09_Amazon_Fine_Food_Reviews_Analysis_RF Final

May 28, 2019

1 Amazon Fine Food Reviews Analysis

Data Source: https://www.kaggle.com/snap/amazon-fine-food-reviews
EDA: https://nycdatascience.com/blog/student-works/amazon-fine-foods-visualization/
The Amazon Fine Food Reviews dataset consists of reviews of fine foods from Amazon.
Number of reviews: 568,454 Number of users: 256,059 Number of products: 74,258 Timespan: Oct 1999 - Oct 2012 Number of Attributes/Columns in data: 10
Attribute Information:

1. Id
2. ProductId - unique identifier for the product
3. UserId - unqiue identifier for the user
4. ProfileName
5. HelpfulnessNumerator - number of users who found the review helpful
6. HelpfulnessDenominator - number of users who indicated whether they found the review helpful or not
7. Score - rating between 1 and 5
8. Time - timestamp for the review
9. Summary - brief summary of the review
10. Text - text of the review

Objective: Given a review, determine whether the review is positive (rating of 4 or 5) or negative (rating of 1 or 2).

[Q] How to determine if a review is positive or negative? [Ans] We could use Score/Rating. A rating of 4 or 5 can be cosnidered as a positive review. A rating of 1 or 2 can be considered as negative one. A review of rating 3 is considered nuetral and such reviews are ignored from our analysis. This is an approximate and proxy way of determining the polarity (positivity/negativity) of a review.

2 [1]. Reading Data

2.1 [1.1] Loading the data

The dataset is available in two forms 1. .csv file 2. SQLite Database

In order to load the data, We have used the SQLITE dataset as it is easier to query the data and visualise the data efficiently.

Here as we only want to get the global sentiment of the recommendations (positive or negative), we will purposefully ignore all Scores equal to 3. If the score is above 3, then the recommendation wil be set to "positive". Otherwise, it will be set to "negative".

```
In [0]: %matplotlib inline
        import warnings
        warnings.filterwarnings("ignore")


        import sqlite3
        import pandas as pd
        import numpy as np
        import nltk
        import string
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.feature_extraction.text import TfidfTransformer
        from sklearn.feature_extraction.text import TfidfVectorizer

        from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.metrics import confusion_matrix
        from sklearn import metrics
        from sklearn.metrics import roc_curve, auc
        from nltk.stem.porter import PorterStemmer

        import re
        # Tutorial about Python regular expressions: https://pymotw.com/2/re/
        import string
        from nltk.corpus import stopwords
        from nltk.stem import PorterStemmer
        from nltk.stem.wordnet import WordNetLemmatizer

        from gensim.models import Word2Vec
        from gensim.models import KeyedVectors
        import pickle

        from tqdm import tqdm
        import os
        import xgboost as xgb
```

```
In [6]: from google.colab import drive
        drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/co
```

```
In [100]: # using SQLite Table to read data.
          con = sqlite3.connect('drive/My Drive/database.sqlite')
```

```
            # filtering only positive and negative reviews i.e.
            # not taking into consideration those reviews with Score=3
            # SELECT * FROM Reviews WHERE Score != 3 LIMIT 500000, will give top 500000 data poi
            # you can change the number to any other number based on your computing power

            # filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Score != 3 LIMIT
            # for tsne assignment you can take 5k data points

            filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Score != 3 LIMIT 1

            # Give reviews with Score>3 a positive rating(1), and reviews with a score<3 a negat
            def partition(x):
                if x < 3:
                    return 0
                return 1

            #changing reviews with score less than 3 to be positive and vice-versa
            actualScore = filtered_data['Score']
            positiveNegative = actualScore.map(partition)
            filtered_data['Score'] = positiveNegative
            print("Number of data points in our data", filtered_data.shape)
            filtered_data.head(3)
```

```
Number of data points in our data (100000, 10)
```

Out[100]:

	Id	...	Text
0	1	...	I have bought several of the Vitality canned d...
1	2	...	Product arrived labeled as Jumbo Salted Peanut...
2	3	...	This is a confection that has been around a fe...

[3 rows x 10 columns]

```
In [0]: display = pd.read_sql_query("""
        SELECT UserId, ProductId, ProfileName, Time, Score, Text, COUNT(*)
        FROM Reviews
        GROUP BY UserId
        HAVING COUNT(*)>1
        """, con)
```

```
In [9]: print(display.shape)
        display.head()
```

```
(80668, 7)
```

Out[9]:

	UserId	...	COUNT(*)
0	#oc-R115TNMSPFT9I7	...	2
1	#oc-R11D9D7SHXIJB9	...	3

```
2  #oc-R11DNU2NBKQ23Z  ...        2
3  #oc-R11O5J5ZVQE25C  ...        3
4  #oc-R12KPBODL2B5ZD  ...        2

[5 rows x 7 columns]
```

```
In [10]: display[display['UserId']=='AZY10LLTJ71NX']
```

```
Out[10]:              UserId  ... COUNT(*)
         80638  AZY10LLTJ71NX  ...        5

         [1 rows x 7 columns]
```

```
In [11]: display['COUNT(*)'].sum()
```

```
Out[11]: 393063
```

3 [2] Exploratory Data Analysis

3.1 [2.1] Data Cleaning: Deduplication

It is observed (as shown in the table below) that the reviews data had many duplicate entries. Hence it was necessary to remove duplicates in order to get unbiased results for the analysis of the data. Following is an example:

```
In [12]: display= pd.read_sql_query("""
         SELECT *
         FROM Reviews
         WHERE Score != 3 AND UserId="AR5J8UI46CURR"
         ORDER BY ProductID
         """, con)
         display.head()
```

```
Out[12]:        Id  ...                                                Text
         0   78445  ...  DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...
         1  138317  ...  DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...
         2  138277  ...  DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...
         3   73791  ...  DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...
         4  155049  ...  DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...

         [5 rows x 10 columns]
```

As it can be seen above that same user has multiple reviews with same values for HelpfulnessNumerator, HelpfulnessDenominator, Score, Time, Summary and Text and on doing analysis it was found that ProductId=B000HDOPZG was Loacker Quadratini Vanilla Wafer Cookies, 8.82-Ounce Packages (Pack of 8) ProductId=B000HDL1RQ was Loacker Quadratini Lemon Wafer Cookies, 8.82-Ounce Packages (Pack of 8) and so on

It was inferred after analysis that reviews with same parameters other than ProductId belonged to the same product just having different flavour or quantity. Hence in order to reduce redundancy it was decided to eliminate the rows having same parameters.

The method used for the same was that we first sort the data according to ProductId and then just keep the first similar product review and delelte the others. for eg. in the above just the review for ProductId=B000HDL1RQ remains. This method ensures that there is only one representative for each product and deduplication without sorting would lead to possibility of different representatives still existing for the same product.

```
In [0]: #Sorting data according to ProductId in ascending order
        sorted_data=filtered_data.sort_values('ProductId', axis=0, ascending=True, inplace=Fals
```

```
In [14]: #Deduplication of entries
         final=sorted_data.drop_duplicates(subset={"UserId","ProfileName","Time","Text"}, keep=
         final.shape
```

```
Out[14]: (87775, 10)
```

```
In [15]: #Checking to see how much % of data still remains
         (final['Id'].size*1.0)/(filtered_data['Id'].size*1.0)*100
```

```
Out[15]: 87.775
```

Observation:- It was also seen that in two rows given below the value of HelpfulnessNumerator is greater than HelpfulnessDenominator which is not practically possible hence these two rows too are removed from calcualtions

```
In [16]: display= pd.read_sql_query("""
         SELECT *
         FROM Reviews
         WHERE Score != 3 AND Id=44737 OR Id=64422
         ORDER BY ProductID
         """, con)

         display.head()
```

```
Out[16]:       Id  ...                                                Text
         0  64422  ...  My son loves spaghetti so I didn't hesitate or...
         1  44737  ...  It was almost a 'love at first bite' - the per...

         [2 rows x 10 columns]
```

```
In [0]: final=final[final.HelpfulnessNumerator<=final.HelpfulnessDenominator]
```

```
In [18]: #Before starting the next phase of preprocessing lets see the number of entries left
         print(final.shape)

         #How many positive and negative reviews are present in our dataset?
         final['Score'].value_counts()
```

```
(87773, 10)
```

```
Out[18]: 1    73592
         0    14181
         Name: Score, dtype: int64

In [19]: final.columns

Out[19]: Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
                'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text'],
               dtype='object')
```

4 [3] Preprocessing

4.1 [3.1]. Preprocessing Review Text

Now that we have finished deduplication our data requires some preprocessing before we go on further with analysis and making the prediction model.

Hence in the Preprocessing phase we do the following in the order below:-

1. Begin by removing the html tags
2. Remove any punctuations or limited set of special characters like , or . or # etc.
3. Check if the word is made up of english letters and is not alpha-numeric
4. Check to see if the length of the word is greater than 2 (as it was researched that there is no adjective in 2-letters)
5. Convert the word to lowercase
6. Remove Stopwords
7. Finally Snowball Stemming the word (it was obsereved to be better than Porter Stemming)

After which we collect the words used to describe positive and negative reviews

```
In [20]: # printing some random reviews
         sent_0 = final['Text'].values[0]
         print(sent_0)
         print("="*50)

         sent_1000 = final['Text'].values[1000]
         print(sent_1000)
         print("="*50)

         sent_1500 = final['Text'].values[1500]
         print(sent_1500)
         print("="*50)

         sent_4900 = final['Text'].values[4900]
         print(sent_4900)
         print("="*50)
```

```
My dogs loves this chicken but its a product from China, so we wont be buying it anymore.  Its
==================================================
The Candy Blocks were a nice visual for the Lego Birthday party but the candy has little taste
```

```
==================================================
was way to hot for my blood, took a bite and did a jig  lol
==================================================
My dog LOVES these treats. They tend to have a very strong fish oil smell. So if you are afraid
==================================================
```

In [21]:
```
# remove urls from text python: https://stackoverflow.com/a/40823105/4084039
sent_0 = re.sub(r"http\S+", "", sent_0)
sent_1000 = re.sub(r"http\S+", "", sent_1000)
sent_150 = re.sub(r"http\S+", "", sent_1500)
sent_4900 = re.sub(r"http\S+", "", sent_4900)

print(sent_0)
```

```
My dogs loves this chicken but its a product from China, so we wont be buying it anymore.  Its
```

In [22]:
```
# https://stackoverflow.com/questions/16206380/python-beautifulsoup-how-to-remove-all-
from bs4 import BeautifulSoup

soup = BeautifulSoup(sent_0, 'lxml')
text = soup.get_text()
print(text)
print("="*50)

soup = BeautifulSoup(sent_1000, 'lxml')
text = soup.get_text()
print(text)
print("="*50)

soup = BeautifulSoup(sent_1500, 'lxml')
text = soup.get_text()
print(text)
print("="*50)

soup = BeautifulSoup(sent_4900, 'lxml')
text = soup.get_text()
print(text)
```

```
My dogs loves this chicken but its a product from China, so we wont be buying it anymore.  Its
==================================================
The Candy Blocks were a nice visual for the Lego Birthday party but the candy has little taste
==================================================
was way to hot for my blood, took a bite and did a jig  lol
==================================================
My dog LOVES these treats. They tend to have a very strong fish oil smell. So if you are afraid
```

```
In [0]: # https://stackoverflow.com/a/47091490/4084039
        import re

        def decontracted(phrase):
            # specific
            phrase = re.sub(r"won't", "will not", phrase)
            phrase = re.sub(r"can\'t", "can not", phrase)

            # general
            phrase = re.sub(r"n\'t", " not", phrase)
            phrase = re.sub(r"\'re", " are", phrase)
            phrase = re.sub(r"\'s", " is", phrase)
            phrase = re.sub(r"\'d", " would", phrase)
            phrase = re.sub(r"\'ll", " will", phrase)
            phrase = re.sub(r"\'t", " not", phrase)
            phrase = re.sub(r"\'ve", " have", phrase)
            phrase = re.sub(r"\'m", " am", phrase)
            return phrase

In [24]: sent_1500 = decontracted(sent_1500)
         print(sent_1500)
         print("="*50)

was way to hot for my blood, took a bite and did a jig  lol
==================================================


In [25]: #remove words with numbers python: https://stackoverflow.com/a/18082370/4084039
         sent_0 = re.sub("\S*\d\S*", "", sent_0).strip()
         print(sent_0)

My dogs loves this chicken but its a product from China, so we wont be buying it anymore.  Its


In [26]: #remove spacial character: https://stackoverflow.com/a/5843547/4084039
         sent_1500 = re.sub('[^A-Za-z0-9]+', ' ', sent_1500)
         print(sent_1500)

was way to hot for my blood took a bite and did a jig lol


In [0]: # https://gist.github.com/sebleier/554280
        # we are removing the words from the stop words list: 'no', 'nor', 'not'
        # <br /><br /> ==> after the above steps, we are getting "br br"
        # we are including them into stop words list
        # instead of <br /> if we have <br/> these tags would have revmoved in the 1st step

        stopwords= set(['br', 'the', 'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves
                    "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him',
```

```
                    'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', '
                    'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "t
                    'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'h
                    'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as
                    'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through
                    'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'o
                    'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'an
                    'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too
                    's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'n
                    've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't"
                    "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mig
                    "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", '
                    'won', "won't", 'wouldn', "wouldn't"])
```

```
In [28]: # Combining all the above stundents
         from tqdm import tqdm
         preprocessed_reviews = []
         # tqdm is for printing the status bar
         for sentance in tqdm(final['Text'].values):
             sentance = re.sub(r"http\S+", "", sentance)
             sentance = BeautifulSoup(sentance, 'lxml').get_text()
             sentance = decontracted(sentance)
             sentance = re.sub("\S*\d\S*", "", sentance).strip()
             sentance = re.sub('[^A-Za-z]+', ' ', sentance)
             # https://gist.github.com/sebleier/554280
             sentance = ' '.join(e.lower() for e in sentance.split() if e.lower() not in stopw
             preprocessed_reviews.append(sentance.strip())
```

```
100%|| 87773/87773 [00:32<00:00, 2664.06it/s]
```

```
In [29]: preprocessed_reviews[87772]
```

```
Out[29]: 'purchased product local store ny kids love quick easy meal put toaster oven toast mi
```

[3.2] Preprocessing Review Summary

```
In [0]: ## Similartly you can do preprocessing for review summary also.
```

```
In [0]: #Before starting the next phase of preprocessing lets see the number of entries left
        print(final.shape)

        #How many positive and negative reviews are present in our dataset?
        final['Summary'].value_counts()
```

```
(160176, 10)
```

```
Out[0]: Delicious!                                                    744
        Delicious                                                     710
        Yummy!                                                        483
        Yummy                                                         380
        Great product                                                 340
        Yum!                                                          339
        Excellent                                                     317
        Great Product                                                 290
        Love it!                                                      285
        Great                                                         239
        Great!                                                        229
        Tasty                                                         183
        Yum                                                           182
        Awesome                                                       179
        Great Coffee                                                  172
        Good stuff                                                    163
        Awesome!                                                      161
        Good Stuff                                                    154
        Excellent!                                                    152
        yummy                                                         152
        Disappointed                                                  151
        Great coffee                                                  145
        great product                                                 143
        Great product!                                                135
        delicious                                                     128
        Wonderful                                                     128
        The Best                                                      124
        Love it                                                       123
        Very good                                                     116
        YUM!                                                          115
                                                                      ...
        High Protein, Low Fat, Tast Great. Expensive.                   1
        A good shampoo, good packaging                                  1
        Fabbri Amarena cherries in syrup                                1
        Caribou Coffee Is Amazing                                       1
        old fashion treat                                               1
        In Limerick Form...                                             1
        Seriously?  This product isn't sugar free.                      1
        Gummi Cherries are WONDERFUL!                                   1
        Aaahhhhhh...                                                    1
        Very nice Christmas chocolates, but they slip                   1
        Finally! Amazing blue cheese olives!!                           1
        Delicious and Rich Flavor                                       1
        Happy Tots                                                      1
        A winner in our house!                                          1
        sad...                                                          1
        dabur vatika olive oil                                          1
        One of the best tasting packaged cookies out there              1
```

```
Works great for energy demanding jobs!                                  1
bad order                                                               1
Wow! You will try to find them - like I am doing -once you taste them.  1
Great on the go food!                                                   1
good substitue for acini de pepi pasta                                  1
good coffee, right amount of pumpkin spice                              1
They are back to being dry and tasteless                                1
U Have Bad Taste Buds If You Don't Like This Stuff                      1
tastes mediacore if not horrible                                        1
Great Chocolate bars!                                                   1
Too much filler material, but chocolate chip the best flavor            1
a really good decaf coffee                                              1
Nothing Miraculous Here                                                 1
Name: Summary, Length: 125515, dtype: int64
```

In [0]:
```
# printing some random reviews
summary_0 = final['Summary'].values[0]
print(summary_0)
print("="*50)

summary_1000 = final['Summary'].values[1000]
print(summary_1000)
print("="*50)

summary_1500 = final['Summary'].values[1500]
print(summary_1500)
print("="*50)

summary_4900 = final['Summary'].values[4900]
print(summary_4900)
print("="*50)
```

```
A classic
==================================================
Diamond dog food
==================================================
Essential for Tonkatsu, etc
==================================================
Better to the last drop
==================================================
```

In [0]:
```
# remove urls from text python: https://stackoverflow.com/a/40823105/4084039
summary_0 = re.sub(r"http\S+", "", summary_0)
summary_1000 = re.sub(r"http\S+", "", summary_1000)
summary_150 = re.sub(r"http\S+", "", summary_1500)
summary_4900 = re.sub(r"http\S+", "", summary_4900)

print(summary_0)
```

```
A classic
```

```
In [0]: # https://stackoverflow.com/questions/16206380/python-beautifulsoup-how-to-remove-all-
        from bs4 import BeautifulSoup

        soup = BeautifulSoup(summary_0, 'lxml')
        text = soup.get_text()
        print(text)
        print("="*50)

        soup = BeautifulSoup(summary_1000, 'lxml')
        text = soup.get_text()
        print(text)
        print("="*50)

        soup = BeautifulSoup(summary_1500, 'lxml')
        text = soup.get_text()
        print(text)
        print("="*50)

        soup = BeautifulSoup(summary_4900, 'lxml')
        text = soup.get_text()
        print(text)
```

```
A classic
==================================================
Diamond dog food
==================================================
Essential for Tonkatsu, etc
==================================================
Better to the last drop
```

```
In [0]: summary_1500 = decontracted(summary_1500)
        print(summary_1500)
        print("="*50)
```

```
Essential for Tonkatsu, etc
==================================================
```

```
In [0]: #remove words with numbers python: https://stackoverflow.com/a/18082370/4084039
        summary_0 = re.sub("\S*\d\S*", "", summary_0).strip()
        print(summary_0)
```

```
A classic
```

```
In [0]: #remove spacial character: https://stackoverflow.com/a/5843547/4084039
        summary_1500 = re.sub('[^A-Za-z0-9]+', ' ', summary_1500)
        print(summary_1500)
```

```
Essential for Tonkatsu etc
```

```
In [0]: # https://gist.github.com/sebleier/554280
        # we are removing the words from the stop words list: 'no', 'nor', 'not'
        # <br /><br /> ==> after the above steps, we are getting "br br"
        # we are including them into stop words list
        # instead of <br /> if we have <br/> these tags would have revmoved in the 1st step

        stopwords= set(['br', 'the', 'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves
                    "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him',
                    'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', '
                    'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "th
                    'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'ha
                    'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as
                    'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through
                    'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'ov
                    'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any
                    'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too
                    's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'no
                    've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't",
                    "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'migh
                    "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'w
                    'won', "won't", 'wouldn', "wouldn't"])
```

```
In [0]: # Combining all the above stundents
        from tqdm import tqdm
        preprocessed_summary = []
        # tqdm is for printing the status bar
        for sentance in tqdm(final['Summary'].values):
            sentance = re.sub(r"http\S+", "", sentance)
            sentance = BeautifulSoup(sentance, 'lxml').get_text()
            sentance = decontracted(sentance)
            sentance = re.sub("\S*\d\S*", "", sentance).strip()
            sentance = re.sub('[^A-Za-z]+', ' ', sentance)
            # https://gist.github.com/sebleier/554280
            sentance = ' '.join(e.lower() for e in sentance.split() if e.lower() not in stopwo
            preprocessed_summary.append(sentance.strip())
```

```
 58%|    | 92203/160176 [00:21<00:14, 4571.66it/s]/usr/local/lib/python3.6/dist-packages/bs4/_
  ' Beautiful Soup.' % markup)
100%|| 160176/160176 [00:36<00:00, 4391.77it/s]
```

Train-Test Split

```
In [0]: X=preprocessed_reviews[:]
        y=final['Score'][:]

In [0]: from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X,y , test_size=0.30, random_state=
```

---

5 [4] Featurization

5.1 [4.1] BAG OF WORDS

```
In [32]: #BoW
         count_vect = CountVectorizer() #in scikit-learn
         count_vect.fit(X_train)
         print("some feature names ", count_vect.get_feature_names()[:10])
         print('='*50)

         final_counts = count_vect.transform(X_train)
         print("the type of count vectorizer ",type(final_counts))
         print("the shape of out text BOW vectorizer ",final_counts.get_shape())
         print("the number of unique words ", final_counts.get_shape()[1])
```

```
some feature names  ['aa', 'aaa', 'aaaa', 'aaaaa', 'aaaaaaaaaaa', 'aaaaaaaaaaaaaa', 'aaaaaaa
==================================================
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
the shape of out text BOW vectorizer  (61441, 46115)
the number of unique words  46115
```

5.2 [4.2] Bi-Grams and n-Grams.

```
In [81]: #bi-gram, tri-gram and n-gram

         #removing stop words like "not" should be avoided before building n-grams
         # count_vect = CountVectorizer(ngram_range=(1,2))
         # please do read the CountVectorizer documentation http://scikit-learn.org/stable/mod

         # you can choose these numebrs min_df=10, max_features=5000, of your choice
         count_vect = CountVectorizer(ngram_range=(1,2), min_df=10, max_features=5000)
         final_bigram_counts = count_vect.fit_transform(X_train)
         print("the type of count vectorizer ",type(final_bigram_counts))
         print("the shape of out text BOW vectorizer ",final_bigram_counts.get_shape())
         print("the number of unique words including both unigrams and bigrams ", final_bigram
```

```
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
the shape of out text BOW vectorizer  (61441, 5000)
```

```
the number of unique words including both unigrams and bigrams  5000
```

```
In [0]: X_test1 =  count_vect.transform(X_test)
```

```
In [35]: from sklearn.ensemble import RandomForestClassifier
         from sklearn.model_selection import GridSearchCV


         grid_params = dict(n_estimators = [10,20,30,50,100],
                            max_depth= [3,6,10,15,20] )

         rf = RandomForestClassifier()
         rf_clf = GridSearchCV(rf, param_grid=grid_params, n_jobs=-1, verbose=30,return_train_s
         rf_clf = rf_clf.fit(final_bigram_counts, y_train)


         results = rf_clf.cv_results_
```

```
Fitting 3 folds for each of 25 candidates, totalling 75 fits


[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done   1 tasks      | elapsed:    2.1s
[Parallel(n_jobs=-1)]: Done   2 tasks      | elapsed:    2.1s
[Parallel(n_jobs=-1)]: Done   3 tasks      | elapsed:    2.6s
[Parallel(n_jobs=-1)]: Done   4 tasks      | elapsed:    2.8s
[Parallel(n_jobs=-1)]: Done   5 tasks      | elapsed:    3.3s
[Parallel(n_jobs=-1)]: Done   6 tasks      | elapsed:    3.4s
[Parallel(n_jobs=-1)]: Done   7 tasks      | elapsed:    4.2s
[Parallel(n_jobs=-1)]: Done   8 tasks      | elapsed:    4.3s
[Parallel(n_jobs=-1)]: Done   9 tasks      | elapsed:    5.1s
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:    5.6s
[Parallel(n_jobs=-1)]: Done  11 tasks      | elapsed:    6.5s
[Parallel(n_jobs=-1)]: Done  12 tasks      | elapsed:    6.8s
[Parallel(n_jobs=-1)]: Done  13 tasks      | elapsed:    9.0s
[Parallel(n_jobs=-1)]: Done  14 tasks      | elapsed:    9.2s
[Parallel(n_jobs=-1)]: Done  15 tasks      | elapsed:    9.8s
[Parallel(n_jobs=-1)]: Done  16 tasks      | elapsed:   10.3s
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:   10.8s
[Parallel(n_jobs=-1)]: Done  18 tasks      | elapsed:   11.5s
[Parallel(n_jobs=-1)]: Done  19 tasks      | elapsed:   11.5s
[Parallel(n_jobs=-1)]: Done  20 tasks      | elapsed:   12.3s
[Parallel(n_jobs=-1)]: Done  21 tasks      | elapsed:   12.3s
[Parallel(n_jobs=-1)]: Done  22 tasks      | elapsed:   13.5s
[Parallel(n_jobs=-1)]: Done  23 tasks      | elapsed:   13.5s
[Parallel(n_jobs=-1)]: Done  24 tasks      | elapsed:   14.7s
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:   15.2s
[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed:   16.6s
```

```
[Parallel(n_jobs=-1)]: Done  27 tasks      | elapsed:   17.1s
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:   20.3s
[Parallel(n_jobs=-1)]: Done  29 tasks      | elapsed:   20.5s
[Parallel(n_jobs=-1)]: Done  30 tasks      | elapsed:   21.3s
[Parallel(n_jobs=-1)]: Done  31 tasks      | elapsed:   22.2s
[Parallel(n_jobs=-1)]: Done  32 tasks      | elapsed:   22.8s
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:   23.8s
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:   24.0s
[Parallel(n_jobs=-1)]: Done  35 tasks      | elapsed:   24.9s
[Parallel(n_jobs=-1)]: Done  36 tasks      | elapsed:   25.1s
[Parallel(n_jobs=-1)]: Done  37 tasks      | elapsed:   26.4s
[Parallel(n_jobs=-1)]: Done  38 tasks      | elapsed:   26.5s
[Parallel(n_jobs=-1)]: Done  39 tasks      | elapsed:   28.1s
[Parallel(n_jobs=-1)]: Done  40 tasks      | elapsed:   28.6s
[Parallel(n_jobs=-1)]: Done  41 tasks      | elapsed:   30.4s
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:   30.9s
[Parallel(n_jobs=-1)]: Done  43 tasks      | elapsed:   34.8s
[Parallel(n_jobs=-1)]: Done  44 tasks      | elapsed:   35.2s
[Parallel(n_jobs=-1)]: Done  45 tasks      | elapsed:   36.2s
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:   37.2s
[Parallel(n_jobs=-1)]: Done  47 tasks      | elapsed:   38.1s
[Parallel(n_jobs=-1)]: Done  48 tasks      | elapsed:   39.0s
[Parallel(n_jobs=-1)]: Done  49 tasks      | elapsed:   39.5s
[Parallel(n_jobs=-1)]: Done  50 tasks      | elapsed:   40.4s
[Parallel(n_jobs=-1)]: Done  51 tasks      | elapsed:   40.8s
[Parallel(n_jobs=-1)]: Done  52 tasks      | elapsed:   42.5s
[Parallel(n_jobs=-1)]: Done  53 tasks      | elapsed:   43.0s
[Parallel(n_jobs=-1)]: Done  54 tasks      | elapsed:   44.5s
[Parallel(n_jobs=-1)]: Done  55 tasks      | elapsed:   46.7s
[Parallel(n_jobs=-1)]: Done  56 tasks      | elapsed:   47.8s
[Parallel(n_jobs=-1)]: Done  57 tasks      | elapsed:   50.0s
[Parallel(n_jobs=-1)]: Done  58 tasks      | elapsed:   54.4s
[Parallel(n_jobs=-1)]: Done  59 tasks      | elapsed:   56.6s
[Parallel(n_jobs=-1)]: Done  60 tasks      | elapsed:   57.9s
[Parallel(n_jobs=-1)]: Done  61 tasks      | elapsed:   59.1s
[Parallel(n_jobs=-1)]: Done  62 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done  63 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done  64 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done  65 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done  66 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done  67 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done  68 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done  69 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done  70 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done  71 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done  72 tasks      | elapsed:  1.3min
[Parallel(n_jobs=-1)]: Done  75 out of  75 | elapsed:  1.5min remaining:    0.0s
[Parallel(n_jobs=-1)]: Done  75 out of  75 | elapsed:  1.5min finished
```

```
In [38]: print("Best parameters: ", rf_clf.best_params_)
         print("Best cross-validation score: {:.3f}".format(rf_clf.best_score_))
```

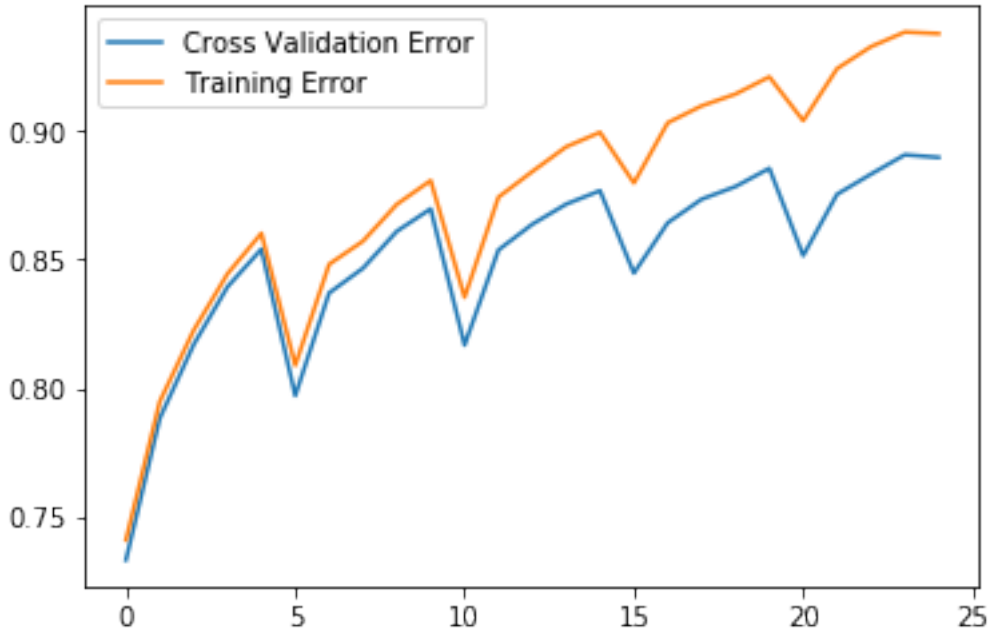
```
Best parameters:  {'max_depth': 20, 'n_estimators': 100}
Best cross-validation score: 0.891
```

```
In [0]: pred = rf_clf.predict(X_test1)
        pred_train = rf_clf.predict(final_bigram_counts)
```

```
In [0]: from sklearn.metrics import accuracy_score
        from sklearn.metrics import roc_curve
        import sklearn.metrics as metrics
        from sklearn.metrics import roc_auc_score

        train_error=rf_clf.cv_results_['mean_train_score']
        cv_error = rf_clf.cv_results_['mean_test_score']
        score=roc_auc_score(y_train, pred_train)

        estimator=rf_clf.best_params_['n_estimators']
        depth=rf_clf.best_params_['max_depth']
```

Error plots

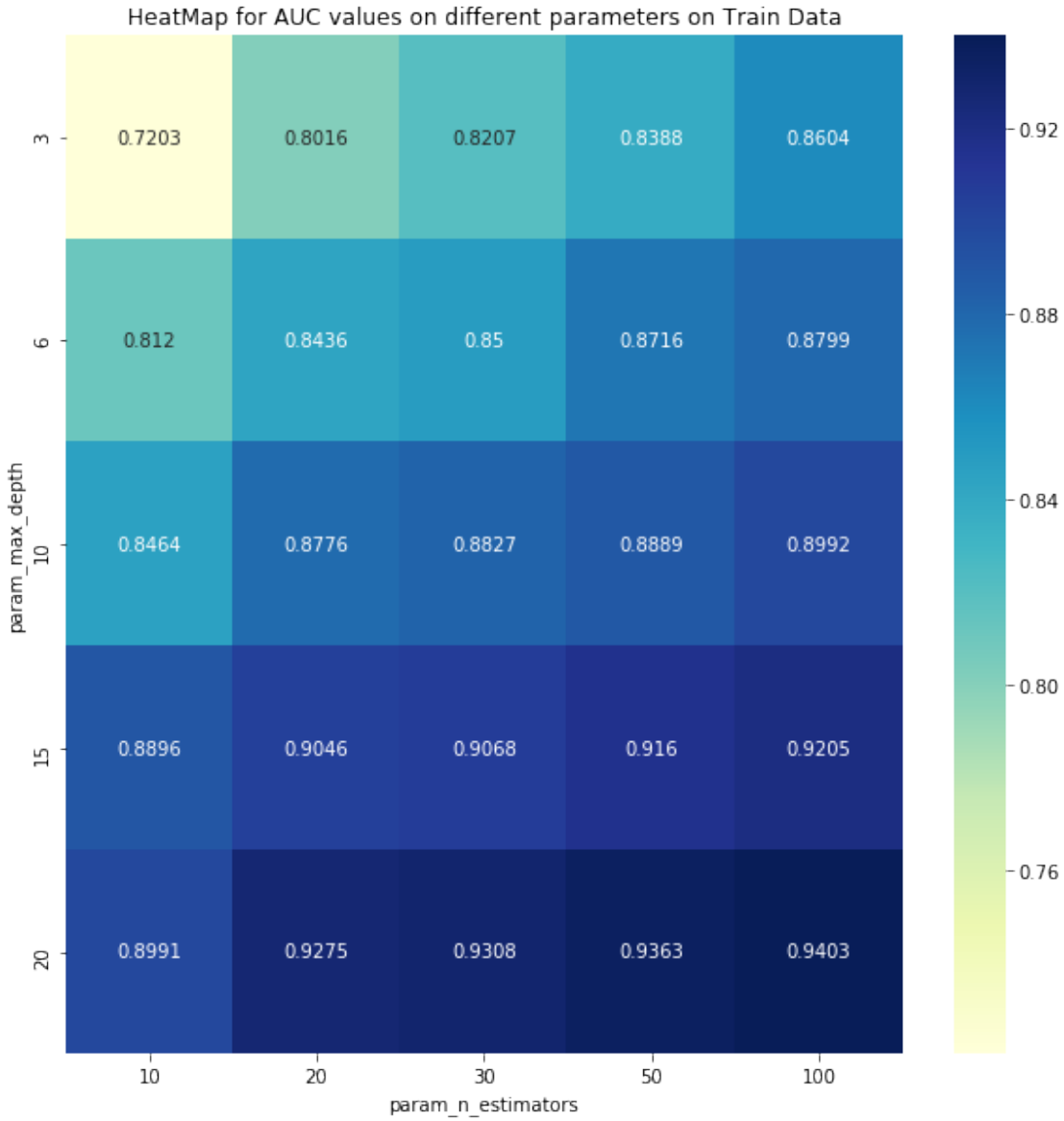
```
In [0]: plt.plot(cv_error, label='Cross Validation Error')
        plt.plot(train_error, label='Training Error')
        plt.legend()
        plt.show()
```



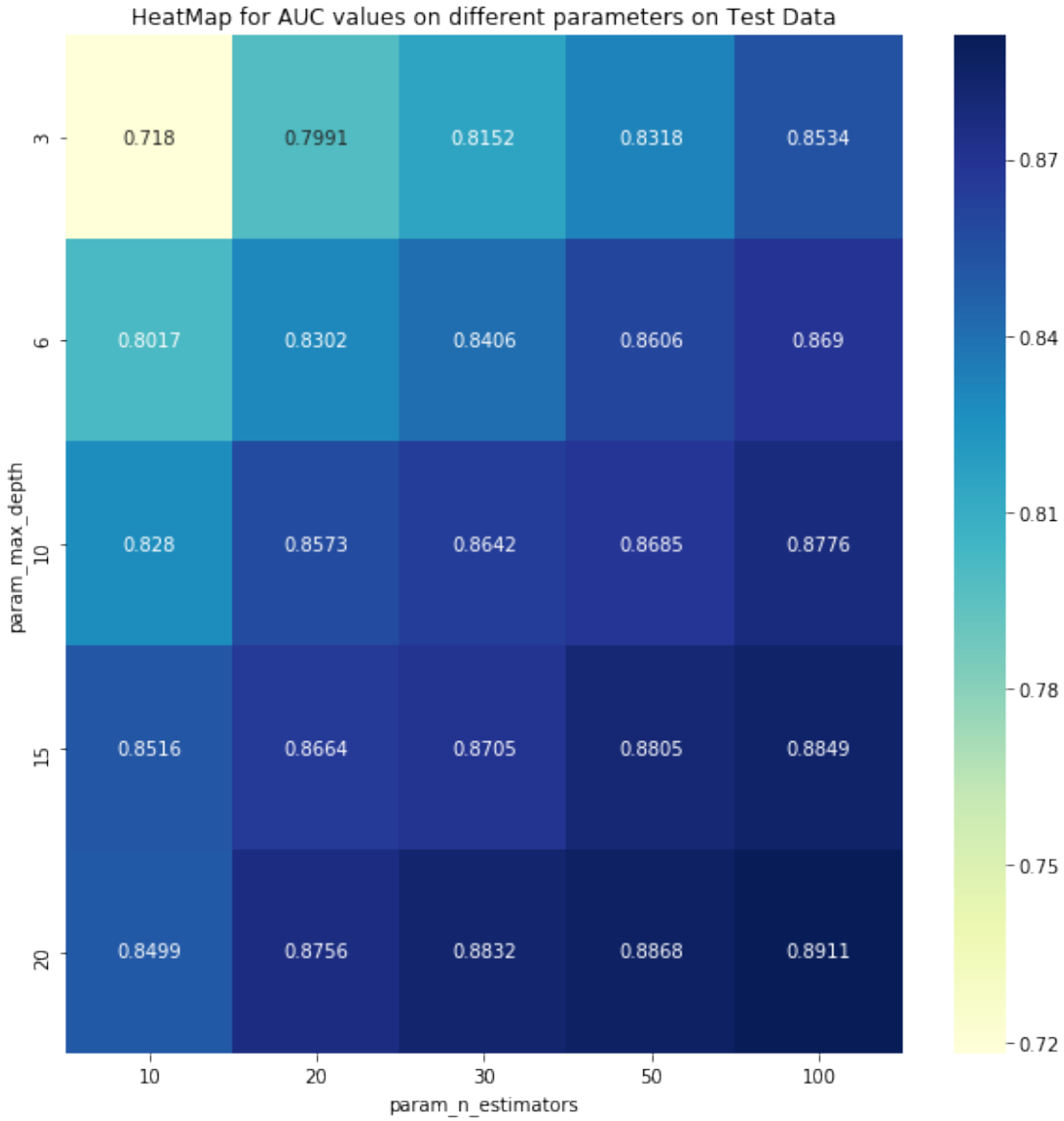
```
In [0]: n_estimators =[10,20,30,50,100]
        max_depth= [3,6,10,15,20]
```

5.2.1 HeatMap for AUC vs N_estimators and Max_Depth

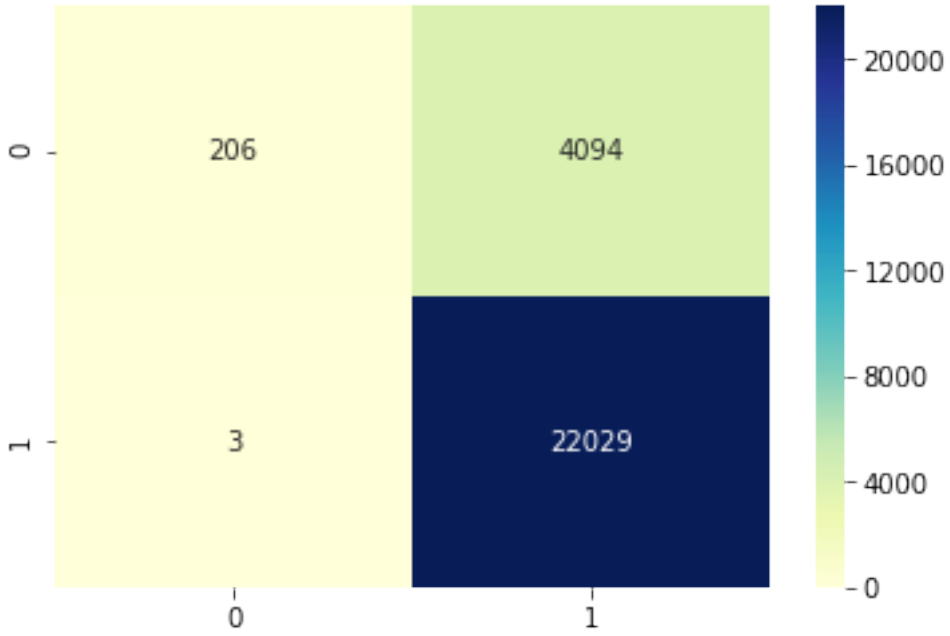
```
In [40]: df_new = pd.DataFrame(rf_clf.cv_results_)
         max_params = df_new.groupby(['param_max_depth','param_n_estimators']).max()
         max_params = max_params.unstack()[['mean_test_score', 'mean_train_score']]

         fig, ax = plt.subplots(figsize=(10,10))         # Sample figsize in inches

         sns.heatmap(max_params.mean_train_score, annot=True,cmap="YlGnBu", fmt='.4g',ax=ax)
         plt.title('HeatMap for AUC values on different parameters on Train Data')
         plt.show()
```




```
In [41]: df_new = pd.DataFrame(rf_clf.cv_results_)
         max_params = df_new.groupby(['param_max_depth','param_n_estimators']).max()
         max_params = max_params.unstack()[['mean_test_score', 'mean_train_score']]

         fig, ax = plt.subplots(figsize=(10,10))         # Sample figsize in inches

         sns.heatmap(max_params.mean_test_score, annot=True,cmap="YlGnBu", fmt='.4g',ax=ax)
         plt.title('HeatMap for AUC values on different parameters on Test Data')
         plt.show()
```

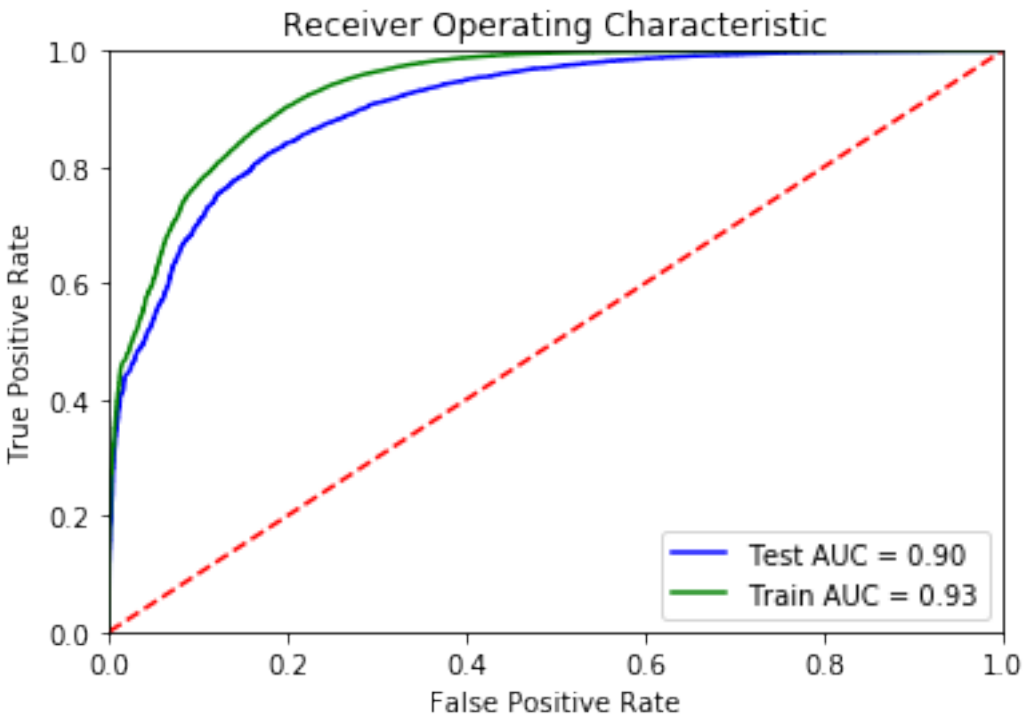



```
In [0]: pred_test = rf_clf.predict_proba(X_test1)[:,1]
        pred_train =  rf_clf.predict_proba(final_bigram_counts)[:,1]

In [0]: from sklearn.metrics import confusion_matrix

        cm = confusion_matrix(y_test, pred_test.round())
        sns.heatmap(cm, annot=True, fmt="d",cmap="YlGnBu")
        plt.show()
```




```
In [0]: fpr, tpr, thresholds = roc_curve(y_test, pred_test)
        fpr2, tpr2, thresholds = roc_curve(y_train, pred_train)

        score_test = roc_auc_score(y_test, pred_test)
        score_train = roc_auc_score(y_train, pred_train)
```

```
In [0]: roc_auc_test = metrics.auc(fpr, tpr)
        roc_auc_train = metrics.auc(fpr2, tpr2)
        plt.title('Receiver Operating Characteristic')
        plt.plot(fpr, tpr, 'b', label='Test AUC = %0.2f' % score_test)
        plt.plot(fpr2, tpr2, 'g', label = 'Train AUC = %0.2f' % score_train)
        plt.legend(loc = 'lower right')
        plt.plot([0, 1], [0, 1],'r--')
        plt.xlim([0, 1])
        plt.ylim([0, 1])
        plt.ylabel('True Positive Rate')
        plt.xlabel('False Positive Rate')
        plt.legend()
        plt.show()
```



