve in water quikly sugar  salt...

         [5 rows x 28 columns]
```

In [32]:
```
y_true = dfnlp.is_duplicate
```

In [33]:
```
X = df_temp.head(100000)
y = y_true.head(100000)
```

In [34]:
```
print(X.shape)
print(y.shape)
```

```
(100000, 28)
(100000,)
```

```
In [36]: #train-test splitting
         X_train_temp, X_test_temp, y_train, y_test = train_test_split(X, y , test_size=0.2, ra

In [37]: print(X_train_temp.shape)
         print(y_train.shape)
         print(X_test_temp.shape)
         print(y_test.shape)

(80000, 28)
(80000,)
(20000, 28)
(20000,)
```

3 2. TFIDF VECTORIZING TEXT DATA

```
In [35]: vect = TfidfVectorizer()

In [40]: tfidf_text_X_train = vect.fit_transform(X_train_temp['Text'])

In [48]: tfidf_text_X_train

Out[48]: <80000x41401 sparse matrix of type '<class 'numpy.float64'>'
                 with 1233358 stored elements in Compressed Sparse Row format>

In [42]: tfidf_text_X_test = vect.transform(X_test_temp['Text'])

In [49]: tfidf_text_X_test

Out[49]: <20000x41401 sparse matrix of type '<class 'numpy.float64'>'
                 with 304774 stored elements in Compressed Sparse Row format>

In [45]: X_train_1 = X_train_temp.drop('Text', axis=1)
         X_test_1 = X_test_temp.drop('Text', axis=1)

In [92]: X_train_1.shape

Out[92]: (80000, 27)

In [93]: X_test_1.shape

Out[93]: (20000, 27)

In [51]: #stacking tfidf vectorized text data & other advanvanced nlp features like ratio
         X_train = hstack((X_train_1,tfidf_text_X_train))
         X_test = hstack((X_test_1,tfidf_text_X_test))

In [95]: print('X_train :',X_train.shape)
         print('X_test :',X_test.shape)
```

```
X_train : (80000, 41428)
X_test : (20000, 41428)
```

```
In [57]: # Standardizing the data
         from sklearn.preprocessing import StandardScaler
         scale = StandardScaler(with_mean=False)
         X_train = scale.fit_transform(X_train)
         X_test = scale.transform(X_test)
```

4 3. Machine Learning Models

```
In [63]: # 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)
```

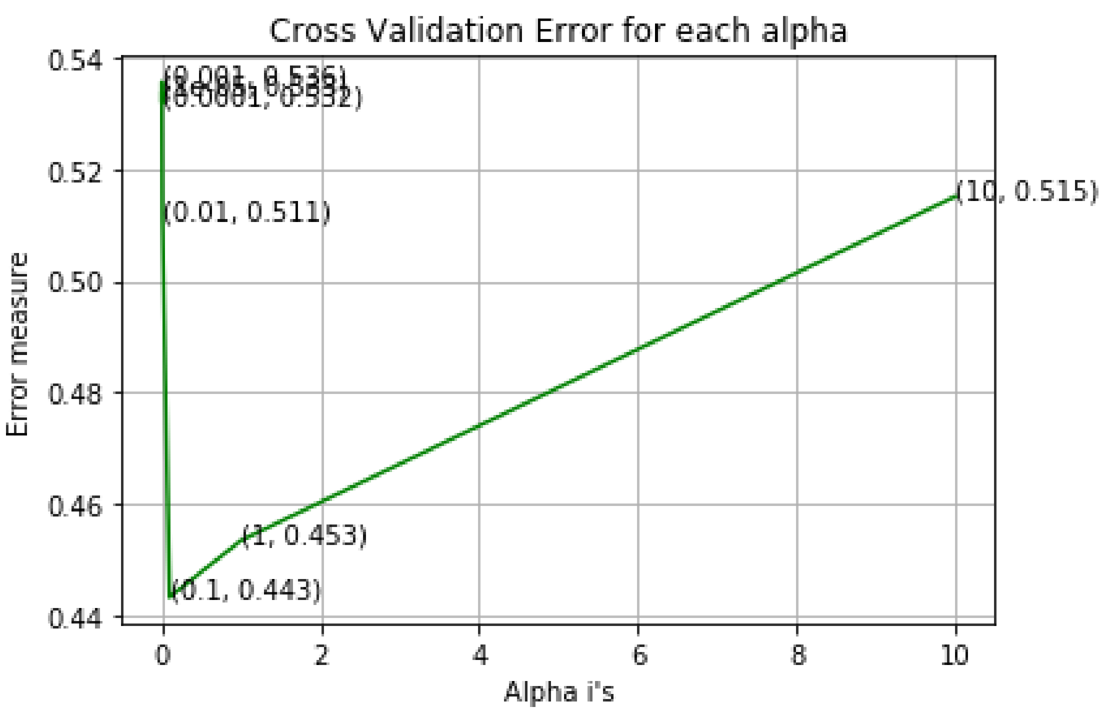
```
    sns.heatmap(C, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=l
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Confusion matrix")

    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()
```

5 3.1 Logistic Regression with TFIDF

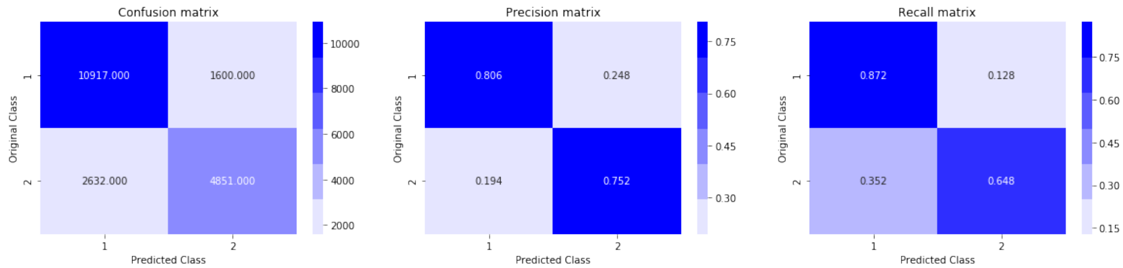
```
In [64]: alpha = [10 ** x for x in range(-5, 2)] # hyperparam for SGD classifier.

         log_error_array=[]
         for i in alpha:
             clf = SGDClassifier(alpha=i, penalty='l2', loss='log', random_state=42)
             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, 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=4
         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: 0.5334702978650625
For values of alpha =  0.0001 The log loss is: 0.5318544476928626
For values of alpha =  0.001 The log loss is: 0.5356305533461118
For values of alpha =  0.01 The log loss is: 0.5109246376180082
For values of alpha =  0.1 The log loss is: 0.44328160929829463
For values of alpha =  1 The log loss is: 0.4534260664278418
For values of alpha =  10 The log loss is: 0.5150349182643443
```



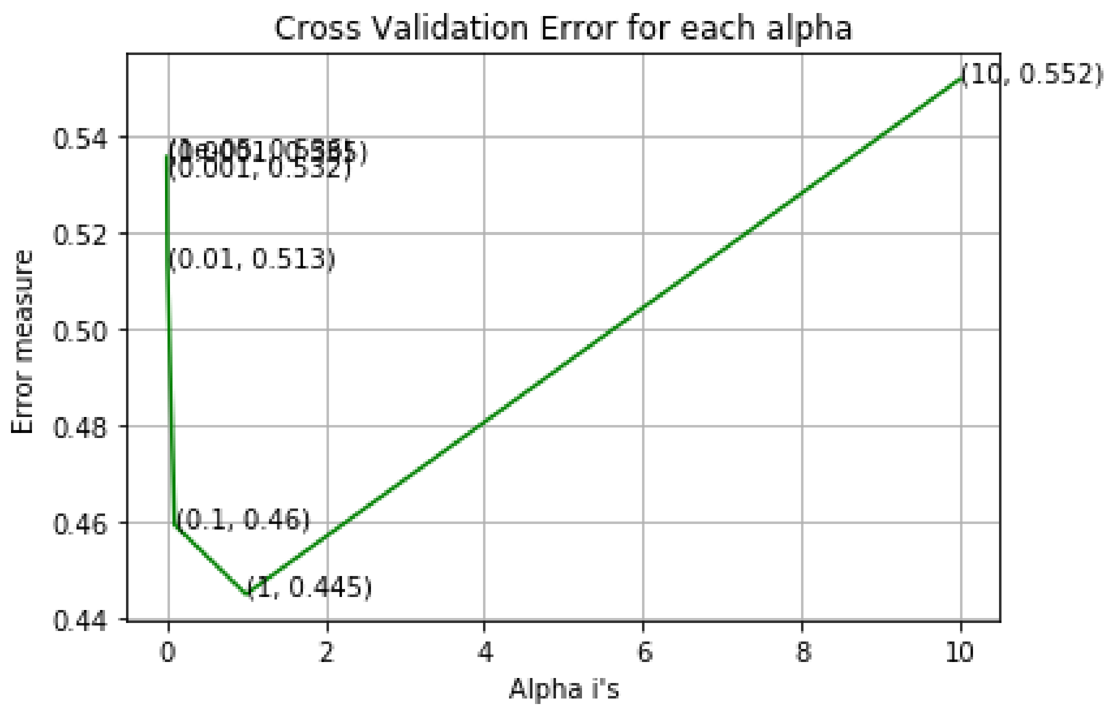

