DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in need of funding. Right now, a large number of volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org website.

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main problems they need to solve:

- How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as efficiently as possible
- How to increase the consistency of project vetting across different volunteers to improve the experience for teachers
- How to focus volunteer time on the applications that need the most assistance

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

About the DonorsChoose Data Set

The `train.csv` data set provided by DonorsChoose contains the following features:

Feature	De
project_id	A unique identifier for the proposed project. **Example:**
project_title	Title of the project. **E** • Art Will Make You • First Gr
project_grade_category	Grade level of students for which the project is targeted. One of the enumerate • Grades • Gra • Gra • Grad
project_subject_categories	One or more (comma-separated) subject categories for the projec following enumerated list • Applied L • Care & • Health & • History & • Literacy & L • Math & • Music & T • Specia • **E** • Music & T • Literacy & Language, Math &
school_state	State where school is located (Two-letter U.S. p (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Posta **Exar**
project_subject_subcategories	One or more (comma-separated) subject subcategories for th **E** • L • Literature & Writing, Social S
project_resource_summary	An explanation of the resources needed for the project. **E** • My students need hands on literacy materials to sensory
project_essay_1	First applicat
project_essay_2	Second applicat
project_essay_3	Third applicat
project_essay_4	Fourth applicat
project_submitted_datetime	Datetime when project application was submitted. **Example:** 201 12:43
teacher_id	A unique identifier for the teacher of the proposed project. bdf8baa8fedef6bfeec7ae4ff

Feature	De
`teacher_prefix`	Teacher's title. One of the following enumerate • • • • • • T
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the sam **Exa**

* See the section **Notes on the Essay Data** for more details about these features.

Additionally, the `resources.csv` data set provides more data about the resources required for each project. Each line in this file represents a resource required by a project:

Feature	Description
`id`	A `project_id` value from the `train.csv` file. **Example:** `p036502`
`description`	Desciption of the resource. **Example:** `Tenor Saxophone Reeds, Box of 25`
`quantity`	Quantity of the resource required. **Example:** `3`
`price`	Price of the resource required. **Example:** `9.95`

Note: Many projects require multiple resources. The `id` value corresponds to a `project_id` in train.csv, so you use it as a key to retrieve all resources needed for a project:

The data set contains the following label (the value you will attempt to predict):

Label	Description
`project_is_approved`	A binary flag indicating whether DonorsChoose approved the project. A value of `0` indicates the

Notes on the Essay Data

Prior to May 17, 2016, the prompts for the essays were as follows:

- __project_essay_1:__ "Introduce us to your classroom"
- __project_essay_2:__ "Tell us more about your students"
- __project_essay_3:__ "Describe how your students will use the materials you're requesting"
- __project_essay_3:__ "Close by sharing why your project will make a difference"

Starting on May 17, 2016, the number of essays was reduced from 4 to 2, and the prompts for the first 2 essays were changed to the following:

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

For all projects with project_submitted_datetime of 2016-05-17 and later, the values of project_essay_3 and project_essay_4 will be NaN.

In [2]:

```
%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

from plotly import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

```
C:\Users\RASHU TYAGI\Anaconda3\lib\site-packages\smart_open\ssh.py:34: Use
rWarning: paramiko missing, opening SSH/SCP/SFTP paths will be disabled.
`pip install paramiko` to suppress
  warnings.warn('paramiko missing, opening SSH/SCP/SFTP paths will be disa
bled.  `pip install paramiko` to suppress')
```

1.1 Reading Data

In [3]:

```
project_data = pd.read_csv(r'D:\Rashu Studies\AppliedAICourse\Assignments\Mandatory Ass
ignments\Mandatory Assignment 3 Donors Choose KNN\train_data.csv')
resource_data = pd.read_csv(r'D:\Rashu Studies\AppliedAICourse\Assignments\Mandatory As
signments\Mandatory Assignment 3 Donors Choose KNN\resources.csv')
```

In [4]:

```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix'
'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [5]:

```
# how to replace elements in list python: https://stackoverflow.com/a/2582163/4084039
cols = ['Date' if x=='project_submitted_datetime' else x for x in list(project_data.col
umns)]


#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/40840
39
project_data['Date'] = pd.to_datetime(project_data['project_submitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)


# how to reorder columns pandas python: https://stackoverflow.com/a/13148611/4084039
project_data = project_data[cols]


project_data.head(2)
```

Out[5]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT

In [6]:

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

```
Number of data points in train data (1541272, 4)
['id' 'description' 'quantity' 'price']
```

Out[6]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

NOW THE MOST IMPORTANT THING HERE IS THAT YOU SHOULD SPLIT OUR DATA INTO TRAIN AND TEST BEFORE APPLYING ANY FIT TECHNIQUE LIKE BOW OR TFIDF BECAUSE OTHERWISE THERE WILL BE DATA LEAKAGE PROBLEM.ALSO FOR PREPROCESSING LIKE STANDARDIZATION AND NORMALIZATION ALSO WE SHOULD KEEP IN MIND THAT TRAIN TEST SPLIT SHOULD BE DONE BEFORE APPLYING THOSE PREPROCESSING TECHNIQUES

In [7]:

```
# REFER THIS SOUNDCLOUD LINK : https://soundcloud.com/applied-ai-course/leakage-bow-and
-tfidf
```

Train_Test Split

In [8]:

```
# train test split
# note that here This stratify parameter makes a split so that the proportion of values
in the sample produced will be the same as the proportion of values provided to paramet
er stratify.
#For example, if variable y is a binary categorical variable with values 0 and 1 and th
ere are 25% of zeros and 75% of ones, stratify=y will make sure that your random split
 has 25% of 0's and 75% of 1's.

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(project_data, project_data['project
_is_approved'], test_size=0.33, stratify = project_data['project_is_approved'])

# now getting the crossvalidation data from our train data
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, strat
ify=y_train)
```

In [9]:

```
# Now we will be removing the column "project_is_approved" because that is the only one
which our model needs to predict

X_train.drop(['project_is_approved'], axis=1, inplace=True)
X_test.drop(['project_is_approved'], axis=1, inplace=True)
X_cv.drop(['project_is_approved'], axis=1, inplace=True)
```

In [ ]:

In [ ]:

In [ ]:

Now we will do all kind of preprocessing required for the train data ,test data,crossvalidation data separately

FOR TRAIN DATA

Preprocessing of `project_subject_categories'

In [10]:

```
catogories = list(X_train['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
ng
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
on
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace
 it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"M
ath & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
ces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

X_train['clean_categories'] = cat_list
X_train.drop(['project_subject_categories'], axis=1, inplace=True)

from collections import Counter
my_counter = Counter()
for word in X_train['clean_categories'].values:
    my_counter.update(word.split())

cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))
```

Preprocessing of project_subject_subcategories

In [11]:

```
sub_catogories = list(X_train['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
ng
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
on

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace
 it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"M
ath & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spa
ces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

X_train['clean_subcategories'] = sub_cat_list
X_train.drop(['project_subject_subcategories'], axis=1, inplace=True)

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
for word in X_train['clean_subcategories'].values:
    my_counter.update(word.split())

sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))
```

Text preprocessing

In [12]:

```
# merge two column text dataframe:
X_train["essay"] = X_train["project_essay_1"].map(str) +\
                        X_train["project_essay_2"].map(str) + \
                        X_train["project_essay_3"].map(str) + \
                        X_train["project_essay_4"].map(str)
```

In [13]:

```
X_train.head(2)
```

Out[13]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
8134	135883	p090948	ade6aad520456e6e23490918ef5ea2d4	Ms.	CA
32738	113417	p140603	e37e94882ace7b10b482892d78787fa4	Ms.	CA