5.2.2 Wordcloud Top-20 Features

```
In [0]: model = RandomForestClassifier(n_estimators= 100 , max_depth=20, class_weight='balanced
        model.fit(final_bigram_counts, y_train)
```

```
Out[0]: RandomForestClassifier(bootstrap=True, class_weight='balanced',
                               criterion='gini', max_depth=20, max_features='auto',
                               max_leaf_nodes=None, min_impurity_decrease=0.0,
                               min_impurity_split=None, min_samples_leaf=1,
                               min_samples_split=2, min_weight_fraction_leaf=0.0,
                               n_estimators=100, n_jobs=None, oob_score=False,
                               random_state=None, verbose=0, warm_start=False)
```

```
In [0]: # Please write all the code with proper documentation
        # Copied from wordcloud documentation
        # https://towardsdatascience.com/running-random-forests-inspect-the-feature-importance
        from wordcloud import WordCloud, STOPWORDS

        feat_imp = model.feature_importances_

        count_features=count_vect.get_feature_names()
        feature_importances = pd.DataFrame(feat_imp,index = count_features, columns=['importan

        a = feature_importances.iloc[0:20]
```

```
comment_words = ' '
for val in a.index:
    val = str(val)
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '

        stopwords = set(STOPWORDS)
wordcloud = WordCloud(width = 600, height = 600,
                background_color ='Black',
                stopwords = stopwords,
                min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```

5.2.3 Applying XGBOOST Classifier

```
In [42]: from sklearn.model_selection import GridSearchCV


         grid_params = {'n_estimators': [5, 10, 15,20,50,100],
                        'max_depth': [2, 5,10, 15]}

         xg_clf = xgb.XGBClassifier()
         xg_clf = GridSearchCV(xg_clf, param_grid=grid_params, n_jobs=-1, verbose=30,return_tra
         xg_clf = xg_clf.fit(final_bigram_counts, y_train)
```

```
        results =xg_clf.cv_results_

Fitting 3 folds for each of 24 candidates, totalling 72 fits


[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done   1 tasks      | elapsed:    4.4s
[Parallel(n_jobs=-1)]: Done   2 tasks      | elapsed:    4.4s
[Parallel(n_jobs=-1)]: Done   3 tasks      | elapsed:    6.6s
[Parallel(n_jobs=-1)]: Done   4 tasks      | elapsed:    7.0s
[Parallel(n_jobs=-1)]: Done   5 tasks      | elapsed:    9.1s
[Parallel(n_jobs=-1)]: Done   6 tasks      | elapsed:    9.4s
[Parallel(n_jobs=-1)]: Done   7 tasks      | elapsed:   11.9s
[Parallel(n_jobs=-1)]: Done   8 tasks      | elapsed:   12.3s
[Parallel(n_jobs=-1)]: Done   9 tasks      | elapsed:   14.8s
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:   15.4s
[Parallel(n_jobs=-1)]: Done  11 tasks      | elapsed:   17.9s
[Parallel(n_jobs=-1)]: Done  12 tasks      | elapsed:   18.5s
[Parallel(n_jobs=-1)]: Done  13 tasks      | elapsed:   23.1s
[Parallel(n_jobs=-1)]: Done  14 tasks      | elapsed:   23.7s
[Parallel(n_jobs=-1)]: Done  15 tasks      | elapsed:   28.2s
[Parallel(n_jobs=-1)]: Done  16 tasks      | elapsed:   32.0s
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:   36.6s
[Parallel(n_jobs=-1)]: Done  18 tasks      | elapsed:   39.2s
[Parallel(n_jobs=-1)]: Done  19 tasks      | elapsed:   40.2s
[Parallel(n_jobs=-1)]: Done  20 tasks      | elapsed:   41.7s
[Parallel(n_jobs=-1)]: Done  21 tasks      | elapsed:   42.8s
[Parallel(n_jobs=-1)]: Done  22 tasks      | elapsed:   45.0s
[Parallel(n_jobs=-1)]: Done  23 tasks      | elapsed:   46.2s
[Parallel(n_jobs=-1)]: Done  24 tasks      | elapsed:   48.3s
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:   50.2s
[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed:   52.3s
[Parallel(n_jobs=-1)]: Done  27 tasks      | elapsed:   54.2s
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:   57.0s
[Parallel(n_jobs=-1)]: Done  29 tasks      | elapsed:   59.0s
[Parallel(n_jobs=-1)]: Done  30 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done  31 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done  32 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:  1.3min
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done  35 tasks      | elapsed:  1.6min
[Parallel(n_jobs=-1)]: Done  36 tasks      | elapsed:  1.7min
[Parallel(n_jobs=-1)]: Done  37 tasks      | elapsed:  1.7min
[Parallel(n_jobs=-1)]: Done  38 tasks      | elapsed:  1.8min
[Parallel(n_jobs=-1)]: Done  39 tasks      | elapsed:  1.8min
[Parallel(n_jobs=-1)]: Done  40 tasks      | elapsed:  1.9min
[Parallel(n_jobs=-1)]: Done  41 tasks      | elapsed:  1.9min
```

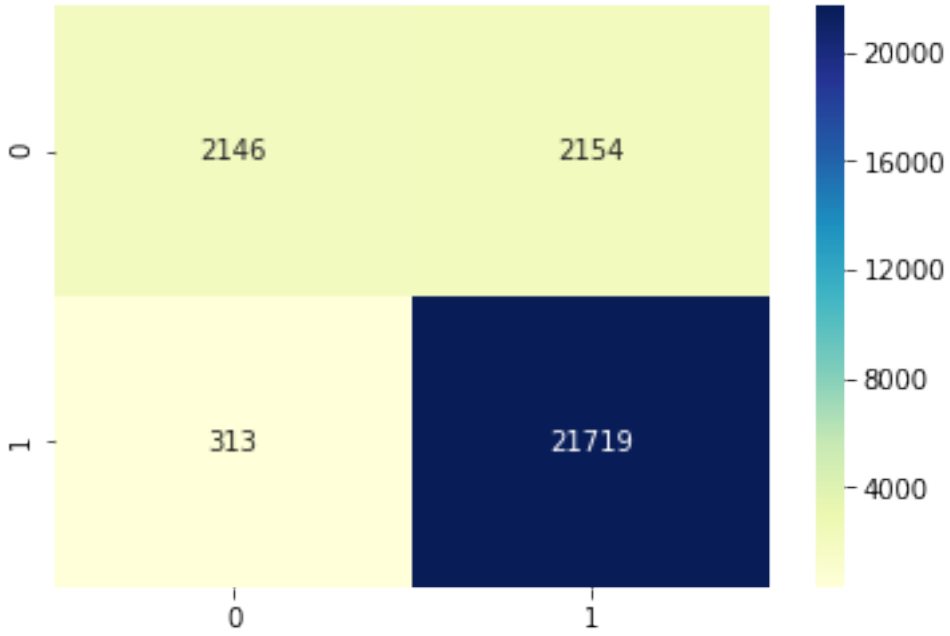
```
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.9min
[Parallel(n_jobs=-1)]: Done  43 tasks      | elapsed:  2.0min
[Parallel(n_jobs=-1)]: Done  44 tasks      | elapsed:  2.1min
[Parallel(n_jobs=-1)]: Done  45 tasks      | elapsed:  2.1min
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:  2.2min
[Parallel(n_jobs=-1)]: Done  47 tasks      | elapsed:  2.2min
[Parallel(n_jobs=-1)]: Done  48 tasks      | elapsed:  2.3min
[Parallel(n_jobs=-1)]: Done  49 tasks      | elapsed:  2.5min
[Parallel(n_jobs=-1)]: Done  50 tasks      | elapsed:  2.7min
[Parallel(n_jobs=-1)]: Done  51 tasks      | elapsed:  2.8min
[Parallel(n_jobs=-1)]: Done  52 tasks      | elapsed:  3.2min
[Parallel(n_jobs=-1)]: Done  53 tasks      | elapsed:  3.4min
[Parallel(n_jobs=-1)]: Done  54 tasks      | elapsed:  3.5min
[Parallel(n_jobs=-1)]: Done  55 tasks      | elapsed:  3.6min
[Parallel(n_jobs=-1)]: Done  56 tasks      | elapsed:  3.6min
[Parallel(n_jobs=-1)]: Done  57 tasks      | elapsed:  3.8min
[Parallel(n_jobs=-1)]: Done  58 tasks      | elapsed:  3.8min
[Parallel(n_jobs=-1)]: Done  59 tasks      | elapsed:  3.9min
[Parallel(n_jobs=-1)]: Done  60 tasks      | elapsed:  3.9min
[Parallel(n_jobs=-1)]: Done  61 tasks      | elapsed:  4.1min
[Parallel(n_jobs=-1)]: Done  62 tasks      | elapsed:  4.1min
[Parallel(n_jobs=-1)]: Done  63 tasks      | elapsed:  4.2min
[Parallel(n_jobs=-1)]: Done  64 tasks      | elapsed:  4.3min
[Parallel(n_jobs=-1)]: Done  65 tasks      | elapsed:  4.4min
[Parallel(n_jobs=-1)]: Done  66 tasks      | elapsed:  4.5min
[Parallel(n_jobs=-1)]: Done  67 tasks      | elapsed:  4.9min
[Parallel(n_jobs=-1)]: Done  68 tasks      | elapsed:  5.0min
[Parallel(n_jobs=-1)]: Done  69 tasks      | elapsed:  5.4min
[Parallel(n_jobs=-1)]: Done  72 out of  72 | elapsed:  6.6min remaining:    0.0s
[Parallel(n_jobs=-1)]: Done  72 out of  72 | elapsed:  6.6min finished
```

```
In [43]: print("Best parameters: ", xg_clf.best_params_)
         print("Best cross-validation score: {:.3f}".format(xg_clf.best_score_))
```

```
Best parameters:  {'max_depth': 15, 'n_estimators': 100}
Best cross-validation score: 0.930
```

```
In [0]: pred_test = xg_clf.predict(X_test1)
        pred_train = xg_clf.predict(final_bigram_counts)
```

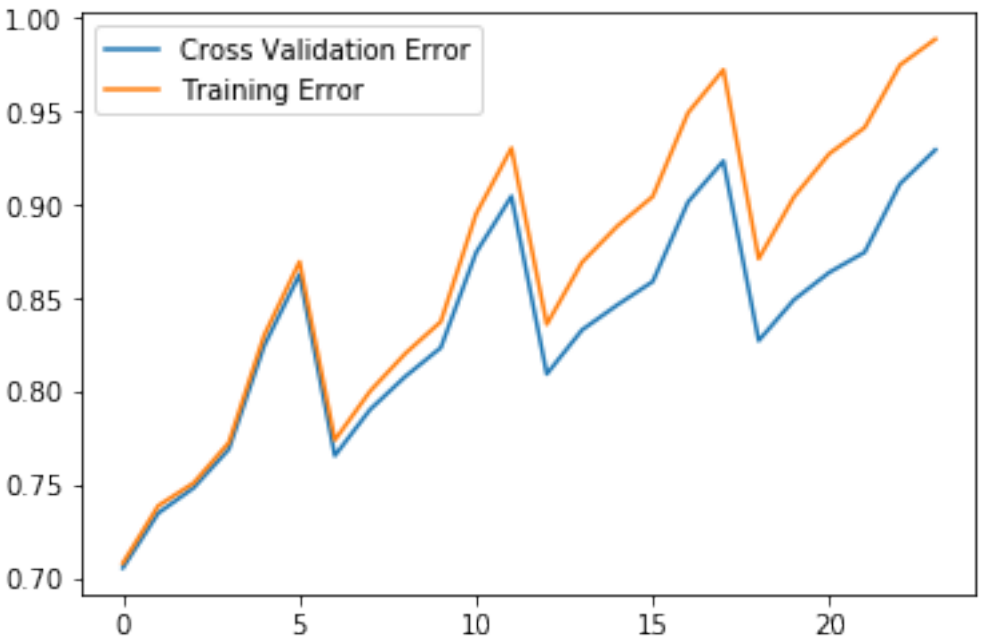
```
In [0]: cm = confusion_matrix(y_test, pred_test.round())
        sns.heatmap(cm, annot=True, fmt="d",cmap="YlGnBu")
        plt.show()
```




```
In [0]: from sklearn.metrics import accuracy_score
        from sklearn.metrics import roc_curve
        import sklearn.metrics as metrics
        from sklearn.metrics import roc_auc_score

        train_error=xg_clf.cv_results_['mean_train_score']
        cv_error = xg_clf.cv_results_['mean_test_score']
        score=roc_auc_score(y_train, pred_train)

        estimator=xg_clf.best_params_['n_estimators']
        depth=xg_clf.best_params_['max_depth']
```

Error Plots

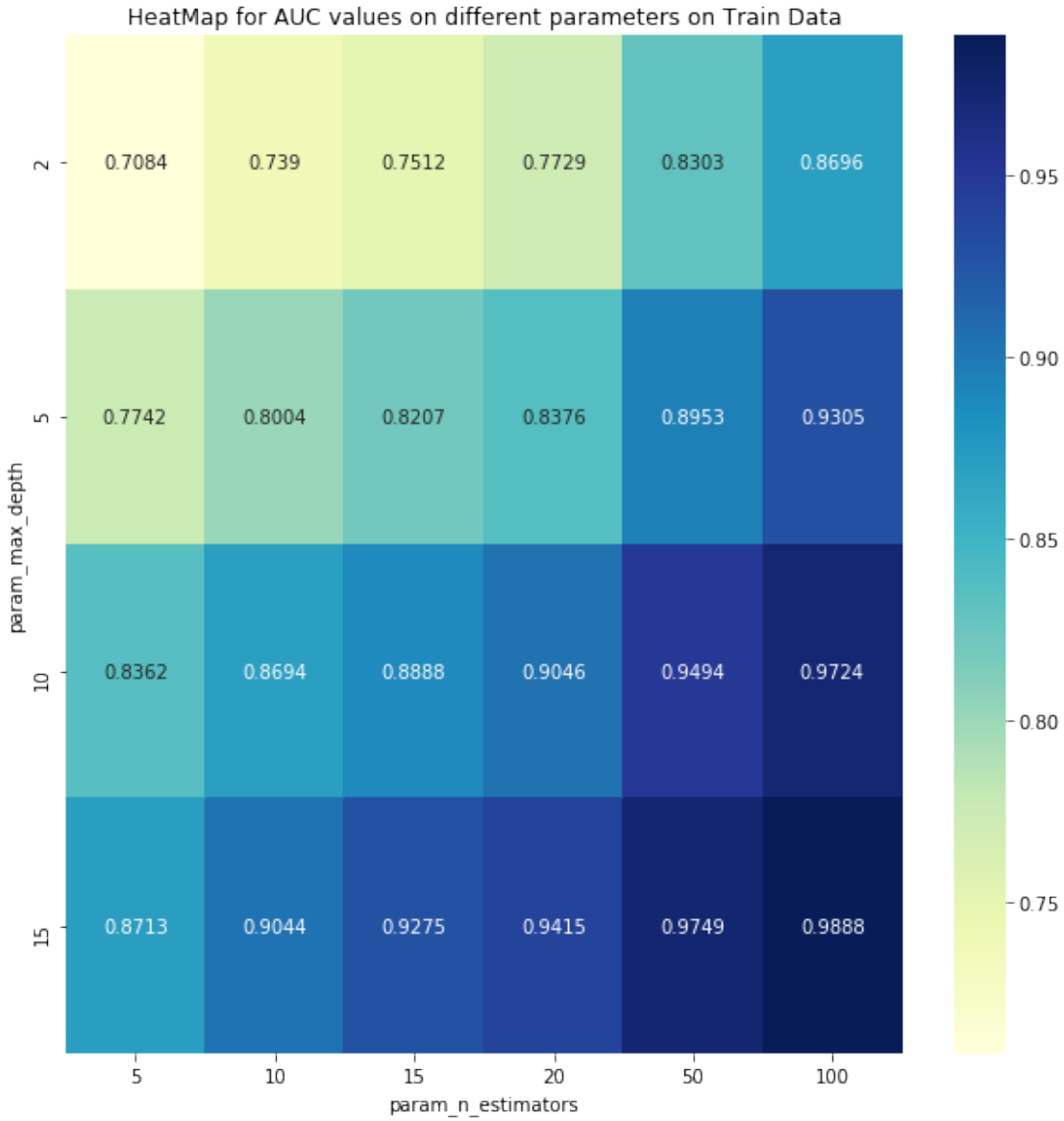
```
In [0]: plt.plot(cv_error, label='Cross Validation Error')
        plt.plot(train_error, label='Training Error')
        plt.legend()
        plt.show()
```



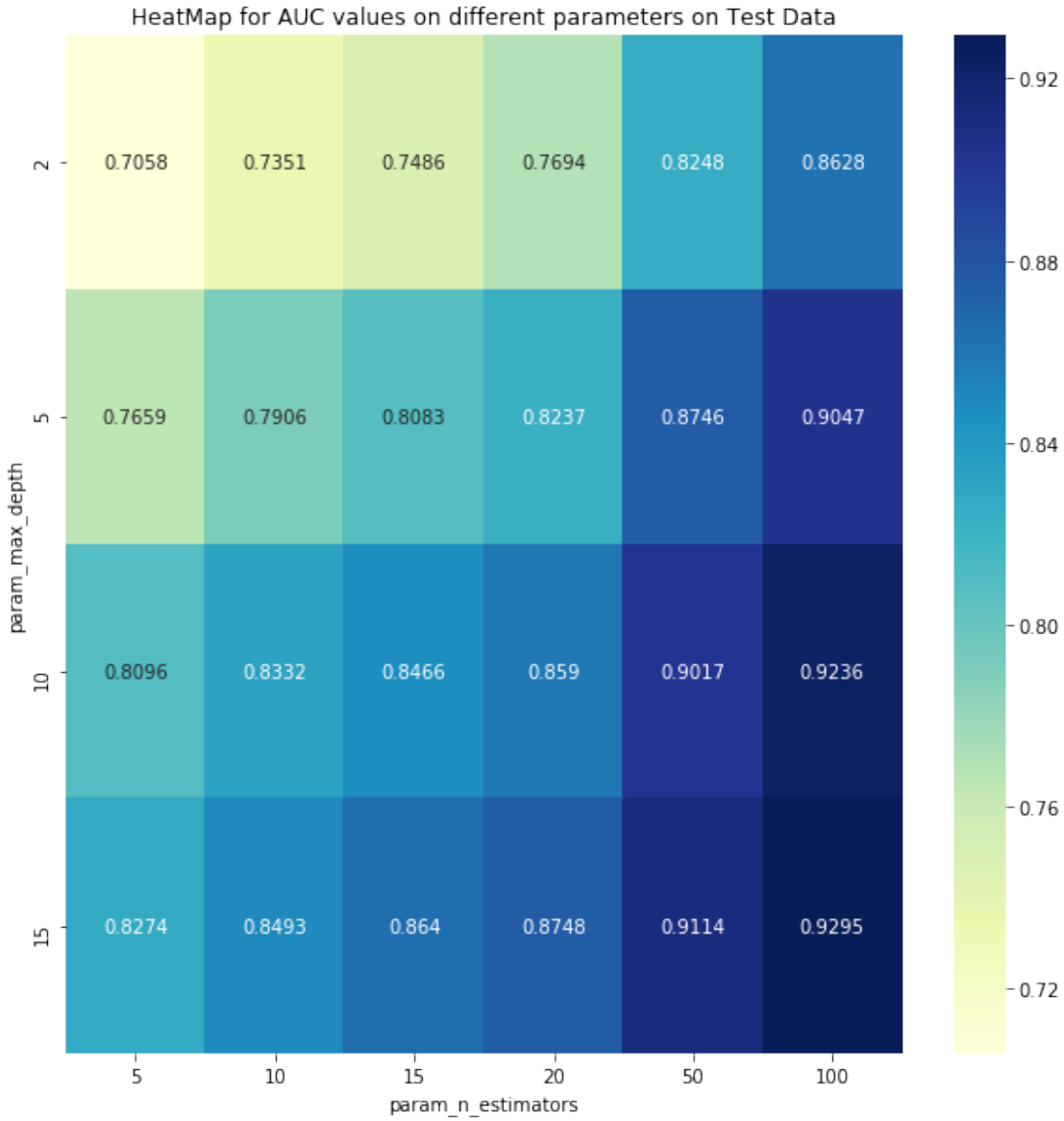

```
In [0]: n_estimators =[5, 10, 15,20,50,100]
        max_depth= [2, 5,10, 15]
```

5.2.4 HeatMap for AUC vs N_estimators and Max_Depth

```
In [44]: df_new = pd.DataFrame(xg_clf.cv_results_)
         max_params = df_new.groupby(['param_max_depth','param_n_estimators']).max()
         max_params = max_params.unstack()[['mean_test_score', 'mean_train_score']]

         fig, ax = plt.subplots(figsize=(10,10))         # Sample figsize in inches

         sns.heatmap(max_params.mean_train_score, annot=True,cmap="YlGnBu", fmt='.4g',ax=ax)
         plt.title('HeatMap for AUC values on different parameters on Train Data')
         plt.show()
```

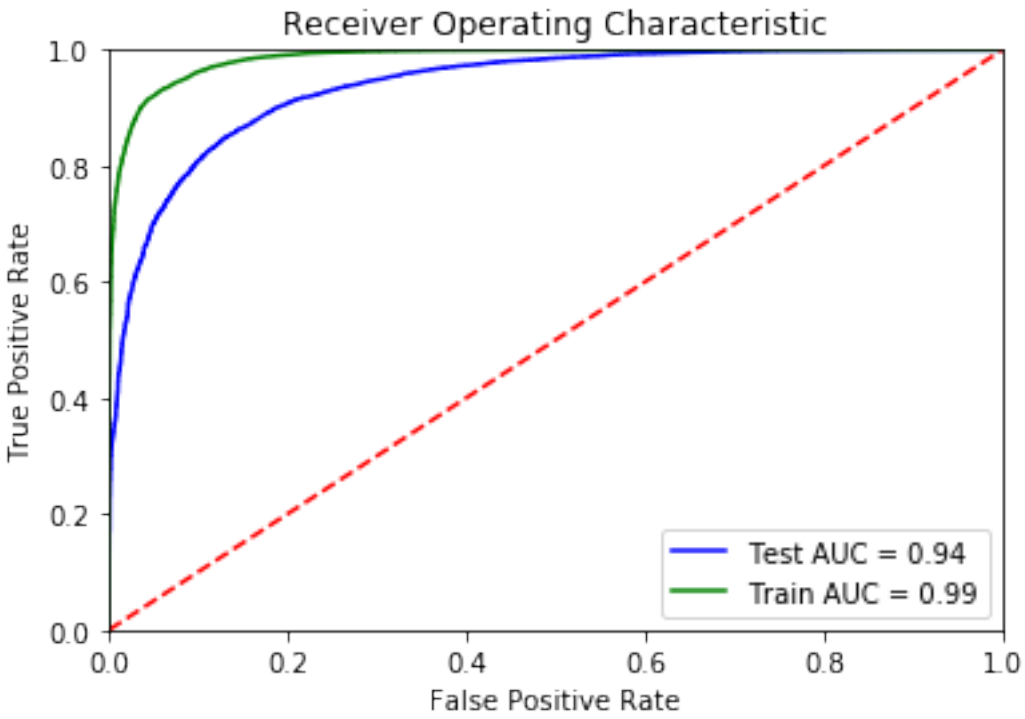



```
In [45]: df_new = pd.DataFrame(xg_clf.cv_results_)
         max_params = df_new.groupby(['param_max_depth','param_n_estimators']).max()
         max_params = max_params.unstack()[['mean_test_score', 'mean_train_score']]

         fig, ax = plt.subplots(figsize=(10,10))         # Sample figsize in inches

         sns.heatmap(max_params.mean_test_score, annot=True,cmap="YlGnBu", fmt='.4g',ax=ax)
         plt.title('HeatMap for AUC values on different parameters on Test Data')
         plt.show()
```




```
In [0]: pred_test = xg_clf.predict_proba(X_test1)[:,1]
        pred_train =  xg_clf.predict_proba(final_bigram_counts)[:,1]

In [0]: fpr, tpr, thresholds = roc_curve(y_test, pred_test)
        fpr2, tpr2, thresholds = roc_curve(y_train, pred_train)

        score_test = roc_auc_score(y_test, pred_test)
        score_train = roc_auc_score(y_train, pred_train)

In [0]: roc_auc_test = metrics.auc(fpr, tpr)
        roc_auc_train = metrics.auc(fpr2, tpr2)
        plt.title('Receiver Operating Characteristic')
```

```
plt.plot(fpr, tpr, 'b', label='Test AUC = %0.2f' % score_test)
plt.plot(fpr2, tpr2, 'g', label = 'Train AUC = %0.2f' % score_train)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend()
plt.show()
```

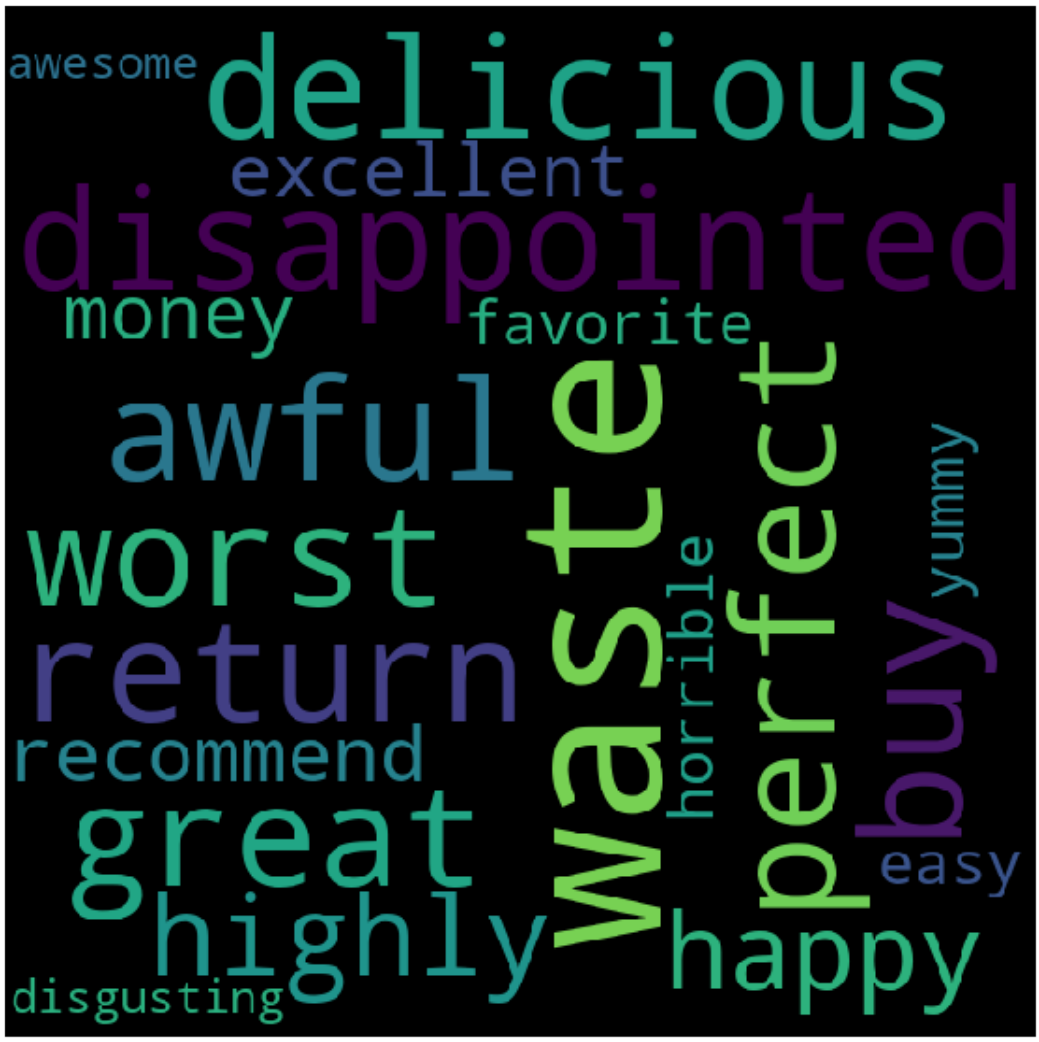


5.2.5 Wordcloud for Top-20 Features

```
In [0]: model = xgb.XGBClassifier(n_estimators= 100 , max_depth=15, class_weight='balanced')
        model.fit(final_bigram_counts, y_train)
```

```
Out[0]: XGBClassifier(base_score=0.5, booster='gbtree', class_weight='balanced',
                      colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
                      gamma=0, learning_rate=0.1, max_delta_step=0, max_depth=15,
                      min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
                      nthread=None, objective='binary:logistic', random_state=0,
                      reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
                      silent=None, subsample=1, verbosity=1)
```

```
In [0]: # Please write all the code with proper documentation
        # Copied from wordcloud documentation
        # https://towardsdatascience.com/running-random-forests-inspect-the-feature-importance
        from wordcloud import WordCloud, STOPWORDS

        feat_imp = model.feature_importances_

        count_features=count_vect.get_feature_names()
        feature_importances = pd.DataFrame(feat_imp,index = count_features, columns=['importan

        a = feature_importances.iloc[0:20]
        comment_words = ' '
        for val in a.index:
            val = str(val)
            tokens = val.split()

            # Converts each token into lowercase
            for i in range(len(tokens)):
                tokens[i] = tokens[i].lower()

            for words in tokens:
                comment_words = comment_words + words + ' '

                stopwords = set(STOPWORDS)
        wordcloud = WordCloud(width = 600, height = 600,
                        background_color ='Black',
                        stopwords = stopwords,
                        min_font_size = 10).generate(comment_words)

        # plot the WordCloud image
        plt.figure(figsize = (8, 8), facecolor = None)
        plt.imshow(wordcloud)
        plt.axis("off")
        plt.tight_layout(pad = 0)

        plt.show()
```



5.3 [4.3] TF-IDF

```
In [179]: tf_idf_vect = TfidfVectorizer(ngram_range=(1,2), min_df=10)
          tf_idf_vect.fit(X_train)
          print("some sample features(unique words in the corpus)",tf_idf_vect.get_feature_name
          print('='*50)

          final_tf_idf = tf_idf_vect.transform(X_train)
          print("the type of count vectorizer ",type(final_tf_idf))
          print("the shape of out text TFIDF vectorizer ",final_tf_idf.get_shape())
          print("the number of unique words including both unigrams and bigrams ", final_tf_id
```

```
some sample features(unique words in the corpus) ['aa', 'abandoned', 'abdominal', 'ability', 'a
==================================================
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
the shape of out text TFIDF vectorizer  (61441, 36458)
the number of unique words including both unigrams and bigrams  36458
```

```
In [0]: X_test2 =  tf_idf_vect.transform(X_test)
```

5.3.1 Applying Random Forest Classifier

```
In [48]: from sklearn.ensemble import RandomForestClassifier
         from sklearn.model_selection import GridSearchCV
         from sklearn.model_selection import TimeSeriesSplit



         grid_params = dict(n_estimators = [10,20,30,50,100],
                            max_depth= [3,6,10,15,20] )


         rf = RandomForestClassifier()
         rf_clf = GridSearchCV(rf, param_grid=grid_params, n_jobs=-1, verbose=30,return_train_s
         rf_clf = rf_clf.fit(final_tf_idf, y_train)


         results = rf_clf.cv_results_
```

```
Fitting 3 folds for each of 25 candidates, totalling 75 fits


[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done   1 tasks      | elapsed:    0.6s
[Parallel(n_jobs=-1)]: Done   2 tasks      | elapsed:    0.6s
[Parallel(n_jobs=-1)]: Done   3 tasks      | elapsed:    1.2s
[Parallel(n_jobs=-1)]: Done   4 tasks      | elapsed:    1.3s
[Parallel(n_jobs=-1)]: Done   5 tasks      | elapsed:    1.9s
[Parallel(n_jobs=-1)]: Done   6 tasks      | elapsed:    2.2s
[Parallel(n_jobs=-1)]: Done   7 tasks      | elapsed:    2.9s
[Parallel(n_jobs=-1)]: Done   8 tasks      | elapsed:    3.2s
[Parallel(n_jobs=-1)]: Done   9 tasks      | elapsed:    4.0s
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:    4.7s
[Parallel(n_jobs=-1)]: Done  11 tasks      | elapsed:    5.5s
[Parallel(n_jobs=-1)]: Done  12 tasks      | elapsed:    6.3s
[Parallel(n_jobs=-1)]: Done  13 tasks      | elapsed:    8.4s
[Parallel(n_jobs=-1)]: Done  14 tasks      | elapsed:    9.2s
[Parallel(n_jobs=-1)]: Done  15 tasks      | elapsed:    9.7s
[Parallel(n_jobs=-1)]: Done  16 tasks      | elapsed:   10.3s
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:   10.8s
```

```
[Parallel(n_jobs=-1)]: Done  18 tasks      | elapsed:   11.3s
[Parallel(n_jobs=-1)]: Done  19 tasks      | elapsed:   11.8s
[Parallel(n_jobs=-1)]: Done  20 tasks      | elapsed:   12.1s
[Parallel(n_jobs=-1)]: Done  21 tasks      | elapsed:   12.7s
[Parallel(n_jobs=-1)]: Done  22 tasks      | elapsed:   13.3s
[Parallel(n_jobs=-1)]: Done  23 tasks      | elapsed:   14.0s
[Parallel(n_jobs=-1)]: Done  24 tasks      | elapsed:   14.6s
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:   15.8s
[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed:   16.4s
[Parallel(n_jobs=-1)]: Done  27 tasks      | elapsed:   17.9s
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:   20.1s
[Parallel(n_jobs=-1)]: Done  29 tasks      | elapsed:   21.4s
[Parallel(n_jobs=-1)]: Done  30 tasks      | elapsed:   22.0s
[Parallel(n_jobs=-1)]: Done  31 tasks      | elapsed:   22.7s
[Parallel(n_jobs=-1)]: Done  32 tasks      | elapsed:   23.4s
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:   23.7s
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:   24.7s
[Parallel(n_jobs=-1)]: Done  35 tasks      | elapsed:   24.9s
[Parallel(n_jobs=-1)]: Done  36 tasks      | elapsed:   26.1s
[Parallel(n_jobs=-1)]: Done  37 tasks      | elapsed:   26.6s
[Parallel(n_jobs=-1)]: Done  38 tasks      | elapsed:   28.0s
[Parallel(n_jobs=-1)]: Done  39 tasks      | elapsed:   28.4s
[Parallel(n_jobs=-1)]: Done  40 tasks      | elapsed:   30.8s
[Parallel(n_jobs=-1)]: Done  41 tasks      | elapsed:   31.5s
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:   33.6s
[Parallel(n_jobs=-1)]: Done  43 tasks      | elapsed:   36.5s
[Parallel(n_jobs=-1)]: Done  44 tasks      | elapsed:   38.4s
[Parallel(n_jobs=-1)]: Done  45 tasks      | elapsed:   39.3s
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:   40.2s
[Parallel(n_jobs=-1)]: Done  47 tasks      | elapsed:   40.9s
[Parallel(n_jobs=-1)]: Done  48 tasks      | elapsed:   41.0s
[Parallel(n_jobs=-1)]: Done  49 tasks      | elapsed:   42.4s
[Parallel(n_jobs=-1)]: Done  50 tasks      | elapsed:   42.4s
[Parallel(n_jobs=-1)]: Done  51 tasks      | elapsed:   43.9s
[Parallel(n_jobs=-1)]: Done  52 tasks      | elapsed:   44.7s
[Parallel(n_jobs=-1)]: Done  53 tasks      | elapsed:   46.0s
[Parallel(n_jobs=-1)]: Done  54 tasks      | elapsed:   46.8s
[Parallel(n_jobs=-1)]: Done  55 tasks      | elapsed:   49.6s
[Parallel(n_jobs=-1)]: Done  56 tasks      | elapsed:   50.4s
[Parallel(n_jobs=-1)]: Done  57 tasks      | elapsed:   53.4s
[Parallel(n_jobs=-1)]: Done  58 tasks      | elapsed:   57.6s
[Parallel(n_jobs=-1)]: Done  59 tasks      | elapsed:   59.7s
[Parallel(n_jobs=-1)]: Done  60 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done  61 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done  62 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done  63 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done  64 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done  65 tasks      | elapsed:  1.1min
```

```
[Parallel(n_jobs=-1)]: Done  66 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done  67 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done  68 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done  69 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done  70 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done  71 tasks      | elapsed:  1.3min
[Parallel(n_jobs=-1)]: Done  72 tasks      | elapsed:  1.3min
[Parallel(n_jobs=-1)]: Done  75 out of  75 | elapsed:  1.5min remaining:    0.0s
[Parallel(n_jobs=-1)]: Done  75 out of  75 | elapsed:  1.5min finished
```

```
In [49]: print("Best parameters: ", rf_clf.best_params_)
         print("Best cross-validation score: {:.3f}".format(rf_clf.best_score_))
```

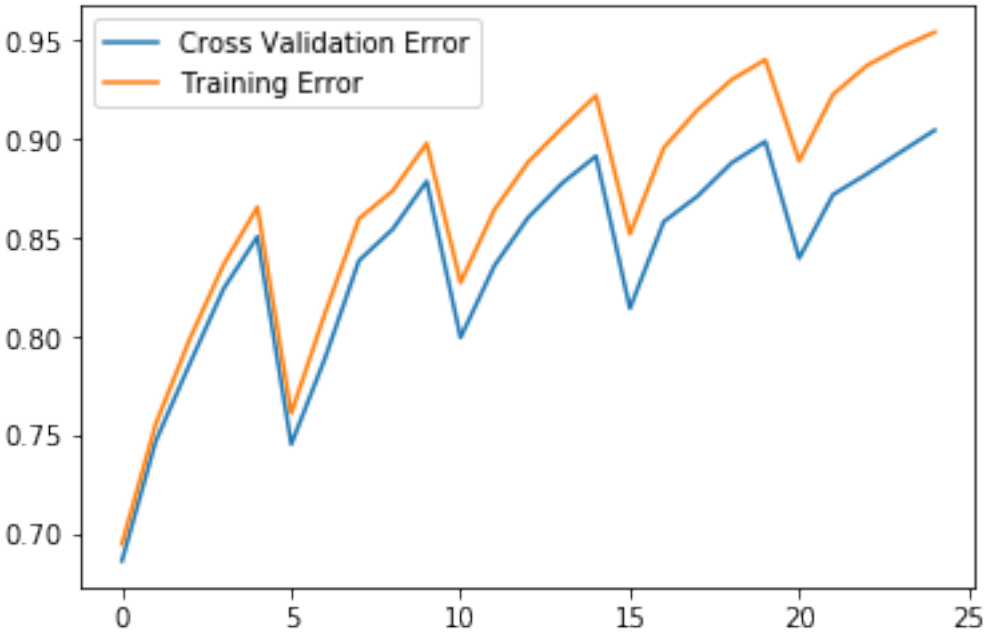
```
Best parameters:  {'max_depth': 20, 'n_estimators': 100}
Best cross-validation score: 0.907
```

```
In [0]: pred = rf_clf.predict(X_test2)
        pred_train = rf_clf.predict(final_tf_idf)
```