```
For values of best alpha =  0.1 The train log loss is: 0.311506376997774
For values of best alpha =  0.1 The test log loss is: 0.44328160929829463
Total number of data points : 20000
```



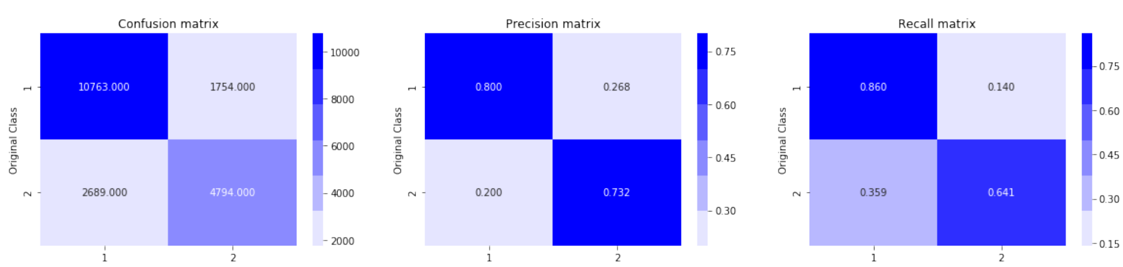
6 3.2 Linear SVM with TFIDF

```
In [65]: 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='hinge', random_state=42)
             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, 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=4
         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: 0.5356218417894919
For values of alpha =  0.0001 The log loss is: 0.5351194839494243
For values of alpha =  0.001 The log loss is: 0.532300638438738
For values of alpha =  0.01 The log loss is: 0.5132107309888126
For values of alpha =  0.1 The log loss is: 0.45950008696797395
For values of alpha =  1 The log loss is: 0.4451276981150781
For values of alpha =  10 The log loss is: 0.5517164546767706
```




```
For values of best alpha =  1 The train log loss is: 0.36867697250115267
For values of best alpha =  1 The test log loss is: 0.4534260664278418
Total number of data points : 20000
```

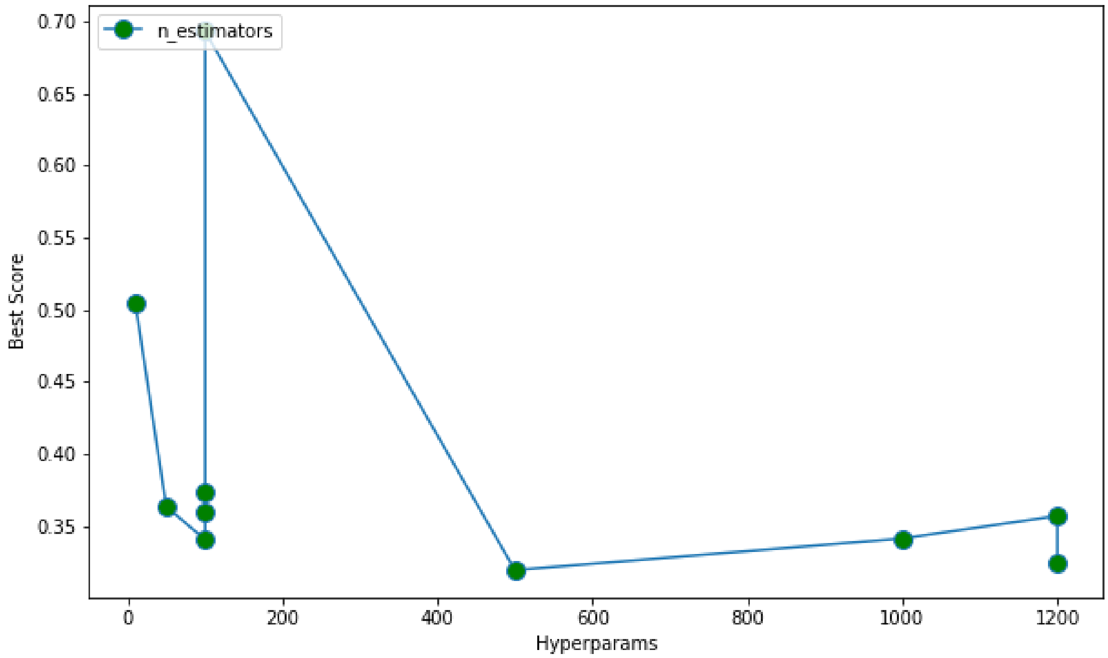


7 3.3 XGBoost with TFIDF

```
In [85]: 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('LOG-LOSS:', 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 [86]: XGB_best_params(X_train, y_train)
```

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


[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed:  8.7min
[Parallel(n_jobs=-1)]: Done  50 out of  50 | elapsed: 10.6min finished


LOG-LOSS: {'n_estimators': 500, 'max_depth': 4, 'learning_rate': 0.2}
best Score: -0.31938490364894384
```