In [14]:

```
# printing some random reviews
print(X_train['essay'].values[0])
print("="*50)
print(X_train['essay'].values[150])
print("="*50)
print(X_train['essay'].values[1000])
print("="*50)
print(X_train['essay'].values[20000])
print("="*50)
print(X_train['essay'].values[9999])
print("="*50)
```

As a teacher in a high-poverty school, I have the unique opportunity to wo
rk with a group of thirty-three students from varying backgrounds who all
contribute unique skills and experiences to our classroom.\r\n\r\nOur clas
s is made up of a multilingual and global community of learners, with over
half the class speaking a language other than English at home.\r\n\r\nThes
e bright, innovative, spunky students are enthusiastic readers, writers, m
athematicians, scientists and historians. Many students do not have access
to writing materials at home and would greatly benefit from having access
to a rich writing center from which they can easily build the worlds and i
deas that are just waiting to be put down on paper.Writing is a critical p
art of 5th grade. It is the beginning of the 5 paragraph essay, and my stu
dents will be doing a lot of work researching different topics and develop
ing their own persuasive essays. They will also have an opportunity to wri
te a memoir at the end of the school year to reflect on their elementary c
areer. \r\n\r\nWriting time is the best opportunity that my students have
to express themselves and their own beliefs. To do this they need a place
to put all of this writing, which is why I am requesting composition books
for all of my students. They also need an organized center where they can
find all of the tools for their writing - from paper trays to pencils bin
s. Your contributions will greatly influence my students experience with w
riting this year. To say thank you, every donor will receive a sample of s
tudent writing from our narrative unit!nannan
==
My students thrive on learning new things and sharing what they know! Our
school has over 70% of students who receive free or reduced lunch and we a
re a Title I school that has a high poverty and low income rate. My studen
ts come to school having to think about more than just learning and it is
my job to make sure that they feel loved and have an environment where the
re are no limits to their dreams! I have a wide variety of students in my
classroom from the language they speak to their learning abilities and dis
abilities! My children are independent, dreamers, helpers and learners.In
first grade, our main focus is to teach our students how to read. There ar
e so many apps that reinforce the reading skills I teach every day! I curr
ently have two iPads in my classroom. One was provided by our county and o
ne was funded through my last Donors Choose project. It is the highlight o
f their day if they get to work on the iPad that day. I have many struggli
ng readers in my class this year and they get excited to learn when they h
ave the opportunity to work in the iPad. If I had three iPads in my classr
oom, students would have double the opportunity to enhance their reading s
kills. Many of my students do not access to the Internet or any technology
at home. Having the opportunity to use technology in the classroom is bene
ficial to each child on so many levels. We use apps such as reading friend
zy, epic, abcmouse, abcya and more. I am able to tailor the needs to each
specific student on the iPad. Your generous donation to our project will i
mprove our First grade classroom by building stronger readers and learner
s.nannan
==
Fifth grade has always been my favorite grade. 5th graders personalities
are all very special and incredibly unique. Their energy is contagious!
They have a wide variety of hobbies, interests, hopes and dreams. Howeve
r, as different as my kids all are, they do have ONE thing in common....\r
\nThey all LOVE graphic novels! \r\nMy class is completely enthralled by
them! The novels fly off of the shelves! Discussions about the books are
vibrant and lively. \r\nRequests for new titles happen daily. \r\n\r\nMy
5th grades had these thoughts and opinions to share about their experience
s reading graphic novels. \r\n\"Graphic novels are the best books ever!\r
\n\"I like graphic novels because they are interesting and because they ar
e so adventurous. You never know what will happen next!\"\r\n\"They take
you to a whole other fantasy world.\"\r\n\"Graphic novels are funny and th
e pictures help to tell the story.\"\r\n\"They are easier to read and they
make me feel more confident.\"\r\n\"The drawings are amazing. The details