```
In [0]: from sklearn.metrics import accuracy_score
        from sklearn.metrics import roc_curve
        import sklearn.metrics as metrics
        from sklearn.metrics import roc_auc_score

        train_error=rf_clf.cv_results_['mean_train_score']
        cv_error = rf_clf.cv_results_['mean_test_score']
        score=roc_auc_score(y_train, pred_train)

        estimator=rf_clf.best_params_['n_estimators']
        depth=rf_clf.best_params_['max_depth']
```

Error Plots

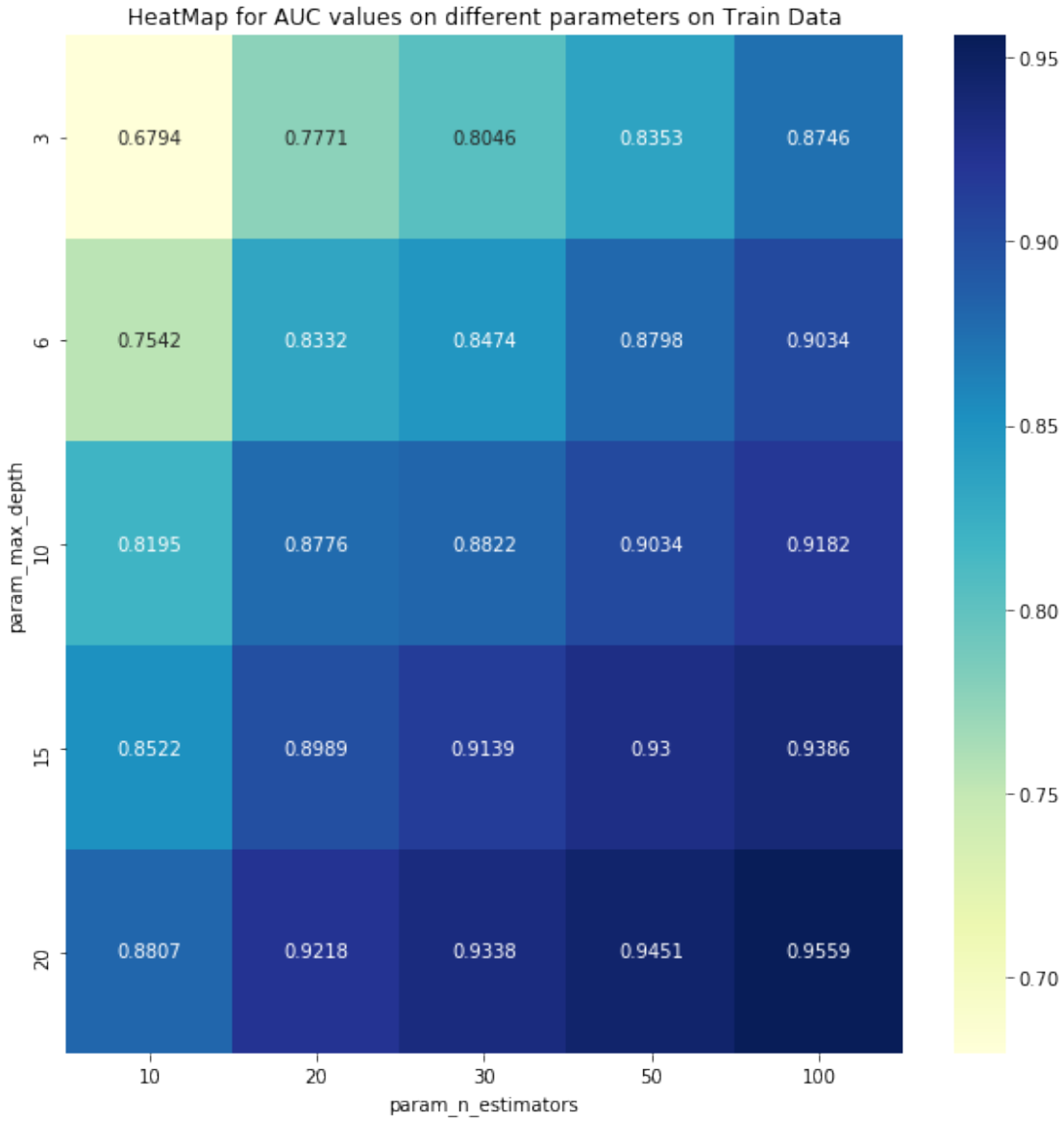
```
In [0]: plt.plot(cv_error, label='Cross Validation Error')
        plt.plot(train_error, label='Training Error')
        plt.legend()
        plt.show()
```



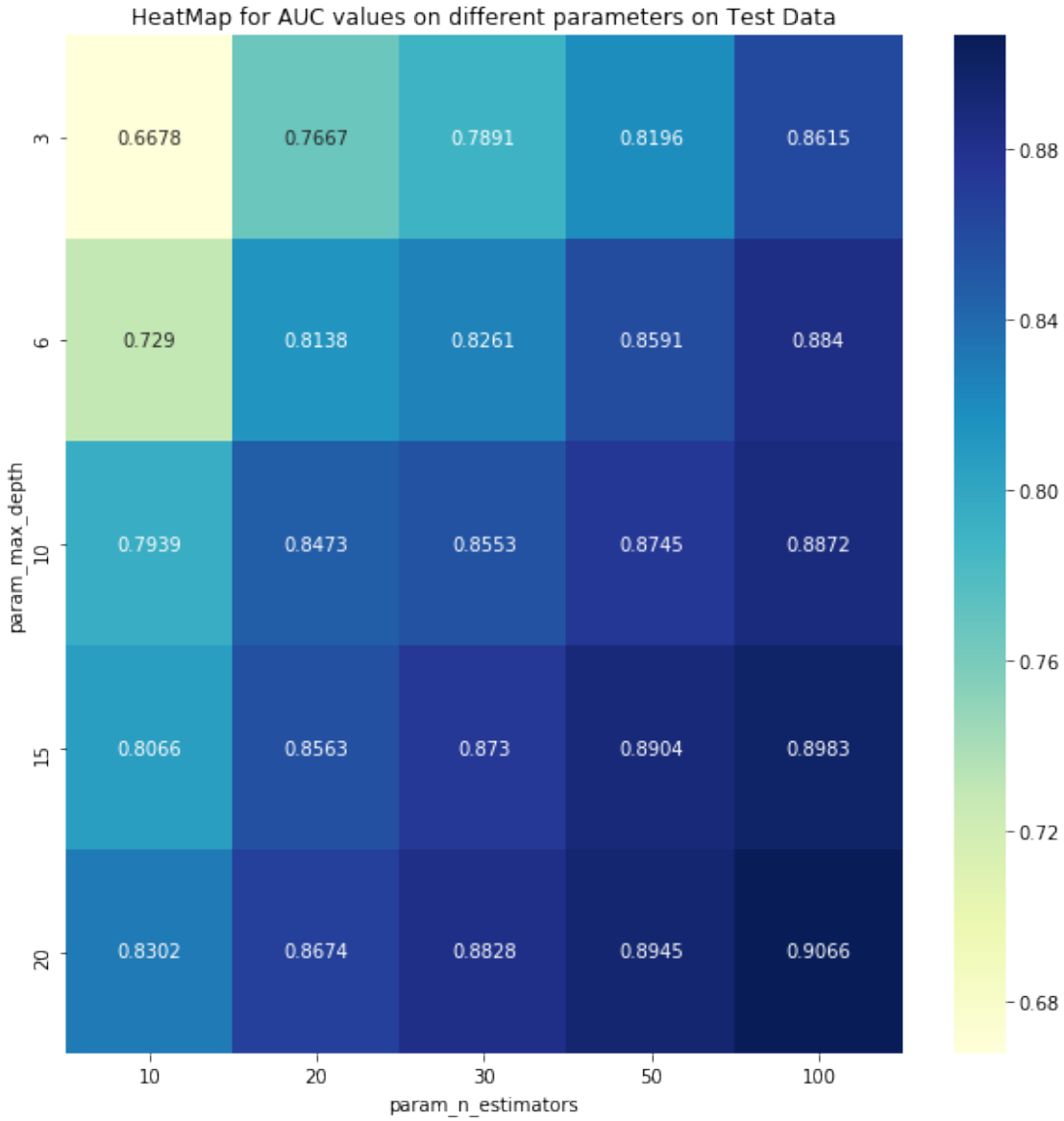

```
In [0]: n_estimators =[10,20,30,50,100]
        max_depth= [3,6,10,15,20]
```

5.3.2 HeatMap for AUC vs N_estimators and Max_Depth

```
In [50]: df_new = pd.DataFrame(rf_clf.cv_results_)
         max_params = df_new.groupby(['param_max_depth','param_n_estimators']).max()
         max_params = max_params.unstack()[['mean_test_score', 'mean_train_score']]

         fig, ax = plt.subplots(figsize=(10,10))         # Sample figsize in inches

         sns.heatmap(max_params.mean_train_score, annot=True,cmap="YlGnBu", fmt='.4g',ax=ax)
         plt.title('HeatMap for AUC values on different parameters on Train Data')
         plt.show()
```

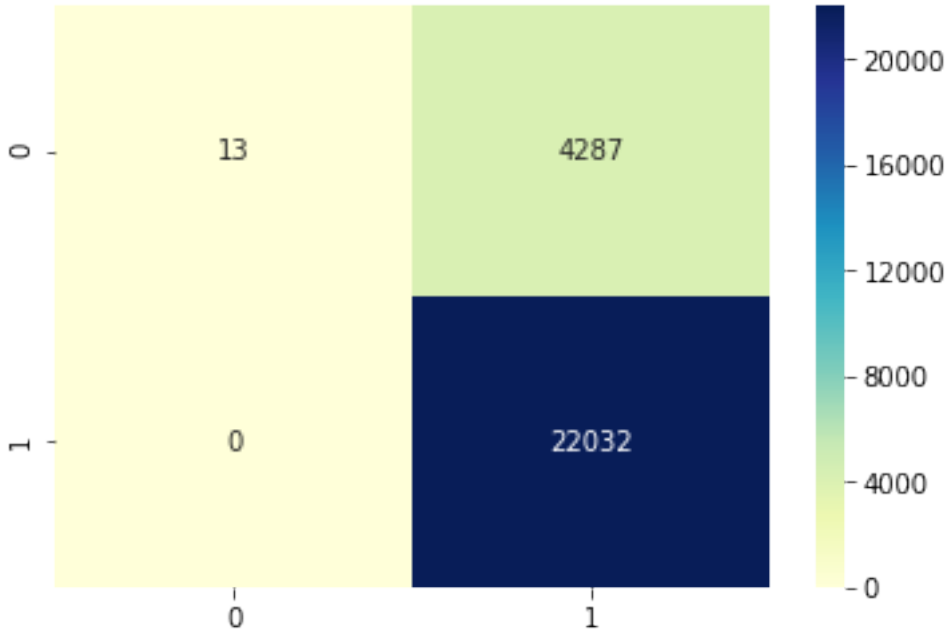



```
In [51]: df_new = pd.DataFrame(rf_clf.cv_results_)
         max_params = df_new.groupby(['param_max_depth','param_n_estimators']).max()
         max_params = max_params.unstack()[['mean_test_score', 'mean_train_score']]

         fig, ax = plt.subplots(figsize=(10,10))         # Sample figsize in inches

         sns.heatmap(max_params.mean_test_score, annot=True,cmap="YlGnBu", fmt='.4g',ax=ax)
         plt.title('HeatMap for AUC values on different parameters on Test Data')
         plt.show()
```



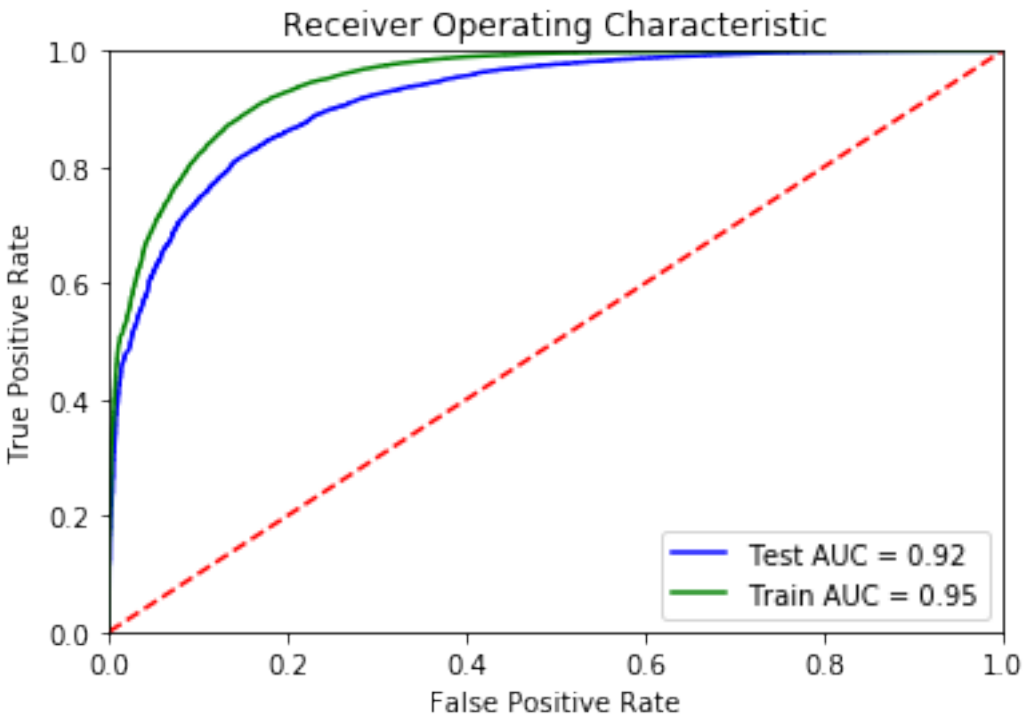

```
In [0]: pred_test = rf_clf.predict_proba(X_test2)[:,1]
        pred_train =  rf_clf.predict_proba(final_tf_idf)[:,1]
```

```
In [0]: cm = confusion_matrix(y_test, pred_test.round())
        sns.heatmap(cm, annot=True, fmt="d",cmap="YlGnBu")
        plt.show()
```




```
In [0]: fpr, tpr, thresholds = roc_curve(y_test, pred_test)
        fpr2, tpr2, thresholds = roc_curve(y_train, pred_train)

        score_test = roc_auc_score(y_test, pred_test)
        score_train = roc_auc_score(y_train, pred_train)
```

```
In [0]: roc_auc_test = metrics.auc(fpr, tpr)
        roc_auc_train = metrics.auc(fpr2, tpr2)
        plt.title('Receiver Operating Characteristic')
        plt.plot(fpr, tpr, 'b', label='Test AUC = %0.2f' % score_test)
        plt.plot(fpr2, tpr2, 'g', label = 'Train AUC = %0.2f' % score_train)
        plt.legend(loc = 'lower right')
        plt.plot([0, 1], [0, 1],'r--')
        plt.xlim([0, 1])
        plt.ylim([0, 1])
        plt.ylabel('True Positive Rate')
        plt.xlabel('False Positive Rate')
        plt.legend()
        plt.show()
```




```
In [0]: model = RandomForestClassifier(n_estimators= 100 , max_depth=20, class_weight='balanced
        model.fit(final_tf_idf, y_train)
```

```
Out[0]: RandomForestClassifier(bootstrap=True, class_weight='balanced',
                               criterion='gini', max_depth=20, max_features='auto',
                               max_leaf_nodes=None, min_impurity_decrease=0.0,
                               min_impurity_split=None, min_samples_leaf=1,
                               min_samples_split=2, min_weight_fraction_leaf=0.0,
                               n_estimators=100, n_jobs=None, oob_score=False,
                               random_state=None, verbose=0, warm_start=False)
```

```
In [0]: # Please write all the code with proper documentation
        # Copied from wordcloud documentation
        # https://towardsdatascience.com/running-random-forests-inspect-the-feature-importance
        from wordcloud import WordCloud, STOPWORDS

        feat_imp = model.feature_importances_

        count_features=tf_idf_vect.get_feature_names()
        feature_importances = pd.DataFrame(feat_imp,index = count_features, columns=['importan

        a = feature_importances.iloc[0:20]
        comment_words = ' '
        for val in a.index:
```

```
    val = str(val)
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '

        stopwords = set(STOPWORDS)
wordcloud = WordCloud(width = 600, height = 600,
                background_color ='Black',
                stopwords = stopwords,
                min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```

5.3.3 Applying XGBOOST Classifier

```
In [52]: from sklearn.model_selection import GridSearchCV
```

```
grid_params = {'n_estimators': [5, 10, 15,20,50,100],
               'max_depth': [2, 5,10, 15]}

xg_clf = xgb.XGBClassifier()
xg_clf = GridSearchCV(xg_clf, param_grid=grid_params, n_jobs=-1, verbose=30,return_tra
xg_clf = xg_clf.fit(final_tf_idf, y_train)

results =xg_clf.cv_results_
```

```
Fitting 3 folds for each of 24 candidates, totalling 72 fits


[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done   1 tasks      | elapsed:    2.4s
[Parallel(n_jobs=-1)]: Done   2 tasks      | elapsed:    2.5s
[Parallel(n_jobs=-1)]: Done   3 tasks      | elapsed:    6.1s
[Parallel(n_jobs=-1)]: Done   4 tasks      | elapsed:    7.5s
[Parallel(n_jobs=-1)]: Done   5 tasks      | elapsed:   11.1s
[Parallel(n_jobs=-1)]: Done   6 tasks      | elapsed:   12.5s
[Parallel(n_jobs=-1)]: Done   7 tasks      | elapsed:   18.1s
[Parallel(n_jobs=-1)]: Done   8 tasks      | elapsed:   19.6s
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py:700: UserWarni
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done   9 tasks      | elapsed:   25.4s
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:   29.8s
[Parallel(n_jobs=-1)]: Done  11 tasks      | elapsed:   32.9s
[Parallel(n_jobs=-1)]: Done  12 tasks      | elapsed:   37.4s
[Parallel(n_jobs=-1)]: Done  13 tasks      | elapsed:   49.0s
[Parallel(n_jobs=-1)]: Done  14 tasks      | elapsed:   54.2s
[Parallel(n_jobs=-1)]: Done  15 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done  16 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done  18 tasks      | elapsed:  1.6min
[Parallel(n_jobs=-1)]: Done  19 tasks      | elapsed:  1.7min
[Parallel(n_jobs=-1)]: Done  20 tasks      | elapsed:  1.8min
[Parallel(n_jobs=-1)]: Done  21 tasks      | elapsed:  1.8min
[Parallel(n_jobs=-1)]: Done  22 tasks      | elapsed:  2.0min
[Parallel(n_jobs=-1)]: Done  23 tasks      | elapsed:  2.0min
[Parallel(n_jobs=-1)]: Done  24 tasks      | elapsed:  2.1min
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:  2.2min
[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed:  2.3min
[Parallel(n_jobs=-1)]: Done  27 tasks      | elapsed:  2.4min
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:  2.6min
[Parallel(n_jobs=-1)]: Done  29 tasks      | elapsed:  2.7min
[Parallel(n_jobs=-1)]: Done  30 tasks      | elapsed:  2.8min
[Parallel(n_jobs=-1)]: Done  31 tasks      | elapsed:  3.2min
[Parallel(n_jobs=-1)]: Done  32 tasks      | elapsed:  3.4min
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:  3.8min
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:  4.5min
[Parallel(n_jobs=-1)]: Done  35 tasks      | elapsed:  4.9min
[Parallel(n_jobs=-1)]: Done  36 tasks      | elapsed:  5.1min
[Parallel(n_jobs=-1)]: Done  37 tasks      | elapsed:  5.2min
[Parallel(n_jobs=-1)]: Done  38 tasks      | elapsed:  5.4min
[Parallel(n_jobs=-1)]: Done  39 tasks      | elapsed:  5.6min
[Parallel(n_jobs=-1)]: Done  40 tasks      | elapsed:  5.7min
[Parallel(n_jobs=-1)]: Done  41 tasks      | elapsed:  5.9min
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  5.9min
```

```
[Parallel(n_jobs=-1)]: Done  43 tasks      | elapsed:  6.3min
[Parallel(n_jobs=-1)]: Done  44 tasks      | elapsed:  6.3min
[Parallel(n_jobs=-1)]: Done  45 tasks      | elapsed:  6.6min
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:  6.8min
[Parallel(n_jobs=-1)]: Done  47 tasks      | elapsed:  7.1min
[Parallel(n_jobs=-1)]: Done  48 tasks      | elapsed:  7.3min
[Parallel(n_jobs=-1)]: Done  49 tasks      | elapsed:  8.3min
[Parallel(n_jobs=-1)]: Done  50 tasks      | elapsed:  8.5min
[Parallel(n_jobs=-1)]: Done  51 tasks      | elapsed:  9.4min
[Parallel(n_jobs=-1)]: Done  52 tasks      | elapsed: 10.7min
[Parallel(n_jobs=-1)]: Done  53 tasks      | elapsed: 11.7min
[Parallel(n_jobs=-1)]: Done  54 tasks      | elapsed: 11.9min
[Parallel(n_jobs=-1)]: Done  55 tasks      | elapsed: 12.1min
[Parallel(n_jobs=-1)]: Done  56 tasks      | elapsed: 12.3min
[Parallel(n_jobs=-1)]: Done  57 tasks      | elapsed: 12.7min
[Parallel(n_jobs=-1)]: Done  58 tasks      | elapsed: 12.9min
[Parallel(n_jobs=-1)]: Done  59 tasks      | elapsed: 13.1min
[Parallel(n_jobs=-1)]: Done  60 tasks      | elapsed: 13.3min
[Parallel(n_jobs=-1)]: Done  61 tasks      | elapsed: 13.7min
[Parallel(n_jobs=-1)]: Done  62 tasks      | elapsed: 13.9min
[Parallel(n_jobs=-1)]: Done  63 tasks      | elapsed: 14.3min
[Parallel(n_jobs=-1)]: Done  64 tasks      | elapsed: 14.7min
[Parallel(n_jobs=-1)]: Done  65 tasks      | elapsed: 15.1min
[Parallel(n_jobs=-1)]: Done  66 tasks      | elapsed: 15.4min
[Parallel(n_jobs=-1)]: Done  67 tasks      | elapsed: 16.9min
[Parallel(n_jobs=-1)]: Done  68 tasks      | elapsed: 17.2min
[Parallel(n_jobs=-1)]: Done  69 tasks      | elapsed: 18.7min
[Parallel(n_jobs=-1)]: Done  72 out of  72 | elapsed: 23.1min remaining:    0.0s
[Parallel(n_jobs=-1)]: Done  72 out of  72 | elapsed: 23.1min finished
```

```
In [53]: print("Best parameters: ", xg_clf.best_params_)
         print("Best cross-validation score: {:.3f}".format(xg_clf.best_score_))
```

```
Best parameters:  {'max_depth': 15, 'n_estimators': 100}
Best cross-validation score: 0.929
```

```
In [0]: pred = xg_clf.predict(X_test2)
        pred_train = xg_clf.predict(final_tf_idf)
```

```
In [0]: from sklearn.metrics import accuracy_score
        from sklearn.metrics import roc_curve
        import sklearn.metrics as metrics
        from sklearn.metrics import roc_auc_score

        train_error=xg_clf.cv_results_['mean_train_score']
        cv_error = xg_clf.cv_results_['mean_test_score']
        score=roc_auc_score(y_train, pred_train)
```

```
estimator=xg_clf.best_params_['n_estimators']
depth=xg_clf.best_params_['max_depth']
```

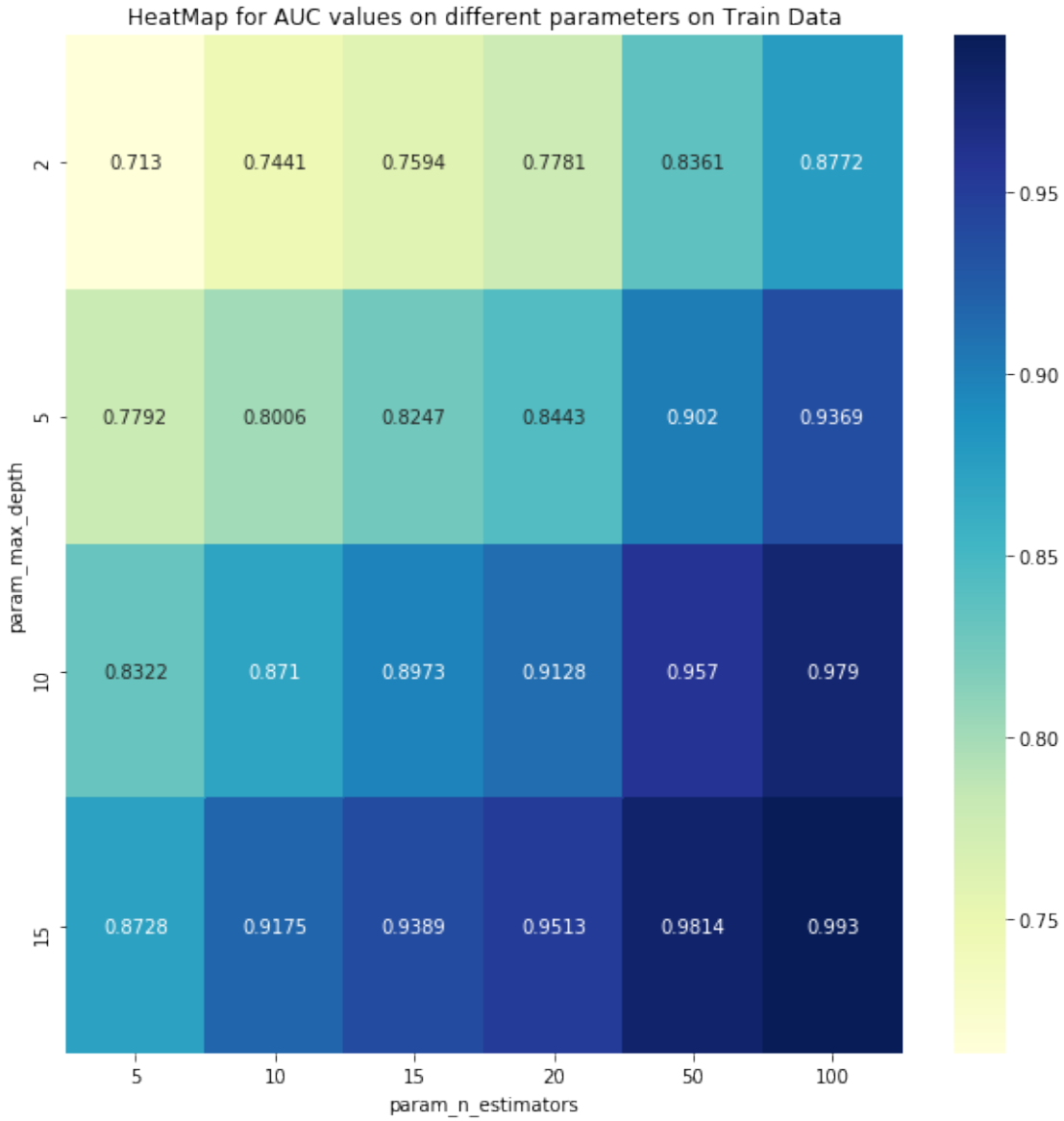
Error plots

```
In [0]: plt.plot(cv_error, label='Cross Validation Error')
        plt.plot(train_error, label='Training Error')
        plt.legend()
        plt.show()
```

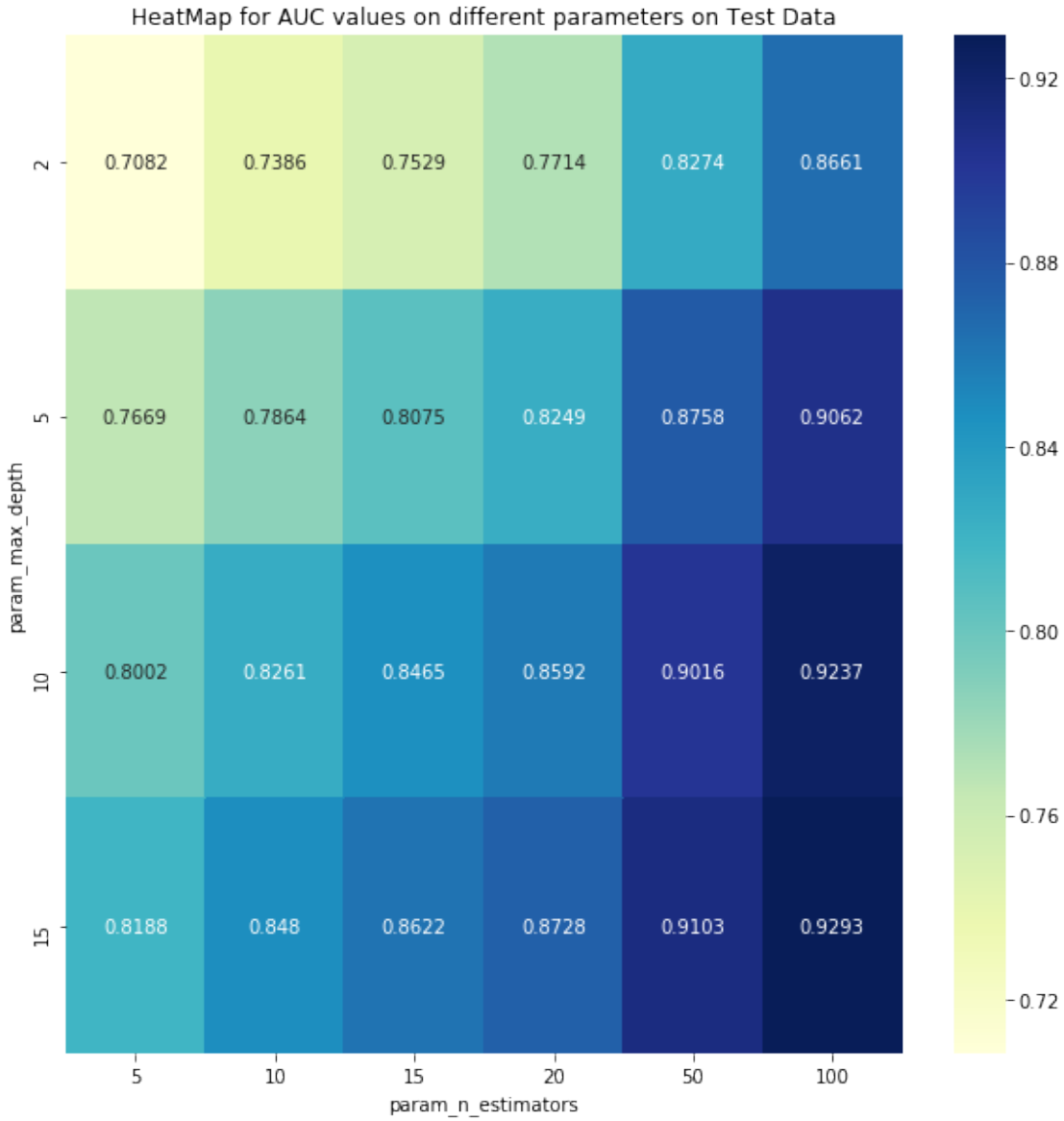

```
In [0]: n_estimators =[5, 10, 15,20,50,100]
        max_depth= [2, 5,10, 15]
```

5.3.4 HeatMap for AUC vs N_estimators and Max_Depth

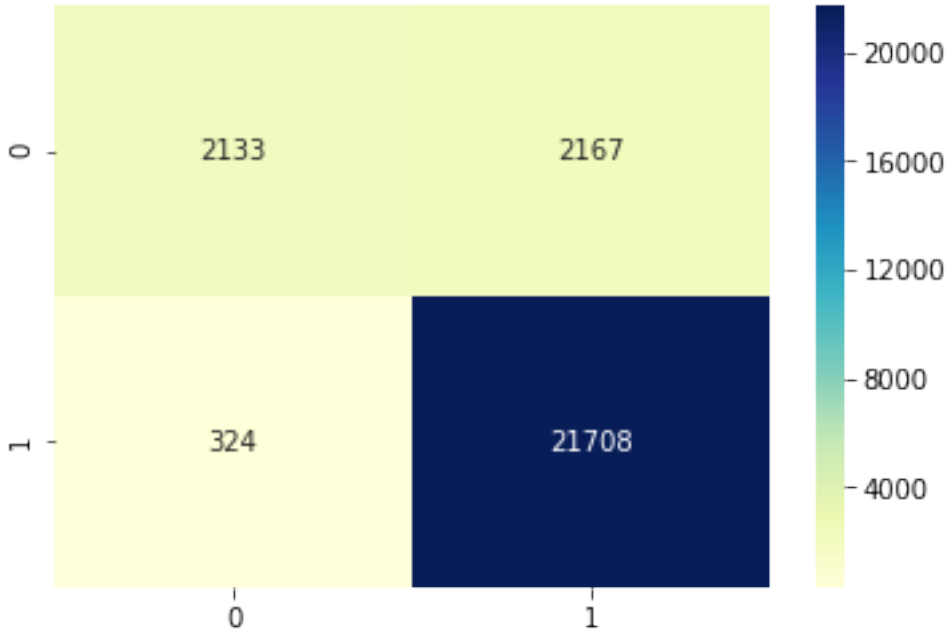
```
In [54]: df_new = pd.DataFrame(xg_clf.cv_results_)
         max_params = df_new.groupby(['param_max_depth','param_n_estimators']).max()
         max_params = max_params.unstack()[['mean_test_score', 'mean_train_score']]

         fig, ax = plt.subplots(figsize=(10,10))         # Sample figsize in inches

         sns.heatmap(max_params.mean_train_score, annot=True,cmap="YlGnBu", fmt='.4g',ax=ax)
         plt.title('HeatMap for AUC values on different parameters on Train Data')
         plt.show()
```




```
In [55]: df_new = pd.DataFrame(xg_clf.cv_results_)
         max_params = df_new.groupby(['param_max_depth','param_n_estimators']).max()
         max_params = max_params.unstack()[['mean_test_score', 'mean_train_score']]

         fig, ax = plt.subplots(figsize=(10,10))         # Sample figsize in inches

         sns.heatmap(max_params.mean_test_score, annot=True,cmap="YlGnBu", fmt='.4g',ax=ax)
         plt.title('HeatMap for AUC values on different parameters on Test Data')
         plt.show()
```

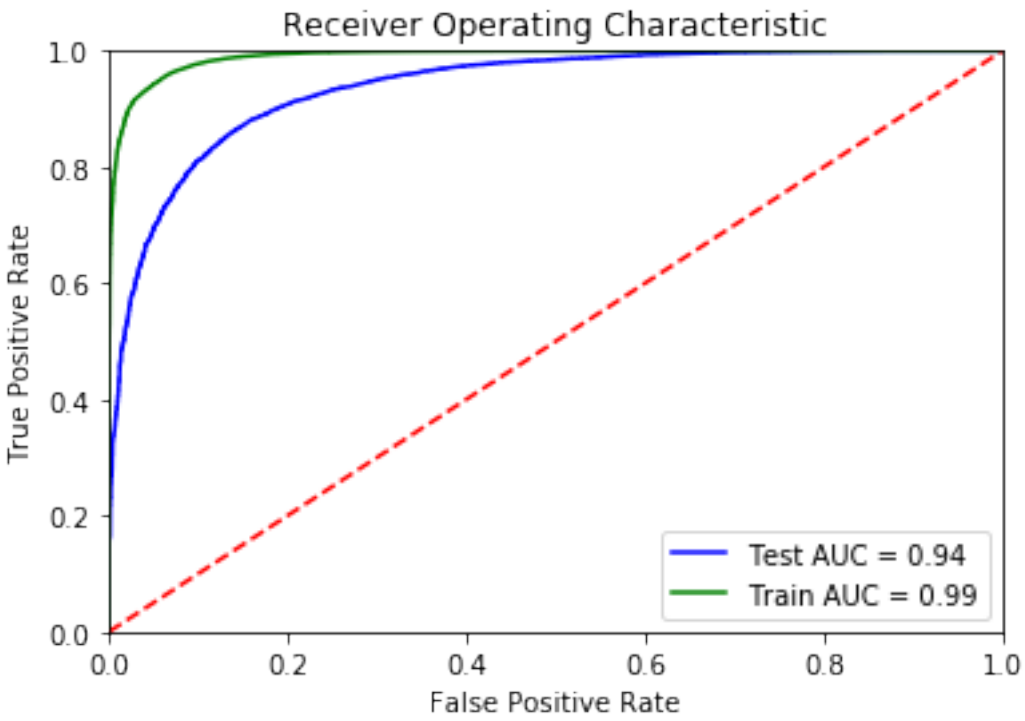



```
In [0]: pred_test = xg_clf.predict_proba(X_test2)[:,1]
        pred_train =  xg_clf.predict_proba(final_tf_idf)[:,1]

In [0]: from sklearn.metrics import confusion_matrix

        cm = confusion_matrix(y_test, pred_test.round())
        sns.heatmap(cm, annot=True, fmt="d",cmap="YlGnBu")
        plt.show()
```




```
In [0]: fpr, tpr, thresholds = roc_curve(y_test, pred_test)
        fpr2, tpr2, thresholds = roc_curve(y_train, pred_train)

        score_test = roc_auc_score(y_test, pred_test)
        score_train = roc_auc_score(y_train, pred_train)
```

```
In [0]: roc_auc_test = metrics.auc(fpr, tpr)
        roc_auc_train = metrics.auc(fpr2, tpr2)
        plt.title('Receiver Operating Characteristic')
        plt.plot(fpr, tpr, 'b', label='Test AUC = %0.2f' % score_test)
        plt.plot(fpr2, tpr2, 'g', label = 'Train AUC = %0.2f' % score_train)
        plt.legend(loc = 'lower right')
        plt.plot([0, 1], [0, 1],'r--')
        plt.xlim([0, 1])
        plt.ylim([0, 1])
        plt.ylabel('True Positive Rate')
        plt.xlabel('False Positive Rate')
        plt.legend()
        plt.show()
```



5.3.5 Wordcloud for Tf-idf Representation

```
In [0]: model = xgb.XGBClassifier(n_estimators= 100 , max_depth=15, class_weight='balanced')
        model.fit(final_tf_idf, y_train)
```

```
Out[0]: XGBClassifier(base_score=0.5, booster='gbtree', class_weight='balanced',
                      colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
                      gamma=0, learning_rate=0.1, max_delta_step=0, max_depth=15,
                      min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
                      nthread=None, objective='binary:logistic', random_state=0,
                      reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
                      silent=None, subsample=1, verbosity=1)
```

```
In [0]: # Please write all the code with proper documentation
        # Copied from wordcloud documentation
        # https://towardsdatascience.com/running-random-forests-inspect-the-feature-importance
        from wordcloud import WordCloud, STOPWORDS

        feat_imp = model.feature_importances_

        count_features = tf_idf_vect.get_feature_names()
        feature_importances = pd.DataFrame(feat_imp,index = count_features, columns=['importan

        a = feature_importances.iloc[0:20]
```

```
comment_words = ' '
for val in a.index:
    val = str(val)
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '

        stopwords = set(STOPWORDS)
wordcloud = WordCloud(width = 600, height = 600,
                background_color ='Black',
                stopwords = stopwords,
                min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```

5.4 [4.4] Word2Vec

```
In [0]: i=0
        list_of_sentance=[]
        for sentance in preprocessed_reviews:
            list_of_sentance.append(sentance.split())
```

```
In [141]: # Using Google News Word2Vectors

          # in this project we are using a pretrained model by google
          # its 3.3G file, once you load this into your memory
          # it occupies ~9Gb, so please do this step only if you have >12G of ram
```

```
        # we will provide a pickle file wich contains a dict ,
        # and it contains all our courpus words as keys and  model[word] as values
        # To use this code-snippet, download "GoogleNews-vectors-negative300.bin"
        # from https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit
        # it's 1.9GB in size.


        # http://kavita-ganesan.com/gensim-word2vec-tutorial-starter-code/#.W17SRFAzZPY
        # you can comment this whole cell
        # or change these varible according to your need

        is_your_ram_gt_16g=False
        want_to_use_google_w2v = False
        want_to_train_w2v = True

        if want_to_train_w2v:
            # min_count = 5 considers only words that occured atleast 5 times
            w2v_model=Word2Vec(list_of_sentance,min_count=5,size=50, workers=4)
            print(w2v_model.wv.most_similar('great'))
            print('='*50)
            print(w2v_model.wv.most_similar('worst'))

        elif want_to_use_google_w2v and is_your_ram_gt_16g:
            if os.path.isfile('GoogleNews-vectors-negative300.bin'):
                w2v_model=KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.
                print(w2v_model.wv.most_similar('great'))
                print(w2v_model.wv.most_similar('worst'))
            else:
                print("you don't have gogole's word2vec file, keep want_to_train_w2v = True,
```

```
[('fantastic', 0.8344917893409729), ('good', 0.808018147945404), ('excellent', 0.807872712612
==================================================
[('greatest', 0.8235164880752563), ('best', 0.718223512172699), ('tastiest', 0.706526041030883
```

```
In [144]: w2v_words = list(w2v_model.wv.vocab)
          print("number of words that occured minimum 5 times ",len(w2v_words))
          print("sample words ", w2v_words[0:50])
```

```
number of words that occured minimum 5 times  17386
sample words  ['dogs', 'loves', 'chicken', 'product', 'china', 'wont', 'buying', 'anymore', 'ha
```

```
In [0]: X = list_of_sentance[:]
        y = final['Score'][:]

        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state
```

```
In [150]: # Then vectorize your train model as
          train_sent = [];
```

```
for sent in tqdm(X_train):
    sent_vec = np.zeros(50)
    cnt_words =0;
    for word in sent:
        if word in w2v_words:
            vec = w2v_model.wv[word]
            sent_vec += vec
            cnt_words += 1
    if cnt_words != 0:
        sent_vec /= cnt_words
    train_sent.append(sent_vec)
```

```
0%|          | 0/61441 [00:00<?, ?it/s]
0%|          | 43/61441 [00:00<02:23, 428.87it/s]
0%|          | 75/61441 [00:00<02:44, 373.84it/s]
0%|          | 114/61441 [00:00<02:43, 376.08it/s]
0%|          | 153/61441 [00:00<02:42, 377.64it/s]
0%|          | 190/61441 [00:00<02:45, 370.42it/s]
0%|          | 238/61441 [00:00<02:34, 395.81it/s]
0%|          | 274/61441 [00:00<02:41, 378.13it/s]
1%|          | 313/61441 [00:00<02:40, 380.26it/s]
1%|          | 362/61441 [00:00<02:30, 407.00it/s]
1%|          | 413/61441 [00:01<02:27, 413.77it/s]
1%|          | 471/61441 [00:01<02:14, 452.64it/s]
1%|          | 521/61441 [00:01<02:10, 465.81it/s]
1%|          | 575/61441 [00:01<02:06, 479.65it/s]
1%|          | 624/61441 [00:01<02:21, 431.04it/s]
1%|          | 673/61441 [00:01<02:17, 441.88it/s]
1%|          | 721/61441 [00:01<02:15, 448.29it/s]
```

```
1%|          | 768/61441 [00:01<02:13, 454.04it/s]
1%|          | 819/61441 [00:01<02:10, 464.52it/s]
1%|          | 866/61441 [00:02<02:17, 442.02it/s]
1%|          | 921/61441 [00:02<02:09, 468.63it/s]
2%|          | 969/61441 [00:02<02:11, 461.45it/s]
2%|          | 1016/61441 [00:02<02:15, 444.58it/s]
2%|          | 1064/61441 [00:02<02:13, 453.47it/s]
2%|          | 1115/61441 [00:02<02:10, 462.38it/s]
2%|          | 1169/61441 [00:02<02:04, 483.00it/s]
2%|          | 1220/61441 [00:02<02:02, 490.35it/s]
2%|          | 1276/61441 [00:02<01:58, 506.96it/s]
2%|          | 1328/61441 [00:02<02:05, 478.26it/s]
2%|          | 1381/61441 [00:03<02:06, 475.83it/s]
2%|          | 1431/61441 [00:03<02:04, 481.40it/s]
2%|          | 1480/61441 [00:03<02:07, 472.10it/s]
2%|          | 1533/61441 [00:03<02:02, 487.40it/s]
3%|          | 1583/61441 [00:03<02:21, 423.80it/s]
3%|          | 1630/61441 [00:03<02:17, 435.38it/s]
3%|          | 1689/61441 [00:03<02:06, 471.00it/s]
3%|          | 1746/61441 [00:03<02:00, 496.29it/s]
3%|          | 1798/61441 [00:04<02:15, 438.57it/s]
3%|          | 1855/61441 [00:04<02:06, 470.55it/s]
3%|          | 1908/61441 [00:04<02:02, 485.47it/s]
3%|          | 1959/61441 [00:04<02:05, 472.49it/s]
```

```
3%|         | 2008/61441 [00:04<02:21, 419.38it/s]
3%|         | 2058/61441 [00:04<02:15, 436.85it/s]
3%|         | 2110/61441 [00:04<02:09, 458.38it/s]
4%|         | 2168/61441 [00:04<02:02, 483.37it/s]
4%|         | 2218/61441 [00:04<02:17, 432.02it/s]
4%|         | 2264/61441 [00:05<02:17, 431.87it/s]
4%|         | 2309/61441 [00:05<02:17, 429.68it/s]
4%|         | 2353/61441 [00:05<02:31, 390.88it/s]
4%|         | 2399/61441 [00:05<02:24, 408.32it/s]
4%|         | 2441/61441 [00:05<02:27, 399.88it/s]
4%|         | 2488/61441 [00:05<02:22, 412.94it/s]
4%|         | 2537/61441 [00:05<02:16, 432.01it/s]
4%|         | 2581/61441 [00:05<02:45, 356.12it/s]
4%|         | 2633/61441 [00:05<02:29, 392.43it/s]
4%|         | 2679/61441 [00:06<02:24, 407.62it/s]
4%|         | 2731/61441 [00:06<02:16, 430.65it/s]
5%|         | 2777/61441 [00:06<02:16, 428.55it/s]
5%|         | 2832/61441 [00:06<02:07, 458.93it/s]
5%|         | 2880/61441 [00:06<02:27, 397.34it/s]
5%|         | 2923/61441 [00:06<02:26, 400.76it/s]
5%|         | 2972/61441 [00:06<02:21, 414.43it/s]
5%|         | 3028/61441 [00:06<02:11, 445.77it/s]
5%|         | 3075/61441 [00:06<02:12, 439.98it/s]
5%|         | 3123/61441 [00:07<02:09, 450.67it/s]
```

```
5%|         | 3170/61441 [00:07<02:07, 455.64it/s]
5%|         | 3217/61441 [00:07<02:16, 425.21it/s]
5%|         | 3269/61441 [00:07<02:09, 448.07it/s]
5%|         | 3324/61441 [00:07<02:02, 472.78it/s]
5%|         | 3373/61441 [00:07<02:15, 429.93it/s]
6%|         | 3418/61441 [00:07<02:28, 391.17it/s]
6%|         | 3468/61441 [00:07<02:18, 418.48it/s]
6%|         | 3515/61441 [00:07<02:14, 430.71it/s]
6%|         | 3564/61441 [00:08<02:10, 443.41it/s]
6%|         | 3610/61441 [00:08<02:10, 444.31it/s]
6%|         | 3656/61441 [00:08<02:17, 421.33it/s]
6%|         | 3710/61441 [00:08<02:08, 450.91it/s]
6%|         | 3757/61441 [00:08<02:15, 424.72it/s]
6%|         | 3801/61441 [00:08<02:33, 374.47it/s]
6%|         | 3845/61441 [00:08<02:27, 391.60it/s]
6%|         | 3887/61441 [00:08<02:24, 398.54it/s]
6%|         | 3941/61441 [00:08<02:13, 431.74it/s]
6%|         | 3986/61441 [00:09<02:19, 412.25it/s]
7%|         | 4036/61441 [00:09<02:12, 432.83it/s]
7%|         | 4088/61441 [00:09<02:06, 453.69it/s]
7%|         | 4135/61441 [00:09<02:08, 445.35it/s]
7%|         | 4182/61441 [00:09<02:07, 450.56it/s]
7%|         | 4228/61441 [00:09<02:16, 418.63it/s]
7%|         | 4290/61441 [00:09<02:04, 458.92it/s]
```