In [97]:
```
#lets viuallise the eval result
clf = XGBClassifier(learning_rate=0.2, n_estimators=500, max_depth=4, njobs=-1)
clf.fit(X_train, y_train,
        eval_set=[(X_train, y_train), (X_test, y_test)],
        eval_metric='logloss',
        verbose=True)
y_pred = clf.predict(X_test)
fi = clf.feature_importances_
```

```
[0]	validation_0-logloss:0.617872	validation_1-logloss:0.617761
[1]	validation_0-logloss:0.563976	validation_1-logloss:0.564293
[2]	validation_0-logloss:0.523963	validation_1-logloss:0.524828
[3]	validation_0-logloss:0.495163	validation_1-logloss:0.495894
[4]	validation_0-logloss:0.471689	validation_1-logloss:0.472717
[5]	validation_0-logloss:0.454709	validation_1-logloss:0.455807
[6]	validation_0-logloss:0.439191	validation_1-logloss:0.44046
[7]	validation_0-logloss:0.428588	validation_1-logloss:0.430075
[8]	validation_0-logloss:0.418899	validation_1-logloss:0.420731
[9]	validation_0-logloss:0.410939	validation_1-logloss:0.412635
[10]	validation_0-logloss:0.404258	validation_1-logloss:0.406137
[11]	validation_0-logloss:0.398847	validation_1-logloss:0.401047
[12]	validation_0-logloss:0.394228	validation_1-logloss:0.39653
[13]	validation_0-logloss:0.390694	validation_1-logloss:0.393188
[14]	validation_0-logloss:0.38677	validation_1-logloss:0.389431
[15]	validation_0-logloss:0.383814	validation_1-logloss:0.386723
```

```
[16]	validation_0-logloss:0.381428	validation_1-logloss:0.384464
[17]	validation_0-logloss:0.378998	validation_1-logloss:0.382156
[18]	validation_0-logloss:0.37682	validation_1-logloss:0.380135
[19]	validation_0-logloss:0.374589	validation_1-logloss:0.378289
[20]	validation_0-logloss:0.373194	validation_1-logloss:0.377164
[21]	validation_0-logloss:0.37152	validation_1-logloss:0.375749
[22]	validation_0-logloss:0.369454	validation_1-logloss:0.373893
[23]	validation_0-logloss:0.367397	validation_1-logloss:0.372023
[24]	validation_0-logloss:0.366316	validation_1-logloss:0.371045
[25]	validation_0-logloss:0.365188	validation_1-logloss:0.370106
[26]	validation_0-logloss:0.363532	validation_1-logloss:0.368761
[27]	validation_0-logloss:0.362404	validation_1-logloss:0.367541
[28]	validation_0-logloss:0.361172	validation_1-logloss:0.366557
[29]	validation_0-logloss:0.360184	validation_1-logloss:0.365609
[30]	validation_0-logloss:0.358872	validation_1-logloss:0.364579
[31]	validation_0-logloss:0.358125	validation_1-logloss:0.363971
[32]	validation_0-logloss:0.356853	validation_1-logloss:0.362763
[33]	validation_0-logloss:0.355767	validation_1-logloss:0.361741
[34]	validation_0-logloss:0.35469	validation_1-logloss:0.360849
[35]	validation_0-logloss:0.353936	validation_1-logloss:0.360312
[36]	validation_0-logloss:0.353229	validation_1-logloss:0.359842
[37]	validation_0-logloss:0.352221	validation_1-logloss:0.359076
[38]	validation_0-logloss:0.351635	validation_1-logloss:0.358406
[39]	validation_0-logloss:0.350974	validation_1-logloss:0.357781
[40]	validation_0-logloss:0.350158	validation_1-logloss:0.357044
[41]	validation_0-logloss:0.34962	validation_1-logloss:0.356794
[42]	validation_0-logloss:0.34912	validation_1-logloss:0.356417
[43]	validation_0-logloss:0.348537	validation_1-logloss:0.356089
[44]	validation_0-logloss:0.348075	validation_1-logloss:0.355701
[45]	validation_0-logloss:0.347597	validation_1-logloss:0.355408
[46]	validation_0-logloss:0.346515	validation_1-logloss:0.354522
[47]	validation_0-logloss:0.34525	validation_1-logloss:0.353357
[48]	validation_0-logloss:0.344648	validation_1-logloss:0.35293
[49]	validation_0-logloss:0.343946	validation_1-logloss:0.352419
[50]	validation_0-logloss:0.343468	validation_1-logloss:0.351955
[51]	validation_0-logloss:0.342783	validation_1-logloss:0.351457
[52]	validation_0-logloss:0.342283	validation_1-logloss:0.351113
[53]	validation_0-logloss:0.34188	validation_1-logloss:0.350854
[54]	validation_0-logloss:0.341409	validation_1-logloss:0.350479
[55]	validation_0-logloss:0.340814	validation_1-logloss:0.349984
[56]	validation_0-logloss:0.340345	validation_1-logloss:0.349701
[57]	validation_0-logloss:0.340032	validation_1-logloss:0.349533
[58]	validation_0-logloss:0.339492	validation_1-logloss:0.349127
[59]	validation_0-logloss:0.339047	validation_1-logloss:0.348811
[60]	validation_0-logloss:0.338741	validation_1-logloss:0.348597
[61]	validation_0-logloss:0.338247	validation_1-logloss:0.348234
[62]	validation_0-logloss:0.337965	validation_1-logloss:0.348052
[63]	validation_0-logloss:0.33763	validation_1-logloss:0.34782
```

```
[64]	validation_0-logloss:0.337228	validation_1-logloss:0.347646
[65]	validation_0-logloss:0.336939	validation_1-logloss:0.347499
[66]	validation_0-logloss:0.336505	validation_1-logloss:0.347193
[67]	validation_0-logloss:0.335784	validation_1-logloss:0.346729
[68]	validation_0-logloss:0.335348	validation_1-logloss:0.346375
[69]	validation_0-logloss:0.3349	validation_1-logloss:0.346193
[70]	validation_0-logloss:0.334457	validation_1-logloss:0.345937
[71]	validation_0-logloss:0.333813	validation_1-logloss:0.345343
[72]	validation_0-logloss:0.333389	validation_1-logloss:0.345023
[73]	validation_0-logloss:0.333105	validation_1-logloss:0.344875
[74]	validation_0-logloss:0.332808	validation_1-logloss:0.344714
[75]	validation_0-logloss:0.33243	validation_1-logloss:0.344459
[76]	validation_0-logloss:0.332069	validation_1-logloss:0.344191
[77]	validation_0-logloss:0.331078	validation_1-logloss:0.343429
[78]	validation_0-logloss:0.330828	validation_1-logloss:0.343224
[79]	validation_0-logloss:0.330546	validation_1-logloss:0.343065
[80]	validation_0-logloss:0.33031	validation_1-logloss:0.342908
[81]	validation_0-logloss:0.329988	validation_1-logloss:0.342763
[82]	validation_0-logloss:0.329601	validation_1-logloss:0.342549
[83]	validation_0-logloss:0.329236	validation_1-logloss:0.342345
[84]	validation_0-logloss:0.328991	validation_1-logloss:0.342197
[85]	validation_0-logloss:0.328749	validation_1-logloss:0.342021
[86]	validation_0-logloss:0.328329	validation_1-logloss:0.341781
[87]	validation_0-logloss:0.328098	validation_1-logloss:0.341688
[88]	validation_0-logloss:0.327903	validation_1-logloss:0.341627
[89]	validation_0-logloss:0.327564	validation_1-logloss:0.341463
[90]	validation_0-logloss:0.327352	validation_1-logloss:0.341334
[91]	validation_0-logloss:0.327054	validation_1-logloss:0.341184
[92]	validation_0-logloss:0.32681	validation_1-logloss:0.341027
[93]	validation_0-logloss:0.326527	validation_1-logloss:0.340844
[94]	validation_0-logloss:0.325962	validation_1-logloss:0.340367
[95]	validation_0-logloss:0.325663	validation_1-logloss:0.340246
[96]	validation_0-logloss:0.325474	validation_1-logloss:0.340197
[97]	validation_0-logloss:0.325235	validation_1-logloss:0.340119
[98]	validation_0-logloss:0.324908	validation_1-logloss:0.339912
[99]	validation_0-logloss:0.324608	validation_1-logloss:0.339794
[100]	validation_0-logloss:0.324388	validation_1-logloss:0.339683
[101]	validation_0-logloss:0.324061	validation_1-logloss:0.339528
[102]	validation_0-logloss:0.323766	validation_1-logloss:0.339303
[103]	validation_0-logloss:0.323449	validation_1-logloss:0.339202
[104]	validation_0-logloss:0.323172	validation_1-logloss:0.338957
[105]	validation_0-logloss:0.322794	validation_1-logloss:0.338672
[106]	validation_0-logloss:0.322213	validation_1-logloss:0.338166
[107]	validation_0-logloss:0.321074	validation_1-logloss:0.337427
[108]	validation_0-logloss:0.320358	validation_1-logloss:0.33686
[109]	validation_0-logloss:0.319776	validation_1-logloss:0.336587
[110]	validation_0-logloss:0.319565	validation_1-logloss:0.336576
[111]	validation_0-logloss:0.319351	validation_1-logloss:0.336552
```

```
[112]           validation_0-logloss:0.319132         validation_1-logloss:0.336416
[113]           validation_0-logloss:0.318919         validation_1-logloss:0.336312
[114]           validation_0-logloss:0.318748         validation_1-logloss:0.336269
[115]           validation_0-logloss:0.318569         validation_1-logloss:0.336261
[116]           validation_0-logloss:0.318416         validation_1-logloss:0.336172
[117]           validation_0-logloss:0.318196         validation_1-logloss:0.336048
[118]           validation_0-logloss:0.317874         validation_1-logloss:0.335862
[119]           validation_0-logloss:0.317452         validation_1-logloss:0.33566
[120]           validation_0-logloss:0.317157         validation_1-logloss:0.335468