```
7%|         | 4338/61441 [00:09<02:03, 460.93it/s]
7%|         | 4386/61441 [00:09<02:09, 440.96it/s]
7%|         | 4434/61441 [00:10<02:06, 450.10it/s]
7%|         | 4480/61441 [00:10<02:06, 451.91it/s]
7%|         | 4542/61441 [00:10<01:56, 489.04it/s]
7%|         | 4593/61441 [00:10<01:59, 475.37it/s]
8%|         | 4651/61441 [00:10<01:54, 496.06it/s]
8%|         | 4702/61441 [00:10<02:23, 394.29it/s]
8%|         | 4746/61441 [00:10<02:19, 405.25it/s]
8%|         | 4790/61441 [00:10<02:27, 383.71it/s]
8%|         | 4847/61441 [00:11<02:13, 424.27it/s]
8%|         | 4893/61441 [00:11<02:12, 427.27it/s]
8%|         | 4945/61441 [00:11<02:06, 446.95it/s]
8%|         | 4992/61441 [00:11<02:05, 449.35it/s]
8%|         | 5039/61441 [00:11<02:07, 443.96it/s]
8%|         | 5085/61441 [00:11<02:11, 427.78it/s]
8%|         | 5129/61441 [00:11<02:11, 428.60it/s]
8%|         | 5175/61441 [00:11<02:08, 436.26it/s]
9%|         | 5226/61441 [00:11<02:04, 453.09it/s]
9%|         | 5276/61441 [00:11<02:00, 465.62it/s]
9%|         | 5336/61441 [00:12<01:52, 498.10it/s]
9%|         | 5387/61441 [00:12<02:22, 392.11it/s]
9%|         | 5438/61441 [00:12<02:14, 416.30it/s]
9%|         | 5491/61441 [00:12<02:08, 434.23it/s]
```

```
 9%|         | 5537/61441 [00:12<02:14, 415.92it/s]
 9%|         | 5581/61441 [00:12<02:19, 399.01it/s]
 9%|         | 5645/61441 [00:12<02:04, 449.41it/s]
 9%|         | 5694/61441 [00:12<02:11, 424.48it/s]
 9%|         | 5743/61441 [00:13<02:06, 441.00it/s]
 9%|         | 5797/61441 [00:13<02:00, 463.62it/s]
10%|         | 5846/61441 [00:13<01:58, 471.05it/s]
10%|         | 5895/61441 [00:13<01:58, 468.17it/s]
10%|         | 5943/61441 [00:13<02:03, 450.50it/s]
10%|         | 5994/61441 [00:13<01:59, 463.46it/s]
10%|         | 6041/61441 [00:13<01:59, 464.49it/s]
10%|         | 6088/61441 [00:13<02:03, 447.44it/s]
10%|         | 6138/61441 [00:13<01:59, 461.99it/s]
10%|         | 6185/61441 [00:13<01:59, 461.67it/s]
10%|         | 6232/61441 [00:14<02:05, 440.59it/s]
10%|         | 6289/61441 [00:14<02:02, 449.13it/s]
10%|         | 6337/61441 [00:14<02:00, 457.87it/s]
10%|         | 6386/61441 [00:14<01:58, 463.16it/s]
10%|         | 6436/61441 [00:14<01:56, 472.44it/s]
11%|         | 6484/61441 [00:14<02:00, 454.47it/s]
11%|         | 6530/61441 [00:14<02:10, 420.51it/s]
11%|         | 6577/61441 [00:14<02:07, 431.38it/s]
11%|         | 6621/61441 [00:15<02:18, 396.01it/s]
11%|         | 6673/61441 [00:15<02:12, 412.87it/s]
```

```
11%|          | 6716/61441 [00:15<02:21, 386.62it/s]
11%|          | 6756/61441 [00:15<02:20, 387.95it/s]
11%|          | 6796/61441 [00:15<02:33, 355.54it/s]
11%|          | 6839/61441 [00:15<02:26, 373.61it/s]
11%|          | 6878/61441 [00:15<02:25, 374.48it/s]
11%|         | 6928/61441 [00:15<02:15, 403.05it/s]
11%|         | 6970/61441 [00:15<02:15, 401.62it/s]
11%|         | 7011/61441 [00:16<02:14, 403.65it/s]
11%|         | 7054/61441 [00:16<02:12, 410.27it/s]
12%|         | 7101/61441 [00:16<02:07, 426.39it/s]
12%|         | 7145/61441 [00:16<02:07, 424.68it/s]
12%|         | 7188/61441 [00:16<02:13, 406.45it/s]
12%|         | 7243/61441 [00:16<02:02, 440.91it/s]
12%|         | 7289/61441 [00:16<03:00, 300.70it/s]
12%|         | 7338/61441 [00:16<02:39, 339.48it/s]
12%|         | 7392/61441 [00:16<02:21, 381.95it/s]
12%|         | 7439/61441 [00:17<02:14, 400.27it/s]
12%|         | 7484/61441 [00:17<02:13, 405.24it/s]
12%|         | 7528/61441 [00:17<02:20, 383.61it/s]
12%|         | 7587/61441 [00:17<02:06, 425.00it/s]
12%|         | 7633/61441 [00:17<02:04, 433.20it/s]
13%|         | 7708/61441 [00:17<01:48, 493.98it/s]
13%|         | 7772/61441 [00:17<01:43, 517.32it/s]
13%|         | 7829/61441 [00:17<01:40, 531.23it/s]
```

```
13%|        | 7885/61441 [00:17<01:45, 508.03it/s]
13%|        | 7938/61441 [00:18<01:51, 480.65it/s]
13%|        | 7988/61441 [00:18<01:57, 454.39it/s]
13%|        | 8043/61441 [00:18<01:51, 479.34it/s]
13%|        | 8094/61441 [00:18<01:49, 486.91it/s]
13%|        | 8144/61441 [00:18<01:53, 469.29it/s]
13%|        | 8200/61441 [00:18<01:48, 491.62it/s]
13%|        | 8251/61441 [00:18<01:51, 476.51it/s]
14%|        | 8300/61441 [00:18<02:16, 388.71it/s]
14%|        | 8343/61441 [00:19<02:12, 399.79it/s]
14%|        | 8388/61441 [00:19<02:08, 412.04it/s]
14%|        | 8431/61441 [00:19<02:08, 413.33it/s]
14%|        | 8485/61441 [00:19<01:59, 443.76it/s]
14%|        | 8538/61441 [00:19<01:54, 461.75it/s]
14%|        | 8586/61441 [00:19<01:56, 452.23it/s]
14%|        | 8633/61441 [00:19<02:01, 433.20it/s]
14%|        | 8684/61441 [00:19<01:56, 453.52it/s]
14%|        | 8731/61441 [00:19<02:01, 433.91it/s]
14%|        | 8776/61441 [00:20<02:02, 431.36it/s]
14%|        | 8827/61441 [00:20<02:02, 430.24it/s]
14%|        | 8871/61441 [00:20<02:05, 420.47it/s]
15%|        | 8925/61441 [00:20<01:56, 449.42it/s]
15%|        | 8972/61441 [00:20<01:55, 452.81it/s]
15%|        | 9020/61441 [00:20<01:54, 456.70it/s]
```

```
15%|        | 9067/61441 [00:20<01:59, 438.82it/s]
15%|        | 9112/61441 [00:20<02:02, 428.89it/s]
15%|        | 9156/61441 [00:20<02:01, 430.67it/s]
15%|        | 9205/61441 [00:20<01:57, 445.76it/s]
15%|        | 9261/61441 [00:21<01:49, 474.47it/s]
15%|        | 9310/61441 [00:21<01:51, 468.62it/s]
15%|        | 9358/61441 [00:21<02:01, 429.11it/s]
15%|        | 9402/61441 [00:21<02:04, 416.40it/s]
15%|        | 9448/61441 [00:21<02:02, 423.15it/s]
15%|        | 9491/61441 [00:21<02:03, 419.73it/s]
16%|        | 9539/61441 [00:21<01:59, 435.61it/s]
16%|        | 9589/61441 [00:21<01:54, 451.56it/s]
16%|        | 9635/61441 [00:21<02:02, 421.22it/s]
16%|        | 9685/61441 [00:22<01:57, 441.25it/s]
16%|        | 9734/61441 [00:22<01:53, 454.51it/s]
16%|        | 9781/61441 [00:22<01:54, 449.85it/s]
16%|        | 9832/61441 [00:22<01:51, 464.28it/s]
16%|        | 9887/61441 [00:22<01:46, 484.55it/s]
16%|        | 9936/61441 [00:22<01:57, 439.36it/s]
16%|        | 9982/61441 [00:22<02:02, 420.32it/s]
16%|        | 10028/61441 [00:22<02:00, 425.01it/s]
16%|        | 10072/61441 [00:22<02:14, 380.52it/s]
16%|        | 10112/61441 [00:23<02:13, 385.16it/s]
17%|        | 10152/61441 [00:23<02:12, 387.50it/s]
```

```
17%|        | 10204/61441 [00:23<02:03, 414.32it/s]
17%|        | 10247/61441 [00:23<02:11, 390.02it/s]
17%|        | 10287/61441 [00:23<02:16, 373.78it/s]
17%|        | 10330/61441 [00:23<02:11, 388.57it/s]
17%|        | 10373/61441 [00:23<02:08, 398.18it/s]
17%|        | 10414/61441 [00:23<02:08, 397.91it/s]
17%|        | 10455/61441 [00:23<02:09, 393.72it/s]
17%|        | 10495/61441 [00:24<02:12, 383.10it/s]
17%|        | 10538/61441 [00:24<02:15, 375.27it/s]
17%|        | 10583/61441 [00:24<02:09, 392.35it/s]
17%|        | 10623/61441 [00:24<02:09, 391.96it/s]
17%|        | 10672/61441 [00:24<02:02, 415.90it/s]
17%|        | 10717/61441 [00:24<01:59, 423.75it/s]
18%|        | 10760/61441 [00:24<02:08, 394.21it/s]
18%|        | 10801/61441 [00:24<02:07, 398.29it/s]
18%|        | 10848/61441 [00:24<02:01, 417.14it/s]
18%|        | 10891/61441 [00:25<02:02, 411.98it/s]
18%|        | 10937/61441 [00:25<01:58, 424.45it/s]
18%|        | 10980/61441 [00:25<02:00, 419.38it/s]
18%|        | 11023/61441 [00:25<01:59, 422.37it/s]
18%|        | 11066/61441 [00:25<01:59, 422.77it/s]
18%|        | 11109/61441 [00:25<02:00, 416.02it/s]
18%|        | 11156/61441 [00:25<01:56, 430.55it/s]
18%|        | 11200/61441 [00:25<01:57, 426.79it/s]
```

```
18%|        | 11243/61441 [00:25<02:02, 408.24it/s]
18%|        | 11286/61441 [00:25<02:01, 413.25it/s]
18%|        | 11338/61441 [00:26<01:54, 439.34it/s]
19%|        | 11383/61441 [00:26<01:56, 429.68it/s]
19%|        | 11427/61441 [00:26<01:56, 429.68it/s]
19%|        | 11471/61441 [00:26<02:04, 402.88it/s]
19%|        | 11512/61441 [00:26<02:21, 353.14it/s]
19%|        | 11553/61441 [00:26<02:15, 366.84it/s]
19%|        | 11607/61441 [00:26<02:03, 403.97it/s]
19%|        | 11650/61441 [00:26<02:06, 392.74it/s]
19%|        | 11691/61441 [00:26<02:12, 374.87it/s]
19%|        | 11730/61441 [00:27<02:12, 374.16it/s]
19%|        | 11769/61441 [00:27<02:18, 358.35it/s]
19%|        | 11807/61441 [00:27<02:16, 363.70it/s]
19%|        | 11848/61441 [00:27<02:12, 375.36it/s]
19%|        | 11896/61441 [00:27<02:04, 397.66it/s]
19%|        | 11937/61441 [00:27<02:06, 390.71it/s]
20%|        | 11983/61441 [00:27<02:01, 405.51it/s]
20%|        | 12027/61441 [00:27<02:00, 411.04it/s]
20%|        | 12072/61441 [00:27<01:58, 416.95it/s]
20%|        | 12122/61441 [00:28<01:52, 437.82it/s]
20%|        | 12167/61441 [00:28<02:05, 393.44it/s]
20%|        | 12215/61441 [00:28<01:58, 414.99it/s]
20%|        | 12258/61441 [00:28<01:59, 411.02it/s]
```

```
20%|        | 12313/61441 [00:28<01:51, 441.53it/s]
20%|        | 12359/61441 [00:28<02:07, 383.49it/s]
20%|        | 12419/61441 [00:28<01:54, 428.38it/s]
20%|        | 12484/61441 [00:28<01:42, 476.59it/s]
20%|        | 12536/61441 [00:28<01:43, 470.50it/s]
20%|        | 12586/61441 [00:29<01:47, 456.26it/s]
21%|        | 12634/61441 [00:29<01:47, 454.24it/s]
21%|        | 12683/61441 [00:29<01:45, 463.77it/s]
21%|        | 12731/61441 [00:29<01:48, 448.44it/s]
21%|        | 12777/61441 [00:29<01:51, 434.89it/s]
21%|        | 12825/61441 [00:29<01:48, 447.43it/s]
21%|        | 12871/61441 [00:29<01:49, 441.69it/s]
21%|        | 12927/61441 [00:29<01:43, 470.54it/s]
21%|        | 12979/61441 [00:29<01:40, 483.75it/s]
21%|        | 13029/61441 [00:30<01:50, 437.99it/s]
21%|       | 13075/61441 [00:30<02:02, 396.26it/s]
21%|       | 13123/61441 [00:30<01:56, 416.38it/s]
21%|       | 13172/61441 [00:30<01:51, 431.48it/s]
22%|       | 13217/61441 [00:30<02:10, 368.55it/s]
22%|       | 13262/61441 [00:30<02:04, 388.25it/s]
22%|       | 13318/61441 [00:30<01:52, 426.13it/s]
22%|       | 13371/61441 [00:30<01:46, 451.61it/s]
22%|       | 13419/61441 [00:31<02:00, 398.73it/s]
22%|       | 13465/61441 [00:31<01:56, 413.12it/s]
```

```
22%|       | 13511/61441 [00:31<01:52, 424.33it/s]
22%|       | 13557/61441 [00:31<01:50, 434.05it/s]
22%|       | 13603/61441 [00:31<01:48, 440.93it/s]
22%|       | 13648/61441 [00:31<01:58, 403.83it/s]
22%|       | 13693/61441 [00:31<01:54, 416.29it/s]
22%|       | 13739/61441 [00:31<01:51, 428.50it/s]
22%|       | 13785/61441 [00:31<01:51, 427.59it/s]
23%|       | 13829/61441 [00:31<01:55, 410.73it/s]
23%|       | 13880/61441 [00:32<01:49, 434.68it/s]
23%|       | 13925/61441 [00:32<01:54, 415.85it/s]
23%|       | 13968/61441 [00:32<02:08, 369.91it/s]
23%|       | 14019/61441 [00:32<01:58, 401.09it/s]
23%|       | 14066/61441 [00:32<01:53, 416.42it/s]
23%|       | 14110/61441 [00:32<01:53, 418.15it/s]
23%|       | 14153/61441 [00:32<01:55, 410.23it/s]
23%|       | 14201/61441 [00:32<01:50, 428.04it/s]
23%|       | 14245/61441 [00:33<01:56, 403.84it/s]
23%|       | 14295/61441 [00:33<01:50, 425.05it/s]
23%|       | 14343/61441 [00:33<01:48, 434.80it/s]
23%|       | 14396/61441 [00:33<01:42, 458.93it/s]
24%|       | 14443/61441 [00:33<01:48, 432.97it/s]
24%|       | 14488/61441 [00:33<01:48, 433.64it/s]
24%|       | 14532/61441 [00:33<02:00, 388.50it/s]
24%|       | 14580/61441 [00:33<01:54, 410.58it/s]
```

```
24%|       | 14628/61441 [00:33<01:49, 427.09it/s]
24%|       | 14672/61441 [00:34<01:58, 395.21it/s]
24%|       | 14724/61441 [00:34<01:50, 423.00it/s]
24%|       | 14780/61441 [00:34<01:42, 456.01it/s]
24%|       | 14828/61441 [00:34<01:41, 458.70it/s]
24%|       | 14876/61441 [00:34<01:49, 425.68it/s]
24%|       | 14930/61441 [00:34<01:43, 450.75it/s]
24%|       | 14977/61441 [00:34<01:42, 453.75it/s]
24%|       | 15028/61441 [00:34<01:39, 466.16it/s]
25%|       | 15076/61441 [00:34<01:42, 452.46it/s]
25%|       | 15122/61441 [00:35<01:54, 406.26it/s]
25%|       | 15164/61441 [00:35<01:59, 386.22it/s]
25%|       | 15219/61441 [00:35<01:49, 423.36it/s]
25%|       | 15274/61441 [00:35<01:41, 453.91it/s]
25%|       | 15322/61441 [00:35<01:42, 451.58it/s]
25%|       | 15372/61441 [00:35<01:39, 465.07it/s]
25%|       | 15420/61441 [00:35<01:45, 435.79it/s]
25%|       | 15474/61441 [00:35<01:39, 461.99it/s]
25%|       | 15522/61441 [00:35<01:38, 465.73it/s]
25%|       | 15570/61441 [00:35<01:40, 455.89it/s]
25%|       | 15617/61441 [00:36<01:44, 436.45it/s]
25%|       | 15662/61441 [00:36<01:55, 396.33it/s]
26%|       | 15703/61441 [00:36<01:58, 387.29it/s]
26%|       | 15758/61441 [00:36<01:49, 417.77it/s]
```

```
26%|       | 15802/61441 [00:36<01:49, 415.80it/s]
26%|       | 15845/61441 [00:36<01:56, 390.04it/s]
26%|       | 15902/61441 [00:36<01:45, 430.36it/s]
26%|       | 15949/61441 [00:36<01:43, 439.22it/s]
26%|       | 15995/61441 [00:36<01:43, 440.70it/s]
26%|       | 16041/61441 [00:37<01:47, 422.11it/s]
26%|       | 16085/61441 [00:37<01:53, 400.19it/s]
26%|       | 16133/61441 [00:37<01:48, 418.84it/s]
26%|       | 16186/61441 [00:37<01:41, 446.29it/s]
26%|       | 16237/61441 [00:37<01:37, 462.40it/s]
27%|       | 16285/61441 [00:37<01:39, 455.73it/s]
27%|       | 16332/61441 [00:37<01:40, 446.83it/s]
27%|       | 16378/61441 [00:37<01:43, 434.71it/s]
27%|       | 16426/61441 [00:37<01:42, 438.14it/s]
27%|       | 16473/61441 [00:38<01:40, 446.31it/s]
27%|       | 16525/61441 [00:38<01:37, 460.43it/s]
27%|       | 16572/61441 [00:38<01:37, 457.90it/s]
27%|       | 16618/61441 [00:38<01:44, 430.01it/s]
27%|       | 16662/61441 [00:38<01:43, 432.17it/s]
27%|       | 16713/61441 [00:38<01:40, 442.97it/s]
27%|       | 16758/61441 [00:38<01:42, 437.21it/s]
27%|       | 16802/61441 [00:38<01:50, 402.62it/s]
27%|       | 16848/61441 [00:38<01:46, 416.97it/s]
27%|       | 16892/61441 [00:39<01:46, 419.13it/s]
```

```
28%|       | 16941/61441 [00:39<01:44, 426.84it/s]
28%|       | 17009/61441 [00:39<01:34, 468.34it/s]
28%|       | 17058/61441 [00:39<01:37, 457.42it/s]
28%|       | 17105/61441 [00:39<01:44, 425.79it/s]
28%|       | 17149/61441 [00:39<01:45, 419.17it/s]
28%|       | 17192/61441 [00:39<01:49, 403.80it/s]
28%|       | 17234/61441 [00:39<01:48, 408.15it/s]
28%|       | 17282/61441 [00:39<01:44, 424.48it/s]
28%|       | 17333/61441 [00:40<01:38, 446.43it/s]
28%|       | 17379/61441 [00:40<01:42, 430.18it/s]
28%|       | 17429/61441 [00:40<01:38, 447.73it/s]
28%|       | 17475/61441 [00:40<01:45, 416.82it/s]
29%|       | 17526/61441 [00:40<01:39, 440.48it/s]
29%|       | 17572/61441 [00:40<01:39, 441.22it/s]
29%|       | 17617/61441 [00:40<01:46, 412.25it/s]
29%|       | 17660/61441 [00:40<01:49, 399.77it/s]
29%|       | 17702/61441 [00:40<01:48, 402.50it/s]
29%|       | 17745/61441 [00:41<01:48, 402.79it/s]
29%|       | 17786/61441 [00:41<01:51, 392.16it/s]
29%|       | 17827/61441 [00:41<01:50, 396.17it/s]
29%|       | 17875/61441 [00:41<01:44, 415.54it/s]
29%|       | 17917/61441 [00:41<01:53, 384.28it/s]
29%|       | 17967/61441 [00:41<01:45, 411.85it/s]
29%|       | 18021/61441 [00:41<01:40, 431.38it/s]
```

```
29%|       | 18067/61441 [00:41<01:39, 438.05it/s]
29%|       | 18120/61441 [00:41<01:34, 456.83it/s]
30%|       | 18171/61441 [00:42<01:31, 471.11it/s]
30%|       | 18229/61441 [00:42<01:27, 494.89it/s]
30%|       | 18288/61441 [00:42<01:23, 518.89it/s]
30%|       | 18341/61441 [00:42<01:35, 453.42it/s]
30%|       | 18389/61441 [00:42<01:45, 409.68it/s]
30%|       | 18433/61441 [00:42<01:49, 394.54it/s]
30%|       | 18490/61441 [00:42<01:39, 432.35it/s]
30%|       | 18536/61441 [00:42<01:39, 431.41it/s]
30%|       | 18585/61441 [00:42<01:36, 445.99it/s]
30%|       | 18641/61441 [00:43<01:30, 474.28it/s]
30%|       | 18690/61441 [00:43<01:34, 452.75it/s]
30%|       | 18737/61441 [00:43<01:38, 435.23it/s]
31%|       | 18791/61441 [00:43<01:32, 459.99it/s]
31%|       | 18839/61441 [00:43<01:38, 433.85it/s]
31%|       | 18887/61441 [00:43<01:35, 445.80it/s]
31%|       | 18939/61441 [00:43<01:31, 462.38it/s]
31%|       | 18986/61441 [00:43<01:33, 451.98it/s]
31%|       | 19032/61441 [00:43<01:40, 420.62it/s]
31%|       | 19078/61441 [00:44<01:38, 429.58it/s]
31%|       | 19129/61441 [00:44<01:33, 450.72it/s]
31%|       | 19175/61441 [00:44<01:37, 434.24it/s]
31%|      | 19230/61441 [00:44<01:31, 459.64it/s]
```

```
31%|      | 19277/61441 [00:44<01:38, 428.43it/s]
31%|      | 19327/61441 [00:44<01:35, 440.77it/s]
32%|      | 19373/61441 [00:44<01:36, 437.70it/s]
32%|      | 19419/61441 [00:44<01:34, 443.52it/s]
32%|      | 19466/61441 [00:44<01:46, 395.93it/s]
32%|      | 19518/61441 [00:45<01:38, 426.20it/s]
32%|      | 19577/61441 [00:45<01:31, 457.91it/s]
32%|      | 19625/61441 [00:45<01:30, 460.09it/s]
32%|      | 19673/61441 [00:45<01:29, 465.11it/s]
32%|      | 19737/61441 [00:45<01:22, 506.02it/s]
32%|      | 19790/61441 [00:45<01:31, 455.06it/s]
32%|      | 19848/61441 [00:45<01:25, 485.36it/s]
32%|      | 19899/61441 [00:45<01:37, 427.10it/s]
32%|      | 19949/61441 [00:45<01:33, 445.33it/s]
33%|      | 19996/61441 [00:46<01:35, 432.03it/s]
33%|      | 20041/61441 [00:46<01:37, 423.30it/s]
33%|      | 20085/61441 [00:46<01:38, 418.22it/s]
33%|      | 20133/61441 [00:46<01:40, 409.78it/s]
33%|      | 20180/61441 [00:46<01:37, 425.23it/s]
33%|      | 20224/61441 [00:46<01:50, 372.75it/s]
33%|      | 20272/61441 [00:46<01:43, 399.40it/s]
33%|      | 20316/61441 [00:46<01:40, 410.30it/s]
33%|      | 20374/61441 [00:46<01:31, 448.54it/s]
33%|      | 20421/61441 [00:47<01:36, 422.95it/s]
```

```
33%|      | 20473/61441 [00:47<01:31, 448.02it/s]
33%|      | 20521/61441 [00:47<01:29, 456.17it/s]
33%|      | 20568/61441 [00:47<01:31, 447.99it/s]
34%|      | 20614/61441 [00:47<01:35, 426.57it/s]
34%|      | 20658/61441 [00:47<01:35, 427.27it/s]
34%|      | 20702/61441 [00:47<01:39, 410.51it/s]
34%|      | 20759/61441 [00:47<01:30, 447.64it/s]
34%|      | 20811/61441 [00:47<01:27, 466.67it/s]
34%|      | 20859/61441 [00:48<01:28, 457.59it/s]
34%|      | 20906/61441 [00:48<01:37, 415.02it/s]
34%|      | 20955/61441 [00:48<01:33, 433.97it/s]
34%|      | 21000/61441 [00:48<01:35, 423.86it/s]
34%|      | 21052/61441 [00:48<01:30, 448.57it/s]
34%|      | 21098/61441 [00:48<01:32, 438.45it/s]
34%|      | 21146/61441 [00:48<01:30, 446.01it/s]
34%|      | 21197/61441 [00:48<01:27, 460.68it/s]
35%|      | 21244/61441 [00:48<01:29, 448.19it/s]
35%|      | 21291/61441 [00:49<01:28, 451.86it/s]
35%|      | 21337/61441 [00:49<01:33, 431.11it/s]
35%|      | 21381/61441 [00:49<01:45, 379.25it/s]
35%|      | 21436/61441 [00:49<01:35, 417.31it/s]
35%|      | 21482/61441 [00:49<01:33, 427.36it/s]
35%|      | 21527/61441 [00:49<01:37, 409.33it/s]
35%|      | 21573/61441 [00:49<01:34, 422.66it/s]
```

```
35%|      | 21617/61441 [00:50<02:17, 289.00it/s]
35%|      | 21662/61441 [00:50<02:03, 321.26it/s]
35%|      | 21704/61441 [00:50<01:57, 338.73it/s]
35%|      | 21749/61441 [00:50<01:48, 365.20it/s]
35%|      | 21792/61441 [00:50<01:44, 380.20it/s]
36%|      | 21839/61441 [00:50<01:38, 400.85it/s]
36%|      | 21883/61441 [00:50<01:36, 411.11it/s]
36%|      | 21926/61441 [00:50<01:35, 413.58it/s]
36%|      | 21969/61441 [00:50<01:48, 365.14it/s]
36%|      | 22011/61441 [00:50<01:46, 369.28it/s]
36%|      | 22054/61441 [00:51<01:42, 383.01it/s]
36%|      | 22094/61441 [00:51<01:42, 383.51it/s]
36%|      | 22134/61441 [00:51<01:55, 340.88it/s]
36%|      | 22181/61441 [00:51<01:47, 364.10it/s]
36%|      | 22233/61441 [00:51<01:38, 397.33it/s]
36%|      | 22275/61441 [00:51<01:40, 390.04it/s]
36%|      | 22316/61441 [00:51<01:45, 370.11it/s]
36%|      | 22369/61441 [00:51<01:36, 406.20it/s]
36%|      | 22412/61441 [00:51<01:36, 403.03it/s]
37%|      | 22454/61441 [00:52<01:38, 397.41it/s]
37%|      | 22498/61441 [00:52<01:35, 408.51it/s]
37%|      | 22556/61441 [00:52<01:27, 446.26it/s]
37%|      | 22603/61441 [00:52<01:33, 415.85it/s]
37%|      | 22653/61441 [00:52<01:29, 434.36it/s]
```

```
37%|      | 22705/61441 [00:52<01:26, 445.36it/s]
37%|      | 22751/61441 [00:52<01:26, 446.08it/s]
37%|      | 22797/61441 [00:52<01:35, 404.46it/s]
37%|      | 22842/61441 [00:52<01:33, 414.82it/s]
37%|      | 22890/61441 [00:53<01:29, 431.23it/s]
37%|      | 22934/61441 [00:53<01:32, 415.42it/s]
37%|      | 22984/61441 [00:53<01:27, 437.46it/s]
38%|      | 23046/61441 [00:53<01:20, 479.68it/s]
38%|      | 23096/61441 [00:53<01:19, 485.27it/s]
38%|      | 23146/61441 [00:53<01:19, 481.02it/s]
38%|      | 23195/61441 [00:53<01:26, 441.19it/s]
38%|      | 23241/61441 [00:53<01:39, 383.44it/s]
38%|      | 23295/61441 [00:54<01:31, 418.90it/s]
38%|      | 23340/61441 [00:54<01:30, 420.89it/s]
38%|      | 23384/61441 [00:54<01:33, 408.89it/s]
38%|      | 23427/61441 [00:54<01:32, 411.81it/s]
38%|      | 23471/61441 [00:54<01:31, 414.40it/s]
38%|      | 23514/61441 [00:54<01:34, 402.64it/s]
38%|      | 23555/61441 [00:54<01:33, 404.71it/s]
38%|      | 23596/61441 [00:54<01:49, 344.12it/s]
38%|      | 23640/61441 [00:54<01:47, 350.08it/s]
39%|      | 23691/61441 [00:55<01:37, 385.60it/s]
39%|      | 23733/61441 [00:55<01:37, 386.47it/s]
39%|      | 23783/61441 [00:55<01:31, 412.13it/s]
```

```
39%|      | 23826/61441 [00:55<01:33, 400.26it/s]
39%|      | 23868/61441 [00:55<01:35, 392.24it/s]
39%|      | 23909/61441 [00:55<01:34, 396.45it/s]
39%|      | 23957/61441 [00:55<01:30, 414.95it/s]
39%|      | 24000/61441 [00:55<01:37, 385.79it/s]
39%|      | 24046/61441 [00:55<01:32, 404.94it/s]
39%|      | 24088/61441 [00:56<01:33, 398.51it/s]
39%|      | 24130/61441 [00:56<01:32, 401.53it/s]
39%|      | 24181/61441 [00:56<01:26, 428.67it/s]
39%|      | 24225/61441 [00:56<01:29, 417.93it/s]
39%|      | 24268/61441 [00:56<01:40, 369.99it/s]
40%|      | 24316/61441 [00:56<01:35, 390.18it/s]
40%|      | 24357/61441 [00:56<01:36, 385.99it/s]
40%|      | 24398/61441 [00:56<01:35, 388.27it/s]
40%|      | 24438/61441 [00:56<01:39, 373.65it/s]
40%|      | 24476/61441 [00:57<01:46, 345.71it/s]
40%|      | 24514/61441 [00:57<01:45, 350.47it/s]
40%|      | 24553/61441 [00:57<01:42, 360.86it/s]
40%|      | 24590/61441 [00:57<01:44, 353.76it/s]
40%|      | 24626/61441 [00:57<01:47, 343.52it/s]
40%|      | 24676/61441 [00:57<01:37, 377.39it/s]
40%|      | 24725/61441 [00:57<01:30, 404.93it/s]
40%|      | 24767/61441 [00:57<01:32, 394.39it/s]
40%|      | 24808/61441 [00:57<01:37, 375.12it/s]
```

```
40%|      | 24852/61441 [00:58<01:33, 391.04it/s]
41%|      | 24899/61441 [00:58<01:29, 408.28it/s]
41%|      | 24941/61441 [00:58<01:30, 404.56it/s]
41%|      | 24986/61441 [00:58<01:30, 403.38it/s]
41%|      | 25027/61441 [00:58<01:37, 374.25it/s]
41%|      | 25069/61441 [00:58<01:34, 384.42it/s]
41%|      | 25108/61441 [00:58<01:35, 381.66it/s]
41%|      | 25147/61441 [00:58<01:36, 375.86it/s]
41%|      | 25185/61441 [00:58<01:40, 359.00it/s]
41%|      | 25224/61441 [00:58<01:39, 365.48it/s]
41%|      | 25265/61441 [00:59<01:35, 377.53it/s]
41%|      | 25304/61441 [00:59<01:38, 366.26it/s]
41%|      | 25341/61441 [00:59<01:40, 360.61it/s]
41%|     | 25390/61441 [00:59<01:32, 390.08it/s]
41%|     | 25430/61441 [00:59<01:34, 381.61it/s]
41%|     | 25469/61441 [00:59<01:37, 367.41it/s]
42%|     | 25519/61441 [00:59<01:30, 398.75it/s]
42%|     | 25561/61441 [00:59<01:31, 393.17it/s]
42%|     | 25602/61441 [00:59<01:33, 384.09it/s]
42%|     | 25642/61441 [01:00<01:34, 377.99it/s]
42%|     | 25686/61441 [01:00<01:30, 393.64it/s]
42%|     | 25728/61441 [01:00<01:29, 401.00it/s]
42%|     | 25776/61441 [01:00<01:28, 403.89it/s]
42%|     | 25821/61441 [01:00<01:25, 416.68it/s]
```

```
42%|     | 25863/61441 [01:00<01:31, 387.89it/s]
42%|     | 25903/61441 [01:00<01:30, 390.87it/s]
42%|     | 25943/61441 [01:00<01:33, 377.70it/s]
42%|     | 25982/61441 [01:00<01:34, 373.46it/s]
42%|     | 26037/61441 [01:01<01:25, 412.86it/s]
42%|     | 26080/61441 [01:01<01:26, 407.88it/s]
43%|     | 26122/61441 [01:01<01:29, 395.32it/s]
43%|     | 26163/61441 [01:01<01:41, 347.46it/s]
43%|     | 26205/61441 [01:01<01:36, 366.25it/s]
43%|     | 26244/61441 [01:01<01:44, 335.89it/s]
43%|     | 26282/61441 [01:01<01:42, 344.64it/s]
43%|     | 26327/61441 [01:01<01:34, 369.86it/s]
43%|     | 26366/61441 [01:01<01:44, 335.48it/s]
43%|     | 26402/61441 [01:02<01:45, 332.42it/s]
43%|     | 26437/61441 [01:02<01:53, 309.56it/s]
43%|     | 26480/61441 [01:02<01:51, 314.90it/s]
43%|     | 26528/61441 [01:02<01:39, 349.44it/s]
43%|     | 26567/61441 [01:02<01:38, 354.63it/s]
43%|     | 26607/61441 [01:02<01:35, 365.35it/s]
43%|     | 26645/61441 [01:02<01:34, 366.83it/s]
43%|     | 26692/61441 [01:02<01:29, 389.54it/s]
44%|     | 26732/61441 [01:03<01:33, 369.48it/s]
44%|     | 26770/61441 [01:03<01:40, 344.49it/s]
44%|     | 26810/61441 [01:03<01:36, 359.41it/s]
```

```
44%|     | 26856/61441 [01:03<01:31, 379.87it/s]
44%|     | 26903/61441 [01:03<01:26, 401.48it/s]
44%|     | 26945/61441 [01:03<01:33, 367.91it/s]
44%|     | 26985/61441 [01:03<01:32, 372.36it/s]
44%|     | 27028/61441 [01:03<01:29, 385.05it/s]
44%|     | 27068/61441 [01:03<01:36, 355.72it/s]
44%|     | 27120/61441 [01:04<01:27, 391.11it/s]
44%|     | 27181/61441 [01:04<01:18, 433.92it/s]
44%|     | 27228/61441 [01:04<01:20, 425.94it/s]
44%|     | 27273/61441 [01:04<01:26, 394.01it/s]
44%|     | 27315/61441 [01:04<01:25, 400.28it/s]
45%|     | 27364/61441 [01:04<01:20, 422.24it/s]
45%|     | 27415/61441 [01:04<01:16, 443.18it/s]
45%|     | 27465/61441 [01:04<01:14, 458.06it/s]
45%|     | 27519/61441 [01:04<01:10, 477.83it/s]
45%|     | 27568/61441 [01:05<01:36, 351.53it/s]
45%|     | 27619/61441 [01:05<01:27, 387.04it/s]
45%|     | 27670/61441 [01:05<01:21, 413.63it/s]
45%|     | 27716/61441 [01:05<01:19, 423.44it/s]
45%|     | 27772/61441 [01:05<01:14, 452.99it/s]
45%|     | 27821/61441 [01:05<01:12, 460.83it/s]
45%|     | 27869/61441 [01:05<01:16, 439.28it/s]
45%|     | 27922/61441 [01:05<01:12, 460.38it/s]
46%|     | 27970/61441 [01:05<01:11, 465.01it/s]
```

```
46%|     | 28018/61441 [01:06<01:11, 465.74it/s]
46%|     | 28066/61441 [01:06<01:14, 448.49it/s]
46%|     | 28117/61441 [01:06<01:12, 458.56it/s]
46%|     | 28164/61441 [01:06<01:17, 431.35it/s]
46%|     | 28208/61441 [01:06<01:24, 392.85it/s]
46%|     | 28251/61441 [01:06<01:22, 401.10it/s]
46%|     | 28314/61441 [01:06<01:15, 440.10it/s]
46%|     | 28360/61441 [01:06<01:15, 440.16it/s]
46%|     | 28416/61441 [01:06<01:10, 469.87it/s]
46%|     | 28468/61441 [01:07<01:08, 481.95it/s]
46%|     | 28518/61441 [01:07<01:12, 453.81it/s]
47%|     | 28574/61441 [01:07<01:08, 480.77it/s]
47%|     | 28628/61441 [01:07<01:07, 487.39it/s]
47%|     | 28681/61441 [01:07<01:05, 498.46it/s]
47%|     | 28732/61441 [01:07<01:13, 446.28it/s]
47%|     | 28784/61441 [01:07<01:10, 465.06it/s]
47%|     | 28832/61441 [01:07<01:12, 452.79it/s]
47%|     | 28879/61441 [01:07<01:16, 427.45it/s]
47%|     | 28932/61441 [01:08<01:12, 450.37it/s]
47%|     | 28979/61441 [01:08<01:17, 417.71it/s]
47%|     | 29022/61441 [01:08<01:22, 392.88it/s]
47%|     | 29063/61441 [01:08<01:25, 378.50it/s]
47%|     | 29106/61441 [01:08<01:23, 388.34it/s]
47%|     | 29146/61441 [01:08<01:27, 370.65it/s]
```

```
47%|     | 29184/61441 [01:08<01:28, 365.24it/s]
48%|     | 29234/61441 [01:08<01:21, 397.21it/s]
48%|     | 29275/61441 [01:08<01:25, 374.56it/s]
48%|     | 29315/61441 [01:09<01:24, 380.14it/s]
48%|     | 29357/61441 [01:09<01:22, 390.10it/s]
48%|     | 29411/61441 [01:09<01:15, 424.47it/s]
48%|     | 29455/61441 [01:09<01:16, 418.93it/s]
48%|     | 29505/61441 [01:09<01:12, 439.55it/s]
48%|     | 29550/61441 [01:09<01:18, 408.66it/s]
48%|     | 29596/61441 [01:09<01:15, 421.86it/s]
48%|     | 29652/61441 [01:09<01:10, 451.56it/s]
48%|     | 29699/61441 [01:09<01:16, 414.07it/s]
48%|     | 29746/61441 [01:10<01:13, 429.09it/s]
48%|     | 29791/61441 [01:10<01:14, 426.90it/s]
49%|     | 29840/61441 [01:10<01:13, 432.27it/s]
49%|     | 29884/61441 [01:10<01:13, 431.02it/s]
49%|     | 29932/61441 [01:10<01:12, 436.85it/s]
49%|     | 29976/61441 [01:10<01:18, 400.34it/s]
49%|     | 30044/61441 [01:10<01:08, 456.63it/s]
49%|     | 30105/61441 [01:10<01:05, 481.40it/s]
49%|     | 30156/61441 [01:10<01:08, 455.53it/s]
49%|     | 30217/61441 [01:11<01:03, 490.69it/s]
49%|     | 30269/61441 [01:11<01:09, 450.51it/s]
49%|     | 30319/61441 [01:11<01:07, 460.98it/s]
```

```
49%|     | 30367/61441 [01:11<01:13, 422.46it/s]
49%|     | 30412/61441 [01:11<01:19, 388.65it/s]
50%|     | 30453/61441 [01:11<01:19, 388.24it/s]
50%|     | 30495/61441 [01:11<01:18, 394.17it/s]
50%|     | 30546/61441 [01:11<01:13, 422.37it/s]
50%|     | 30598/61441 [01:11<01:09, 445.27it/s]
50%|     | 30644/61441 [01:12<01:10, 438.84it/s]
50%|     | 30689/61441 [01:12<01:10, 439.22it/s]
50%|     | 30734/61441 [01:12<01:13, 418.25it/s]
50%|     | 30777/61441 [01:12<01:15, 407.57it/s]
50%|     | 30827/61441 [01:12<01:11, 425.98it/s]
50%|     | 30880/61441 [01:12<01:08, 445.21it/s]
50%|     | 30926/61441 [01:12<01:08, 443.20it/s]
50%|     | 30984/61441 [01:12<01:04, 474.81it/s]
51%|     | 31033/61441 [01:12<01:05, 467.42it/s]
51%|     | 31081/61441 [01:13<01:09, 439.82it/s]
51%|     | 31126/61441 [01:13<01:10, 428.72it/s]
51%|     | 31170/61441 [01:13<01:14, 408.23it/s]
51%|     | 31221/61441 [01:13<01:09, 432.93it/s]
51%|     | 31275/61441 [01:13<01:10, 428.51it/s]
51%|     | 31319/61441 [01:13<01:14, 403.14it/s]
51%|     | 31361/61441 [01:13<01:16, 392.26it/s]
51%|     | 31402/61441 [01:13<01:15, 396.68it/s]
51%|     | 31451/61441 [01:13<01:11, 417.46it/s]
```

```
51%|    | 31502/61441 [01:14<01:08, 434.06it/s]
51%|    | 31550/61441 [01:14<01:07, 443.53it/s]
51%|    | 31595/61441 [01:14<01:14, 398.27it/s]
52%|    | 31644/61441 [01:14<01:10, 419.83it/s]
52%|    | 31700/61441 [01:14<01:05, 453.49it/s]
52%|    | 31747/61441 [01:14<01:09, 428.08it/s]
52%|    | 31795/61441 [01:14<01:07, 440.17it/s]
52%|    | 31844/61441 [01:14<01:05, 453.45it/s]
52%|    | 31891/61441 [01:14<01:08, 428.68it/s]
52%|    | 31943/61441 [01:15<01:05, 451.61it/s]
52%|    | 31990/61441 [01:15<01:06, 444.67it/s]
52%|    | 32036/61441 [01:15<01:05, 447.36it/s]
52%|    | 32082/61441 [01:15<01:05, 446.68it/s]
52%|    | 32128/61441 [01:15<01:12, 401.99it/s]
52%|    | 32170/61441 [01:15<01:13, 399.86it/s]
52%|    | 32211/61441 [01:15<01:19, 369.87it/s]
53%|    | 32264/61441 [01:15<01:12, 402.38it/s]
53%|    | 32306/61441 [01:16<01:16, 383.25it/s]
53%|    | 32363/61441 [01:16<01:08, 424.90it/s]
53%|    | 32416/61441 [01:16<01:04, 449.22it/s]
53%|    | 32463/61441 [01:16<01:04, 449.03it/s]
53%|    | 32511/61441 [01:16<01:03, 453.92it/s]
53%|    | 32559/61441 [01:16<01:03, 456.99it/s]
53%|    | 32606/61441 [01:16<01:02, 459.59it/s]
```