[121]           validation_0-logloss:0.316856         validation_1-logloss:0.335332
[122]           validation_0-logloss:0.316668         validation_1-logloss:0.335257
[123]           validation_0-logloss:0.316181         validation_1-logloss:0.335079
[124]           validation_0-logloss:0.315922         validation_1-logloss:0.334838
[125]           validation_0-logloss:0.315637         validation_1-logloss:0.334534
[126]           validation_0-logloss:0.315412         validation_1-logloss:0.334429
[127]           validation_0-logloss:0.315163         validation_1-logloss:0.334372
[128]           validation_0-logloss:0.314934         validation_1-logloss:0.334264
[129]           validation_0-logloss:0.314788         validation_1-logloss:0.334248
[130]           validation_0-logloss:0.314466         validation_1-logloss:0.334112
[131]           validation_0-logloss:0.314231         validation_1-logloss:0.334038
[132]           validation_0-logloss:0.313985         validation_1-logloss:0.333899
[133]           validation_0-logloss:0.313769         validation_1-logloss:0.333796
[134]           validation_0-logloss:0.31361         validation_1-logloss:0.333837
[135]           validation_0-logloss:0.313431         validation_1-logloss:0.333828
[136]           validation_0-logloss:0.313198         validation_1-logloss:0.333757
[137]           validation_0-logloss:0.312979         validation_1-logloss:0.333617
[138]           validation_0-logloss:0.312733         validation_1-logloss:0.333519
[139]           validation_0-logloss:0.312598         validation_1-logloss:0.333433
[140]           validation_0-logloss:0.312099         validation_1-logloss:0.333149
[141]           validation_0-logloss:0.311879         validation_1-logloss:0.333077
[142]           validation_0-logloss:0.311735         validation_1-logloss:0.333011
[143]           validation_0-logloss:0.311456         validation_1-logloss:0.332985
[144]           validation_0-logloss:0.311242         validation_1-logloss:0.332894
[145]           validation_0-logloss:0.311055         validation_1-logloss:0.332806
[146]           validation_0-logloss:0.310851         validation_1-logloss:0.332736
[147]           validation_0-logloss:0.310609         validation_1-logloss:0.332728
[148]           validation_0-logloss:0.310395         validation_1-logloss:0.332606
[149]           validation_0-logloss:0.310225         validation_1-logloss:0.332509
[150]           validation_0-logloss:0.310104         validation_1-logloss:0.33246
[151]           validation_0-logloss:0.309809         validation_1-logloss:0.332205
[152]           validation_0-logloss:0.309686         validation_1-logloss:0.33214
[153]           validation_0-logloss:0.309283         validation_1-logloss:0.331876
[154]           validation_0-logloss:0.309138         validation_1-logloss:0.33187
[155]           validation_0-logloss:0.309008         validation_1-logloss:0.331805
[156]           validation_0-logloss:0.308862         validation_1-logloss:0.331768
[157]           validation_0-logloss:0.308709         validation_1-logloss:0.331692
[158]           validation_0-logloss:0.308466         validation_1-logloss:0.331578
[159]           validation_0-logloss:0.308346         validation_1-logloss:0.331623
```

```
[160]	validation_0-logloss:0.30804	validation_1-logloss:0.331437
[161]	validation_0-logloss:0.307892	validation_1-logloss:0.331378
[162]	validation_0-logloss:0.307679	validation_1-logloss:0.331294
[163]	validation_0-logloss:0.307472	validation_1-logloss:0.331244
[164]	validation_0-logloss:0.307309	validation_1-logloss:0.331157
[165]	validation_0-logloss:0.306936	validation_1-logloss:0.330924
[166]	validation_0-logloss:0.306754	validation_1-logloss:0.330795
[167]	validation_0-logloss:0.30659	validation_1-logloss:0.330751
[168]	validation_0-logloss:0.306112	validation_1-logloss:0.330487
[169]	validation_0-logloss:0.305865	validation_1-logloss:0.330316
[170]	validation_0-logloss:0.305694	validation_1-logloss:0.33024
[171]	validation_0-logloss:0.305573	validation_1-logloss:0.330193
[172]	validation_0-logloss:0.305319	validation_1-logloss:0.330133
[173]	validation_0-logloss:0.305161	validation_1-logloss:0.330086
[174]	validation_0-logloss:0.305003	validation_1-logloss:0.330075
[175]	validation_0-logloss:0.304847	validation_1-logloss:0.329991
[176]	validation_0-logloss:0.304629	validation_1-logloss:0.329876
[177]	validation_0-logloss:0.304519	validation_1-logloss:0.329829
[178]	validation_0-logloss:0.304357	validation_1-logloss:0.329758
[179]	validation_0-logloss:0.304213	validation_1-logloss:0.32972
[180]	validation_0-logloss:0.304065	validation_1-logloss:0.329696
[181]	validation_0-logloss:0.303969	validation_1-logloss:0.329689
[182]	validation_0-logloss:0.303848	validation_1-logloss:0.329671
[183]	validation_0-logloss:0.303696	validation_1-logloss:0.329592
[184]	validation_0-logloss:0.303598	validation_1-logloss:0.329536
[185]	validation_0-logloss:0.303482	validation_1-logloss:0.329498
[186]	validation_0-logloss:0.303297	validation_1-logloss:0.329337
[187]	validation_0-logloss:0.303186	validation_1-logloss:0.329283
[188]	validation_0-logloss:0.303077	validation_1-logloss:0.329233
[189]	validation_0-logloss:0.302929	validation_1-logloss:0.329168
[190]	validation_0-logloss:0.302743	validation_1-logloss:0.329076
[191]	validation_0-logloss:0.302547	validation_1-logloss:0.32896
[192]	validation_0-logloss:0.302447	validation_1-logloss:0.328925
[193]	validation_0-logloss:0.302242	validation_1-logloss:0.328781
[194]	validation_0-logloss:0.302089	validation_1-logloss:0.328722
[195]	validation_0-logloss:0.301987	validation_1-logloss:0.328681
[196]	validation_0-logloss:0.301654	validation_1-logloss:0.328496
[197]	validation_0-logloss:0.301514	validation_1-logloss:0.328443
[198]	validation_0-logloss:0.30142	validation_1-logloss:0.328375
[199]	validation_0-logloss:0.301204	validation_1-logloss:0.328285
[200]	validation_0-logloss:0.300932	validation_1-logloss:0.328229
[201]	validation_0-logloss:0.300826	validation_1-logloss:0.328218
[202]	validation_0-logloss:0.300696	validation_1-logloss:0.328145
[203]	validation_0-logloss:0.300393	validation_1-logloss:0.328038
[204]	validation_0-logloss:0.300236	validation_1-logloss:0.327989
[205]	validation_0-logloss:0.299908	validation_1-logloss:0.327828
[206]	validation_0-logloss:0.299733	validation_1-logloss:0.327783
[207]	validation_0-logloss:0.299639	validation_1-logloss:0.327766
```

```
[208]	validation_0-logloss:0.29945	validation_1-logloss:0.327654
[209]	validation_0-logloss:0.298824	validation_1-logloss:0.327307
[210]	validation_0-logloss:0.298655	validation_1-logloss:0.327229
[211]	validation_0-logloss:0.29849	validation_1-logloss:0.327114
[212]	validation_0-logloss:0.298101	validation_1-logloss:0.326853
[213]	validation_0-logloss:0.297506	validation_1-logloss:0.3264
[214]	validation_0-logloss:0.297394	validation_1-logloss:0.326361
[215]	validation_0-logloss:0.297249	validation_1-logloss:0.326353
[216]	validation_0-logloss:0.2971	validation_1-logloss:0.32629
[217]	validation_0-logloss:0.296968	validation_1-logloss:0.326263
[218]	validation_0-logloss:0.296773	validation_1-logloss:0.32622
[219]	validation_0-logloss:0.296669	validation_1-logloss:0.326251
[220]	validation_0-logloss:0.296507	validation_1-logloss:0.326158
[221]	validation_0-logloss:0.296297	validation_1-logloss:0.326106
[222]	validation_0-logloss:0.296143	validation_1-logloss:0.326007
[223]	validation_0-logloss:0.295989	validation_1-logloss:0.325927
[224]	validation_0-logloss:0.295823	validation_1-logloss:0.325916
[225]	validation_0-logloss:0.295737	validation_1-logloss:0.325904
[226]	validation_0-logloss:0.295587	validation_1-logloss:0.325836
[227]	validation_0-logloss:0.295464	validation_1-logloss:0.32584
[228]	validation_0-logloss:0.295259	validation_1-logloss:0.325744
[229]	validation_0-logloss:0.295052	validation_1-logloss:0.325634
[230]	validation_0-logloss:0.294914	validation_1-logloss:0.32559
[231]	validation_0-logloss:0.294783	validation_1-logloss:0.325637
[232]	validation_0-logloss:0.294607	validation_1-logloss:0.325637
[233]	validation_0-logloss:0.29447	validation_1-logloss:0.325576
[234]	validation_0-logloss:0.294318	validation_1-logloss:0.325502
[235]	validation_0-logloss:0.294196	validation_1-logloss:0.325463
[236]	validation_0-logloss:0.294095	validation_1-logloss:0.325432
[237]	validation_0-logloss:0.293883	validation_1-logloss:0.325389