```
53%|    | 32653/61441 [01:16<01:04, 445.86it/s]
53%|    | 32701/61441 [01:16<01:07, 425.69it/s]
53%|    | 32745/61441 [01:16<01:10, 408.90it/s]
53%|    | 32795/61441 [01:17<01:06, 429.23it/s]
53%|    | 32839/61441 [01:17<01:07, 426.31it/s]
54%|    | 32900/61441 [01:17<01:01, 466.31it/s]
54%|    | 32949/61441 [01:17<01:02, 457.04it/s]
54%|    | 33003/61441 [01:17<00:59, 478.64it/s]
54%|    | 33052/61441 [01:17<00:59, 478.74it/s]
54%|    | 33106/61441 [01:17<00:57, 494.06it/s]
54%|    | 33157/61441 [01:17<00:58, 481.17it/s]
54%|    | 33206/61441 [01:17<01:03, 443.24it/s]
54%|    | 33255/61441 [01:18<01:02, 453.97it/s]
54%|    | 33302/61441 [01:18<01:04, 436.38it/s]
54%|    | 33360/61441 [01:18<00:59, 468.63it/s]
54%|    | 33408/61441 [01:18<01:06, 421.86it/s]
54%|    | 33460/61441 [01:18<01:03, 440.76it/s]
55%|    | 33507/61441 [01:18<01:02, 447.92it/s]
55%|    | 33558/61441 [01:18<01:00, 461.64it/s]
55%|    | 33615/61441 [01:18<00:56, 489.10it/s]
55%|    | 33665/61441 [01:18<01:01, 453.43it/s]
55%|    | 33712/61441 [01:19<01:04, 428.44it/s]
55%|    | 33760/61441 [01:19<01:03, 437.07it/s]
55%|    | 33821/61441 [01:19<00:58, 475.50it/s]
```

```
55%|    | 33875/61441 [01:19<00:56, 491.54it/s]
55%|    | 33926/61441 [01:19<01:00, 458.53it/s]
55%|    | 33974/61441 [01:19<01:01, 447.07it/s]
55%|    | 34020/61441 [01:19<01:03, 431.70it/s]
55%|    | 34064/61441 [01:19<01:09, 394.61it/s]
56%|    | 34105/61441 [01:19<01:15, 363.54it/s]
56%|    | 34155/61441 [01:20<01:09, 394.58it/s]
56%|    | 34211/61441 [01:20<01:02, 432.31it/s]
56%|    | 34257/61441 [01:20<01:02, 435.25it/s]
56%|    | 34303/61441 [01:20<01:07, 402.24it/s]
56%|    | 34349/61441 [01:20<01:04, 417.59it/s]
56%|    | 34393/61441 [01:20<01:05, 415.35it/s]
56%|    | 34451/61441 [01:20<00:59, 453.30it/s]
56%|    | 34498/61441 [01:20<01:00, 444.80it/s]
56%|    | 34544/61441 [01:21<01:08, 391.27it/s]
56%|    | 34594/61441 [01:21<01:06, 405.02it/s]
56%|    | 34637/61441 [01:21<01:05, 407.78it/s]
56%|    | 34684/61441 [01:21<01:03, 424.19it/s]
57%|    | 34732/61441 [01:21<01:01, 432.44it/s]
57%|    | 34776/61441 [01:21<01:01, 432.22it/s]
57%|    | 34820/61441 [01:21<01:05, 403.94it/s]
57%|    | 34867/61441 [01:21<01:03, 418.65it/s]
57%|    | 34910/61441 [01:21<01:03, 418.47it/s]
57%|    | 34953/61441 [01:21<01:05, 406.26it/s]
```

```
57%|    | 34996/61441 [01:22<01:05, 403.13it/s]
57%|    | 35055/61441 [01:22<00:59, 444.92it/s]
57%|    | 35108/61441 [01:22<00:56, 463.84it/s]
57%|    | 35156/61441 [01:22<01:01, 426.26it/s]
57%|    | 35204/61441 [01:22<00:59, 440.81it/s]
57%|    | 35250/61441 [01:22<01:01, 425.42it/s]
57%|    | 35298/61441 [01:22<00:59, 436.82it/s]
58%|    | 35353/61441 [01:22<00:57, 453.73it/s]
58%|    | 35400/61441 [01:22<01:00, 433.37it/s]
58%|    | 35449/61441 [01:23<00:57, 448.40it/s]
58%|    | 35495/61441 [01:23<01:00, 429.84it/s]
58%|    | 35544/61441 [01:23<00:58, 446.12it/s]
58%|    | 35601/61441 [01:23<00:54, 477.01it/s]
58%|    | 35650/61441 [01:23<00:56, 458.55it/s]
58%|    | 35702/61441 [01:23<00:54, 474.15it/s]
58%|    | 35751/61441 [01:23<00:55, 459.70it/s]
58%|    | 35798/61441 [01:23<00:56, 456.49it/s]
58%|    | 35847/61441 [01:23<00:54, 465.94it/s]
58%|    | 35894/61441 [01:24<01:01, 414.83it/s]
58%|    | 35937/61441 [01:24<01:01, 415.50it/s]
59%|    | 35980/61441 [01:24<01:01, 411.56it/s]
59%|    | 36022/61441 [01:24<01:07, 376.66it/s]
59%|    | 36074/61441 [01:24<01:03, 402.15it/s]
59%|    | 36135/61441 [01:24<00:56, 444.01it/s]
```

```
59%|    | 36182/61441 [01:24<01:00, 419.34it/s]
59%|    | 36226/61441 [01:24<01:01, 409.22it/s]
59%|    | 36279/61441 [01:24<00:57, 437.26it/s]
59%|    | 36326/61441 [01:25<00:56, 445.41it/s]
59%|    | 36372/61441 [01:25<00:57, 435.03it/s]
59%|    | 36418/61441 [01:25<00:56, 440.64it/s]
59%|    | 36463/61441 [01:25<00:57, 436.51it/s]
59%|    | 36508/61441 [01:25<01:01, 402.28it/s]
59%|    | 36550/61441 [01:25<01:02, 397.86it/s]
60%|    | 36597/61441 [01:25<00:59, 416.35it/s]
60%|    | 36661/61441 [01:25<00:53, 462.87it/s]
60%|    | 36710/61441 [01:25<00:54, 450.14it/s]
60%|    | 36770/61441 [01:26<00:50, 486.44it/s]
60%|    | 36821/61441 [01:26<00:52, 466.69it/s]
60%|    | 36870/61441 [01:26<00:56, 433.60it/s]
60%|    | 36919/61441 [01:26<00:54, 447.06it/s]
60%|    | 36965/61441 [01:26<00:56, 436.48it/s]
60%|    | 37011/61441 [01:26<00:55, 439.40it/s]
60%|    | 37065/61441 [01:26<00:52, 460.62it/s]
60%|    | 37112/61441 [01:26<00:52, 463.06it/s]
60%|    | 37159/61441 [01:26<00:55, 440.33it/s]
61%|    | 37204/61441 [01:27<00:56, 425.96it/s]
61%|    | 37248/61441 [01:27<00:57, 423.44it/s]
61%|    | 37293/61441 [01:27<00:56, 427.36it/s]
```

```
61%|    | 37336/61441 [01:27<00:59, 406.23it/s]
61%|    | 37379/61441 [01:27<00:58, 412.24it/s]
61%|    | 37421/61441 [01:27<00:58, 413.71it/s]
61%|    | 37470/61441 [01:27<00:55, 431.57it/s]
61%|    | 37514/61441 [01:27<01:10, 338.11it/s]
61%|    | 37552/61441 [01:27<01:11, 332.54it/s]
61%|    | 37601/61441 [01:28<01:05, 366.61it/s]
61%|   | 37642/61441 [01:28<01:03, 375.96it/s]
61%|   | 37682/61441 [01:28<01:04, 365.74it/s]
61%|   | 37726/61441 [01:28<01:01, 385.08it/s]
61%|   | 37771/61441 [01:28<00:58, 401.99it/s]
62%|   | 37818/61441 [01:28<00:56, 419.16it/s]
62%|   | 37878/61441 [01:28<00:51, 459.05it/s]
62%|   | 37935/61441 [01:28<00:48, 486.22it/s]
62%|   | 37986/61441 [01:28<00:47, 491.96it/s]
62%|   | 38037/61441 [01:29<00:49, 475.15it/s]
62%|   | 38095/61441 [01:29<00:47, 496.33it/s]
62%|   | 38146/61441 [01:29<00:50, 461.48it/s]
62%|   | 38194/61441 [01:29<00:50, 463.82it/s]
62%|   | 38242/61441 [01:29<00:55, 414.87it/s]
62%|   | 38286/61441 [01:29<00:58, 398.34it/s]
62%|   | 38345/61441 [01:29<00:52, 440.81it/s]
62%|   | 38397/61441 [01:29<00:50, 460.48it/s]
63%|   | 38454/61441 [01:29<00:47, 484.61it/s]
```

```
63%|   | 38505/61441 [01:30<00:47, 481.09it/s]
63%|   | 38555/61441 [01:30<00:48, 467.87it/s]
63%|   | 38603/61441 [01:30<00:56, 404.76it/s]
63%|   | 38660/61441 [01:30<00:51, 442.50it/s]
63%|   | 38707/61441 [01:30<00:56, 402.40it/s]
63%|   | 38770/61441 [01:30<00:50, 450.75it/s]
63%|   | 38819/61441 [01:30<00:50, 450.86it/s]
63%|   | 38867/61441 [01:30<00:52, 433.98it/s]
63%|   | 38923/61441 [01:30<00:48, 461.82it/s]
63%|   | 38972/61441 [01:31<00:48, 460.08it/s]
64%|   | 39020/61441 [01:31<00:48, 464.94it/s]
64%|   | 39068/61441 [01:31<00:49, 452.06it/s]
64%|   | 39120/61441 [01:31<00:47, 467.09it/s]
64%|   | 39168/61441 [01:31<00:48, 463.54it/s]
64%|   | 39221/61441 [01:31<00:47, 472.65it/s]
64%|   | 39269/61441 [01:31<00:48, 457.68it/s]
64%|   | 39316/61441 [01:31<00:50, 434.33it/s]
64%|   | 39379/61441 [01:31<00:46, 477.61it/s]
64%|   | 39429/61441 [01:32<00:50, 438.63it/s]
64%|   | 39475/61441 [01:32<00:49, 440.34it/s]
64%|   | 39521/61441 [01:32<00:52, 414.83it/s]
64%|   | 39578/61441 [01:32<00:48, 450.40it/s]
65%|   | 39637/61441 [01:32<00:45, 480.61it/s]
65%|   | 39687/61441 [01:32<00:46, 466.95it/s]
```

```
65%|   | 39736/61441 [01:32<00:50, 428.86it/s]
65%|   | 39795/61441 [01:32<00:47, 455.57it/s]
65%|   | 39854/61441 [01:33<00:44, 482.64it/s]
65%|   | 39904/61441 [01:33<00:45, 477.91it/s]
65%|   | 39953/61441 [01:33<00:49, 437.34it/s]
65%|   | 39999/61441 [01:33<00:50, 425.60it/s]
65%|   | 40043/61441 [01:33<00:52, 405.76it/s]
65%|   | 40085/61441 [01:33<00:53, 399.52it/s]
65%|   | 40138/61441 [01:33<00:49, 430.42it/s]
65%|   | 40183/61441 [01:33<00:50, 420.62it/s]
65%|   | 40226/61441 [01:33<00:52, 407.58it/s]
66%|   | 40280/61441 [01:34<00:48, 434.31it/s]
66%|   | 40325/61441 [01:34<00:53, 395.17it/s]
66%|   | 40369/61441 [01:34<00:51, 405.88it/s]
66%|   | 40411/61441 [01:34<00:51, 407.59it/s]
66%|   | 40462/61441 [01:34<00:48, 430.02it/s]
66%|   | 40513/61441 [01:34<00:46, 450.85it/s]
66%|   | 40559/61441 [01:34<00:46, 452.18it/s]
66%|   | 40605/61441 [01:34<00:46, 452.00it/s]
66%|   | 40651/61441 [01:34<00:46, 444.16it/s]
66%|   | 40696/61441 [01:34<00:48, 426.46it/s]
66%|   | 40752/61441 [01:35<00:45, 458.07it/s]
66%|   | 40799/61441 [01:35<00:45, 450.14it/s]
66%|   | 40849/61441 [01:35<00:46, 444.03it/s]
```

67%| | 40894/61441 [01:35<00:50, 409.70it/s]

67%| | 40936/61441 [01:35<00:51, 397.91it/s]

67%| | 40987/61441 [01:35<00:48, 424.67it/s]

67%| | 41043/61441 [01:35<00:44, 457.13it/s]

67%| | 41093/61441 [01:35<00:43, 467.22it/s]

67%| | 41141/61441 [01:36<00:51, 392.95it/s]

67%| | 41184/61441 [01:36<00:57, 354.10it/s]

67%| | 41227/61441 [01:36<00:54, 372.93it/s]

67%| | 41267/61441 [01:36<00:53, 374.68it/s]

67%| | 41306/61441 [01:36<00:58, 343.20it/s]

67%| | 41358/61441 [01:36<00:54, 369.60it/s]

67%| | 41413/61441 [01:36<00:48, 409.01it/s]

67%| | 41458/61441 [01:36<00:48, 412.97it/s]

68%| | 41502/61441 [01:36<00:47, 418.22it/s]

68%| | 41558/61441 [01:37<00:44, 451.15it/s]

68%| | 41607/61441 [01:37<00:42, 461.79it/s]

68%| | 41660/61441 [01:37<00:41, 477.01it/s]

68%| | 41716/61441 [01:37<00:39, 493.51it/s]

68%| | 41767/61441 [01:37<00:41, 474.51it/s]

68%| | 41816/61441 [01:37<00:41, 475.69it/s]

68%| | 41865/61441 [01:37<00:45, 428.37it/s]

68%| | 41910/61441 [01:37<00:49, 393.18it/s]

68%| | 41951/61441 [01:37<00:49, 397.68it/s]

68%| | 42004/61441 [01:38<00:45, 428.09it/s]

```
68%|   | 42049/61441 [01:38<00:46, 417.69it/s]
69%|   | 42093/61441 [01:38<00:45, 422.77it/s]
69%|   | 42137/61441 [01:38<00:45, 423.59it/s]
69%|   | 42180/61441 [01:38<00:45, 425.22it/s]
69%|   | 42229/61441 [01:38<00:44, 435.67it/s]
69%|   | 42273/61441 [01:38<00:46, 409.78it/s]
69%|   | 42327/61441 [01:38<00:43, 435.42it/s]
69%|   | 42372/61441 [01:38<00:45, 423.09it/s]
69%|   | 42417/61441 [01:39<00:44, 428.89it/s]
69%|   | 42461/61441 [01:39<00:45, 417.01it/s]
69%|   | 42505/61441 [01:39<00:44, 422.35it/s]
69%|   | 42548/61441 [01:39<00:45, 413.88it/s]
69%|   | 42592/61441 [01:39<00:45, 418.09it/s]
69%|   | 42634/61441 [01:39<00:47, 399.90it/s]
69%|   | 42680/61441 [01:39<00:45, 415.61it/s]
70%|   | 42722/61441 [01:39<00:45, 409.48it/s]
70%|   | 42777/61441 [01:39<00:42, 441.97it/s]
70%|   | 42823/61441 [01:39<00:42, 436.14it/s]
70%|   | 42873/61441 [01:40<00:40, 453.02it/s]
70%|   | 42937/61441 [01:40<00:37, 496.03it/s]
70%|   | 42990/61441 [01:40<00:36, 505.66it/s]
70%|   | 43042/61441 [01:40<00:39, 467.71it/s]
70%|   | 43091/61441 [01:40<00:39, 469.37it/s]
70%|   | 43139/61441 [01:40<00:40, 446.39it/s]
```

```
70%|   | 43188/61441 [01:40<00:40, 455.45it/s]
70%|   | 43235/61441 [01:40<00:40, 454.83it/s]
70%|   | 43282/61441 [01:40<00:39, 457.03it/s]
71%|   | 43329/61441 [01:41<00:40, 442.91it/s]
71%|   | 43377/61441 [01:41<00:39, 452.96it/s]
71%|   | 43423/61441 [01:41<00:40, 446.62it/s]
71%|   | 43468/61441 [01:41<00:42, 421.95it/s]
71%|   | 43511/61441 [01:41<00:45, 392.85it/s]
71%|   | 43565/61441 [01:41<00:41, 426.36it/s]
71%|   | 43615/61441 [01:41<00:40, 442.95it/s]
71%|   | 43661/61441 [01:41<00:40, 434.45it/s]
71%|   | 43718/61441 [01:41<00:37, 467.34it/s]
71%|   | 43767/61441 [01:42<00:40, 440.58it/s]
71%|  | 43813/61441 [01:42<00:41, 427.29it/s]
71%|  | 43866/61441 [01:42<00:38, 452.90it/s]
71%|  | 43913/61441 [01:42<00:39, 449.00it/s]
72%|  | 43959/61441 [01:42<00:44, 395.73it/s]
72%|  | 44017/61441 [01:42<00:39, 435.83it/s]
72%|  | 44064/61441 [01:42<00:39, 442.32it/s]
72%|  | 44114/61441 [01:42<00:38, 448.00it/s]
72%|  | 44161/61441 [01:42<00:38, 447.02it/s]
72%|  | 44207/61441 [01:43<00:43, 393.84it/s]
72%|  | 44258/61441 [01:43<00:40, 422.55it/s]
72%|  | 44303/61441 [01:43<00:41, 416.09it/s]
```

```
72%|  | 44346/61441 [01:43<00:41, 416.87it/s]
72%|  | 44389/61441 [01:43<00:40, 416.17it/s]
72%|  | 44432/61441 [01:43<00:41, 405.83it/s]
72%|  | 44474/61441 [01:43<00:41, 404.19it/s]
72%|  | 44515/61441 [01:43<00:42, 397.19it/s]
73%|  | 44565/61441 [01:43<00:39, 422.25it/s]
73%|  | 44608/61441 [01:44<00:40, 416.92it/s]
73%|  | 44652/61441 [01:44<00:40, 419.31it/s]
73%|  | 44695/61441 [01:44<00:40, 415.32it/s]
73%|  | 44749/61441 [01:44<00:37, 440.33it/s]
73%|  | 44794/61441 [01:44<00:39, 419.90it/s]
73%|  | 44837/61441 [01:44<00:40, 413.59it/s]
73%|  | 44882/61441 [01:44<00:39, 420.33it/s]
73%|  | 44930/61441 [01:44<00:38, 433.74it/s]
73%|  | 44974/61441 [01:44<00:43, 382.88it/s]
73%|  | 45023/61441 [01:45<00:40, 409.65it/s]
73%|  | 45066/61441 [01:45<00:41, 394.88it/s]
73%|  | 45108/61441 [01:45<00:41, 398.24it/s]
73%|  | 45149/61441 [01:45<00:40, 397.81it/s]
74%|  | 45200/61441 [01:45<00:38, 422.66it/s]
74%|  | 45244/61441 [01:45<00:39, 404.99it/s]
74%|  | 45299/61441 [01:45<00:36, 438.87it/s]
74%|  | 45348/61441 [01:45<00:36, 439.55it/s]
74%|  | 45402/61441 [01:45<00:34, 463.41it/s]
```

```
74%|  | 45461/61441 [01:45<00:32, 490.56it/s]
74%|  | 45512/61441 [01:46<00:34, 456.65it/s]
74%|  | 45573/61441 [01:46<00:32, 491.14it/s]
74%|  | 45624/61441 [01:46<00:32, 491.28it/s]
74%|  | 45675/61441 [01:46<00:36, 436.82it/s]
74%|  | 45721/61441 [01:46<00:37, 419.84it/s]
74%|  | 45765/61441 [01:46<00:38, 411.98it/s]
75%|  | 45818/61441 [01:46<00:35, 439.27it/s]
75%|  | 45868/61441 [01:46<00:34, 451.80it/s]
75%|  | 45917/61441 [01:47<00:33, 460.45it/s]
75%|  | 45964/61441 [01:47<00:33, 461.80it/s]
75%|  | 46018/61441 [01:47<00:32, 480.61it/s]
75%|  | 46071/61441 [01:47<00:31, 494.22it/s]
75%|  | 46121/61441 [01:47<00:36, 417.81it/s]
75%|  | 46166/61441 [01:47<00:36, 421.44it/s]
75%|  | 46211/61441 [01:47<00:35, 428.47it/s]
75%|  | 46256/61441 [01:47<00:35, 427.05it/s]
75%|  | 46300/61441 [01:47<00:35, 423.74it/s]
75%|  | 46343/61441 [01:48<00:38, 393.73it/s]
76%|  | 46395/61441 [01:48<00:35, 423.49it/s]
76%|  | 46440/61441 [01:48<00:34, 429.96it/s]
76%|  | 46489/61441 [01:48<00:33, 446.28it/s]
76%|  | 46536/61441 [01:48<00:32, 452.95it/s]
76%|  | 46582/61441 [01:48<00:36, 412.68it/s]
```

```
76%|  | 46625/61441 [01:48<00:36, 410.31it/s]
76%|  | 46686/61441 [01:48<00:33, 438.66it/s]
76%|  | 46731/61441 [01:48<00:33, 436.53it/s]
76%|  | 46777/61441 [01:48<00:33, 439.59it/s]
76%|  | 46823/61441 [01:49<00:33, 437.84it/s]
76%|  | 46868/61441 [01:49<00:33, 434.01it/s]
76%|  | 46912/61441 [01:49<00:35, 414.07it/s]
76%|  | 46961/61441 [01:49<00:33, 433.38it/s]
77%|  | 47005/61441 [01:49<00:33, 429.55it/s]
77%|  | 47049/61441 [01:49<00:33, 423.84it/s]
77%|  | 47101/61441 [01:49<00:32, 448.01it/s]
77%|  | 47147/61441 [01:49<00:33, 429.42it/s]
77%|  | 47201/61441 [01:49<00:31, 457.07it/s]
77%|  | 47250/61441 [01:50<00:30, 465.39it/s]
77%|  | 47307/61441 [01:50<00:29, 487.27it/s]
77%|  | 47357/61441 [01:50<00:29, 478.51it/s]
77%|  | 47406/61441 [01:50<00:29, 476.59it/s]
77%|  | 47458/61441 [01:50<00:29, 481.91it/s]
77%|  | 47507/61441 [01:50<00:29, 467.35it/s]
77%|  | 47555/61441 [01:50<00:31, 442.52it/s]
77%|  | 47600/61441 [01:50<00:31, 433.09it/s]
78%|  | 47659/61441 [01:50<00:29, 467.97it/s]
78%|  | 47709/61441 [01:51<00:28, 473.60it/s]
78%|  | 47758/61441 [01:51<00:30, 442.51it/s]
```

78%| | 47804/61441 [01:51<00:34, 397.01it/s]

78%| | 47855/61441 [01:51<00:33, 410.86it/s]

78%| | 47900/61441 [01:51<00:32, 417.78it/s]

78%| | 47943/61441 [01:51<00:33, 400.09it/s]

78%| | 48001/61441 [01:51<00:30, 439.71it/s]

78%| | 48047/61441 [01:51<00:32, 408.27it/s]

78%| | 48095/61441 [01:51<00:31, 426.62it/s]

78%| | 48140/61441 [01:52<00:32, 407.01it/s]

78%| | 48190/61441 [01:52<00:30, 431.03it/s]

79%| | 48235/61441 [01:52<00:32, 409.51it/s]

79%| | 48278/61441 [01:52<00:33, 397.74it/s]

79%| | 48319/61441 [01:52<00:32, 399.99it/s]

79%| | 48369/61441 [01:52<00:30, 424.36it/s]

79%| | 48421/61441 [01:52<00:29, 447.85it/s]

79%| | 48467/61441 [01:52<00:31, 407.38it/s]

79%| | 48531/61441 [01:52<00:28, 454.81it/s]

79%| | 48580/61441 [01:53<00:30, 426.69it/s]

79%| | 48627/61441 [01:53<00:29, 438.35it/s]

79%| | 48673/61441 [01:53<00:29, 429.15it/s]

79%| | 48718/61441 [01:53<00:29, 424.68it/s]

79%| | 48762/61441 [01:53<00:30, 415.21it/s]

79%| | 48805/61441 [01:53<00:31, 399.47it/s]

80%| | 48846/61441 [01:53<00:32, 392.02it/s]

80%| | 48894/61441 [01:53<00:30, 411.83it/s]

80%| | 48941/61441 [01:53<00:29, 426.55it/s]

80%| | 48987/61441 [01:54<00:28, 432.80it/s]

80%| | 49032/61441 [01:54<00:28, 433.41it/s]

80%| | 49076/61441 [01:54<00:30, 408.95it/s]

80%| | 49118/61441 [01:54<00:30, 409.58it/s]

80%| | 49160/61441 [01:54<00:30, 396.81it/s]

80%| | 49201/61441 [01:54<00:31, 385.01it/s]

80%| | 49240/61441 [01:54<00:32, 373.36it/s]

80%| | 49282/61441 [01:54<00:31, 383.74it/s]

80%| | 49325/61441 [01:54<00:31, 384.40it/s]

80%| | 49373/61441 [01:55<00:29, 404.74it/s]

80%| | 49426/61441 [01:55<00:27, 434.36it/s]

81%| | 49471/61441 [01:55<00:27, 434.17it/s]

81%| | 49516/61441 [01:55<00:27, 431.99it/s]

81%| | 49560/61441 [01:55<00:28, 418.03it/s]

81%| | 49603/61441 [01:55<00:28, 418.09it/s]

81%| | 49646/61441 [01:55<00:28, 408.51it/s]

81%| | 49691/61441 [01:55<00:28, 412.64it/s]

81%| | 49750/61441 [01:55<00:26, 446.86it/s]

81%| | 49801/61441 [01:56<00:25, 463.82it/s]

81%| | 49849/61441 [01:56<00:28, 409.55it/s]

81%| | 49894/61441 [01:56<00:28, 411.41it/s]

81%| | 49937/61441 [01:56<00:27, 415.66it/s]

81%| | 49982/61441 [01:56<00:27, 421.81it/s]

81%| | 50028/61441 [01:56<00:26, 431.90it/s]

81%| | 50074/61441 [01:56<00:25, 439.66it/s]

82%| | 50125/61441 [01:56<00:25, 451.42it/s]

82%| | 50171/61441 [01:56<00:26, 420.79it/s]

82%| | 50214/61441 [01:56<00:26, 423.27it/s]

82%| | 50264/61441 [01:57<00:25, 442.03it/s]

82%| | 50309/61441 [01:57<00:26, 412.59it/s]

82%| | 50366/61441 [01:57<00:24, 449.25it/s]

82%| | 50413/61441 [01:57<00:25, 436.36it/s]

82%| | 50458/61441 [01:57<00:25, 439.25it/s]

82%| | 50506/61441 [01:57<00:24, 450.36it/s]

82%| | 50552/61441 [01:57<00:24, 437.70it/s]

82%| | 50599/61441 [01:57<00:24, 445.99it/s]

82%| | 50644/61441 [01:57<00:25, 430.82it/s]

82%| | 50688/61441 [01:58<00:25, 428.04it/s]

83%| | 50732/61441 [01:58<00:26, 400.78it/s]

83%| | 50773/61441 [01:58<00:27, 387.73it/s]

83%| | 50813/61441 [01:58<00:27, 385.29it/s]

83%| | 50861/61441 [01:58<00:25, 409.05it/s]

83%| | 50903/61441 [01:58<00:28, 366.20it/s]

83%| | 50948/61441 [01:58<00:27, 378.85it/s]

83%| | 50991/61441 [01:58<00:26, 392.23it/s]

83%| | 51041/61441 [01:58<00:25, 414.26it/s]

83%| | 51096/61441 [01:59<00:23, 443.99it/s]

83%| | 51142/61441 [01:59<00:24, 420.75it/s]

83%| | 51192/61441 [01:59<00:23, 436.76it/s]

83%| | 51237/61441 [01:59<00:23, 430.38it/s]

83%| | 51281/61441 [01:59<00:24, 419.96it/s]

84%| | 51324/61441 [01:59<00:24, 420.57it/s]

84%| | 51367/61441 [01:59<00:24, 418.91it/s]

84%| | 51414/61441 [01:59<00:23, 429.88it/s]

84%| | 51458/61441 [01:59<00:23, 425.73it/s]

84%| | 51501/61441 [02:00<00:23, 423.01it/s]

84%| | 51544/61441 [02:00<00:24, 404.34it/s]

84%| | 51592/61441 [02:00<00:23, 420.75it/s]

84%| | 51641/61441 [02:00<00:22, 438.26it/s]

84%| | 51686/61441 [02:00<00:23, 419.03it/s]

84%| | 51729/61441 [02:00<00:23, 417.54it/s]

84%| | 51772/61441 [02:00<00:24, 399.75it/s]

84%| | 51817/61441 [02:00<00:23, 413.01it/s]

84%| | 51877/61441 [02:00<00:21, 453.54it/s]

85%| | 51924/61441 [02:01<00:21, 451.99it/s]

85%| | 51972/61441 [02:01<00:20, 459.38it/s]

85%| | 52019/61441 [02:01<00:20, 460.61it/s]

85%| | 52066/61441 [02:01<00:20, 452.12it/s]

85%| | 52117/61441 [02:01<00:19, 467.10it/s]

85%| | 52165/61441 [02:01<00:20, 460.24it/s]

85%| | 52212/61441 [02:01<00:22, 411.96it/s]

85%| | 52255/61441 [02:01<00:22, 412.54it/s]

85%| | 52298/61441 [02:01<00:22, 412.16it/s]

85%| | 52340/61441 [02:01<00:22, 397.83it/s]

85%| | 52390/61441 [02:02<00:21, 423.31it/s]

85%| | 52434/61441 [02:02<00:21, 425.23it/s]

85%| | 52478/61441 [02:02<00:21, 418.12it/s]

85%| | 52521/61441 [02:02<00:21, 415.16it/s]

86%| | 52563/61441 [02:02<00:24, 355.87it/s]

86%| | 52609/61441 [02:02<00:23, 376.58it/s]

86%| | 52659/61441 [02:02<00:21, 405.87it/s]

86%| | 52709/61441 [02:02<00:20, 428.56it/s]

86%| | 52754/61441 [02:02<00:21, 410.42it/s]

86%| | 52799/61441 [02:03<00:20, 419.37it/s]

86%| | 52849/61441 [02:03<00:19, 437.35it/s]

86%| | 52897/61441 [02:03<00:19, 446.83it/s]

86%| | 52946/61441 [02:03<00:18, 458.81it/s]

86%| | 52998/61441 [02:03<00:17, 470.83it/s]

86%| | 53046/61441 [02:03<00:18, 449.04it/s]

86%| | 53092/61441 [02:03<00:19, 436.63it/s]

87%| | 53149/61441 [02:03<00:17, 468.68it/s]

87%| | 53197/61441 [02:03<00:17, 459.70it/s]

87%| | 53247/61441 [02:04<00:17, 470.09it/s]

87%| | 53295/61441 [02:04<00:17, 467.89it/s]

87%| | 53343/61441 [02:04<00:18, 447.28it/s]

87%| | 53389/61441 [02:04<00:17, 450.26it/s]

87%| | 53439/61441 [02:04<00:17, 463.15it/s]

87%| | 53486/61441 [02:04<00:18, 441.25it/s]

87%| | 53531/61441 [02:04<00:18, 430.03it/s]

87%| | 53577/61441 [02:04<00:18, 436.13it/s]

87%| | 53621/61441 [02:04<00:19, 394.41it/s]

87%| | 53669/61441 [02:05<00:18, 416.50it/s]

87%| | 53712/61441 [02:05<00:19, 406.15it/s]

87%| | 53754/61441 [02:05<00:18, 407.10it/s]

88%| | 53796/61441 [02:05<00:18, 406.88it/s]

88%| | 53845/61441 [02:05<00:17, 427.42it/s]

88%| | 53896/61441 [02:05<00:17, 440.65it/s]

88%| | 53943/61441 [02:05<00:16, 446.42it/s]

88%| | 53996/61441 [02:05<00:16, 448.65it/s]

88%| | 54042/61441 [02:05<00:18, 394.56it/s]

88%| | 54083/61441 [02:06<00:19, 368.51it/s]

88%| | 54128/61441 [02:06<00:18, 387.26it/s]

88%| | 54183/61441 [02:06<00:17, 423.76it/s]

88%| | 54239/61441 [02:06<00:15, 454.80it/s]

88%| | 54287/61441 [02:06<00:16, 435.00it/s]

88%| | 54333/61441 [02:06<00:16, 426.43it/s]

89%| | 54377/61441 [02:06<00:17, 410.25it/s]

89%| | 54424/61441 [02:06<00:16, 425.62it/s]

89%| | 54468/61441 [02:06<00:17, 406.94it/s]

89%| | 54510/61441 [02:07<00:17, 405.89it/s]

89%| | 54552/61441 [02:07<00:17, 395.12it/s]

89%| | 54597/61441 [02:07<00:16, 409.75it/s]

89%| | 54639/61441 [02:07<00:16, 402.05it/s]

89%| | 54680/61441 [02:07<00:17, 388.36it/s]

89%| | 54722/61441 [02:07<00:17, 392.28it/s]

89%| | 54762/61441 [02:07<00:17, 391.36it/s]

89%| | 54802/61441 [02:07<00:17, 376.74it/s]

89%| | 54841/61441 [02:07<00:17, 380.60it/s]

89%| | 54890/61441 [02:08<00:16, 401.38it/s]

89%| | 54931/61441 [02:08<00:16, 388.33it/s]

89%| | 54971/61441 [02:08<00:16, 383.49it/s]

90%| | 55013/61441 [02:08<00:16, 393.09it/s]

90%| | 55053/61441 [02:08<00:17, 368.01it/s]

90%| | 55091/61441 [02:08<00:17, 361.08it/s]

90%| | 55130/61441 [02:08<00:17, 367.12it/s]

90%| | 55178/61441 [02:08<00:15, 394.90it/s]

90%| | 55219/61441 [02:08<00:15, 395.74it/s]

90%| | 55278/61441 [02:08<00:14, 434.38it/s]

90%| | 55326/61441 [02:09<00:13, 443.13it/s]

90%| | 55372/61441 [02:09<00:13, 445.27it/s]

90%| | 55418/61441 [02:09<00:13, 434.89it/s]

90%| | 55463/61441 [02:09<00:13, 434.04it/s]

90%| | 55513/61441 [02:09<00:13, 451.16it/s]

90%| | 55565/61441 [02:09<00:12, 462.79it/s]

91%| | 55612/61441 [02:09<00:13, 446.78it/s]

91%| | 55658/61441 [02:09<00:13, 441.28it/s]

91%| | 55703/61441 [02:09<00:13, 416.10it/s]

91%| | 55747/61441 [02:10<00:13, 422.96it/s]

91%| | 55793/61441 [02:10<00:13, 432.86it/s]

91%| | 55837/61441 [02:10<00:13, 416.20it/s]

91%| | 55879/61441 [02:10<00:13, 413.90it/s]

91%| | 55923/61441 [02:10<00:13, 420.26it/s]

91%| | 55966/61441 [02:10<00:14, 386.02it/s]

91%| | 56006/61441 [02:10<00:14, 371.44it/s]

91%| | 56060/61441 [02:10<00:13, 409.54it/s]

91%|| 56113/61441 [02:10<00:12, 439.02it/s]

91%|| 56162/61441 [02:11<00:11, 453.06it/s]

91%|| 56210/61441 [02:11<00:11, 456.50it/s]

92%|| 56257/61441 [02:11<00:11, 432.49it/s]

92%|| 56302/61441 [02:11<00:12, 417.06it/s]

92%|| 56345/61441 [02:11<00:13, 379.86it/s]

92%|| 56392/61441 [02:11<00:12, 402.00it/s]

92%|| 56434/61441 [02:11<00:13, 359.01it/s]

92%|| 56475/61441 [02:11<00:13, 367.90it/s]

92%|| 56514/61441 [02:11<00:14, 344.23it/s]

92%|| 56550/61441 [02:12<00:14, 335.40it/s]

92%|| 56596/61441 [02:12<00:13, 363.91it/s]

92%|| 56640/61441 [02:12<00:12, 382.70it/s]

92%|| 56691/61441 [02:12<00:11, 412.26it/s]

92%|| 56744/61441 [02:12<00:10, 441.36it/s]

92%|| 56790/61441 [02:12<00:11, 406.79it/s]

93%|| 56833/61441 [02:12<00:12, 376.21it/s]

93%|| 56873/61441 [02:12<00:13, 338.04it/s]

93%|| 56930/61441 [02:13<00:11, 384.99it/s]

93%|| 56973/61441 [02:13<00:11, 383.89it/s]

93%|| 57014/61441 [02:13<00:11, 381.69it/s]

93%|| 57060/61441 [02:13<00:10, 402.04it/s]

93%|| 57102/61441 [02:13<00:10, 406.50it/s]

93%|| 57157/61441 [02:13<00:09, 433.75it/s]

93%|| 57205/61441 [02:13<00:09, 445.89it/s]

93%|| 57251/61441 [02:13<00:09, 443.65it/s]

93%|| 57297/61441 [02:13<00:09, 418.55it/s]

93%|| 57340/61441 [02:13<00:09, 414.12it/s]

93%|| 57385/61441 [02:14<00:09, 421.75it/s]

93%|| 57443/61441 [02:14<00:08, 454.75it/s]

94%|| 57496/61441 [02:14<00:08, 472.83it/s]

94%|| 57545/61441 [02:14<00:08, 436.23it/s]

94%|| 57591/61441 [02:14<00:08, 442.49it/s]

94%|| 57637/61441 [02:14<00:08, 435.99it/s]

94%|| 57682/61441 [02:14<00:09, 397.88it/s]

94%|| 57723/61441 [02:14<00:09, 397.81it/s]

94%|| 57764/61441 [02:15<00:10, 360.55it/s]

94%|| 57802/61441 [02:15<00:10, 355.48it/s]

94%|| 57839/61441 [02:15<00:10, 350.40it/s]

94%|| 57879/61441 [02:15<00:09, 362.44it/s]

94%|| 57927/61441 [02:15<00:09, 379.91it/s]

94%|| 57966/61441 [02:15<00:09, 376.84it/s]

94%|| 58005/61441 [02:15<00:09, 361.29it/s]

94%|| 58045/61441 [02:15<00:09, 371.92it/s]

95%|| 58083/61441 [02:15<00:09, 347.06it/s]

95%|| 58130/61441 [02:15<00:08, 373.27it/s]

95%|| 58173/61441 [02:16<00:08, 388.65it/s]

95%|| 58213/61441 [02:16<00:08, 389.30it/s]

95%|| 58265/61441 [02:16<00:07, 419.93it/s]

95%|| 58309/61441 [02:16<00:07, 424.72it/s]

95%|| 58353/61441 [02:16<00:07, 387.85it/s]

95%|| 58393/61441 [02:16<00:08, 370.55it/s]

95%|| 58434/61441 [02:16<00:07, 380.83it/s]

95%|| 58483/61441 [02:16<00:07, 407.66it/s]

95%|| 58525/61441 [02:16<00:07, 372.77it/s]

95%|| 58570/61441 [02:17<00:07, 392.84it/s]

95%|| 58611/61441 [02:17<00:07, 364.89it/s]

95%|| 58649/61441 [02:17<00:08, 339.08it/s]

96%|| 58689/61441 [02:17<00:07, 352.54it/s]

96%|| 58726/61441 [02:17<00:08, 302.60it/s]

96%|| 58773/61441 [02:17<00:07, 337.84it/s]

96%|| 58810/61441 [02:17<00:07, 336.07it/s]

96%|| 58846/61441 [02:17<00:07, 332.00it/s]

96%|| 58891/61441 [02:18<00:07, 360.34it/s]

96%|| 58933/61441 [02:18<00:06, 374.99it/s]

96%|| 58985/61441 [02:18<00:06, 408.82it/s]

96%|| 59034/61441 [02:18<00:05, 430.07it/s]

96%|| 59080/61441 [02:18<00:05, 437.70it/s]

96%|| 59125/61441 [02:18<00:05, 423.97it/s]

96%|| 59169/61441 [02:18<00:05, 388.60it/s]

96%|| 59226/61441 [02:18<00:05, 426.99it/s]

96%|| 59271/61441 [02:18<00:05, 382.69it/s]

97%|| 59312/61441 [02:19<00:06, 350.73it/s]

97%|| 59354/61441 [02:19<00:05, 367.93it/s]

97%|| 59397/61441 [02:19<00:05, 382.26it/s]

97%|| 59441/61441 [02:19<00:05, 394.50it/s]

97%|| 59482/61441 [02:19<00:05, 384.09it/s]

97%|| 59524/61441 [02:19<00:04, 389.54it/s]

97%|| 59564/61441 [02:19<00:04, 389.09it/s]

97%|| 59617/61441 [02:19<00:04, 422.61it/s]

97%|| 59661/61441 [02:19<00:04, 409.38it/s]

97%|| 59703/61441 [02:20<00:04, 381.43it/s]

97%|| 59743/61441 [02:20<00:04, 378.29it/s]

97%|| 59782/61441 [02:20<00:04, 375.07it/s]

97%|| 59821/61441 [02:20<00:04, 373.55it/s]

97%|| 59859/61441 [02:20<00:04, 341.34it/s]

97%|| 59894/61441 [02:20<00:04, 320.99it/s]

98%|| 59930/61441 [02:20<00:04, 330.79it/s]

98%|| 59964/61441 [02:20<00:04, 309.75it/s]

98%|| 60012/61441 [02:20<00:04, 346.22it/s]

98%|| 60049/61441 [02:21<00:03, 349.79it/s]

98%|| 60086/61441 [02:21<00:04, 335.69it/s]

98%|| 60127/61441 [02:21<00:03, 354.36it/s]

98%|| 60176/61441 [02:21<00:03, 386.20it/s]

98%|| 60217/61441 [02:21<00:03, 384.24it/s]

98%|| 60257/61441 [02:21<00:03, 352.31it/s]

98%|| 60300/61441 [02:21<00:03, 371.71it/s]

98%|| 60348/61441 [02:21<00:02, 395.11it/s]

98%|| 60397/61441 [02:21<00:02, 417.63it/s]

98%|| 60440/61441 [02:22<00:02, 404.32it/s]

98%|| 60483/61441 [02:22<00:02, 409.17it/s]

99%|| 60525/61441 [02:22<00:02, 381.73it/s]

99%|| 60565/61441 [02:22<00:02, 383.74it/s]

99%|| 60604/61441 [02:22<00:02, 349.51it/s]

99%|| 60653/61441 [02:22<00:02, 378.35it/s]

99%|| 60693/61441 [02:22<00:01, 382.98it/s]

99%|| 60740/61441 [02:22<00:01, 401.61it/s]

99%|| 60782/61441 [02:22<00:01, 402.73it/s]

```
 99%|| 60823/61441 [02:23<00:01, 385.52it/s]

 99%|| 60863/61441 [02:23<00:01, 358.30it/s]

 99%|| 60908/61441 [02:23<00:01, 381.62it/s]

 99%|| 60948/61441 [02:23<00:01, 356.19it/s]

 99%|| 60985/61441 [02:23<00:01, 345.86it/s]

 99%|| 61038/61441 [02:23<00:01, 383.88it/s]

 99%|| 61079/61441 [02:23<00:00, 382.17it/s]

 99%|| 61119/61441 [02:23<00:00, 381.88it/s]

100%|| 61159/61441 [02:23<00:00, 367.36it/s]

100%|| 61202/61441 [02:24<00:00, 382.02it/s]

100%|| 61242/61441 [02:24<00:00, 383.52it/s]

100%|| 61285/61441 [02:24<00:00, 385.52it/s]

100%|| 61324/61441 [02:24<00:00, 381.24it/s]

100%|| 61368/61441 [02:24<00:00, 395.59it/s]

100%|| 61417/61441 [02:24<00:00, 414.55it/s]

100%|| 61441/61441 [02:24<00:00, 424.67it/s]
```

```
In [151]: test_sent = [];
          for sent in tqdm(X_test):
              sent_vec = np.zeros(50)
              cnt_words =0;
              for word in sent: #
                  if word in w2v_words:
                      vec = w2v_model.wv[word]
                      sent_vec += vec
                      cnt_words += 1
              if cnt_words != 0:
                  sent_vec /= cnt_words
              test_sent.append(sent_vec)
```