[238]	validation_0-logloss:0.293796	validation_1-logloss:0.325358
[239]	validation_0-logloss:0.293689	validation_1-logloss:0.325331
[240]	validation_0-logloss:0.293603	validation_1-logloss:0.325259
[241]	validation_0-logloss:0.293446	validation_1-logloss:0.32533
[242]	validation_0-logloss:0.293312	validation_1-logloss:0.325293
[243]	validation_0-logloss:0.293224	validation_1-logloss:0.325239
[244]	validation_0-logloss:0.293133	validation_1-logloss:0.325232
[245]	validation_0-logloss:0.292998	validation_1-logloss:0.325163
[246]	validation_0-logloss:0.292849	validation_1-logloss:0.325199
[247]	validation_0-logloss:0.292715	validation_1-logloss:0.325187
[248]	validation_0-logloss:0.292548	validation_1-logloss:0.325119
[249]	validation_0-logloss:0.2924	validation_1-logloss:0.325147
[250]	validation_0-logloss:0.29232	validation_1-logloss:0.325109
[251]	validation_0-logloss:0.292166	validation_1-logloss:0.325075
[252]	validation_0-logloss:0.292032	validation_1-logloss:0.32505
[253]	validation_0-logloss:0.291897	validation_1-logloss:0.324976
[254]	validation_0-logloss:0.291786	validation_1-logloss:0.324957
[255]	validation_0-logloss:0.291581	validation_1-logloss:0.324901
```

```
[256]	validation_0-logloss:0.291174	validation_1-logloss:0.32476
[257]	validation_0-logloss:0.290996	validation_1-logloss:0.324766
[258]	validation_0-logloss:0.290914	validation_1-logloss:0.324726
[259]	validation_0-logloss:0.290788	validation_1-logloss:0.324695
[260]	validation_0-logloss:0.290676	validation_1-logloss:0.324679
[261]	validation_0-logloss:0.290561	validation_1-logloss:0.324645
[262]	validation_0-logloss:0.290442	validation_1-logloss:0.324649
[263]	validation_0-logloss:0.290342	validation_1-logloss:0.324635
[264]	validation_0-logloss:0.290219	validation_1-logloss:0.324593
[265]	validation_0-logloss:0.290043	validation_1-logloss:0.324565
[266]	validation_0-logloss:0.289897	validation_1-logloss:0.32455
[267]	validation_0-logloss:0.289746	validation_1-logloss:0.324538
[268]	validation_0-logloss:0.289635	validation_1-logloss:0.324487
[269]	validation_0-logloss:0.289508	validation_1-logloss:0.324471
[270]	validation_0-logloss:0.289417	validation_1-logloss:0.324398
[271]	validation_0-logloss:0.289304	validation_1-logloss:0.324343
[272]	validation_0-logloss:0.289127	validation_1-logloss:0.324324
[273]	validation_0-logloss:0.289011	validation_1-logloss:0.324305
[274]	validation_0-logloss:0.288856	validation_1-logloss:0.324283
[275]	validation_0-logloss:0.288788	validation_1-logloss:0.324286
[276]	validation_0-logloss:0.288661	validation_1-logloss:0.324267
[277]	validation_0-logloss:0.288586	validation_1-logloss:0.324231
[278]	validation_0-logloss:0.288507	validation_1-logloss:0.324199
[279]	validation_0-logloss:0.288403	validation_1-logloss:0.324196
[280]	validation_0-logloss:0.288291	validation_1-logloss:0.324138
[281]	validation_0-logloss:0.288171	validation_1-logloss:0.324137
[282]	validation_0-logloss:0.288045	validation_1-logloss:0.324032
[283]	validation_0-logloss:0.287905	validation_1-logloss:0.323949
[284]	validation_0-logloss:0.287759	validation_1-logloss:0.323868
[285]	validation_0-logloss:0.28765	validation_1-logloss:0.323835
[286]	validation_0-logloss:0.287512	validation_1-logloss:0.323752
[287]	validation_0-logloss:0.287437	validation_1-logloss:0.32373
[288]	validation_0-logloss:0.28734	validation_1-logloss:0.323717
[289]	validation_0-logloss:0.287144	validation_1-logloss:0.323655
[290]	validation_0-logloss:0.286939	validation_1-logloss:0.323667
[291]	validation_0-logloss:0.286872	validation_1-logloss:0.323652
[292]	validation_0-logloss:0.286784	validation_1-logloss:0.323604
[293]	validation_0-logloss:0.286677	validation_1-logloss:0.323551
[294]	validation_0-logloss:0.286604	validation_1-logloss:0.32355
[295]	validation_0-logloss:0.286474	validation_1-logloss:0.323485
[296]	validation_0-logloss:0.286399	validation_1-logloss:0.323497
[297]	validation_0-logloss:0.286205	validation_1-logloss:0.32346
[298]	validation_0-logloss:0.28612	validation_1-logloss:0.323456
[299]	validation_0-logloss:0.286045	validation_1-logloss:0.323403
[300]	validation_0-logloss:0.285882	validation_1-logloss:0.323247
[301]	validation_0-logloss:0.285784	validation_1-logloss:0.323219
[302]	validation_0-logloss:0.285695	validation_1-logloss:0.32326
[303]	validation_0-logloss:0.285544	validation_1-logloss:0.323155
```

```
[304]           validation_0-logloss:0.285445         validation_1-logloss:0.323083
[305]           validation_0-logloss:0.284918         validation_1-logloss:0.322726
[306]           validation_0-logloss:0.28483         validation_1-logloss:0.322677
[307]           validation_0-logloss:0.284725         validation_1-logloss:0.3227
[308]           validation_0-logloss:0.284637         validation_1-logloss:0.322673
[309]           validation_0-logloss:0.2845         validation_1-logloss:0.322635
[310]           validation_0-logloss:0.284397         validation_1-logloss:0.322646
[311]           validation_0-logloss:0.284295         validation_1-logloss:0.322597
[312]           validation_0-logloss:0.284157         validation_1-logloss:0.322583
[313]           validation_0-logloss:0.283995         validation_1-logloss:0.322546
[314]           validation_0-logloss:0.283878         validation_1-logloss:0.322571
[315]           validation_0-logloss:0.283799         validation_1-logloss:0.322562
[316]           validation_0-logloss:0.283725         validation_1-logloss:0.322555
[317]           validation_0-logloss:0.283627         validation_1-logloss:0.322488
[318]           validation_0-logloss:0.28353         validation_1-logloss:0.322446
[319]           validation_0-logloss:0.283444         validation_1-logloss:0.322436
[320]           validation_0-logloss:0.283362         validation_1-logloss:0.32237
[321]           validation_0-logloss:0.283295         validation_1-logloss:0.322366
[322]           validation_0-logloss:0.283153         validation_1-logloss:0.322304
[323]           validation_0-logloss:0.283036         validation_1-logloss:0.322318
[324]           validation_0-logloss:0.28297         validation_1-logloss:0.322325
[325]           validation_0-logloss:0.282871         validation_1-logloss:0.3223
[326]           validation_0-logloss:0.282778         validation_1-logloss:0.322324
[327]           validation_0-logloss:0.282702         validation_1-logloss:0.322304
[328]           validation_0-logloss:0.282533         validation_1-logloss:0.322276
[329]           validation_0-logloss:0.282471         validation_1-logloss:0.322264
[330]           validation_0-logloss:0.282366         validation_1-logloss:0.322238
[331]           validation_0-logloss:0.282101         validation_1-logloss:0.322175
[332]           validation_0-logloss:0.281756         validation_1-logloss:0.322029
[333]           validation_0-logloss:0.281608         validation_1-logloss:0.322031
[334]           validation_0-logloss:0.281498         validation_1-logloss:0.322039
[335]           validation_0-logloss:0.281377         validation_1-logloss:0.321969
[336]           validation_0-logloss:0.281282         validation_1-logloss:0.321938
[337]           validation_0-logloss:0.281216         validation_1-logloss:0.321878
[338]           validation_0-logloss:0.281134         validation_1-logloss:0.321856
[339]           validation_0-logloss:0.281057         validation_1-logloss:0.321847
[340]           validation_0-logloss:0.280984         validation_1-logloss:0.321818
[341]           validation_0-logloss:0.280891         validation_1-logloss:0.321827
[342]           validation_0-logloss:0.280796         validation_1-logloss:0.321797
[343]           validation_0-logloss:0.280719         validation_1-logloss:0.321793
[344]           validation_0-logloss:0.280661         validation_1-logloss:0.321793
[345]           validation_0-logloss:0.280512         validation_1-logloss:0.321764
[346]           validation_0-logloss:0.280414         validation_1-logloss:0.321766
[347]           validation_0-logloss:0.280246         validation_1-logloss:0.321696
[348]           validation_0-logloss:0.280145         validation_1-logloss:0.32165
[349]           validation_0-logloss:0.28005         validation_1-logloss:0.32158
[350]           validation_0-logloss:0.279988         validation_1-logloss:0.321583
[351]           validation_0-logloss:0.279923         validation_1-logloss:0.321553
```