```
  0%|          | 0/26332 [00:00<?, ?it/s]
```

```
0%|          | 53/26332 [00:00<00:49, 525.84it/s]
0%|          | 108/26332 [00:00<00:49, 532.21it/s]
1%|          | 159/26332 [00:00<00:49, 524.03it/s]
1%|          | 203/26332 [00:00<00:53, 491.95it/s]
1%|          | 240/26332 [00:00<01:03, 410.86it/s]
1%|          | 283/26332 [00:00<01:02, 414.07it/s]
1%|         | 331/26332 [00:00<01:00, 431.10it/s]
1%|         | 387/26332 [00:00<00:56, 457.56it/s]
2%|         | 441/26332 [00:00<00:54, 476.97it/s]
2%|         | 488/26332 [00:01<00:54, 474.77it/s]
2%|         | 539/26332 [00:01<00:53, 484.63it/s]
2%|         | 588/26332 [00:01<00:53, 477.34it/s]
2%|         | 637/26332 [00:01<00:53, 479.72it/s]
3%|         | 685/26332 [00:01<00:55, 466.06it/s]
3%|         | 732/26332 [00:01<00:55, 460.31it/s]
3%|         | 779/26332 [00:01<00:56, 453.32it/s]
3%|         | 831/26332 [00:01<00:54, 471.28it/s]
3%|         | 894/26332 [00:01<00:50, 502.75it/s]
4%|         | 945/26332 [00:01<00:51, 496.73it/s]
4%|         | 996/26332 [00:02<00:53, 473.20it/s]
4%|         | 1044/26332 [00:02<00:54, 466.88it/s]
4%|         | 1092/26332 [00:02<00:55, 453.35it/s]
4%|         | 1138/26332 [00:02<00:56, 448.64it/s]
5%|         | 1194/26332 [00:02<00:52, 475.06it/s]
```

```
5%|         | 1256/26332 [00:02<00:49, 510.75it/s]
5%|         | 1309/26332 [00:02<00:52, 479.89it/s]
5%|         | 1359/26332 [00:02<00:53, 469.84it/s]
5%|         | 1407/26332 [00:02<00:55, 451.43it/s]
6%|         | 1465/26332 [00:03<00:51, 481.00it/s]
6%|         | 1517/26332 [00:03<00:50, 488.92it/s]
6%|         | 1570/26332 [00:03<00:49, 495.32it/s]
6%|         | 1621/26332 [00:03<00:50, 485.12it/s]
6%|         | 1670/26332 [00:03<00:50, 485.91it/s]
7%|         | 1719/26332 [00:03<00:55, 444.85it/s]
7%|         | 1765/26332 [00:03<00:57, 427.44it/s]
7%|         | 1809/26332 [00:03<00:58, 421.12it/s]
7%|         | 1852/26332 [00:03<00:59, 410.64it/s]
7%|         | 1894/26332 [00:04<01:03, 386.62it/s]
7%|         | 1934/26332 [00:04<01:02, 389.83it/s]
7%|         | 1974/26332 [00:04<01:04, 374.82it/s]
8%|         | 2030/26332 [00:04<00:58, 415.11it/s]
8%|         | 2074/26332 [00:04<00:59, 409.94it/s]
8%|         | 2120/26332 [00:04<00:57, 422.89it/s]
8%|         | 2164/26332 [00:04<00:57, 416.87it/s]
8%|         | 2223/26332 [00:04<00:52, 455.36it/s]
9%|         | 2271/26332 [00:04<00:57, 417.48it/s]
9%|         | 2319/26332 [00:05<00:55, 434.27it/s]
9%|         | 2364/26332 [00:05<00:56, 427.67it/s]
```

```
 9%|         | 2408/26332 [00:05<00:55, 430.46it/s]
 9%|         | 2460/26332 [00:05<00:52, 453.51it/s]
10%|         | 2507/26332 [00:05<00:53, 443.39it/s]
10%|         | 2558/26332 [00:05<00:51, 460.16it/s]
10%|         | 2605/26332 [00:05<00:55, 431.24it/s]
10%|         | 2652/26332 [00:05<00:53, 441.79it/s]
10%|         | 2697/26332 [00:05<00:53, 443.22it/s]
10%|         | 2750/26332 [00:06<00:50, 464.77it/s]
11%|         | 2801/26332 [00:06<00:49, 475.67it/s]
11%|         | 2850/26332 [00:06<00:50, 460.50it/s]
11%|         | 2904/26332 [00:06<00:48, 479.55it/s]
11%|         | 2953/26332 [00:06<00:52, 449.51it/s]
11%|        | 3005/26332 [00:06<00:50, 466.52it/s]
12%|        | 3063/26332 [00:06<00:48, 475.05it/s]
12%|        | 3112/26332 [00:06<00:56, 407.46it/s]
12%|        | 3163/26332 [00:06<00:53, 430.33it/s]
12%|        | 3208/26332 [00:07<01:00, 380.67it/s]
12%|        | 3249/26332 [00:07<01:07, 340.38it/s]
12%|        | 3286/26332 [00:07<01:11, 322.63it/s]
13%|        | 3335/26332 [00:07<01:04, 358.69it/s]
13%|        | 3376/26332 [00:07<01:02, 369.56it/s]
13%|        | 3432/26332 [00:07<00:56, 404.89it/s]
13%|        | 3476/26332 [00:07<00:55, 414.02it/s]
13%|        | 3520/26332 [00:07<00:54, 419.81it/s]
```

```
14%|        | 3568/26332 [00:07<00:52, 433.35it/s]
14%|        | 3613/26332 [00:08<00:56, 401.45it/s]
14%|        | 3662/26332 [00:08<00:53, 423.59it/s]
14%|        | 3706/26332 [00:08<00:59, 382.35it/s]
14%|        | 3766/26332 [00:08<00:52, 429.02it/s]
14%|        | 3816/26332 [00:08<00:50, 446.73it/s]
15%|        | 3863/26332 [00:08<00:51, 433.23it/s]
15%|        | 3908/26332 [00:08<00:54, 408.07it/s]
15%|        | 3951/26332 [00:08<00:58, 383.17it/s]
15%|        | 4014/26332 [00:09<00:51, 433.66it/s]
15%|        | 4061/26332 [00:09<00:51, 429.63it/s]
16%|        | 4107/26332 [00:09<00:51, 434.93it/s]
16%|        | 4158/26332 [00:09<00:48, 453.94it/s]
16%|        | 4211/26332 [00:09<00:48, 460.09it/s]
16%|        | 4259/26332 [00:09<00:47, 463.57it/s]
16%|        | 4319/26332 [00:09<00:44, 496.71it/s]
17%|        | 4370/26332 [00:09<00:49, 439.84it/s]
17%|        | 4416/26332 [00:09<00:50, 434.53it/s]
17%|        | 4473/26332 [00:10<00:46, 467.54it/s]
17%|        | 4522/26332 [00:10<00:48, 448.99it/s]
17%|        | 4569/26332 [00:10<00:53, 403.98it/s]
18%|        | 4613/26332 [00:10<00:52, 412.07it/s]
18%|        | 4665/26332 [00:10<00:49, 435.63it/s]
18%|        | 4710/26332 [00:10<00:51, 420.06it/s]
```

```
18%|        | 4759/26332 [00:10<00:50, 428.45it/s]
18%|        | 4803/26332 [00:10<00:52, 409.94it/s]
18%|        | 4848/26332 [00:10<00:51, 420.07it/s]
19%|        | 4896/26332 [00:11<00:49, 436.39it/s]
19%|        | 4941/26332 [00:11<00:51, 418.47it/s]
19%|        | 4987/26332 [00:11<00:49, 428.33it/s]
19%|        | 5031/26332 [00:11<00:50, 425.72it/s]
19%|        | 5093/26332 [00:11<00:45, 466.34it/s]
20%|        | 5142/26332 [00:11<00:45, 460.76it/s]
20%|        | 5190/26332 [00:11<00:48, 437.46it/s]
20%|        | 5235/26332 [00:11<00:50, 421.32it/s]
20%|        | 5279/26332 [00:11<00:49, 423.45it/s]
20%|        | 5322/26332 [00:12<00:50, 413.46it/s]
20%|        | 5377/26332 [00:12<00:47, 444.51it/s]
21%|        | 5426/26332 [00:12<00:46, 449.22it/s]
21%|        | 5478/26332 [00:12<00:44, 466.52it/s]
21%|        | 5526/26332 [00:12<00:50, 410.06it/s]
21%|        | 5570/26332 [00:12<00:49, 417.99it/s]
21%|       | 5623/26332 [00:12<00:46, 445.75it/s]
22%|       | 5669/26332 [00:12<00:47, 433.50it/s]
22%|       | 5714/26332 [00:12<00:50, 411.19it/s]
22%|       | 5759/26332 [00:13<00:48, 420.94it/s]
22%|       | 5807/26332 [00:13<00:46, 436.97it/s]
22%|       | 5861/26332 [00:13<00:44, 459.30it/s]
```

```
22%|       | 5908/26332 [00:13<00:45, 453.70it/s]
23%|       | 5954/26332 [00:13<00:47, 426.62it/s]
23%|       | 5998/26332 [00:13<00:48, 415.10it/s]
23%|       | 6041/26332 [00:13<00:50, 400.47it/s]
23%|       | 6085/26332 [00:13<00:49, 409.07it/s]
23%|       | 6127/26332 [00:13<00:49, 410.24it/s]
23%|       | 6176/26332 [00:13<00:46, 429.77it/s]
24%|       | 6220/26332 [00:14<00:48, 417.72it/s]
24%|       | 6263/26332 [00:14<00:48, 409.65it/s]
24%|       | 6305/26332 [00:14<00:52, 380.82it/s]
24%|       | 6344/26332 [00:14<00:52, 382.47it/s]
24%|       | 6391/26332 [00:14<00:49, 404.96it/s]
24%|       | 6433/26332 [00:14<00:55, 357.18it/s]
25%|       | 6486/26332 [00:14<00:50, 395.43it/s]
25%|       | 6528/26332 [00:14<00:49, 398.61it/s]
25%|       | 6570/26332 [00:15<00:52, 376.20it/s]
25%|       | 6625/26332 [00:15<00:47, 414.99it/s]
25%|       | 6683/26332 [00:15<00:43, 452.65it/s]
26%|       | 6734/26332 [00:15<00:42, 465.84it/s]
26%|       | 6789/26332 [00:15<00:40, 480.41it/s]
26%|       | 6841/26332 [00:15<00:39, 489.32it/s]
26%|       | 6891/26332 [00:15<00:45, 425.88it/s]
26%|       | 6936/26332 [00:15<00:45, 427.29it/s]
27%|       | 6981/26332 [00:15<00:51, 376.55it/s]
```

```
27%|       | 7035/26332 [00:16<00:46, 414.01it/s]
27%|       | 7085/26332 [00:16<00:44, 435.92it/s]
27%|       | 7131/26332 [00:16<00:45, 425.08it/s]
27%|       | 7179/26332 [00:16<00:43, 439.75it/s]
27%|       | 7225/26332 [00:16<00:42, 445.06it/s]
28%|       | 7271/26332 [00:16<00:43, 440.37it/s]
28%|       | 7317/26332 [00:16<00:42, 445.43it/s]
28%|       | 7362/26332 [00:16<00:49, 381.84it/s]
28%|       | 7419/26332 [00:16<00:44, 422.95it/s]
28%|       | 7465/26332 [00:17<00:47, 395.82it/s]
29%|       | 7507/26332 [00:17<00:48, 388.69it/s]
29%|       | 7559/26332 [00:17<00:44, 420.44it/s]
29%|       | 7603/26332 [00:17<00:45, 415.45it/s]
29%|       | 7651/26332 [00:17<00:43, 432.79it/s]
29%|       | 7696/26332 [00:17<00:46, 399.92it/s]
29%|       | 7738/26332 [00:17<00:46, 396.85it/s]
30%|       | 7779/26332 [00:17<00:46, 396.47it/s]
30%|       | 7820/26332 [00:17<00:46, 398.80it/s]
30%|       | 7861/26332 [00:18<00:47, 392.36it/s]
30%|       | 7901/26332 [00:18<00:49, 375.94it/s]
30%|       | 7946/26332 [00:18<00:47, 387.06it/s]
30%|       | 7988/26332 [00:18<00:46, 396.28it/s]
31%|       | 8046/26332 [00:18<00:42, 431.98it/s]
31%|       | 8091/26332 [00:18<00:51, 357.42it/s]
```

```
31%|       | 8130/26332 [00:18<00:49, 364.35it/s]
31%|       | 8182/26332 [00:18<00:45, 400.17it/s]
31%|      | 8235/26332 [00:18<00:42, 426.85it/s]
31%|      | 8280/26332 [00:19<00:42, 424.28it/s]
32%|      | 8337/26332 [00:19<00:39, 457.87it/s]
32%|      | 8385/26332 [00:19<00:40, 438.71it/s]
32%|      | 8431/26332 [00:19<00:40, 441.57it/s]
32%|      | 8477/26332 [00:19<00:43, 413.41it/s]
32%|      | 8520/26332 [00:19<00:43, 408.25it/s]
33%|      | 8563/26332 [00:19<00:43, 412.65it/s]
33%|      | 8611/26332 [00:19<00:41, 422.45it/s]
33%|      | 8654/26332 [00:19<00:42, 412.45it/s]
33%|      | 8701/26332 [00:20<00:41, 427.70it/s]
33%|      | 8749/26332 [00:20<00:40, 439.21it/s]
33%|      | 8796/26332 [00:20<00:39, 443.28it/s]
34%|      | 8842/26332 [00:20<00:39, 441.38it/s]
34%|      | 8890/26332 [00:20<00:38, 452.06it/s]
34%|      | 8936/26332 [00:20<00:39, 441.69it/s]
34%|      | 8981/26332 [00:20<00:40, 425.76it/s]
34%|      | 9024/26332 [00:20<00:40, 422.38it/s]
34%|      | 9070/26332 [00:20<00:39, 432.76it/s]
35%|      | 9117/26332 [00:20<00:38, 442.96it/s]
35%|      | 9165/26332 [00:21<00:38, 449.16it/s]
35%|      | 9211/26332 [00:21<00:38, 443.63it/s]
```

```
35%|      | 9267/26332 [00:21<00:36, 472.99it/s]
35%|      | 9320/26332 [00:21<00:34, 488.34it/s]
36%|      | 9371/26332 [00:21<00:34, 493.24it/s]
36%|      | 9421/26332 [00:21<00:37, 445.04it/s]
36%|      | 9472/26332 [00:21<00:37, 452.05it/s]
36%|      | 9522/26332 [00:21<00:36, 465.20it/s]
36%|      | 9571/26332 [00:21<00:35, 468.05it/s]
37%|      | 9632/26332 [00:22<00:33, 502.35it/s]
37%|      | 9684/26332 [00:22<00:35, 474.50it/s]
37%|      | 9733/26332 [00:22<00:38, 436.06it/s]
37%|      | 9787/26332 [00:22<00:36, 458.48it/s]
37%|      | 9840/26332 [00:22<00:34, 477.39it/s]
38%|      | 9893/26332 [00:22<00:33, 491.93it/s]
38%|      | 9944/26332 [00:22<00:33, 494.74it/s]
38%|      | 9995/26332 [00:22<00:34, 479.14it/s]
38%|      | 10044/26332 [00:22<00:34, 468.22it/s]
38%|      | 10097/26332 [00:23<00:33, 484.25it/s]
39%|      | 10146/26332 [00:23<00:34, 469.05it/s]
39%|      | 10194/26332 [00:23<00:36, 445.62it/s]
39%|      | 10240/26332 [00:23<00:38, 418.90it/s]
39%|      | 10283/26332 [00:23<00:40, 396.51it/s]
39%|      | 10324/26332 [00:23<00:40, 392.53it/s]
39%|      | 10383/26332 [00:23<00:36, 434.75it/s]
40%|      | 10433/26332 [00:23<00:35, 449.17it/s]
```

```
40%|      | 10480/26332 [00:23<00:36, 428.60it/s]
40%|      | 10525/26332 [00:24<00:39, 402.64it/s]
40%|      | 10567/26332 [00:24<00:41, 380.72it/s]
40%|      | 10611/26332 [00:24<00:39, 395.42it/s]
40%|      | 10652/26332 [00:24<00:39, 394.35it/s]
41%|      | 10700/26332 [00:24<00:38, 411.26it/s]
41%|      | 10755/26332 [00:24<00:35, 444.43it/s]
41%|      | 10804/26332 [00:24<00:34, 452.00it/s]
41%|      | 10851/26332 [00:24<00:36, 420.16it/s]
41%|     | 10895/26332 [00:24<00:38, 405.77it/s]
42%|     | 10939/26332 [00:25<00:37, 414.72it/s]
42%|     | 10982/26332 [00:25<00:41, 367.22it/s]
42%|     | 11021/26332 [00:25<00:41, 372.54it/s]
42%|     | 11060/26332 [00:25<00:41, 366.98it/s]
42%|     | 11117/26332 [00:25<00:37, 402.58it/s]
42%|     | 11165/26332 [00:25<00:35, 422.54it/s]
43%|     | 11209/26332 [00:25<00:35, 423.90it/s]
43%|     | 11253/26332 [00:25<00:36, 411.66it/s]
43%|     | 11302/26332 [00:25<00:34, 432.10it/s]
43%|     | 11346/26332 [00:26<00:36, 409.66it/s]
43%|     | 11399/26332 [00:26<00:34, 437.83it/s]
43%|     | 11447/26332 [00:26<00:33, 440.33it/s]
44%|     | 11493/26332 [00:26<00:33, 440.16it/s]
44%|     | 11538/26332 [00:26<00:33, 439.28it/s]
```

```
44%|     | 11583/26332 [00:26<00:33, 438.00it/s]
44%|     | 11639/26332 [00:26<00:31, 462.73it/s]
44%|     | 11686/26332 [00:26<00:33, 441.39it/s]
45%|     | 11731/26332 [00:26<00:33, 440.62it/s]
45%|     | 11776/26332 [00:27<00:34, 424.48it/s]
45%|     | 11819/26332 [00:27<00:34, 415.39it/s]
45%|     | 11865/26332 [00:27<00:33, 426.35it/s]
45%|     | 11913/26332 [00:27<00:32, 439.77it/s]
45%|     | 11967/26332 [00:27<00:31, 457.26it/s]
46%|     | 12014/26332 [00:27<00:31, 455.56it/s]
46%|     | 12060/26332 [00:27<00:34, 410.02it/s]
46%|     | 12104/26332 [00:27<00:33, 418.55it/s]
46%|     | 12147/26332 [00:27<00:39, 356.91it/s]
46%|     | 12209/26332 [00:28<00:34, 408.40it/s]
47%|     | 12254/26332 [00:28<00:35, 393.62it/s]
47%|     | 12302/26332 [00:28<00:33, 415.50it/s]
47%|     | 12346/26332 [00:28<00:34, 405.99it/s]
47%|     | 12389/26332 [00:28<00:35, 389.30it/s]
47%|     | 12435/26332 [00:28<00:34, 402.54it/s]
47%|     | 12477/26332 [00:28<00:35, 395.22it/s]
48%|     | 12533/26332 [00:28<00:32, 430.11it/s]
48%|     | 12578/26332 [00:28<00:32, 419.23it/s]
48%|     | 12621/26332 [00:29<00:34, 395.55it/s]
48%|     | 12662/26332 [00:29<00:35, 389.86it/s]
```

```
48%|     | 12714/26332 [00:29<00:32, 418.22it/s]
48%|     | 12757/26332 [00:29<00:34, 389.20it/s]
49%|     | 12804/26332 [00:29<00:32, 410.10it/s]
49%|     | 12853/26332 [00:29<00:31, 422.69it/s]
49%|     | 12901/26332 [00:29<00:30, 437.39it/s]
49%|     | 12946/26332 [00:29<00:30, 434.54it/s]
49%|     | 12990/26332 [00:29<00:32, 408.40it/s]
50%|     | 13039/26332 [00:30<00:30, 429.41it/s]
50%|     | 13089/26332 [00:30<00:29, 445.15it/s]
50%|     | 13135/26332 [00:30<00:29, 445.39it/s]
50%|     | 13206/26332 [00:30<00:26, 501.37it/s]
50%|     | 13259/26332 [00:30<00:28, 462.28it/s]
51%|     | 13308/26332 [00:30<00:29, 442.86it/s]
51%|     | 13355/26332 [00:30<00:29, 435.01it/s]
51%|     | 13400/26332 [00:30<00:29, 434.13it/s]
51%|     | 13445/26332 [00:30<00:29, 432.07it/s]
51%|     | 13493/26332 [00:31<00:29, 441.99it/s]
51%|    | 13546/26332 [00:31<00:27, 464.23it/s]
52%|    | 13594/26332 [00:31<00:28, 453.91it/s]
52%|    | 13640/26332 [00:31<00:28, 448.20it/s]
52%|    | 13688/26332 [00:31<00:27, 452.69it/s]
52%|    | 13734/26332 [00:31<00:28, 440.44it/s]
52%|    | 13779/26332 [00:31<00:30, 416.62it/s]
53%|    | 13837/26332 [00:31<00:27, 454.71it/s]
```

```
53%|    | 13885/26332 [00:31<00:27, 456.65it/s]
53%|    | 13932/26332 [00:31<00:27, 444.57it/s]
53%|    | 13979/26332 [00:32<00:27, 450.38it/s]
53%|    | 14025/26332 [00:32<00:27, 448.23it/s]
53%|    | 14079/26332 [00:32<00:26, 463.49it/s]
54%|    | 14126/26332 [00:32<00:26, 463.02it/s]
54%|    | 14173/26332 [00:32<00:28, 432.66it/s]
54%|    | 14217/26332 [00:32<00:29, 417.68it/s]
54%|    | 14260/26332 [00:32<00:29, 414.00it/s]
54%|    | 14303/26332 [00:32<00:28, 415.30it/s]
54%|    | 14345/26332 [00:32<00:29, 407.33it/s]
55%|    | 14386/26332 [00:33<00:30, 390.97it/s]
55%|    | 14436/26332 [00:33<00:28, 418.19it/s]
55%|    | 14486/26332 [00:33<00:27, 435.47it/s]
55%|    | 14539/26332 [00:33<00:25, 460.01it/s]
55%|    | 14586/26332 [00:33<00:26, 451.77it/s]
56%|    | 14632/26332 [00:33<00:25, 451.74it/s]
56%|    | 14680/26332 [00:33<00:25, 455.04it/s]
56%|    | 14726/26332 [00:33<00:27, 421.61it/s]
56%|    | 14769/26332 [00:33<00:27, 417.57it/s]
56%|    | 14815/26332 [00:34<00:26, 427.73it/s]
56%|    | 14859/26332 [00:34<00:27, 422.13it/s]
57%|    | 14902/26332 [00:34<00:29, 388.05it/s]
57%|    | 14949/26332 [00:34<00:27, 408.98it/s]
```

```
57%|    | 14991/26332 [00:34<00:28, 402.31it/s]
57%|    | 15032/26332 [00:34<00:28, 401.10it/s]
57%|    | 15073/26332 [00:34<00:28, 394.00it/s]
57%|    | 15115/26332 [00:34<00:27, 400.92it/s]
58%|    | 15156/26332 [00:34<00:28, 390.17it/s]
58%|    | 15204/26332 [00:35<00:27, 400.86it/s]
58%|    | 15251/26332 [00:35<00:27, 405.60it/s]
58%|    | 15292/26332 [00:35<00:28, 381.04it/s]
58%|    | 15334/26332 [00:35<00:28, 390.67it/s]
58%|    | 15387/26332 [00:35<00:26, 408.83it/s]
59%|    | 15431/26332 [00:35<00:26, 417.38it/s]
59%|    | 15474/26332 [00:35<00:28, 381.97it/s]
59%|    | 15517/26332 [00:35<00:27, 393.22it/s]
59%|    | 15558/26332 [00:35<00:27, 394.28it/s]
59%|    | 15598/26332 [00:36<00:29, 358.34it/s]
59%|    | 15642/26332 [00:36<00:28, 378.39it/s]
60%|    | 15689/26332 [00:36<00:26, 398.48it/s]
60%|    | 15730/26332 [00:36<00:28, 371.52it/s]
60%|    | 15803/26332 [00:36<00:24, 435.51it/s]
60%|    | 15852/26332 [00:36<00:23, 448.16it/s]
60%|    | 15901/26332 [00:36<00:23, 450.97it/s]
61%|    | 15949/26332 [00:36<00:23, 448.79it/s]
61%|    | 15996/26332 [00:36<00:24, 415.26it/s]
61%|    | 16040/26332 [00:37<00:24, 417.01it/s]
```

```
61%|    | 16083/26332 [00:37<00:24, 420.14it/s]
61%|    | 16134/26332 [00:37<00:23, 443.16it/s]
61%|    | 16180/26332 [00:37<00:23, 427.08it/s]
62%|    | 16230/26332 [00:37<00:23, 436.57it/s]
62%|    | 16275/26332 [00:37<00:23, 435.42it/s]
62%|    | 16319/26332 [00:37<00:24, 413.45it/s]
62%|    | 16361/26332 [00:37<00:24, 410.38it/s]
62%|    | 16418/26332 [00:37<00:22, 444.73it/s]
63%|    | 16464/26332 [00:37<00:22, 431.46it/s]
63%|    | 16508/26332 [00:38<00:23, 419.80it/s]
63%|    | 16551/26332 [00:38<00:23, 414.57it/s]
63%|    | 16606/26332 [00:38<00:21, 446.82it/s]
63%|    | 16652/26332 [00:38<00:22, 433.33it/s]
63%|    | 16697/26332 [00:38<00:22, 430.69it/s]
64%|    | 16741/26332 [00:38<00:22, 423.36it/s]
64%|    | 16784/26332 [00:38<00:24, 391.28it/s]
64%|    | 16831/26332 [00:38<00:23, 407.35it/s]
64%|    | 16873/26332 [00:38<00:23, 398.98it/s]
64%|    | 16917/26332 [00:39<00:23, 409.31it/s]
64%|    | 16959/26332 [00:39<00:22, 410.11it/s]
65%|    | 17001/26332 [00:39<00:22, 406.25it/s]
65%|    | 17042/26332 [00:39<00:26, 350.48it/s]
65%|    | 17098/26332 [00:39<00:23, 394.58it/s]
65%|    | 17144/26332 [00:39<00:22, 411.49it/s]
```

```
65%|   | 17190/26332 [00:39<00:21, 422.59it/s]
65%|   | 17234/26332 [00:39<00:22, 407.28it/s]
66%|   | 17276/26332 [00:39<00:23, 386.84it/s]
66%|   | 17337/26332 [00:40<00:20, 430.46it/s]
66%|   | 17383/26332 [00:40<00:20, 430.94it/s]
66%|   | 17428/26332 [00:40<00:21, 414.99it/s]
66%|   | 17497/26332 [00:40<00:18, 471.29it/s]
67%|   | 17548/26332 [00:40<00:18, 470.72it/s]
67%|   | 17598/26332 [00:40<00:19, 454.18it/s]
67%|   | 17646/26332 [00:40<00:19, 452.11it/s]
67%|   | 17693/26332 [00:40<00:22, 390.75it/s]
67%|   | 17745/26332 [00:41<00:20, 422.20it/s]
68%|   | 17792/26332 [00:41<00:19, 435.16it/s]
68%|   | 17838/26332 [00:41<00:19, 435.19it/s]
68%|   | 17896/26332 [00:41<00:17, 469.67it/s]
68%|   | 17948/26332 [00:41<00:17, 478.59it/s]
68%|   | 17998/26332 [00:41<00:18, 451.68it/s]
69%|   | 18045/26332 [00:41<00:19, 424.55it/s]
69%|   | 18089/26332 [00:41<00:19, 428.21it/s]
69%|   | 18135/26332 [00:41<00:18, 434.49it/s]
69%|   | 18185/26332 [00:41<00:18, 449.97it/s]
69%|   | 18233/26332 [00:42<00:17, 451.80it/s]
69%|   | 18279/26332 [00:42<00:17, 451.55it/s]
70%|   | 18336/26332 [00:42<00:16, 481.46it/s]
```

```
70%|   | 18395/26332 [00:42<00:15, 504.13it/s]
70%|   | 18447/26332 [00:42<00:16, 479.22it/s]
70%|   | 18496/26332 [00:42<00:16, 467.33it/s]
70%|   | 18558/26332 [00:42<00:15, 502.86it/s]
71%|   | 18610/26332 [00:42<00:16, 454.95it/s]
71%|   | 18661/26332 [00:42<00:16, 469.79it/s]
71%|   | 18710/26332 [00:43<00:17, 426.94it/s]
71%|   | 18758/26332 [00:43<00:17, 440.30it/s]
71%|  | 18809/26332 [00:43<00:16, 443.62it/s]
72%|  | 18857/26332 [00:43<00:16, 452.99it/s]
72%|  | 18903/26332 [00:43<00:16, 438.76it/s]
72%|  | 18948/26332 [00:43<00:18, 409.30it/s]
72%|  | 18992/26332 [00:43<00:17, 417.64it/s]
72%|  | 19051/26332 [00:43<00:15, 455.99it/s]
73%|  | 19099/26332 [00:43<00:16, 436.08it/s]
73%|  | 19144/26332 [00:44<00:16, 432.67it/s]
73%|  | 19197/26332 [00:44<00:15, 455.78it/s]
73%|  | 19244/26332 [00:44<00:15, 459.44it/s]
73%|  | 19291/26332 [00:44<00:15, 457.72it/s]
73%|  | 19338/26332 [00:44<00:15, 456.65it/s]
74%|  | 19384/26332 [00:44<00:17, 396.71it/s]
74%|  | 19437/26332 [00:44<00:16, 428.70it/s]
74%|  | 19482/26332 [00:44<00:16, 426.37it/s]
74%|  | 19527/26332 [00:44<00:15, 429.60it/s]
```

```
74%|  | 19571/26332 [00:45<00:15, 428.05it/s]
75%|  | 19623/26332 [00:45<00:14, 447.46it/s]
75%|  | 19676/26332 [00:45<00:14, 467.49it/s]
75%|  | 19738/26332 [00:45<00:13, 494.95it/s]
75%|  | 19789/26332 [00:45<00:14, 466.67it/s]
75%|  | 19837/26332 [00:45<00:13, 464.96it/s]
76%|  | 19885/26332 [00:45<00:14, 453.50it/s]
76%|  | 19931/26332 [00:45<00:14, 443.03it/s]
76%|  | 19983/26332 [00:45<00:13, 463.11it/s]
76%|  | 20030/26332 [00:46<00:13, 461.91it/s]
76%|  | 20077/26332 [00:46<00:14, 436.67it/s]
76%|  | 20132/26332 [00:46<00:13, 446.84it/s]
77%|  | 20179/26332 [00:46<00:13, 452.97it/s]
77%|  | 20225/26332 [00:46<00:14, 416.14it/s]
77%|  | 20283/26332 [00:46<00:13, 452.11it/s]
77%|  | 20339/26332 [00:46<00:12, 475.10it/s]
77%|  | 20389/26332 [00:46<00:12, 480.37it/s]
78%|  | 20439/26332 [00:46<00:12, 485.17it/s]
78%|  | 20489/26332 [00:47<00:12, 470.11it/s]
78%|  | 20537/26332 [00:47<00:12, 459.63it/s]
78%|  | 20590/26332 [00:47<00:12, 477.74it/s]
78%|  | 20639/26332 [00:47<00:13, 430.69it/s]
79%|  | 20684/26332 [00:47<00:13, 404.23it/s]
79%|  | 20726/26332 [00:47<00:14, 380.13it/s]
```

```
79%|  | 20773/26332 [00:47<00:13, 397.85it/s]
79%|  | 20814/26332 [00:47<00:14, 388.03it/s]
79%|  | 20854/26332 [00:47<00:14, 381.25it/s]
79%|  | 20899/26332 [00:48<00:13, 399.07it/s]
80%|  | 20940/26332 [00:48<00:14, 371.79it/s]
80%|  | 20981/26332 [00:48<00:14, 381.37it/s]
80%|  | 21026/26332 [00:48<00:13, 392.03it/s]
80%|  | 21066/26332 [00:48<00:14, 375.95it/s]
80%|  | 21106/26332 [00:48<00:13, 382.15it/s]
80%|  | 21145/26332 [00:48<00:13, 378.23it/s]
80%|  | 21184/26332 [00:48<00:13, 368.53it/s]
81%|  | 21231/26332 [00:48<00:12, 393.24it/s]
81%|  | 21271/26332 [00:49<00:13, 381.84it/s]
81%|  | 21310/26332 [00:49<00:13, 382.82it/s]
81%|  | 21352/26332 [00:49<00:12, 385.95it/s]
81%|  | 21391/26332 [00:49<00:13, 372.14it/s]
81%| | 21430/26332 [00:49<00:12, 377.24it/s]
82%| | 21468/26332 [00:49<00:13, 361.17it/s]
82%| | 21506/26332 [00:49<00:13, 366.24it/s]
82%| | 21543/26332 [00:49<00:13, 354.27it/s]
82%| | 21579/26332 [00:49<00:14, 329.36it/s]
82%| | 21625/26332 [00:50<00:13, 359.83it/s]
82%| | 21673/26332 [00:50<00:12, 386.64it/s]
82%| | 21714/26332 [00:50<00:11, 386.60it/s]
```

```
83%| | 21754/26332 [00:50<00:11, 385.67it/s]
83%| | 21794/26332 [00:50<00:11, 381.88it/s]
83%| | 21833/26332 [00:50<00:11, 380.47it/s]
83%| | 21879/26332 [00:50<00:11, 399.64it/s]
83%| | 21920/26332 [00:50<00:12, 355.61it/s]
83%| | 21966/26332 [00:50<00:11, 381.23it/s]
84%| | 22006/26332 [00:50<00:11, 369.14it/s]
84%| | 22051/26332 [00:51<00:11, 388.66it/s]
84%| | 22091/26332 [00:51<00:11, 362.23it/s]
84%| | 22139/26332 [00:51<00:10, 384.09it/s]
84%| | 22179/26332 [00:51<00:12, 329.06it/s]
84%| | 22215/26332 [00:51<00:12, 329.49it/s]
85%| | 22263/26332 [00:51<00:11, 360.30it/s]
85%| | 22302/26332 [00:51<00:11, 365.05it/s]
85%| | 22340/26332 [00:51<00:10, 369.38it/s]
85%| | 22379/26332 [00:52<00:10, 375.02it/s]
85%| | 22429/26332 [00:52<00:09, 405.18it/s]
85%| | 22471/26332 [00:52<00:10, 376.58it/s]
86%| | 22523/26332 [00:52<00:09, 409.66it/s]
86%| | 22566/26332 [00:52<00:09, 414.57it/s]
86%| | 22609/26332 [00:52<00:09, 396.39it/s]
86%| | 22654/26332 [00:52<00:08, 409.59it/s]
86%| | 22696/26332 [00:52<00:09, 381.53it/s]
86%| | 22736/26332 [00:52<00:09, 374.30it/s]
```

87%| | 22785/26332 [00:53<00:08, 402.17it/s]

87%| | 22827/26332 [00:53<00:08, 397.25it/s]

87%| | 22868/26332 [00:53<00:08, 388.27it/s]

87%| | 22908/26332 [00:53<00:08, 383.60it/s]

87%| | 22949/26332 [00:53<00:08, 388.51it/s]

87%| | 22990/26332 [00:53<00:08, 391.51it/s]

87%| | 23030/26332 [00:53<00:09, 356.94it/s]

88%| | 23072/26332 [00:53<00:08, 369.52it/s]

88%| | 23110/26332 [00:53<00:08, 360.43it/s]

88%| | 23157/26332 [00:53<00:08, 381.91it/s]

88%| | 23196/26332 [00:54<00:08, 364.91it/s]

88%| | 23234/26332 [00:54<00:09, 337.38it/s]

88%| | 23269/26332 [00:54<00:08, 340.93it/s]

89%| | 23313/26332 [00:54<00:08, 360.52it/s]

89%| | 23350/26332 [00:54<00:08, 356.25it/s]

89%| | 23388/26332 [00:54<00:08, 360.63it/s]

89%| | 23431/26332 [00:54<00:07, 378.57it/s]

89%| | 23477/26332 [00:54<00:07, 399.65it/s]

89%| | 23527/26332 [00:54<00:06, 419.25it/s]

90%| | 23570/26332 [00:55<00:06, 396.62it/s]

90%| | 23611/26332 [00:55<00:07, 351.60it/s]

90%| | 23657/26332 [00:55<00:07, 377.55it/s]

90%| | 23697/26332 [00:55<00:07, 368.68it/s]

90%| | 23736/26332 [00:55<00:07, 353.52it/s]

```
90%| | 23782/26332 [00:55<00:06, 378.99it/s]
90%| | 23822/26332 [00:55<00:06, 375.82it/s]
91%| | 23871/26332 [00:55<00:06, 402.03it/s]
91%| | 23917/26332 [00:55<00:05, 417.11it/s]
91%| | 23960/26332 [00:56<00:05, 406.31it/s]
91%| | 24002/26332 [00:56<00:05, 400.10it/s]
91%|| 24044/26332 [00:56<00:05, 404.82it/s]
91%|| 24085/26332 [00:56<00:05, 384.76it/s]
92%|| 24124/26332 [00:56<00:05, 383.55it/s]
92%|| 24163/26332 [00:56<00:05, 381.06it/s]
92%|| 24202/26332 [00:56<00:05, 375.12it/s]
92%|| 24240/26332 [00:56<00:05, 362.02it/s]
92%|| 24277/26332 [00:56<00:06, 332.36it/s]
92%|| 24322/26332 [00:57<00:05, 359.51it/s]
93%|| 24368/26332 [00:57<00:05, 376.81it/s]
93%|| 24417/26332 [00:57<00:04, 403.21it/s]
93%|| 24466/26332 [00:57<00:04, 422.83it/s]
93%|| 24511/26332 [00:57<00:04, 426.35it/s]
93%|| 24556/26332 [00:57<00:04, 431.22it/s]
93%|| 24606/26332 [00:57<00:03, 449.70it/s]
94%|| 24652/26332 [00:57<00:03, 451.22it/s]
94%|| 24698/26332 [00:57<00:03, 448.77it/s]
94%|| 24751/26332 [00:58<00:03, 466.83it/s]
94%|| 24799/26332 [00:58<00:03, 448.97it/s]
```

94%|| 24847/26332 [00:58<00:03, 454.05it/s]

95%|| 24894/26332 [00:58<00:03, 458.24it/s]

95%|| 24941/26332 [00:58<00:03, 407.11it/s]

95%|| 24983/26332 [00:58<00:03, 397.23it/s]

95%|| 25029/26332 [00:58<00:03, 411.13it/s]

95%|| 25078/26332 [00:58<00:02, 427.91it/s]

95%|| 25122/26332 [00:58<00:03, 397.11it/s]

96%|| 25163/26332 [00:59<00:03, 374.76it/s]

96%|| 25205/26332 [00:59<00:02, 387.13it/s]

96%|| 25269/26332 [00:59<00:02, 438.08it/s]

96%|| 25316/26332 [00:59<00:02, 427.61it/s]

96%|| 25361/26332 [00:59<00:02, 421.23it/s]

97%|| 25416/26332 [00:59<00:02, 452.56it/s]

97%|| 25463/26332 [00:59<00:02, 431.34it/s]

97%|| 25508/26332 [00:59<00:01, 429.18it/s]

97%|| 25552/26332 [00:59<00:01, 415.01it/s]

97%|| 25595/26332 [01:00<00:01, 389.92it/s]

97%|| 25635/26332 [01:00<00:01, 383.54it/s]

98%|| 25674/26332 [01:00<00:01, 373.39it/s]

98%|| 25712/26332 [01:00<00:01, 369.11it/s]

98%|| 25750/26332 [01:00<00:01, 360.88it/s]

98%|| 25807/26332 [01:00<00:01, 404.37it/s]

98%|| 25852/26332 [01:00<00:01, 411.06it/s]

98%|| 25896/26332 [01:00<00:01, 417.40it/s]

```
 99%|| 25949/26332 [01:00<00:00, 439.58it/s]

 99%|| 25999/26332 [01:00<00:00, 450.03it/s]

 99%|| 26050/26332 [01:01<00:00, 464.42it/s]

 99%|| 26098/26332 [01:01<00:00, 444.41it/s]

 99%|| 26156/26332 [01:01<00:00, 476.89it/s]

100%|| 26205/26332 [01:01<00:00, 470.24it/s]

100%|| 26253/26332 [01:01<00:00, 466.41it/s]

100%|| 26307/26332 [01:01<00:00, 481.66it/s]

100%|| 26332/26332 [01:01<00:00, 426.84it/s]
```

```
In [0]: X_train3 = train_sent
        X_test3 = test_sent
```

5.5 [4.4.1] Converting text into vectors using Avg W2V, TFIDF-W2V

5.6 [4.4.1.1] Avg W2v

5.6.1 Applying RandomForest Classifier

```
In [153]: from sklearn.ensemble import RandomForestClassifier
          from sklearn.model_selection import GridSearchCV


          grid_params = dict(n_estimators = [2,5,20,50,75,100],
                             max_depth= [2,4,5,8,10] )

          rf = RandomForestClassifier()
          rf_clf = GridSearchCV(rf, param_grid=grid_params, n_jobs=-1, verbose=30,return_train
          rf_clf = rf_clf.fit(X_train3, y_train)


          results = rf_clf.cv_results_
```

```
Fitting 3 folds for each of 30 candidates, totalling 90 fits


[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done   1 tasks      | elapsed:    2.9s
[Parallel(n_jobs=-1)]: Done   2 tasks      | elapsed:    3.9s
[Parallel(n_jobs=-1)]: Done   3 tasks      | elapsed:    4.4s
```

```
[Parallel(n_jobs=-1)]: Done   4 tasks      | elapsed:    5.7s
[Parallel(n_jobs=-1)]: Done   5 tasks      | elapsed:    6.9s
[Parallel(n_jobs=-1)]: Done   6 tasks      | elapsed:    8.2s
[Parallel(n_jobs=-1)]: Done   7 tasks      | elapsed:   10.9s
[Parallel(n_jobs=-1)]: Done   8 tasks      | elapsed:   12.9s
[Parallel(n_jobs=-1)]: Done   9 tasks      | elapsed:   15.0s
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:   18.9s
[Parallel(n_jobs=-1)]: Done  11 tasks      | elapsed:   21.2s
[Parallel(n_jobs=-1)]: Done  12 tasks      | elapsed:   24.5s
[Parallel(n_jobs=-1)]: Done  13 tasks      | elapsed:   28.8s
[Parallel(n_jobs=-1)]: Done  14 tasks      | elapsed:   32.3s
[Parallel(n_jobs=-1)]: Done  15 tasks      | elapsed:   36.4s
[Parallel(n_jobs=-1)]: Done  16 tasks      | elapsed:   41.9s
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:   46.3s
[Parallel(n_jobs=-1)]: Done  18 tasks      | elapsed:   47.9s
[Parallel(n_jobs=-1)]: Done  19 tasks      | elapsed:   49.9s
[Parallel(n_jobs=-1)]: Done  20 tasks      | elapsed:   51.5s
[Parallel(n_jobs=-1)]: Done  21 tasks      | elapsed:   51.7s
[Parallel(n_jobs=-1)]: Done  22 tasks      | elapsed:   53.5s
[Parallel(n_jobs=-1)]: Done  23 tasks      | elapsed:   54.6s
[Parallel(n_jobs=-1)]: Done  24 tasks      | elapsed:   56.1s
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done  27 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done  29 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done  30 tasks      | elapsed:  1.3min
[Parallel(n_jobs=-1)]: Done  31 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done  32 tasks      | elapsed:  1.6min
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:  1.7min
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:  1.8min
[Parallel(n_jobs=-1)]: Done  35 tasks      | elapsed:  1.9min
[Parallel(n_jobs=-1)]: Done  36 tasks      | elapsed:  2.0min
[Parallel(n_jobs=-1)]: Done  37 tasks      | elapsed:  2.0min
[Parallel(n_jobs=-1)]: Done  38 tasks      | elapsed:  2.0min
[Parallel(n_jobs=-1)]: Done  39 tasks      | elapsed:  2.1min
[Parallel(n_jobs=-1)]: Done  40 tasks      | elapsed:  2.1min
[Parallel(n_jobs=-1)]: Done  41 tasks      | elapsed:  2.1min
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  2.1min
[Parallel(n_jobs=-1)]: Done  43 tasks      | elapsed:  2.2min
[Parallel(n_jobs=-1)]: Done  44 tasks      | elapsed:  2.2min
[Parallel(n_jobs=-1)]: Done  45 tasks      | elapsed:  2.3min
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:  2.4min
[Parallel(n_jobs=-1)]: Done  47 tasks      | elapsed:  2.5min
[Parallel(n_jobs=-1)]: Done  48 tasks      | elapsed:  2.6min
[Parallel(n_jobs=-1)]: Done  49 tasks      | elapsed:  2.7min
[Parallel(n_jobs=-1)]: Done  50 tasks      | elapsed:  2.8min
[Parallel(n_jobs=-1)]: Done  51 tasks      | elapsed:  3.0min
```