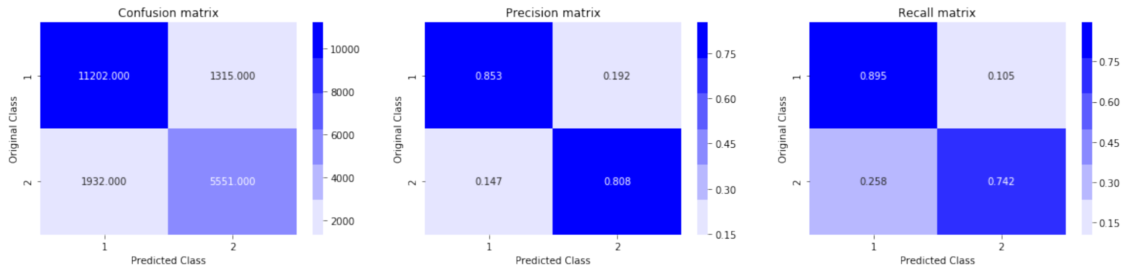
```
[352]           validation_0-logloss:0.27983        validation_1-logloss:0.321511
[353]           validation_0-logloss:0.279703        validation_1-logloss:0.321475
[354]           validation_0-logloss:0.27962        validation_1-logloss:0.321451
[355]           validation_0-logloss:0.279535        validation_1-logloss:0.321431
[356]           validation_0-logloss:0.279449        validation_1-logloss:0.321404
[357]           validation_0-logloss:0.279362        validation_1-logloss:0.321401
[358]           validation_0-logloss:0.279114        validation_1-logloss:0.321204
[359]           validation_0-logloss:0.279004        validation_1-logloss:0.321198
[360]           validation_0-logloss:0.278929        validation_1-logloss:0.321175
[361]           validation_0-logloss:0.278806        validation_1-logloss:0.321124
[362]           validation_0-logloss:0.278749        validation_1-logloss:0.321121
[363]           validation_0-logloss:0.278654        validation_1-logloss:0.321094
[364]           validation_0-logloss:0.278584        validation_1-logloss:0.321129
[365]           validation_0-logloss:0.27847        validation_1-logloss:0.321065
[366]           validation_0-logloss:0.278325        validation_1-logloss:0.321079
[367]           validation_0-logloss:0.278178        validation_1-logloss:0.321081
[368]           validation_0-logloss:0.278052        validation_1-logloss:0.321066
[369]           validation_0-logloss:0.277901        validation_1-logloss:0.321002
[370]           validation_0-logloss:0.277841        validation_1-logloss:0.320951
[371]           validation_0-logloss:0.277775        validation_1-logloss:0.320964
[372]           validation_0-logloss:0.27769        validation_1-logloss:0.320967
[373]           validation_0-logloss:0.277604        validation_1-logloss:0.320949
[374]           validation_0-logloss:0.277467        validation_1-logloss:0.320933
[375]           validation_0-logloss:0.277306        validation_1-logloss:0.320903
[376]           validation_0-logloss:0.277185        validation_1-logloss:0.320891
[377]           validation_0-logloss:0.277093        validation_1-logloss:0.320906
[378]           validation_0-logloss:0.277006        validation_1-logloss:0.320865
[379]           validation_0-logloss:0.276915        validation_1-logloss:0.320826
[380]           validation_0-logloss:0.276858        validation_1-logloss:0.320791
[381]           validation_0-logloss:0.2768        validation_1-logloss:0.320817
[382]           validation_0-logloss:0.27669        validation_1-logloss:0.320787
[383]           validation_0-logloss:0.276623        validation_1-logloss:0.320767
[384]           validation_0-logloss:0.276528        validation_1-logloss:0.320729
[385]           validation_0-logloss:0.276456        validation_1-logloss:0.320713
[386]           validation_0-logloss:0.276402        validation_1-logloss:0.320671
[387]           validation_0-logloss:0.276307        validation_1-logloss:0.320666
[388]           validation_0-logloss:0.276242        validation_1-logloss:0.320657
[389]           validation_0-logloss:0.276084        validation_1-logloss:0.320629
[390]           validation_0-logloss:0.275681        validation_1-logloss:0.320363
[391]           validation_0-logloss:0.275582        validation_1-logloss:0.320317
[392]           validation_0-logloss:0.275466        validation_1-logloss:0.320265
[393]           validation_0-logloss:0.275384        validation_1-logloss:0.320224
[394]           validation_0-logloss:0.275246        validation_1-logloss:0.320197
[395]           validation_0-logloss:0.275167        validation_1-logloss:0.320148
[396]           validation_0-logloss:0.275062        validation_1-logloss:0.320132
[397]           validation_0-logloss:0.274901        validation_1-logloss:0.320011
[398]           validation_0-logloss:0.274843        validation_1-logloss:0.319974
[399]           validation_0-logloss:0.274736        validation_1-logloss:0.31995
```

```
[400]	validation_0-logloss:0.274618	validation_1-logloss:0.319994
[401]	validation_0-logloss:0.274525	validation_1-logloss:0.319992
[402]	validation_0-logloss:0.274402	validation_1-logloss:0.320012
[403]	validation_0-logloss:0.274333	validation_1-logloss:0.320023
[404]	validation_0-logloss:0.274247	validation_1-logloss:0.320019
[405]	validation_0-logloss:0.274187	validation_1-logloss:0.320013
[406]	validation_0-logloss:0.274117	validation_1-logloss:0.319975
[407]	validation_0-logloss:0.274064	validation_1-logloss:0.319982
[408]	validation_0-logloss:0.274006	validation_1-logloss:0.319968
[409]	validation_0-logloss:0.273947	validation_1-logloss:0.319941
[410]	validation_0-logloss:0.273846	validation_1-logloss:0.319941
[411]	validation_0-logloss:0.273644	validation_1-logloss:0.319773
[412]	validation_0-logloss:0.273493	validation_1-logloss:0.319668
[413]	validation_0-logloss:0.273229	validation_1-logloss:0.319545
[414]	validation_0-logloss:0.273147	validation_1-logloss:0.319532
[415]	validation_0-logloss:0.273048	validation_1-logloss:0.319529
[416]	validation_0-logloss:0.272972	validation_1-logloss:0.31952
[417]	validation_0-logloss:0.272842	validation_1-logloss:0.319502
[418]	validation_0-logloss:0.272735	validation_1-logloss:0.31956
[419]	validation_0-logloss:0.272657	validation_1-logloss:0.319552
[420]	validation_0-logloss:0.272587	validation_1-logloss:0.319576
[421]	validation_0-logloss:0.272521	validation_1-logloss:0.319553
[422]	validation_0-logloss:0.272421	validation_1-logloss:0.319576
[423]	validation_0-logloss:0.272355	validation_1-logloss:0.319576
[424]	validation_0-logloss:0.27222	validation_1-logloss:0.319538
[425]	validation_0-logloss:0.272151	validation_1-logloss:0.319527
[426]	validation_0-logloss:0.272042	validation_1-logloss:0.319552
[427]	validation_0-logloss:0.271989	validation_1-logloss:0.319549
[428]	validation_0-logloss:0.271929	validation_1-logloss:0.319538
[429]	validation_0-logloss:0.271874	validation_1-logloss:0.319546
[430]	validation_0-logloss:0.271811	validation_1-logloss:0.319526
[431]	validation_0-logloss:0.271739	validation_1-logloss:0.319516
[432]	validation_0-logloss:0.271658	validation_1-logloss:0.319533
[433]	validation_0-logloss:0.271602	validation_1-logloss:0.319498
[434]	validation_0-logloss:0.271496	validation_1-logloss:0.319488
[435]	validation_0-logloss:0.271413	validation_1-logloss:0.319502
[436]	validation_0-logloss:0.271311	validation_1-logloss:0.319473
[437]	validation_0-logloss:0.271147	validation_1-logloss:0.319361
[438]	validation_0-logloss:0.271011	validation_1-logloss:0.319394
[439]	validation_0-logloss:0.270921	validation_1-logloss:0.319371
[440]	validation_0-logloss:0.27086	validation_1-logloss:0.31936
[441]	validation_0-logloss:0.270791	validation_1-logloss:0.319341
[442]	validation_0-logloss:0.270717	validation_1-logloss:0.319313
[443]	validation_0-logloss:0.270661	validation_1-logloss:0.319303
[444]	validation_0-logloss:0.270581	validation_1-logloss:0.319304
[445]	validation_0-logloss:0.270466	validation_1-logloss:0.319314
[446]	validation_0-logloss:0.270418	validation_1-logloss:0.3193
[447]	validation_0-logloss:0.270337	validation_1-logloss:0.319279
```

```
[448]	validation_0-logloss:0.270243	validation_1-logloss:0.319256
[449]	validation_0-logloss:0.270121	validation_1-logloss:0.319276
[450]	validation_0-logloss:0.270003	validation_1-logloss:0.319208
[451]	validation_0-logloss:0.269948	validation_1-logloss:0.3192
[452]	validation_0-logloss:0.269893	validation_1-logloss:0.319158
[453]	validation_0-logloss:0.269839	validation_1-logloss:0.319123
[454]	validation_0-logloss:0.269757	validation_1-logloss:0.319105
[455]	validation_0-logloss:0.269661	validation_1-logloss:0.319074
[456]	validation_0-logloss:0.269361	validation_1-logloss:0.318864
[457]	validation_0-logloss:0.26925	validation_1-logloss:0.318754
[458]	validation_0-logloss:0.268947	validation_1-logloss:0.318641
[459]	validation_0-logloss:0.268879	validation_1-logloss:0.318579
[460]	validation_0-logloss:0.268786	validation_1-logloss:0.318595
[461]	validation_0-logloss:0.268711	validation_1-logloss:0.318588
[462]	validation_0-logloss:0.268564	validation_1-logloss:0.318495
[463]	validation_0-logloss:0.268484	validation_1-logloss:0.31845
[464]	validation_0-logloss:0.268343	validation_1-logloss:0.318412
[465]	validation_0-logloss:0.267959	validation_1-logloss:0.31822
[466]	validation_0-logloss:0.267855	validation_1-logloss:0.318203
[467]	validation_0-logloss:0.267636	validation_1-logloss:0.318085
[468]	validation_0-logloss:0.267316	validation_1-logloss:0.317862
[469]	validation_0-logloss:0.267164	validation_1-logloss:0.317846
[470]	validation_0-logloss:0.26709	validation_1-logloss:0.31784
[471]	validation_0-logloss:0.267028	validation_1-logloss:0.317778
[472]	validation_0-logloss:0.266931	validation_1-logloss:0.31774
[473]	validation_0-logloss:0.266857	validation_1-logloss:0.31775
[474]	validation_0-logloss:0.26677	validation_1-logloss:0.317756
[475]	validation_0-logloss:0.266716	validation_1-logloss:0.317718
[476]	validation_0-logloss:0.266671	validation_1-logloss:0.317704
[477]	validation_0-logloss:0.266624	validation_1-logloss:0.317707
[478]	validation_0-logloss:0.266567	validation_1-logloss:0.317673
[479]	validation_0-logloss:0.266487	validation_1-logloss:0.317689
[480]	validation_0-logloss:0.266382	validation_1-logloss:0.317669
[481]	validation_0-logloss:0.266281	validation_1-logloss:0.317675
[482]	validation_0-logloss:0.266234	validation_1-logloss:0.317665
[483]	validation_0-logloss:0.266074	validation_1-logloss:0.317584
[484]	validation_0-logloss:0.266017	validation_1-logloss:0.317599
[485]	validation_0-logloss:0.265817	validation_1-logloss:0.317473
[486]	validation_0-logloss:0.265739	validation_1-logloss:0.317469
[487]	validation_0-logloss:0.265628	validation_1-logloss:0.317459
[488]	validation_0-logloss:0.265507	validation_1-logloss:0.317444
[489]	validation_0-logloss:0.265437	validation_1-logloss:0.317425
[490]	validation_0-logloss:0.26535	validation_1-logloss:0.317419
[491]	validation_0-logloss:0.26529	validation_1-logloss:0.317406
[492]	validation_0-logloss:0.265156	validation_1-logloss:0.317285
[493]	validation_0-logloss:0.26505	validation_1-logloss:0.317281
[494]	validation_0-logloss:0.264971	validation_1-logloss:0.31725
[495]	validation_0-logloss:0.264848	validation_1-logloss:0.317189
```

```
[496]       validation_0-logloss:0.264753       validation_1-logloss:0.317149
[497]       validation_0-logloss:0.264704       validation_1-logloss:0.317139
[498]       validation_0-logloss:0.264651       validation_1-logloss:0.317118
[499]       validation_0-logloss:0.264559       validation_1-logloss:0.317139
```