```
[Parallel(n_jobs=-1)]: Done  52 tasks      | elapsed:  3.2min
[Parallel(n_jobs=-1)]: Done  53 tasks      | elapsed:  3.3min
[Parallel(n_jobs=-1)]: Done  54 tasks      | elapsed:  3.3min
[Parallel(n_jobs=-1)]: Done  55 tasks      | elapsed:  3.3min
[Parallel(n_jobs=-1)]: Done  56 tasks      | elapsed:  3.4min
[Parallel(n_jobs=-1)]: Done  57 tasks      | elapsed:  3.4min
[Parallel(n_jobs=-1)]: Done  58 tasks      | elapsed:  3.5min
[Parallel(n_jobs=-1)]: Done  59 tasks      | elapsed:  3.5min
[Parallel(n_jobs=-1)]: Done  60 tasks      | elapsed:  3.5min
[Parallel(n_jobs=-1)]: Done  61 tasks      | elapsed:  3.6min
[Parallel(n_jobs=-1)]: Done  62 tasks      | elapsed:  3.6min
[Parallel(n_jobs=-1)]: Done  63 tasks      | elapsed:  3.7min
[Parallel(n_jobs=-1)]: Done  64 tasks      | elapsed:  3.9min
[Parallel(n_jobs=-1)]: Done  65 tasks      | elapsed:  4.0min
[Parallel(n_jobs=-1)]: Done  66 tasks      | elapsed:  4.1min
[Parallel(n_jobs=-1)]: Done  67 tasks      | elapsed:  4.3min
[Parallel(n_jobs=-1)]: Done  68 tasks      | elapsed:  4.5min
[Parallel(n_jobs=-1)]: Done  69 tasks      | elapsed:  4.7min
[Parallel(n_jobs=-1)]: Done  70 tasks      | elapsed:  4.9min
[Parallel(n_jobs=-1)]: Done  71 tasks      | elapsed:  5.1min
[Parallel(n_jobs=-1)]: Done  72 tasks      | elapsed:  5.2min
[Parallel(n_jobs=-1)]: Done  73 tasks      | elapsed:  5.2min
[Parallel(n_jobs=-1)]: Done  74 tasks      | elapsed:  5.2min
[Parallel(n_jobs=-1)]: Done  75 tasks      | elapsed:  5.3min
[Parallel(n_jobs=-1)]: Done  76 tasks      | elapsed:  5.4min
[Parallel(n_jobs=-1)]: Done  77 tasks      | elapsed:  5.4min
[Parallel(n_jobs=-1)]: Done  78 tasks      | elapsed:  5.4min
[Parallel(n_jobs=-1)]: Done  79 tasks      | elapsed:  5.5min
[Parallel(n_jobs=-1)]: Done  80 tasks      | elapsed:  5.6min
[Parallel(n_jobs=-1)]: Done  81 tasks      | elapsed:  5.7min
[Parallel(n_jobs=-1)]: Done  82 tasks      | elapsed:  5.8min
[Parallel(n_jobs=-1)]: Done  83 tasks      | elapsed:  5.9min
[Parallel(n_jobs=-1)]: Done  84 tasks      | elapsed:  6.1min
[Parallel(n_jobs=-1)]: Done  85 tasks      | elapsed:  6.4min
[Parallel(n_jobs=-1)]: Done  86 tasks      | elapsed:  6.6min
[Parallel(n_jobs=-1)]: Done  87 tasks      | elapsed:  6.8min
[Parallel(n_jobs=-1)]: Done  90 out of  90 | elapsed:  7.4min finished
```

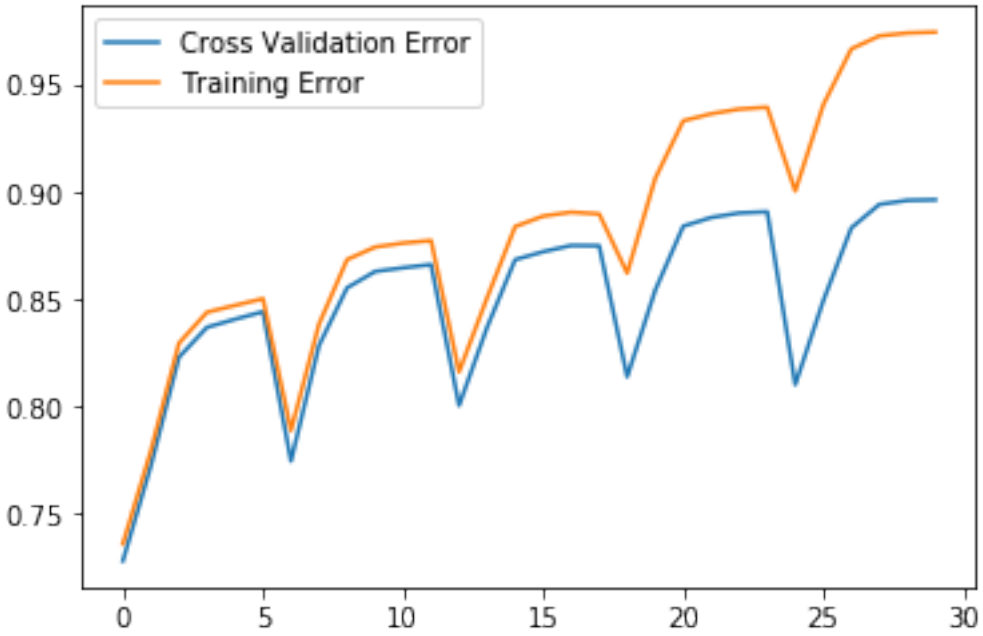
```
In [154]: print("Best parameters: ", rf_clf.best_params_)
          print("Best cross-validation score: {:.3f}".format(rf_clf.best_score_))

Best parameters:  {'max_depth': 10, 'n_estimators': 100}
Best cross-validation score: 0.897
```

```
In [0]: pred = rf_clf.predict(X_test3)
        pred_train = rf_clf.predict(X_train3)
```

```
In [0]: from sklearn.metrics import accuracy_score
        from sklearn.metrics import roc_curve
        import sklearn.metrics as metrics
        from sklearn.metrics import roc_auc_score

        train_error=rf_clf.cv_results_['mean_train_score']
        cv_error = rf_clf.cv_results_['mean_test_score']
        score=roc_auc_score(y_train, pred_train)

        estimator=rf_clf.best_params_['n_estimators']
        depth=rf_clf.best_params_['max_depth']
```

Error plots

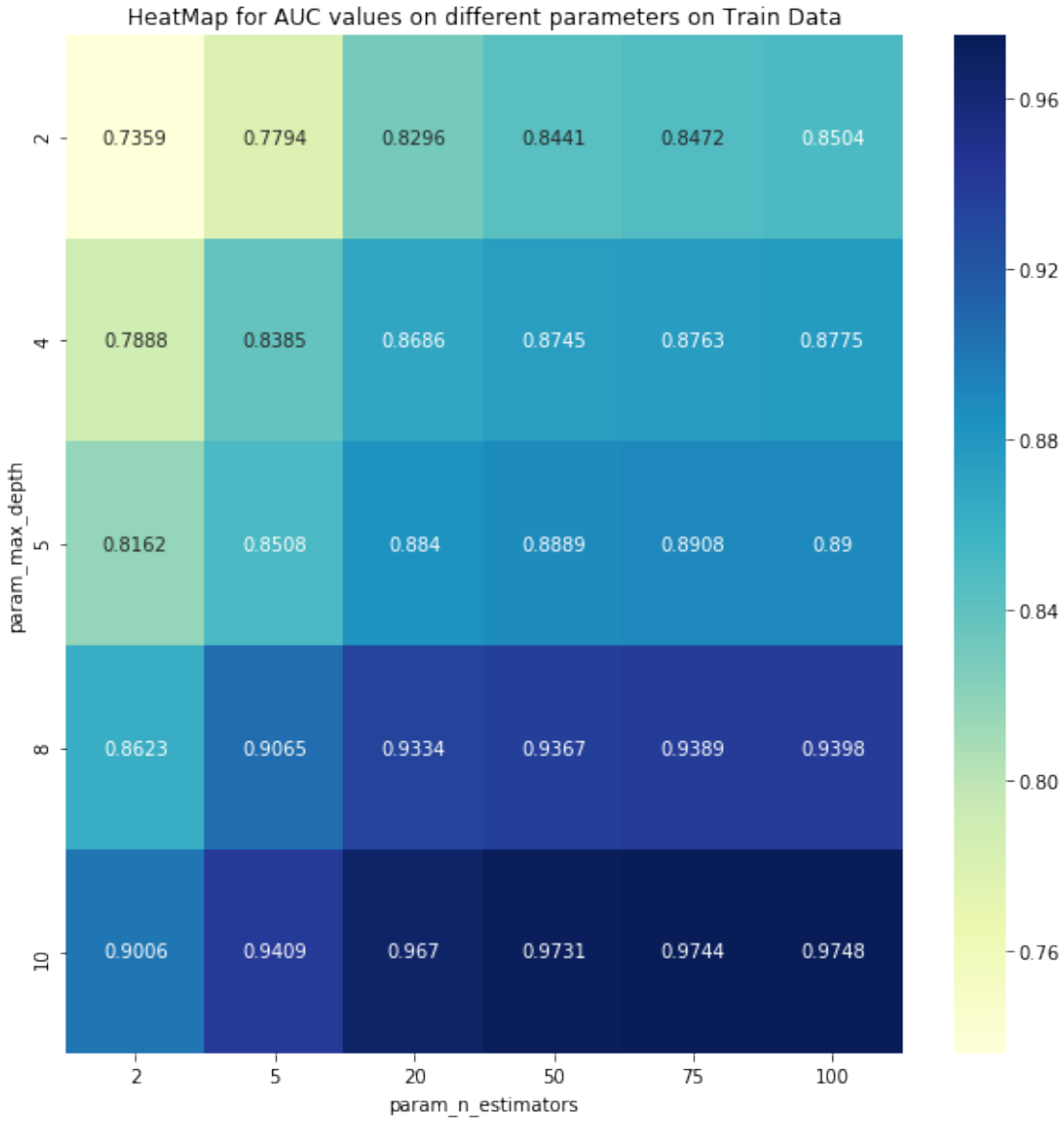
```
In [157]: plt.plot(cv_error, label='Cross Validation Error')
          plt.plot(train_error, label='Training Error')
          plt.legend()
          plt.show()
```



5.6.2 HeatMap for AUC vs N_estimators and Max_Depth

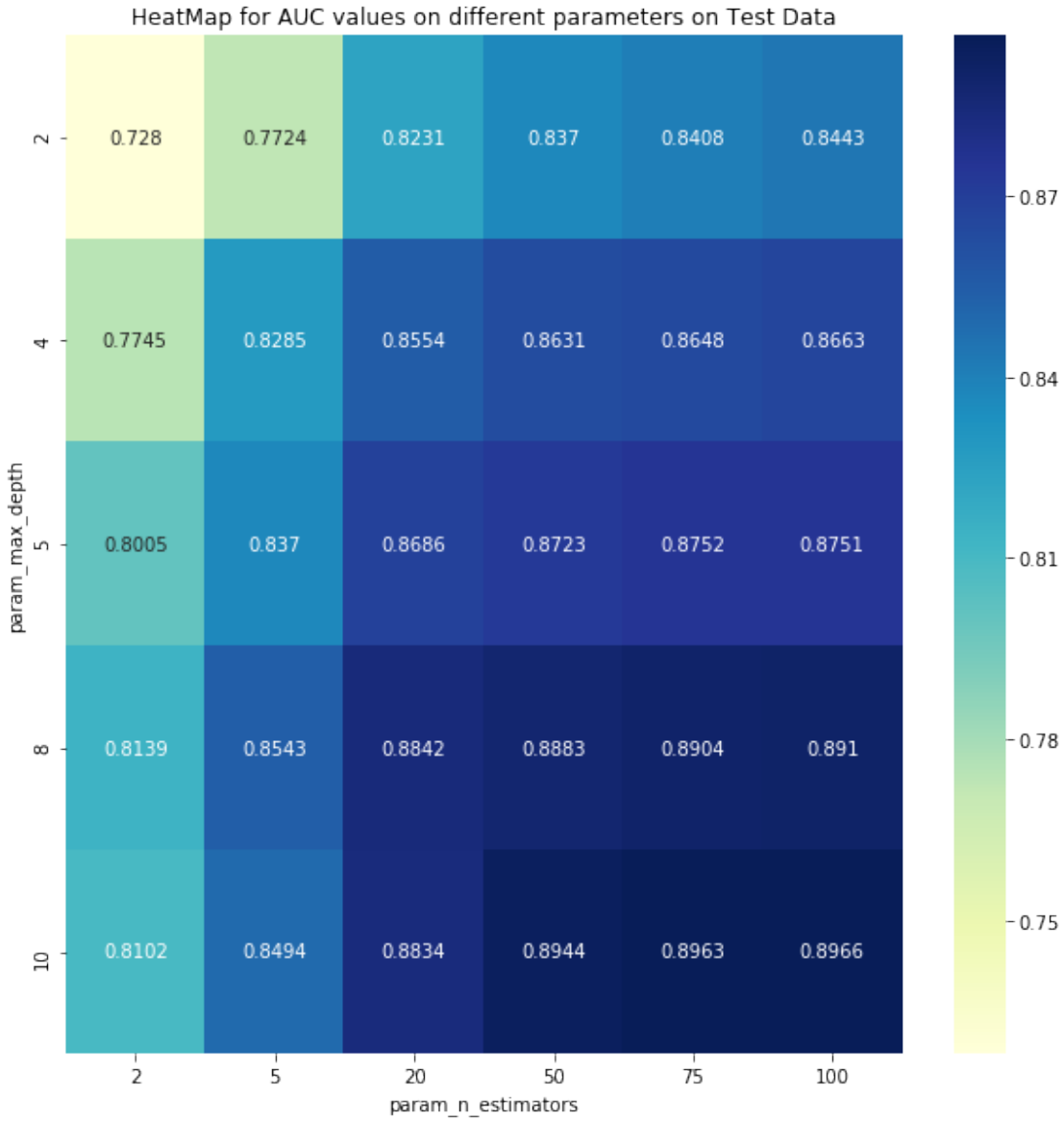
```
In [159]: df_new = pd.DataFrame(rf_clf.cv_results_)
          max_params = df_new.groupby(['param_max_depth','param_n_estimators']).max()
          max_params = max_params.unstack()[['mean_test_score', 'mean_train_score']]

          fig, ax = plt.subplots(figsize=(10,10))         # Sample figsize in inches
```

```
sns.heatmap(max_params.mean_train_score, annot=True,cmap="YlGnBu", fmt='.4g',ax=ax)
plt.title('HeatMap for AUC values on different parameters on Train Data')
plt.show()
```



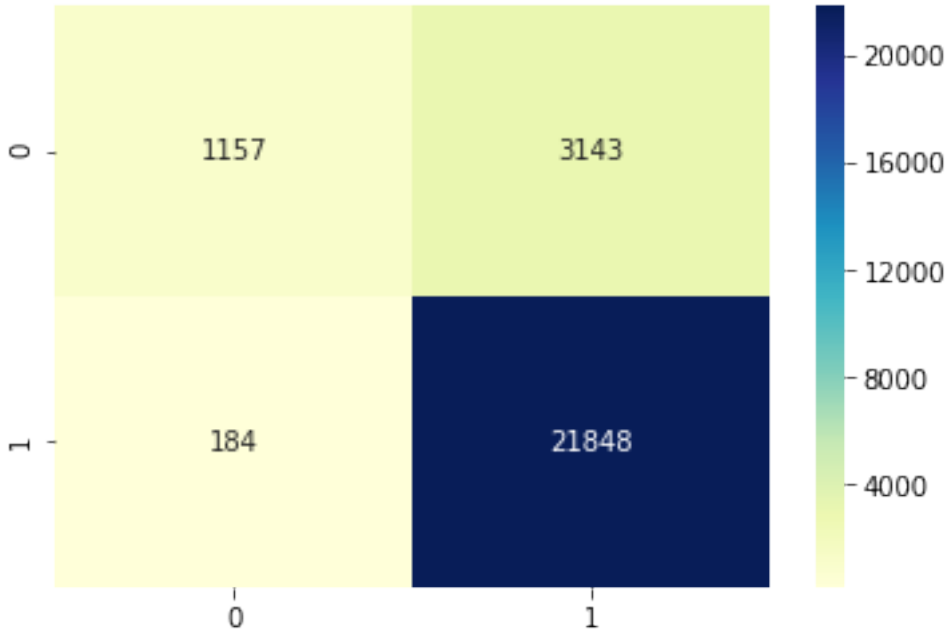
In [160]:
```
df_new = pd.DataFrame(rf_clf.cv_results_)
max_params = df_new.groupby(['param_max_depth','param_n_estimators']).max()
max_params = max_params.unstack()[['mean_test_score', 'mean_train_score']]

fig, ax = plt.subplots(figsize=(10,10))          # Sample figsize in inches

sns.heatmap(max_params.mean_test_score, annot=True,cmap="YlGnBu", fmt='.4g',ax=ax)
```

```
plt.title('HeatMap for AUC values on different parameters on Test Data')
plt.show()
```



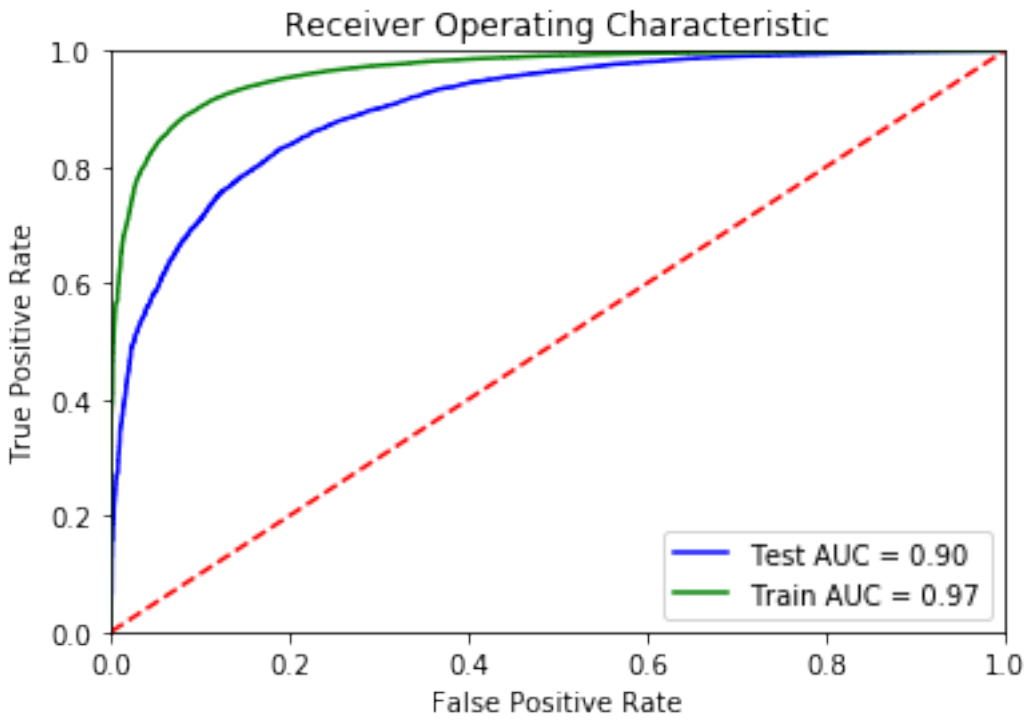

```
In [0]: pred_test = rf_clf.predict_proba(X_test3)[:,1]
        pred_train =  rf_clf.predict_proba(X_train3)[:,1]
```

```
In [162]: from sklearn.metrics import confusion_matrix

          cm = confusion_matrix(y_test, pred_test.round())
          sns.heatmap(cm, annot=True, fmt="d",cmap="YlGnBu")
          plt.show()
```




```
In [0]: fpr, tpr, thresholds = roc_curve(y_test, pred_test)
        fpr2, tpr2, thresholds = roc_curve(y_train, pred_train)

        score_test = roc_auc_score(y_test, pred_test)
        score_train = roc_auc_score(y_train, pred_train)
```

```
In [164]: roc_auc_test = metrics.auc(fpr, tpr)
          roc_auc_train = metrics.auc(fpr2, tpr2)
          plt.title('Receiver Operating Characteristic')
          plt.plot(fpr, tpr, 'b', label='Test AUC = %0.2f' % score_test)
          plt.plot(fpr2, tpr2, 'g', label = 'Train AUC = %0.2f' % score_train)
          plt.legend(loc = 'lower right')
          plt.plot([0, 1], [0, 1],'r--')
          plt.xlim([0, 1])
          plt.ylim([0, 1])
          plt.ylabel('True Positive Rate')
          plt.xlabel('False Positive Rate')
          plt.legend()
          plt.show()
```



5.6.3 Applying XGBOOST Classifier

```
In [166]: from sklearn.model_selection import GridSearchCV

          grid_params = {'n_estimators': [5, 10, 15,20],
                         'max_depth': [2, 5, 15,25,50]}

          xg_clf = xgb.XGBClassifier()
          xg_clf = GridSearchCV(xg_clf, param_grid=grid_params, n_jobs=-1, verbose=30,return_t
          xg_clf = xg_clf.fit(np.array(X_train3), y_train)

          results =xg_clf.cv_results_
```

```
Fitting 3 folds for each of 20 candidates, totalling 60 fits


[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done   1 tasks      | elapsed:    2.8s
[Parallel(n_jobs=-1)]: Done   2 tasks      | elapsed:    2.8s
[Parallel(n_jobs=-1)]: Done   3 tasks      | elapsed:    3.9s
[Parallel(n_jobs=-1)]: Done   4 tasks      | elapsed:    4.7s
[Parallel(n_jobs=-1)]: Done   5 tasks      | elapsed:    5.8s
[Parallel(n_jobs=-1)]: Done   6 tasks      | elapsed:    6.5s
[Parallel(n_jobs=-1)]: Done   7 tasks      | elapsed:    8.3s
```

```
[Parallel(n_jobs=-1)]: Done   8 tasks      | elapsed:    9.0s
[Parallel(n_jobs=-1)]: Done   9 tasks      | elapsed:   10.8s
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:   12.2s
[Parallel(n_jobs=-1)]: Done  11 tasks      | elapsed:   14.0s
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py:700: UserWarni
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done  12 tasks      | elapsed:   15.4s
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py:700: UserWarni
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done  13 tasks      | elapsed:   18.0s
[Parallel(n_jobs=-1)]: Done  14 tasks      | elapsed:   19.4s
[Parallel(n_jobs=-1)]: Done  15 tasks      | elapsed:   20.2s
[Parallel(n_jobs=-1)]: Done  16 tasks      | elapsed:   23.3s
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:   23.9s
[Parallel(n_jobs=-1)]: Done  18 tasks      | elapsed:   27.0s
[Parallel(n_jobs=-1)]: Done  19 tasks      | elapsed:   29.3s
[Parallel(n_jobs=-1)]: Done  20 tasks      | elapsed:   32.5s
[Parallel(n_jobs=-1)]: Done  21 tasks      | elapsed:   34.8s
[Parallel(n_jobs=-1)]: Done  22 tasks      | elapsed:   39.6s
[Parallel(n_jobs=-1)]: Done  23 tasks      | elapsed:   41.8s
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py:700: UserWarni
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done  24 tasks      | elapsed:   46.8s
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py:700: UserWarni
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:   50.0s
[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed:   55.6s
[Parallel(n_jobs=-1)]: Done  27 tasks      | elapsed:   57.4s
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done  29 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done  30 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done  31 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done  32 tasks      | elapsed:  1.7min
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:  1.8min
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:  2.1min
[Parallel(n_jobs=-1)]: Done  35 tasks      | elapsed:  2.2min
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py:700: UserWarni
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done  36 tasks      | elapsed:  2.4min
[Parallel(n_jobs=-1)]: Done  37 tasks      | elapsed:  2.5min
[Parallel(n_jobs=-1)]: Done  38 tasks      | elapsed:  2.5min
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py:700: UserWarni
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done  39 tasks      | elapsed:  2.6min
[Parallel(n_jobs=-1)]: Done  40 tasks      | elapsed:  2.8min
[Parallel(n_jobs=-1)]: Done  41 tasks      | elapsed:  2.9min
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  3.0min
[Parallel(n_jobs=-1)]: Done  43 tasks      | elapsed:  3.2min
```

```
[Parallel(n_jobs=-1)]: Done  44 tasks      | elapsed:  3.4min
[Parallel(n_jobs=-1)]: Done  45 tasks      | elapsed:  3.6min
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py:700: UserWarni
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:  3.8min
[Parallel(n_jobs=-1)]: Done  47 tasks      | elapsed:  4.1min
[Parallel(n_jobs=-1)]: Done  48 tasks      | elapsed:  4.2min
[Parallel(n_jobs=-1)]: Done  49 tasks      | elapsed:  4.3min
[Parallel(n_jobs=-1)]: Done  50 tasks      | elapsed:  4.3min
[Parallel(n_jobs=-1)]: Done  51 tasks      | elapsed:  4.4min
[Parallel(n_jobs=-1)]: Done  52 tasks      | elapsed:  4.6min
[Parallel(n_jobs=-1)]: Done  53 tasks      | elapsed:  4.7min
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py:700: UserWarni
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done  54 tasks      | elapsed:  4.8min
[Parallel(n_jobs=-1)]: Done  55 tasks      | elapsed:  5.1min
[Parallel(n_jobs=-1)]: Done  56 tasks      | elapsed:  5.2min
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py:700: UserWarni
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done  57 tasks      | elapsed:  5.5min
[Parallel(n_jobs=-1)]: Done  60 out of  60 | elapsed:  6.1min remaining:    0.0s
[Parallel(n_jobs=-1)]: Done  60 out of  60 | elapsed:  6.1min finished
```

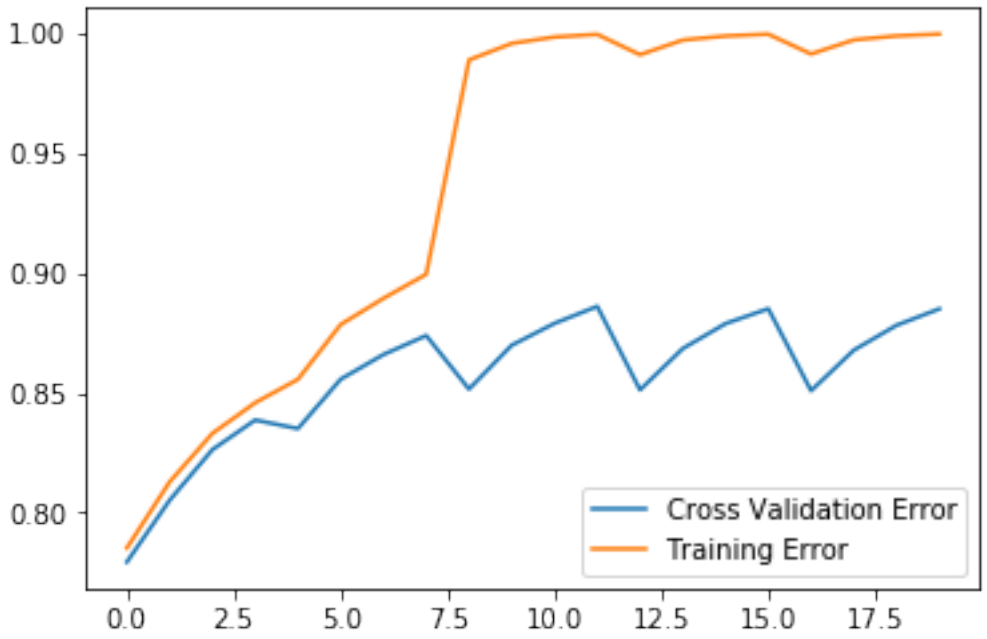
```
In [167]: print("Best parameters: ", xg_clf.best_params_)
          print("Best cross-validation score: {:.3f}".format(xg_clf.best_score_))

Best parameters:  {'max_depth': 15, 'n_estimators': 20}
Best cross-validation score: 0.886
```

```
In [0]: pred = xg_clf.predict(X_test3)
        pred_train = xg_clf.predict(X_train3)
```

```
In [0]: from sklearn.metrics import accuracy_score
        from sklearn.metrics import roc_curve
        import sklearn.metrics as metrics
        from sklearn.metrics import roc_auc_score

        train_error=xg_clf.cv_results_['mean_train_score']
        cv_error = xg_clf.cv_results_['mean_test_score']
        score=roc_auc_score(y_train, pred_train)

        estimator=xg_clf.best_params_['n_estimators']
        depth=xg_clf.best_params_['max_depth']
```

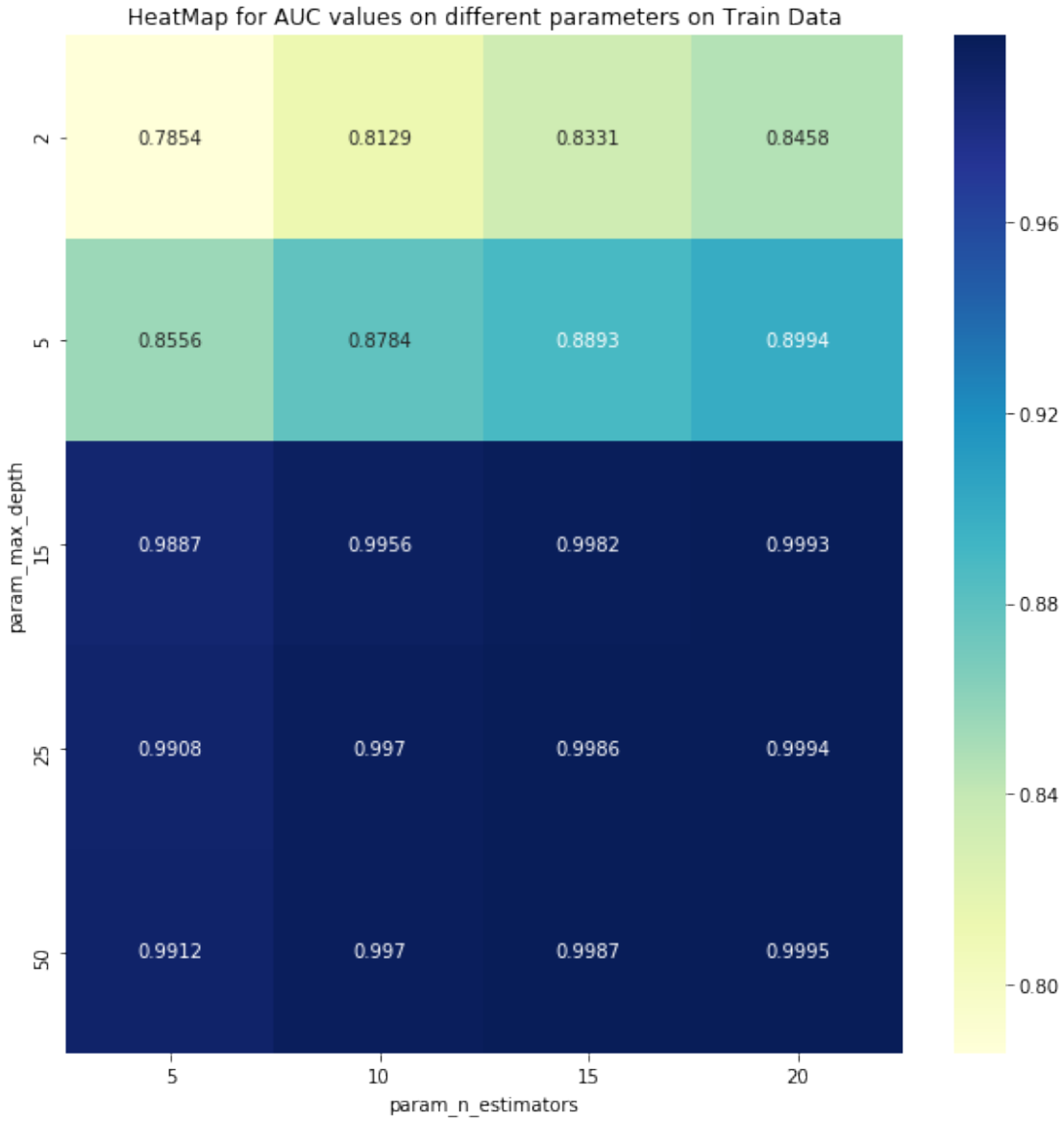
Error Plots

```
In [170]: plt.plot(cv_error, label='Cross Validation Error')
          plt.plot(train_error, label='Training Error')
          plt.legend()
          plt.show()
```

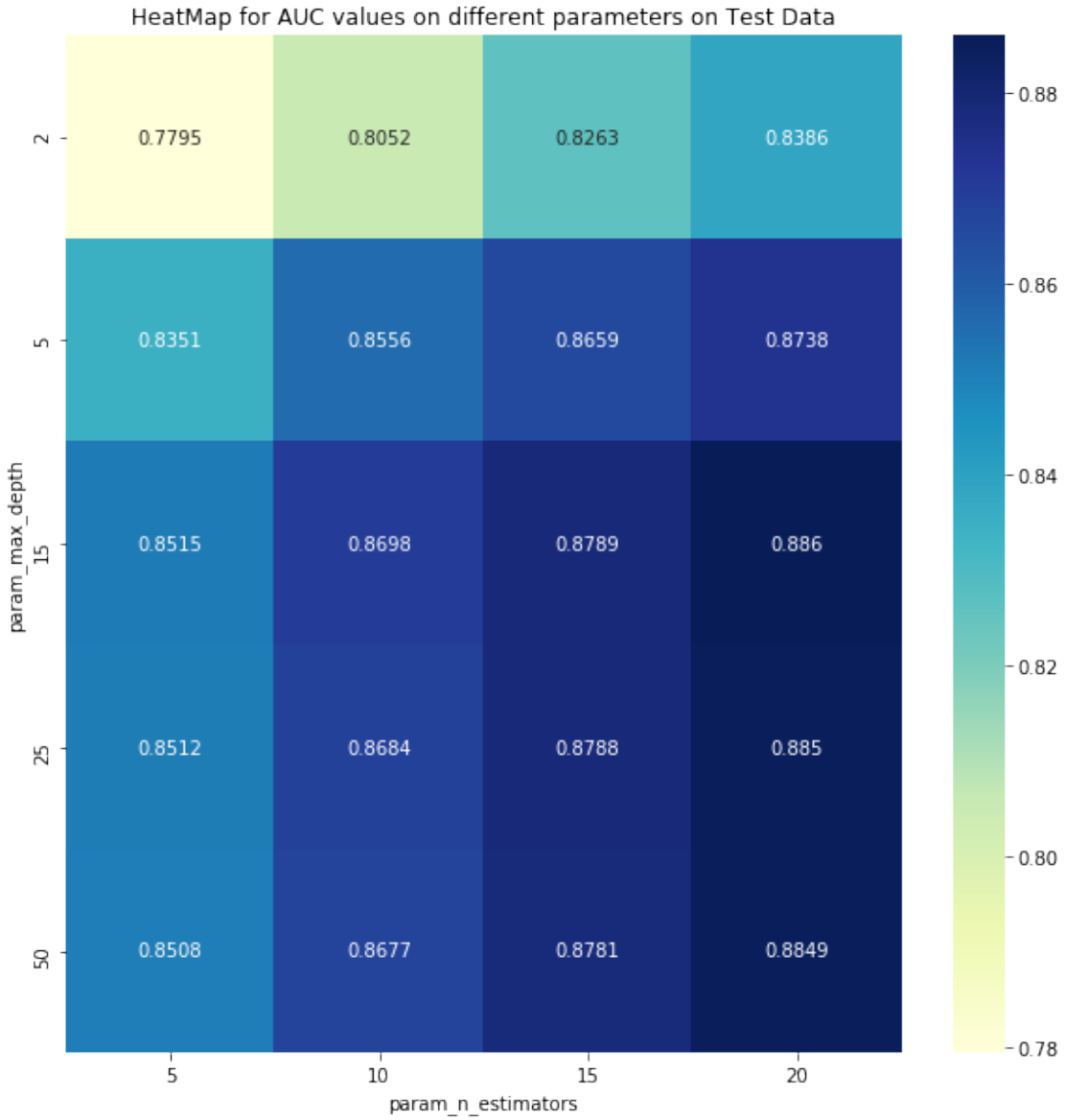


5.6.4 HeatMap for AUC vs N_estimators and Max_Depth

```
In [171]: df_new = pd.DataFrame(xg_clf.cv_results_)
          max_params = df_new.groupby(['param_max_depth','param_n_estimators']).max()
          max_params = max_params.unstack()[['mean_test_score', 'mean_train_score']]

          fig, ax = plt.subplots(figsize=(10,10))         # Sample figsize in inches

          sns.heatmap(max_params.mean_train_score, annot=True,cmap="YlGnBu", fmt='.4g',ax=ax)
          plt.title('HeatMap for AUC values on different parameters on Train Data')
          plt.show()
```

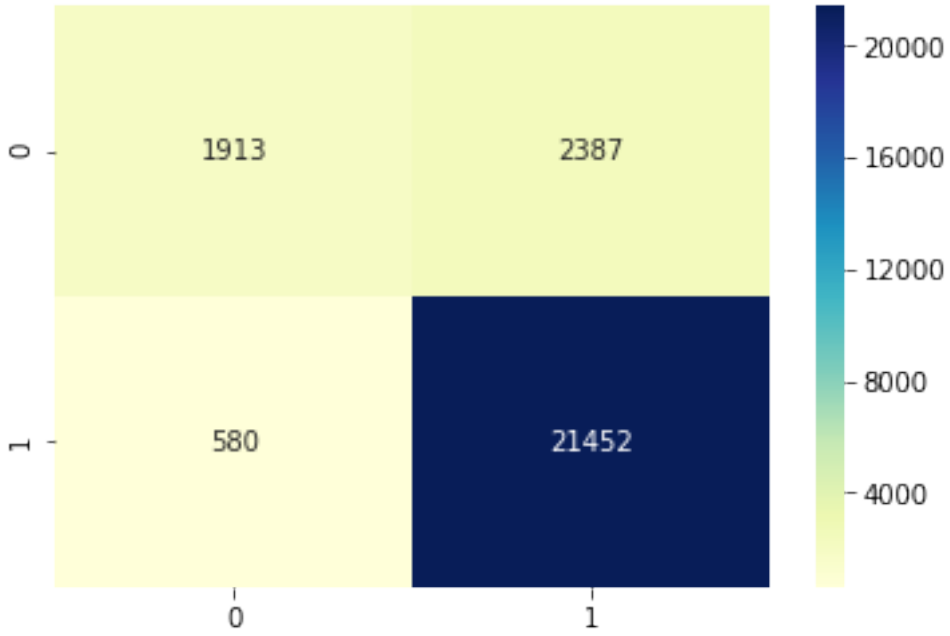



```
In [172]: df_new = pd.DataFrame(xg_clf.cv_results_)
          max_params = df_new.groupby(['param_max_depth','param_n_estimators']).max()
          max_params = max_params.unstack()[['mean_test_score', 'mean_train_score']]

          fig, ax = plt.subplots(figsize=(10,10))         # Sample figsize in inches

          sns.heatmap(max_params.mean_test_score, annot=True,cmap="YlGnBu", fmt='.4g',ax=ax)
          plt.title('HeatMap for AUC values on different parameters on Test Data')
          plt.show()
```



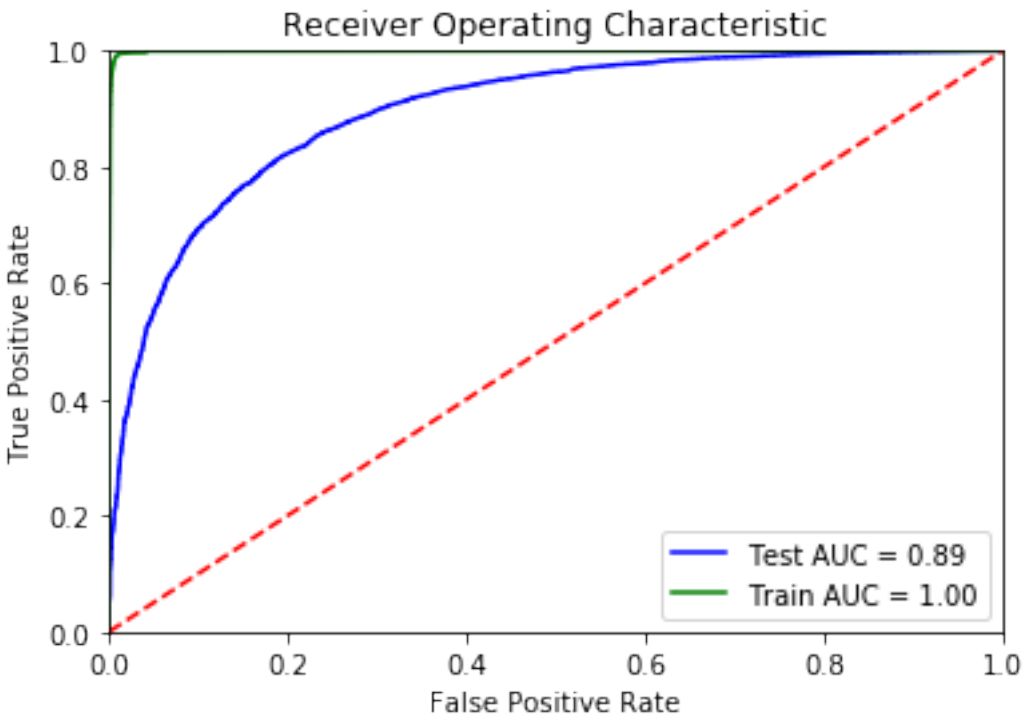

```
In [0]: pred_test = xg_clf.predict_proba(X_test3)[:,1]
        pred_train =  xg_clf.predict_proba(X_train3)[:,1]
```

```
In [174]: from sklearn.metrics import confusion_matrix

          cm = confusion_matrix(y_test, pred_test.round())
          sns.heatmap(cm, annot=True, fmt="d",cmap="YlGnBu")
          plt.show()
```




```
In [0]: fpr, tpr, thresholds = roc_curve(y_test, pred_test)
        fpr2, tpr2, thresholds = roc_curve(y_train, pred_train)

        score_test = roc_auc_score(y_test, pred_test)
        score_train = roc_auc_score(y_train, pred_train)
```

```
In [176]: roc_auc_test = metrics.auc(fpr, tpr)
          roc_auc_train = metrics.auc(fpr2, tpr2)
          plt.title('Receiver Operating Characteristic')
          plt.plot(fpr, tpr, 'b', label='Test AUC = %0.2f' % score_test)
          plt.plot(fpr2, tpr2, 'g', label = 'Train AUC = %0.2f' % score_train)
          plt.legend(loc = 'lower right')
          plt.plot([0, 1], [0, 1],'r--')
          plt.xlim([0, 1])
          plt.ylim([0, 1])
          plt.ylabel('True Positive Rate')
          plt.xlabel('False Positive Rate')
          plt.legend()
          plt.show()
```



5.6.5 [4.4.1.2] TFIDF weighted W2v

In [180]:
```
X = list_sent[:]
y = final['Score'][:]
X_train, X_test, y_train, y_test = train_test_split(X,y , test_size=0.30, random_sta

print('done')
```

```
done
```

In [0]:
```
dictionary = dict(zip(tf_idf_vect.get_feature_names(), list(tf_idf_vect.idf_)))

tfidf_features = tf_idf_vect.get_feature_names()

train_vectors = []; # the tfidf-w2v for each sentence/review is stored in this list
row=0;
for sent in X_train:
    # for each review/sentence
    sent_vec = np.zeros(50) # as word vectors are of zero length
    weight_sum =0; # num of words with a valid vector in the sentence/review
    for word in sent: # for each word in a review/sentence
        if word in w2v_words and word in tfidf_features:
```