In [96]:
```
plot_confusion_matrix(y_test, y_pred)
```



In [98]:
```
evals_result = clf.evals_result()
evals_result #to find the minimum of train and test log loss
```

Out[98]:
```
{'validation_0': {'logloss': [0.617872,
   0.563976,
   0.523963,
   0.495163,
   0.471689,
   0.454709,
   0.439191,
   0.428588,
   0.418899,
   0.410939,
   0.404258,
   0.398847,
   0.394228,
   0.390694,
   0.38677,
   0.383814,
   0.381428,
   0.378998,
   0.37682,
   0.374589,
   0.373194,
   0.37152,
   0.369454,
   0.367397,
   0.366316,
   0.365188,
```

```
0.363532,
0.362404,
0.361172,
0.360184,
0.358872,
0.358125,
0.356853,
0.355767,
0.35469,
0.353936,
0.353229,
0.352221,
0.351635,
0.350974,
0.350158,
0.34962,
0.34912,
0.348537,
0.348075,
0.347597,
0.346515,
0.34525,
0.344648,
0.343946,
0.343468,
0.342783,
0.342283,
0.34188,
0.341409,
0.340814,
0.340345,
0.340032,
0.339492,
0.339047,
0.338741,
0.338247,
0.337965,
0.33763,
0.337228,
0.336939,
0.336505,
0.335784,
0.335348,
0.3349,
0.334457,
0.333813,
0.333389,
0.333105,
```

```
0.332808,
0.33243,
0.332069,
0.331078,
0.330828,
0.330546,
0.33031,
0.329988,
0.329601,
0.329236,
0.328991,
0.328749,
0.328329,
0.328098,
0.327903,
0.327564,
0.327352,
0.327054,
0.32681,
0.326527,
0.325962,
0.325663,
0.325474,
0.325235,
0.324908,
0.324608,
0.324388,
0.324061,
0.323766,
0.323449,
0.323172,
0.322794,
0.322213,
0.321074,
0.320358,
0.319776,
0.319565,
0.319351,
0.319132,
0.318919,
0.318748,
0.318569,
0.318416,
0.318196,
0.317874,
0.317452,
0.317157,
0.316856,
```

```
0.316668,
0.316181,
0.315922,
0.315637,
0.315412,
0.315163,
0.314934,
0.314788,
0.314466,
0.314231,
0.313985,
0.313769,
0.31361,
0.313431,
0.313198,
0.312979,
0.312733,
0.312598,
0.312099,
0.311879,
0.311735,
0.311456,
0.311242,
0.311055,
0.310851,
0.310609,
0.310395,
0.310225,
0.310104,
0.309809,
0.309686,
0.309283,
0.309138,
0.309008,
0.308862,
0.308709,
0.308466,
0.308346,
0.30804,
0.307892,
0.307679,
0.307472,
0.307309,
0.306936,
0.306754,
0.30659,
0.306112,
0.305865,
```

0.305694,
0.305573,
0.305319,
0.305161,
0.305003,
0.304847,
0.304629,
0.304519,
0.304357,
0.304213,
0.304065,
0.303969,
0.303848,
0.303696,
0.303598,
0.303482,
0.303297,
0.303186,
0.303077,
0.302929,
0.302743,
0.302547,
0.302447,
0.302242,
0.302089,
0.301987,
0.301654,
0.301514,
0.30142,
0.301204,
0.300932,
0.300826,
0.300696,
0.300393,
0.300236,
0.299908,
0.299733,
0.299639,
0.29945,
0.298824,
0.298655,
0.29849,
0.298101,
0.297506,
0.297394,
0.297249,
0.2971,
0.296968,

0.296773,
0.296669,
0.296507,
0.296297,
0.296143,
0.295989,
0.295823,
0.295737,
0.295587,
0.295464,
0.295259,
0.295052,
0.294914,
0.294783,
0.294607,
0.29447,
0.294318,
0.294196,
0.294095,
0.293883,
0.293796,
0.293689,
0.293603,
0.293446,
0.293312,
0.293224,
0.293133,
0.292998,
0.292849,
0.292715,
0.292548,
0.2924,
0.29232,
0.292166,
0.292032,
0.291897,
0.291786,
0.291581,
0.291174,
0.290996,
0.290914,
0.290788,
0.290676,
0.290561,
0.290442,
0.290342,
0.290219,
0.290043,

```
0.289897,
0.289746,
0.289635,
0.289508,
0.289417,
0.289304,
0.289127,
0.289011,
0.288856,
0.288788,
0.288661,
0.288586,
0.288507,
0.288403,
0.288291,
0.288171,
0.288045,
0.287905,
0.287759,
0.28765,
0.287512,
0.287437,
0.28734,
0.287144,
0.286939,
0.286872,
0.286784,
0.286677,
0.286604,
0.286474,
0.286399,
0.286205,
0.28612,
0.286045,
0.285882,
0.285784,
0.285695,
0.285544,
0.285445,
0.284918,
0.28483,
0.284725,
0.284637,
0.2845,
0.284397,
0.284295,
0.284157,
0.283995,
```

```
0.283878,
0.283799,
0.283725,
0.283627,
0.28353,
0.283444,
0.283362,
0.283295,
0.283153,
0.283036,
0.28297,
0.282871,
0.282778,
0.282702,
0.282533,
0.282471,
0.282366,
0.282101,
0.281756,
0.281608,
0.281498,
0.281377,
0.281282,
0.281216,
0.281134,
0.281057,
0.280984,
0.280891,
0.280796,
0.280719,
0.280661,
0.280512,
0.280414,
0.280246,
0.280145,
0.28005,
0.279988,
0.279923,
0.27983,
0.279703,
0.27962,
0.279535,
0.279449,
0.279362,
0.279114,
0.279004,
0.278929,
0.278806,
```

0.278749,
0.278654,
0.278584,
0.27847,
0.278325,
0.278178,
0.278052,
0.277901,
0.277841,
0.277775,
0.27769,
0.277604,
0.277467,
0.277306,
0.277185,
0.277093,
0.277006,
0.276915,
0.276858,
0.2768,
0.27669,
0.276623,
0.276528,
0.276456,
0.276402,
0.276307,
0.276242,
0.276084,
0.275681,
0.275582,
0.275466,
0.275384,
0.275246,
0.275167,
0.275062,
0.274901,
0.274843,
0.274736,
0.274618,
0.274525,
0.274402,
0.274333,
0.274247,
0.274187,
0.274117,
0.274064,
0.274006,
0.273947,

0.273846,
0.273644,
0.273493,
0.273229,
0.273147,
0.273048,
0.272972,
0.272842,
0.272735,
0.272657,
0.272587,
0.272521,
0.272421,
0.272355,
0.27222,
0.272151,
0.272042,
0.271989,
0.271929,
0.271874,
0.271811,
0.271739,
0.271658,
0.271602,
0.271496,
0.271413,
0.271311,
0.271147,
0.271011,
0.270921,
0.27086,
0.270791,
0.270717,
0.270661,
0.270581,
0.270466,
0.270418,
0.270337,
0.270243,
0.270121,
0.270003,
0.269948,
0.269893,
0.269839,
0.269757,
0.269661,
0.269361,
0.26925,

```
  0.268947,
  0.268879,
  0.268786,
  0.268711,
  0.268564,
  0.268484,
  0.268343,
  0.267959,
  0.267855,
  0.267636,
  0.267316,
  0.267164,
  0.26709,
  0.267028,
  0.266931,
  0.266857,
  0.26677,
  0.266716,
  0.266671,
  0.266624,
  0.266567,
  0.266487,
  0.266382,
  0.266281,
  0.266234,
  0.266074,
  0.266017,
  0.265817,
  0.265739,
  0.265628,
  0.265507,
  0.265437,
  0.26535,
  0.26529,
  0.265156,
  0.26505,
  0.264971,
  0.264848,
  0.264753,
  0.264704,
  0.264651,
  0.264559]},
'validation_1': {'logloss': [0.617761,
  0.564293,
  0.524828,
  0.495894,
  0.472717,
  0.455807,
```

0.44046,
0.430075,
0.420731,
0.412635,
0.406137,
0.401047,
0.39653,
0.393188,
0.389431,
0.386723,
0.384464,
0.382156,
0.380135,
0.378289,
0.377164,
0.375749,
0.373893,
0.372023,
0.371045,
0.370106,
0.368761,
0.367541,
0.366557,
0.365609,
0.364579,
0.363971,
0.362763,
0.361741,
0.360849,
0.360312,
0.359842,
0.359076,
0.358406,
0.357781,
0.357044,
0.356794,
0.356417,
0.356089,
0.355701,
0.355408,
0.354522,
0.353357,
0.35293,
0.352419,
0.351955,
0.351457,
0.351113,
0.350854,

0.350479,
0.349984,
0.349701,
0.349533,
0.349127,
0.348811,
0.348597,
0.348234,
0.348052,
0.34782,
0.347646,
0.347499,
0.347193,
0.346729,
0.346375,
0.346193,
0.345937,
0.345343,
0.345023,
0.344875,
0.344714,
0.344459,
0.344191,
0.343429,
0.343224,
0.343065,
0.342908,
0.342763,
0.342549,
0.342345,
0.342197,
0.342021,
0.341781,
0.341688,
0.341627,
0.341463,
0.341334,
0.341184,
0.341027,
0.340844,
0.340367,
0.340246,
0.340197,
0.340119,
0.339912,
0.339794,
0.339683,
0.339528,

```
0.339303,
0.339202,
0.338957,
0.338672,
0.338166,
0.337427,
0.33686,
0.336587,
0.336576,
0.336552,
0.336416,
0.336312,
0.336269,
0.336261,
0.336172,
0.336048,
0.335862,
0.33566,
0.335468,
0.335332,
0.335257,
0.335079,
0.334838,
0.334534,
0.334429,
0.334372,
0.334264,
0.334248,
0.334112,
0.334038,
0.333899,
0.333796,
0.333837,
0.333828,
0.333757,
0.333617,
0.333519,
0.333433,
0.333149,
0.333077,
0.333011,
0.332985,
0.332894,
0.332806,
0.332736,
0.332728,
0.332606,
0.332509,
```