```
                vec = w2v_model.wv[word]
                #tf_idf = tf_idf_matrix[row, tfidf_feat.index(word)]
                # to reduce the computation we are
                # dictionary[word] = idf value of word in whole courpus
                # sent.count(word) = tf valeus of word in this review
                tf_idf = dictionary[word]*(sent.count(word)/len(sent))
                sent_vec += (vec * tf_idf)
                weight_sum += tf_idf
        if weight_sum != 0:
            sent_vec /= weight_sum
        train_vectors.append(sent_vec)
        row += 1
```

```
In [0]: test_vectors = [];

        row=0;
        for sent in X_test:
            sent_vec = np.zeros(50)
            weight_sum =0;
            for word in sent:
                if word in w2v_words and word in tfidf_features:
                    vec = w2v_model.wv[word]
                    tf_idf = dictionary[word]*(sent.count(word)/len(sent))
                    sent_vec += (vec * tf_idf)
                    weight_sum += tf_idf
            if weight_sum != 0:
                sent_vec /= weight_sum
            test_vectors.append(sent_vec)
            row += 1
```

```
In [0]: X_train4 = train_vectors
        X_test4 = test_vectors
```

5.6.6 Applying Random Forest Classifier

```
In [184]: from sklearn.ensemble import RandomForestClassifier
          from sklearn.model_selection import GridSearchCV


          grid_params = dict(n_estimators = [10,20,50,100],
                             max_depth= [6,10,15,20] )

          rf = RandomForestClassifier()
          rf_clf = GridSearchCV(rf, param_grid=grid_params, n_jobs=-1, verbose=30,return_train
          rf_clf = rf_clf.fit(X_train4, y_train)


          results = rf_clf.cv_results_
```

```
Fitting 3 folds for each of 16 candidates, totalling 48 fits


[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done   1 tasks      | elapsed:    5.3s
[Parallel(n_jobs=-1)]: Done   2 tasks      | elapsed:    6.9s
[Parallel(n_jobs=-1)]: Done   3 tasks      | elapsed:    8.8s
[Parallel(n_jobs=-1)]: Done   4 tasks      | elapsed:   12.6s
[Parallel(n_jobs=-1)]: Done   5 tasks      | elapsed:   15.1s
[Parallel(n_jobs=-1)]: Done   6 tasks      | elapsed:   18.4s
[Parallel(n_jobs=-1)]: Done   7 tasks      | elapsed:   27.1s
[Parallel(n_jobs=-1)]: Done   8 tasks      | elapsed:   30.3s
[Parallel(n_jobs=-1)]: Done   9 tasks      | elapsed:   39.0s
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:   52.6s
[Parallel(n_jobs=-1)]: Done  11 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done  12 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done  13 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done  14 tasks      | elapsed:  1.3min
[Parallel(n_jobs=-1)]: Done  15 tasks      | elapsed:  1.3min
[Parallel(n_jobs=-1)]: Done  16 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done  18 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done  19 tasks      | elapsed:  1.7min
[Parallel(n_jobs=-1)]: Done  20 tasks      | elapsed:  1.8min
[Parallel(n_jobs=-1)]: Done  21 tasks      | elapsed:  2.0min
[Parallel(n_jobs=-1)]: Done  22 tasks      | elapsed:  2.4min
[Parallel(n_jobs=-1)]: Done  23 tasks      | elapsed:  2.6min
[Parallel(n_jobs=-1)]: Done  24 tasks      | elapsed:  2.7min
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:  2.8min
[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed:  2.9min
[Parallel(n_jobs=-1)]: Done  27 tasks      | elapsed:  3.0min
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:  3.1min
[Parallel(n_jobs=-1)]: Done  29 tasks      | elapsed:  3.1min
[Parallel(n_jobs=-1)]: Done  30 tasks      | elapsed:  3.2min
[Parallel(n_jobs=-1)]: Done  31 tasks      | elapsed:  3.5min
[Parallel(n_jobs=-1)]: Done  32 tasks      | elapsed:  3.6min
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:  3.9min
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py:700: UserWarni
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:  4.3min
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py:700: UserWarni
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done  35 tasks      | elapsed:  4.6min
[Parallel(n_jobs=-1)]: Done  36 tasks      | elapsed:  4.7min
[Parallel(n_jobs=-1)]: Done  37 tasks      | elapsed:  4.8min
[Parallel(n_jobs=-1)]: Done  38 tasks      | elapsed:  4.9min
[Parallel(n_jobs=-1)]: Done  39 tasks      | elapsed:  5.0min
[Parallel(n_jobs=-1)]: Done  40 tasks      | elapsed:  5.1min
```

```
[Parallel(n_jobs=-1)]: Done  41 tasks      | elapsed:  5.2min
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  5.3min
[Parallel(n_jobs=-1)]: Done  43 tasks      | elapsed:  5.6min
[Parallel(n_jobs=-1)]: Done  44 tasks      | elapsed:  5.7min
[Parallel(n_jobs=-1)]: Done  45 tasks      | elapsed:  6.0min
[Parallel(n_jobs=-1)]: Done  48 out of  48 | elapsed:  7.0min finished
```

```
In [185]: print("Best parameters: ", rf_clf.best_params_)
          print("Best cross-validation score: {:.3f}".format(rf_clf.best_score_))
```

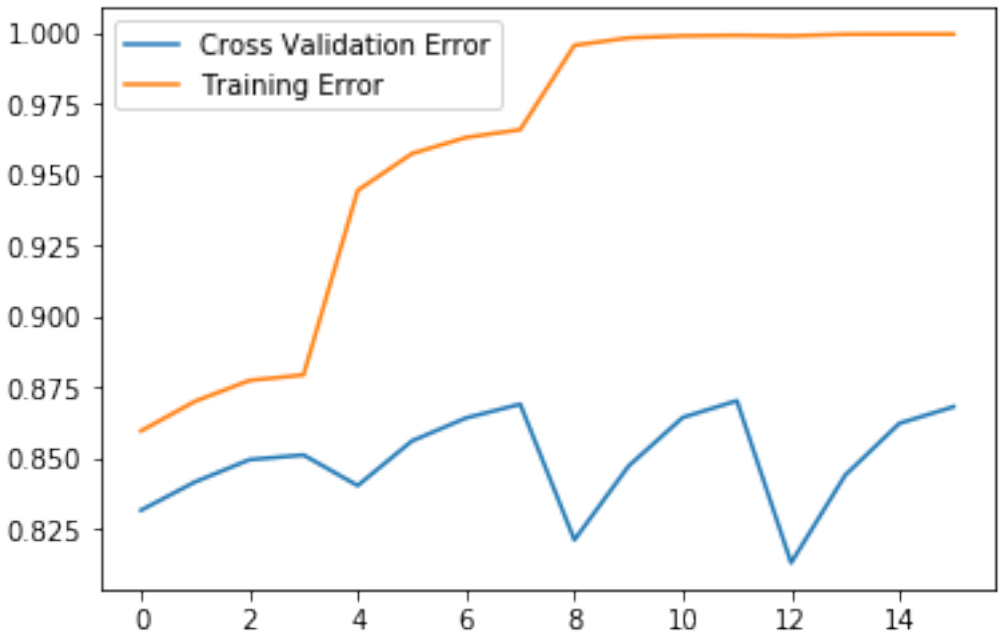
```
Best parameters:  {'max_depth': 15, 'n_estimators': 100}
Best cross-validation score: 0.870
```

```
In [0]: pred = rf_clf.predict(X_test4)
        pred_train = rf_clf.predict(X_train4)
```

```
In [0]: from sklearn.metrics import accuracy_score
        from sklearn.metrics import roc_curve
        import sklearn.metrics as metrics
        from sklearn.metrics import roc_auc_score

        train_error=rf_clf.cv_results_['mean_train_score']
        cv_error = rf_clf.cv_results_['mean_test_score']
        score=roc_auc_score(y_train, pred_train)

        estimator=rf_clf.best_params_['n_estimators']
        depth=rf_clf.best_params_['max_depth']
```

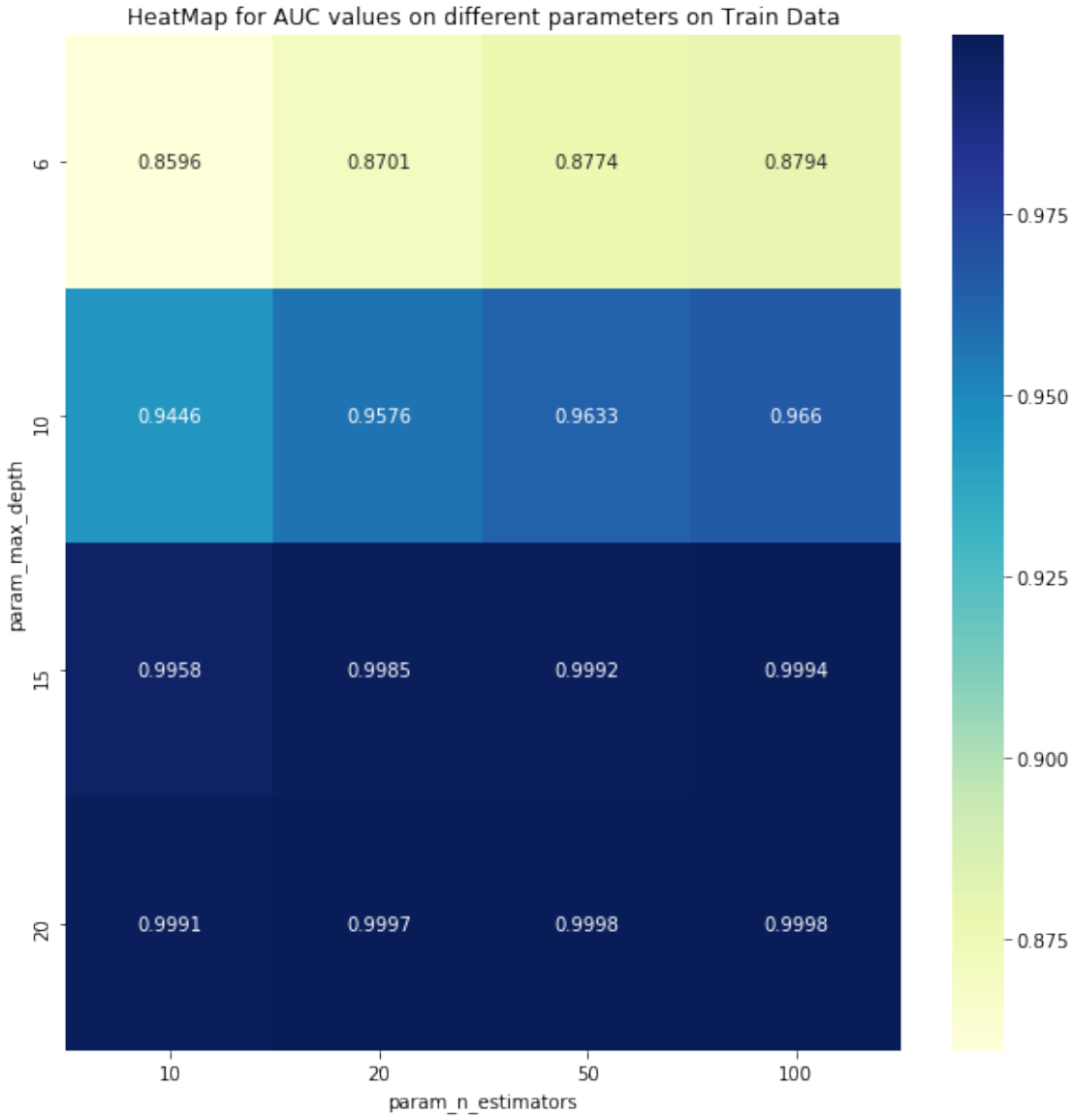
Error plots

```
In [188]: plt.plot(cv_error, label='Cross Validation Error')
          plt.plot(train_error, label='Training Error')
          plt.legend()
          plt.show()
```

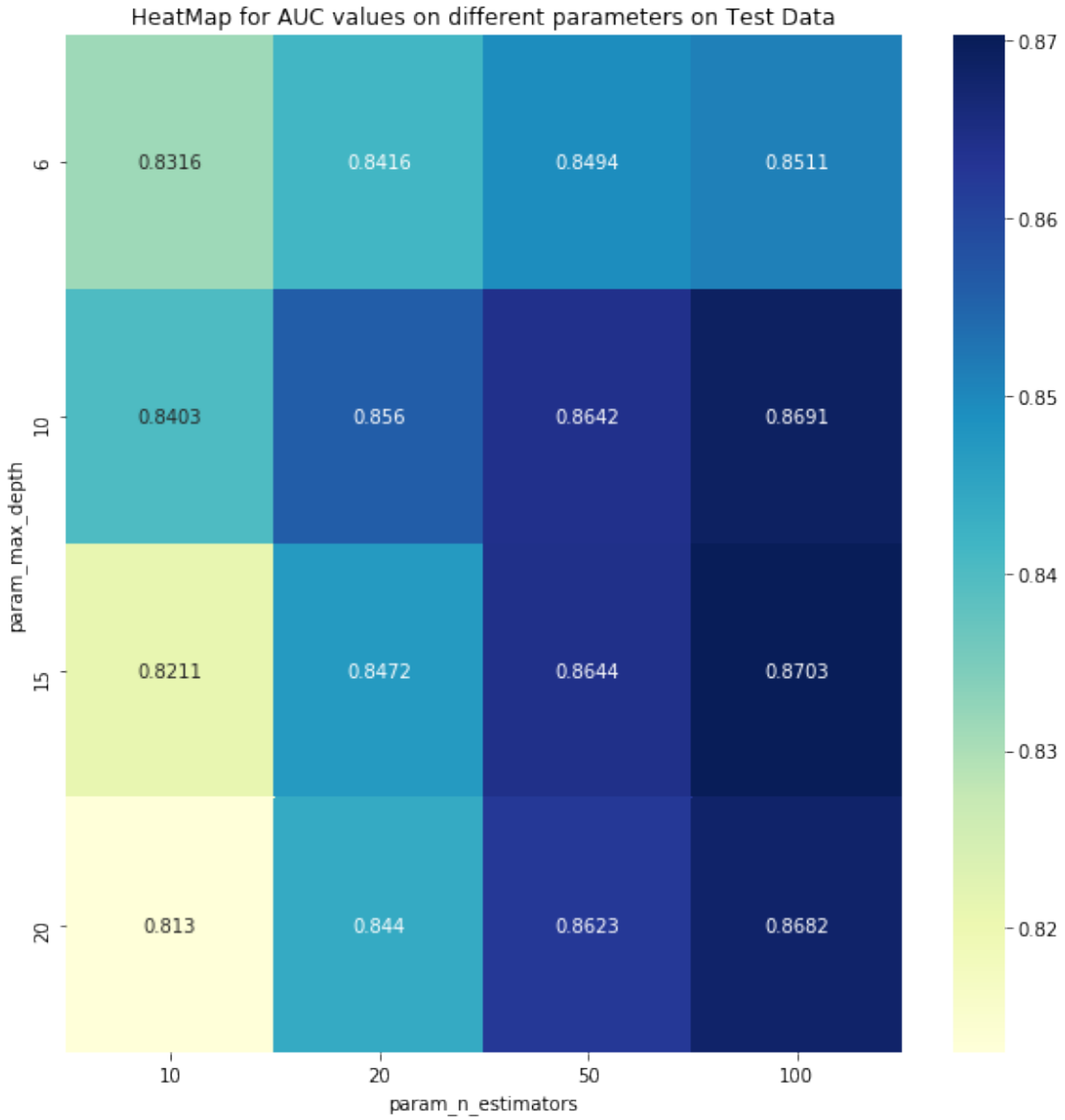


5.6.7 HeatMap for AUC vs N_estimators and Max_Depth

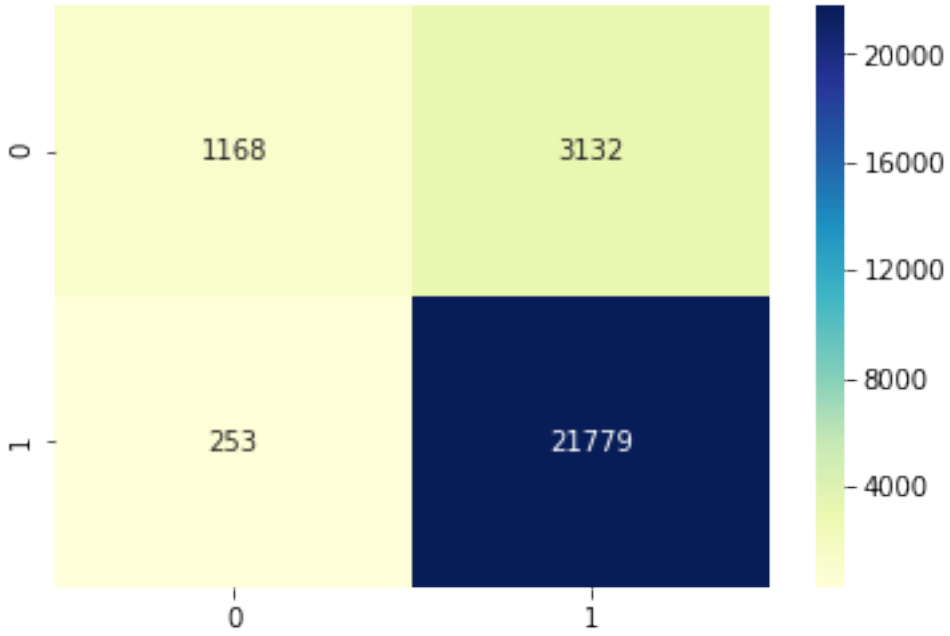
```
In [189]: df_new = pd.DataFrame(rf_clf.cv_results_)
          max_params = df_new.groupby(['param_max_depth','param_n_estimators']).max()
          max_params = max_params.unstack()[['mean_test_score', 'mean_train_score']]

          fig, ax = plt.subplots(figsize=(10,10))         # Sample figsize in inches

          sns.heatmap(max_params.mean_train_score, annot=True,cmap="YlGnBu", fmt='.4g',ax=ax)
          plt.title('HeatMap for AUC values on different parameters on Train Data')
          plt.show()
```




```
In [190]: df_new = pd.DataFrame(rf_clf.cv_results_)
          max_params = df_new.groupby(['param_max_depth','param_n_estimators']).max()
          max_params = max_params.unstack()[['mean_test_score', 'mean_train_score']]

          fig, ax = plt.subplots(figsize=(10,10))         # Sample figsize in inches

          sns.heatmap(max_params.mean_test_score, annot=True,cmap="YlGnBu", fmt='.4g',ax=ax)
          plt.title('HeatMap for AUC values on different parameters on Test Data')
          plt.show()
```

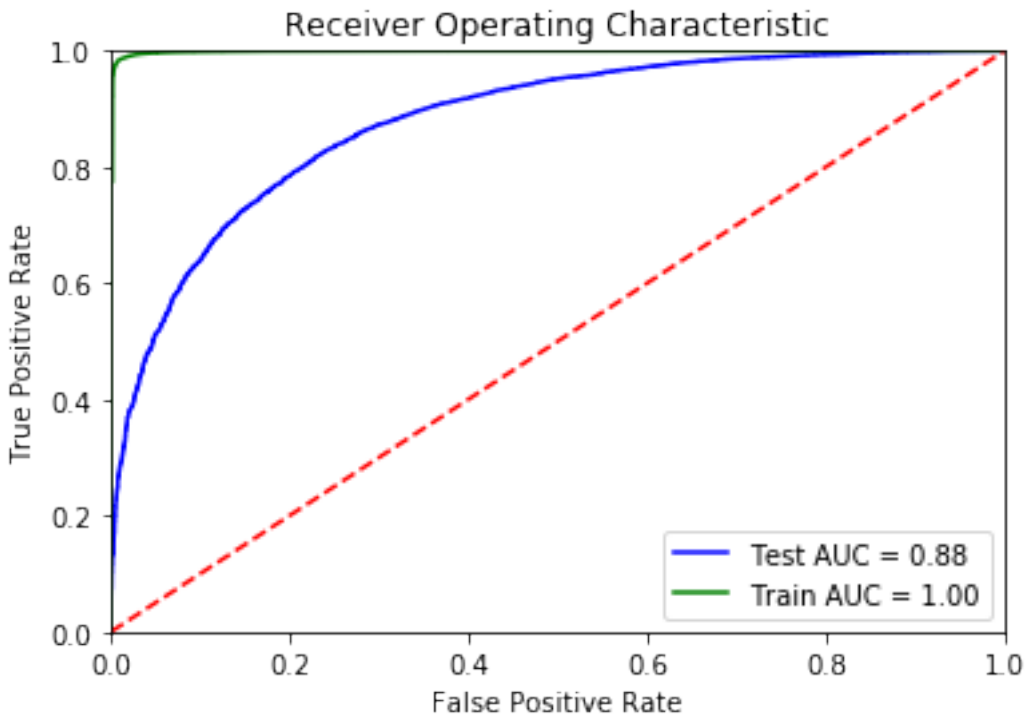



```
In [0]: pred_test = rf_clf.predict_proba(X_test4)[:,1]
        pred_train =  rf_clf.predict_proba(X_train4)[:,1]

In [192]: from sklearn.metrics import confusion_matrix

          cm = confusion_matrix(y_test, pred_test.round())
          sns.heatmap(cm, annot=True, fmt="d",cmap="YlGnBu")
          plt.show()
```




```
In [0]: fpr, tpr, thresholds = roc_curve(y_test, pred_test)
        fpr2, tpr2, thresholds = roc_curve(y_train, pred_train)

        score_test = roc_auc_score(y_test, pred_test)
        score_train = roc_auc_score(y_train, pred_train)
```

```
In [194]: roc_auc_test = metrics.auc(fpr, tpr)
          roc_auc_train = metrics.auc(fpr2, tpr2)
          plt.title('Receiver Operating Characteristic')
          plt.plot(fpr, tpr, 'b', label='Test AUC = %0.2f' % score_test)
          plt.plot(fpr2, tpr2, 'g', label = 'Train AUC = %0.2f' % score_train)
          plt.legend(loc = 'lower right')
          plt.plot([0, 1], [0, 1],'r--')
          plt.xlim([0, 1])
          plt.ylim([0, 1])
          plt.ylabel('True Positive Rate')
          plt.xlabel('False Positive Rate')
          plt.legend()
          plt.show()
```



5.6.8 Applying XGBOOST Classifier

```
In [195]: from sklearn.model_selection import GridSearchCV


          grid_params = {'n_estimators': [5, 10, 15,20],
                         'max_depth': [2, 5, 15]}

          xg_clf = xgb.XGBClassifier()
          xg_clf = GridSearchCV(xg_clf, param_grid=grid_params, n_jobs=-1, verbose=30,return_t
          xg_clf = xg_clf.fit(np.array(X_train4), y_train)

          results =xg_clf.cv_results_
```

```
Fitting 3 folds for each of 12 candidates, totalling 36 fits


[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done   1 tasks      | elapsed:    1.7s
[Parallel(n_jobs=-1)]: Done   2 tasks      | elapsed:    1.7s
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py:700: UserWarni
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py:700: UserWarni
  "timeout or by a memory leak.", UserWarning
```

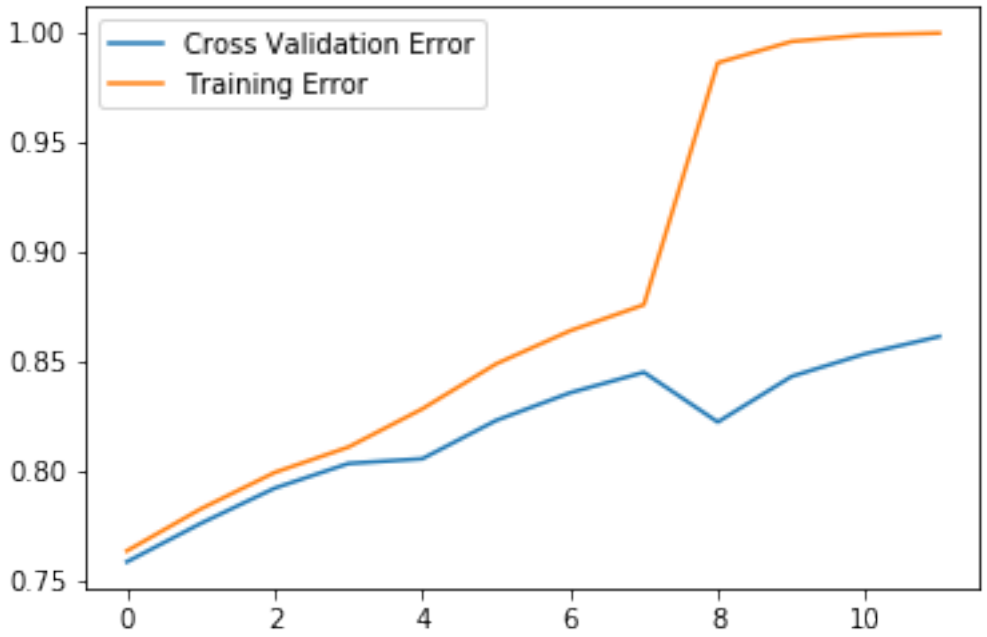
```
[Parallel(n_jobs=-1)]: Done   3 tasks      | elapsed:    5.4s
[Parallel(n_jobs=-1)]: Done   4 tasks      | elapsed:    6.5s
[Parallel(n_jobs=-1)]: Done   5 tasks      | elapsed:    7.6s
[Parallel(n_jobs=-1)]: Done   6 tasks      | elapsed:    8.7s
[Parallel(n_jobs=-1)]: Done   7 tasks      | elapsed:   10.7s
[Parallel(n_jobs=-1)]: Done   8 tasks      | elapsed:   11.7s
[Parallel(n_jobs=-1)]: Done   9 tasks      | elapsed:   13.5s
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:   15.2s
[Parallel(n_jobs=-1)]: Done  11 tasks      | elapsed:   16.9s
[Parallel(n_jobs=-1)]: Done  12 tasks      | elapsed:   18.5s
[Parallel(n_jobs=-1)]: Done  13 tasks      | elapsed:   19.1s
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py:700: UserWarni
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done  14 tasks      | elapsed:   20.9s
[Parallel(n_jobs=-1)]: Done  15 tasks      | elapsed:   23.2s
[Parallel(n_jobs=-1)]: Done  16 tasks      | elapsed:   24.9s
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py:700: UserWarni
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:   27.5s
[Parallel(n_jobs=-1)]: Done  18 tasks      | elapsed:   30.8s
[Parallel(n_jobs=-1)]: Done  19 tasks      | elapsed:   33.3s
[Parallel(n_jobs=-1)]: Done  20 tasks      | elapsed:   36.7s
[Parallel(n_jobs=-1)]: Done  21 tasks      | elapsed:   39.1s
[Parallel(n_jobs=-1)]: Done  22 tasks      | elapsed:   44.4s
[Parallel(n_jobs=-1)]: Done  23 tasks      | elapsed:   47.0s
[Parallel(n_jobs=-1)]: Done  24 tasks      | elapsed:   52.1s
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:   53.8s
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py:700: UserWarni
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed:   58.9s
[Parallel(n_jobs=-1)]: Done  27 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done  29 tasks      | elapsed:  1.3min
[Parallel(n_jobs=-1)]: Done  30 tasks      | elapsed:  1.4min
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py:700: UserWarni
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done  31 tasks      | elapsed:  1.6min
[Parallel(n_jobs=-1)]: Done  32 tasks      | elapsed:  1.8min
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:  1.9min
[Parallel(n_jobs=-1)]: Done  36 out of  36 | elapsed:  2.5min finished
```

```
In [196]: print("Best parameters: ", xg_clf.best_params_)
          print("Best cross-validation score: {:.3f}".format(xg_clf.best_score_))

Best parameters:  {'max_depth': 15, 'n_estimators': 20}
Best cross-validation score: 0.861
```

```
In [0]: pred = xg_clf.predict(X_test4)
        pred_train = xg_clf.predict(X_train4)
```

```
In [0]: from sklearn.metrics import accuracy_score
        from sklearn.metrics import roc_curve
        import sklearn.metrics as metrics
        from sklearn.metrics import roc_auc_score

        train_error=xg_clf.cv_results_['mean_train_score']
        cv_error = xg_clf.cv_results_['mean_test_score']
        score=roc_auc_score(y_train, pred_train)

        estimator=xg_clf.best_params_['n_estimators']
        depth=xg_clf.best_params_['max_depth']
```

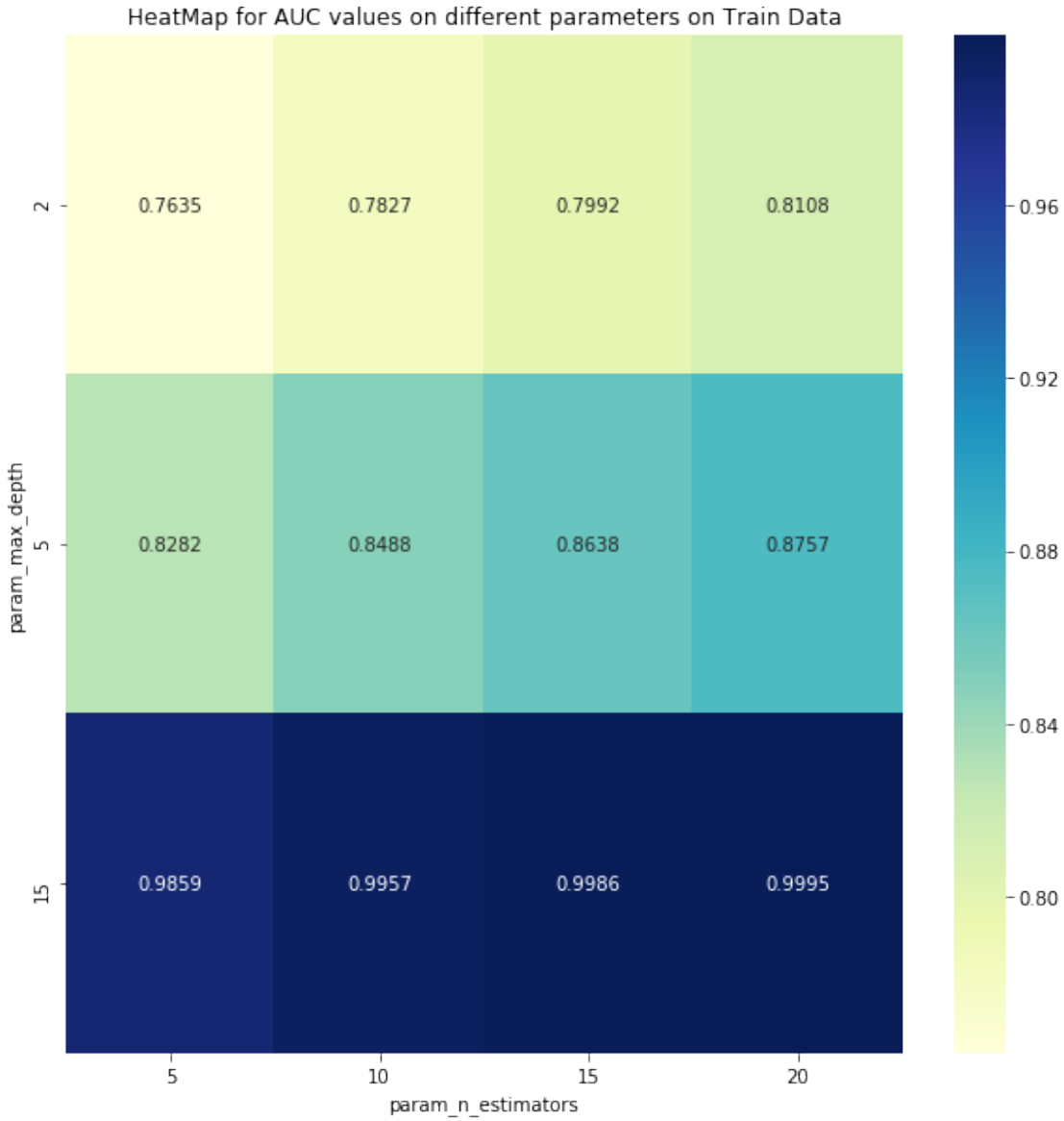
```
In [199]: plt.plot(cv_error, label='Cross Validation Error')
          plt.plot(train_error, label='Training Error')
          plt.legend()
          plt.show()
```



5.6.9 HeatMap for AUC vs N_estimators and Max_Depth

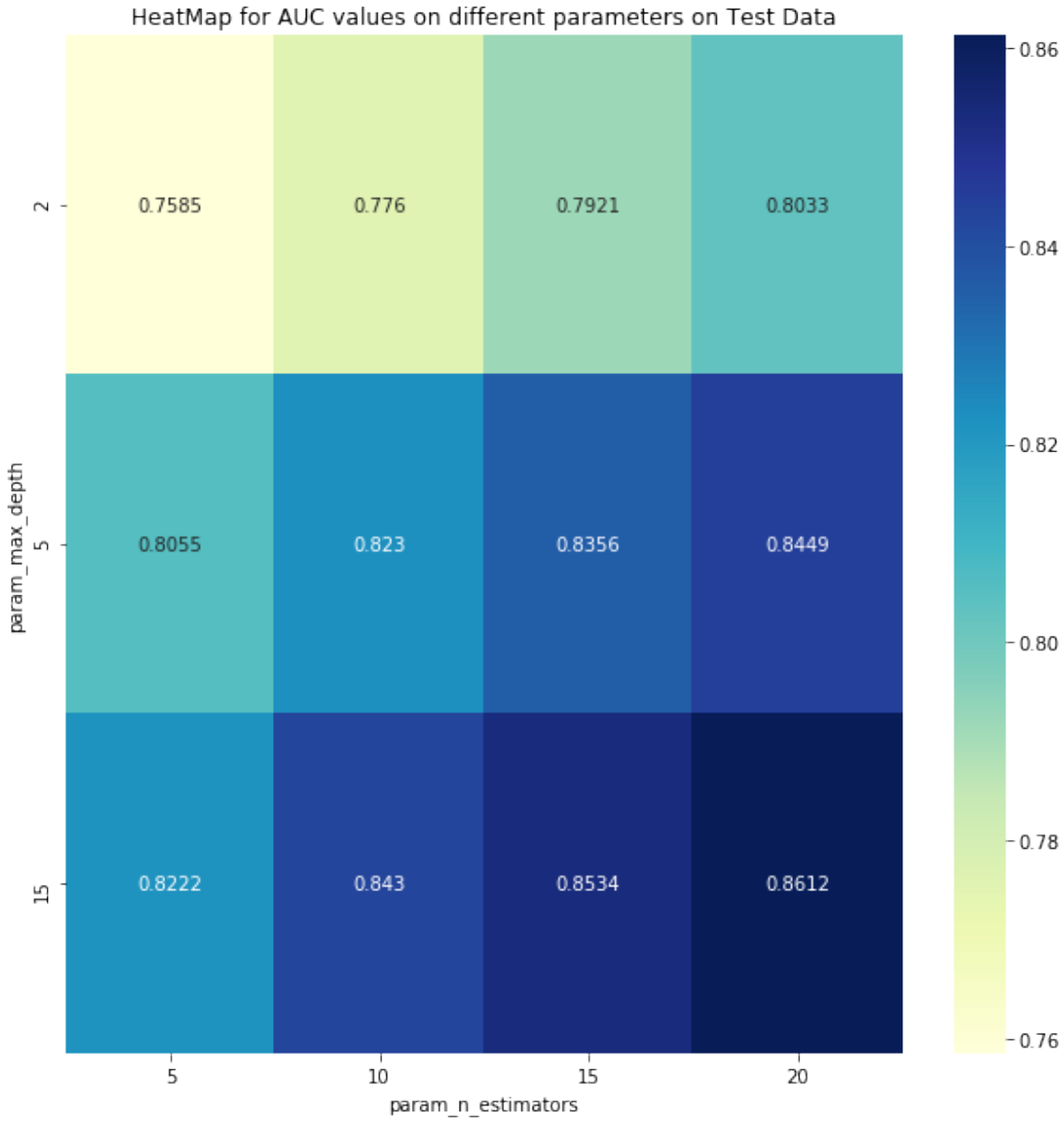
```
In [200]: df_new = pd.DataFrame(xg_clf.cv_results_)
          max_params = df_new.groupby(['param_max_depth','param_n_estimators']).max()
          max_params = max_params.unstack()[['mean_test_score', 'mean_train_score']]
```

```
fig, ax = plt.subplots(figsize=(10,10))         # Sample figsize in inches

sns.heatmap(max_params.mean_train_score, annot=True,cmap="YlGnBu", fmt='.4g',ax=ax)
plt.title('HeatMap for AUC values on different parameters on Train Data')
plt.show()
```

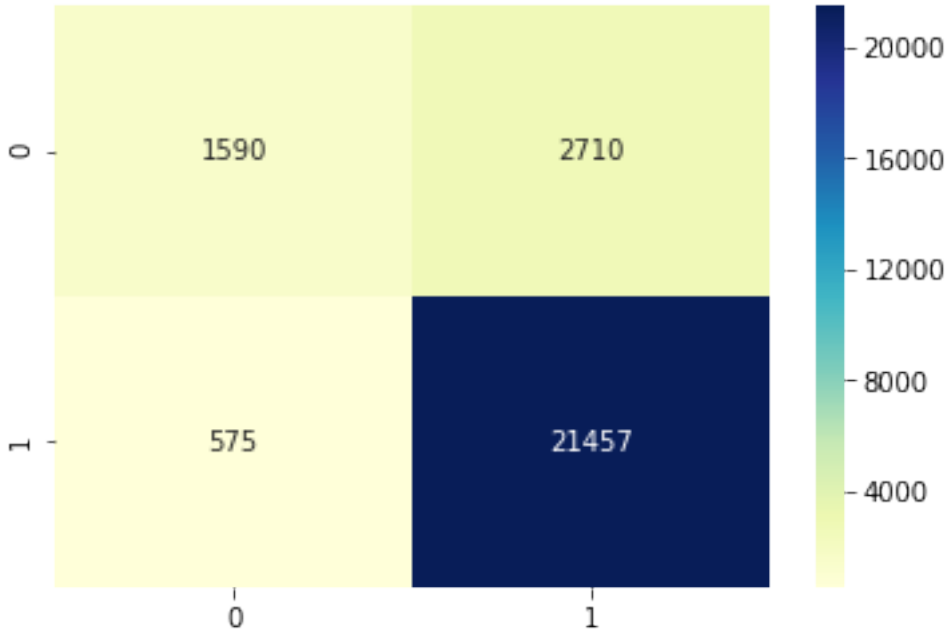


In [201]:
```
df_new = pd.DataFrame(xg_clf.cv_results_)
max_params = df_new.groupby(['param_max_depth','param_n_estimators']).max()
max_params = max_params.unstack()[['mean_test_score', 'mean_train_score']]

fig, ax = plt.subplots(figsize=(10,10))         # Sample figsize in inches
```

```
sns.heatmap(max_params.mean_test_score, annot=True,cmap="YlGnBu", fmt='.4g',ax=ax)
plt.title('HeatMap for AUC values on different parameters on Test Data')
plt.show()
```

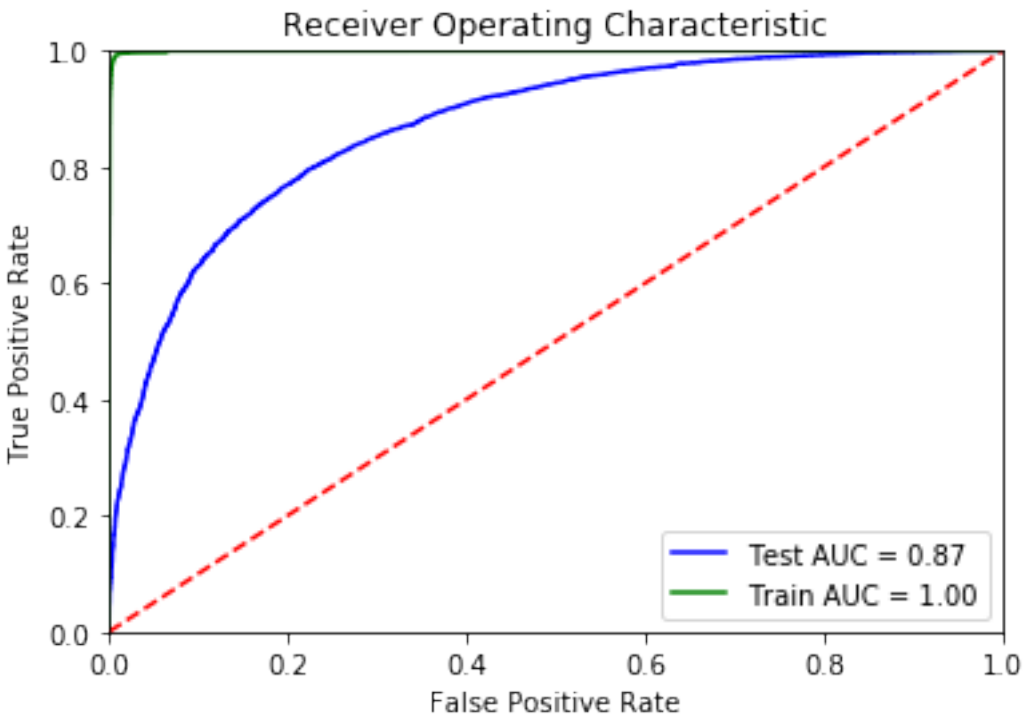



```
In [0]: pred_test = xg_clf.predict_proba(X_test4)[:,1]
        pred_train =  xg_clf.predict_proba(X_train4)[:,1]

In [203]: from sklearn.metrics import confusion_matrix

          cm = confusion_matrix(y_test, pred_test.round())
          sns.heatmap(cm, annot=True, fmt="d",cmap="YlGnBu")
          plt.show()
```




```
In [0]: fpr, tpr, thresholds = roc_curve(y_test, pred_test)
        fpr2, tpr2, thresholds = roc_curve(y_train, pred_train)

        score_test = roc_auc_score(y_test, pred_test)
        score_train = roc_auc_score(y_train, pred_train)
```

```
In [205]: roc_auc_test = metrics.auc(fpr, tpr)
          roc_auc_train = metrics.auc(fpr2, tpr2)
          plt.title('Receiver Operating Characteristic')
          plt.plot(fpr, tpr, 'b', label='Test AUC = %0.2f' % score_test)
          plt.plot(fpr2, tpr2, 'g', label = 'Train AUC = %0.2f' % score_train)
          plt.legend(loc = 'lower right')
          plt.plot([0, 1], [0, 1],'r--')
          plt.xlim([0, 1])
          plt.ylim([0, 1])
          plt.ylabel('True Positive Rate')
          plt.xlabel('False Positive Rate')
          plt.legend()
          plt.show()
```



---

6 [5] Conclusions

```
In [206]: from prettytable import PrettyTable

          x = PrettyTable()

          x.field_names = ["1", "2", "Train AUC", "Test AUC"]

          x.add_row(["Random Forest", "Bag-of-Words", 0.93, 0.90])
          x.add_row(["XGBoost", "Bag-of-Words", 0.99, 0.94])
          x.add_row(["Random Forest", "TF-IDF", 0.95, 0.92])
          x.add_row(["XGBoost", "TF-IDF", 0.99, 0.94])
          x.add_row(["Random Forest", "AVG-W2V", 0.97, 0.90])
          x.add_row(["XGBoost", "AVG-W2V", 1.0, 0.89])
          x.add_row(["Random Forest", "W2V-TFIDF", 1.0, 0.88])
          x.add_row(["XGBoost", "W2V-TFIDF", 1.0, 0.87])


          print(x)
```

```
+---------------+--------------+-----------+----------+
|       1       |      2       | Train AUC | Test AUC |
```

Random Forest	Bag-of-Words	0.93	0.9
XGBoost	Bag-of-Words	0.99	0.94
Random Forest	TF-IDF	0.95	0.92
XGBoost	TF-IDF	0.99	0.94
Random Forest	AVG-W2V	0.97	0.9
XGBoost	AVG-W2V	1.0	0.89
Random Forest	W2V-TFIDF	1.0	0.88
XGBoost	W2V-TFIDF	1.0	0.87