0.33246,
0.332205,
0.33214,
0.331876,
0.33187,
0.331805,
0.331768,
0.331692,
0.331578,
0.331623,
0.331437,
0.331378,
0.331294,
0.331244,
0.331157,
0.330924,
0.330795,
0.330751,
0.330487,
0.330316,
0.33024,
0.330193,
0.330133,
0.330086,
0.330075,
0.329991,
0.329876,
0.329829,
0.329758,
0.32972,
0.329696,
0.329689,
0.329671,
0.329592,
0.329536,
0.329498,
0.329337,
0.329283,
0.329233,
0.329168,
0.329076,
0.32896,
0.328925,
0.328781,
0.328722,
0.328681,
0.328496,
0.328443,

```
0.328375,
0.328285,
0.328229,
0.328218,
0.328145,
0.328038,
0.327989,
0.327828,
0.327783,
0.327766,
0.327654,
0.327307,
0.327229,
0.327114,
0.326853,
0.3264,
0.326361,
0.326353,
0.32629,
0.326263,
0.32622,
0.326251,
0.326158,
0.326106,
0.326007,
0.325927,
0.325916,
0.325904,
0.325836,
0.32584,
0.325744,
0.325634,
0.32559,
0.325637,
0.325637,
0.325576,
0.325502,
0.325463,
0.325432,
0.325389,
0.325358,
0.325331,
0.325259,
0.32533,
0.325293,
0.325239,
0.325232,
0.325163,
```

```
0.325199,
0.325187,
0.325119,
0.325147,
0.325109,
0.325075,
0.32505,
0.324976,
0.324957,
0.324901,
0.32476,
0.324766,
0.324726,
0.324695,
0.324679,
0.324645,
0.324649,
0.324635,
0.324593,
0.324565,
0.32455,
0.324538,
0.324487,
0.324471,
0.324398,
0.324343,
0.324324,
0.324305,
0.324283,
0.324286,
0.324267,
0.324231,
0.324199,
0.324196,
0.324138,
0.324137,
0.324032,
0.323949,
0.323868,
0.323835,
0.323752,
0.32373,
0.323717,
0.323655,
0.323667,
0.323652,
0.323604,
0.323551,
```

```
0.32355,
0.323485,
0.323497,
0.32346,
0.323456,
0.323403,
0.323247,
0.323219,
0.32326,
0.323155,
0.323083,
0.322726,
0.322677,
0.3227,
0.322673,
0.322635,
0.322646,
0.322597,
0.322583,
0.322546,
0.322571,
0.322562,
0.322555,
0.322488,
0.322446,
0.322436,
0.32237,
0.322366,
0.322304,
0.322318,
0.322325,
0.3223,
0.322324,
0.322304,
0.322276,
0.322264,
0.322238,
0.322175,
0.322029,
0.322031,
0.322039,
0.321969,
0.321938,
0.321878,
0.321856,
0.321847,
0.321818,
0.321827,
```

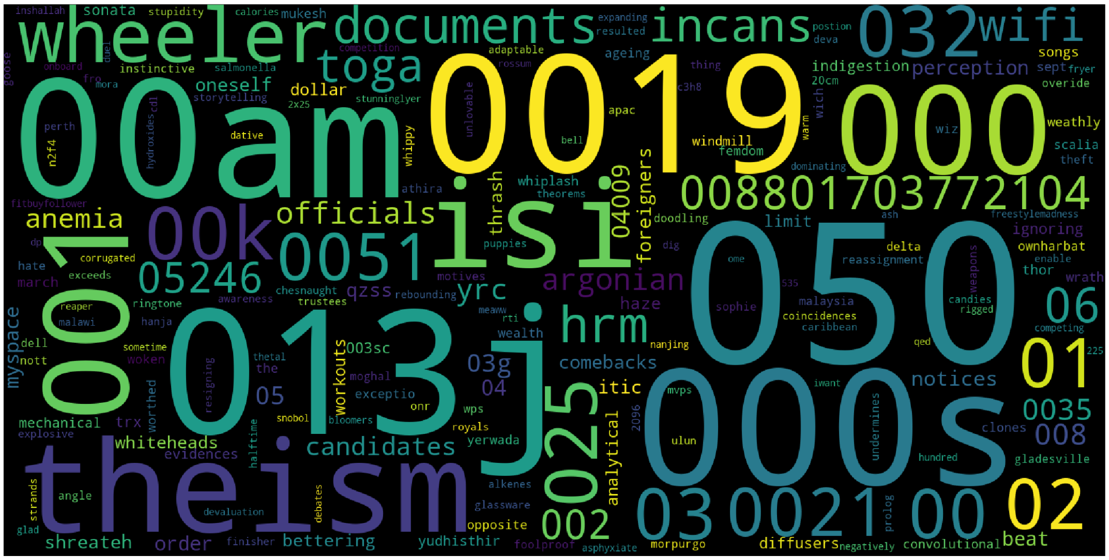
0.321797,
0.321793,
0.321793,
0.321764,
0.321766,
0.321696,
0.32165,
0.32158,
0.321583,
0.321553,
0.321511,
0.321475,
0.321451,
0.321431,
0.321404,
0.321401,
0.321204,
0.321198,
0.321175,
0.321124,
0.321121,
0.321094,
0.321129,
0.321065,
0.321079,
0.321081,
0.321066,
0.321002,
0.320951,
0.320964,
0.320967,
0.320949,
0.320933,
0.320903,
0.320891,
0.320906,
0.320865,
0.320826,
0.320791,
0.320817,
0.320787,
0.320767,
0.320729,
0.320713,
0.320671,
0.320666,
0.320657,
0.320629,

0.320363,
0.320317,
0.320265,
0.320224,
0.320197,
0.320148,
0.320132,
0.320011,
0.319974,
0.31995,
0.319994,
0.319992,
0.320012,
0.320023,
0.320019,
0.320013,
0.319975,
0.319982,
0.319968,
0.319941,
0.319941,
0.319773,
0.319668,
0.319545,
0.319532,
0.319529,
0.31952,
0.319502,
0.31956,
0.319552,
0.319576,
0.319553,
0.319576,
0.319576,
0.319538,
0.319527,
0.319552,
0.319549,
0.319538,
0.319546,
0.319526,
0.319516,
0.319533,
0.319498,
0.319488,
0.319502,
0.319473,
0.319361,

```
0.319394,
0.319371,
0.31936,
0.319341,
0.319313,
0.319303,
0.319304,
0.319314,
0.3193,
0.319279,
0.319256,
0.319276,
0.319208,
0.3192,
0.319158,
0.319123,
0.319105,
0.319074,
0.318864,
0.318754,
0.318641,
0.318579,
0.318595,
0.318588,
0.318495,
0.31845,
0.318412,
0.31822,
0.318203,
0.318085,
0.317862,
0.317846,
0.31784,
0.317778,
0.31774,
0.31775,
0.317756,
0.317718,
0.317704,
0.317707,
0.317673,
0.317689,
0.317669,
0.317675,
0.317665,
0.317584,
0.317599,
0.317473,
```

```
    0.317469,
    0.317459,
    0.317444,
    0.317425,
    0.317419,
    0.317406,
    0.317285,
    0.317281,
    0.31725,
    0.317189,
    0.317149,
    0.317139,
    0.317118,
    0.317139]}}
```

In [99]:
```
# Ploting word cloud
from wordcloud import WordCloud

freq = fi
words = vect.get_feature_names()
result = dict(zip(words, freq))

# Lets first convert the 'result' dictionary to 'list of tuples'
tup = dict(result.items())
#Initializing WordCloud using frequencies of tags.
wordcloud = WordCloud(background_color='black',
                          width=1600,
                          height=800,
                    ).generate_from_frequencies(tup)

fig = plt.figure(figsize=(30,20))
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
fig.savefig("tag.png")
plt.show()
```



8 4. Results

```
In [102]: from prettytable import PrettyTable
          x = PrettyTable()
          x.field_names = ["MODEL", "Hyperparameters", "Train-log-loss", "Test-log-loss"]

          #TFIDFW2V
          x.add_row(['TFIDFW2V with Logistic Regression', 'Alpha=0.1', 0.31, 0.45])
          x.add_row(['--'*5,'--'*5,'--'*5,'--'*5])
          x.add_row(['TFIDFW2V with Linear SVM', 'Alpha=1', 0.36, 0.45])
          x.add_row(['--'*5,'--'*5,'--'*5,'--'*5])
          x.add_row(['TFIDFW2V with XGBOOST', 'n_estimators = 500 \n Tree-max_depth = 4 \n Lea
          x.add_row(['--'*5,'-'*8,'-'*8,'-'*5])
          print(x)
```

```
+-----------------------------------+----------------------+-----------------+----------------+
|               MODEL               |   Hyperparameters    | Train-log-loss  | Test-log-loss  |
+-----------------------------------+----------------------+-----------------+----------------+
| TFIDFW2V with Logistic Regression |      Alpha=0.1       |       0.31      |      0.45      |
|             ----------            |      ----------      |    ----------   |   ----------   |
|      TFIDFW2V with Linear SVM     |       Alpha=1        |       0.36      |      0.45      |
|             ----------            |      ----------      |    ----------   |   ----------   |
|       TFIDFW2V with XGBOOST       | n_estimators = 500   |       0.32      |      0.31      |
|                                   |  Tree-max_depth = 4  |                 |                |
|                                   |  Learning Rate = 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 TFIDF 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.
4. though the results are good with TFIDF but XGBoost with TFIDFW2v still wins with test loss of 0.2 and also trained on much lesser data