are great.\"\r\nBased on these glowing reviews, my curiosity was peaked. \r\nThe kids have converted me also! \r\n\r\n\r\nnannan
==========
My students are brand new students - kindergarten students with such an ea gerness to learn. We are located on a military installation and have stud ents from all over the country and/or world. My students are diverse and bright. Some of these students do not have access to technology at home a nd look forward to the time that is given in class to explore technology a nd all the fun (and academic) things that we can do with it. Our goal is not only to learn, but to have fun doing so.Some of my students do not hav e access to technology at home and we would like to enhance student learni ng by adding tablets to our classroom resources. With these tablets, the students will be able to access educational games that are paired with our curriculum and will expand their learning. Students will be able to pract ice using technology while listening to a book online, reading an e-book, playing math games, looking at pictures of famous presidents, and so much more. The possibilities are endless by adding technology to our classroom and that is what we strive to do - to aim for the stars and then some!nann an
==========
\"Good Morning Mrs. A\" is the greeting I hear every morning from my stude nts. Along with hugs and high fives, it is important they say hello becaus e it sets the tone for their day. By greeting me they know I am here for t hem. I am here to care for them, teach them, and guide them along the way. \r\n\r\nMy school is 95% free and reduced. My class is 95% African America n children. The students in my class often don't have their basic needs me t before they come to school. Clean socks, breakfast, and lotion are some of the things I take care of before we start our learning day. I thrive on seeing their smiling faces and the eagerness they have to learn.Imagine yo u are on a plane and passengers around you are watching a video or listeni ng to music without headphones. It causes you to be annoyed and distracte d. In my classroom each student has their own computer provided by the dis trict. With my school being very transit and low income, most students can not provide their own headphones. \r\n\r\nStudents in my classroom create their own personal learning path with their Chromebooks. With access to th eir own headphones they will be able to complete personal task efficientl y. They will not become confused or frustrated by hearing other students p rograms. They will be able to concentrate and focus on their own task. Thi s will ultimately give them a higher level of personal success.nannan
==========

In [15]:

```
# creating a function named as decontracted which does the job of decontraction

# 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 [16]:

```
sent = decontracted(X_train['essay'].values[20000])
print(sent)
print("="*50)
```

```
My students are brand new students - kindergarten students with such an ea
gerness to learn.  We are located on a military installation and have stud
ents from all over the country and/or world.  My students are diverse and
bright.  Some of these students do not have access to technology at home a
nd look forward to the time that is given in class to explore technology a
nd all the fun (and academic) things that we can do with it.  Our goal is
not only to learn, but to have fun doing so.Some of my students do not hav
e access to technology at home and we would like to enhance student learni
ng by adding tablets to our classroom resources.  With these tablets, the
students will be able to access educational games that are paired with our
curriculum and will expand their learning.  Students will be able to pract
ice using technology while listening to a book online, reading an e-book,
playing math games, looking at pictures of famous presidents, and so much
more.  The possibilities are endless by adding technology to our classroom
and that is what we strive to do - to aim for the stars and then some!nann
an
==================================================
```

In [17]:

```
#\r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-pyt
hon/
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
My students are brand new students - kindergarten students with such an ea
gerness to learn.  We are located on a military installation and have stud
ents from all over the country and/or world.  My students are diverse and
bright.  Some of these students do not have access to technology at home a
nd look forward to the time that is given in class to explore technology a
nd all the fun (and academic) things that we can do with it.  Our goal is
not only to learn, but to have fun doing so.Some of my students do not hav
e access to technology at home and we would like to enhance student learni
ng by adding tablets to our classroom resources.  With these tablets, the
students will be able to access educational games that are paired with our
curriculum and will expand their learning.  Students will be able to pract
ice using technology while listening to a book online, reading an e-book,
playing math games, looking at pictures of famous presidents, and so much
more.  The possibilities are endless by adding technology to our classroom
and that is what we strive to do - to aim for the stars and then some!nann
an
```

In [18]:

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

```
My students are brand new students kindergarten students with such an eage
rness to learn We are located on a military installation and have students
from all over the country and or world My students are diverse and bright
Some of these students do not have access to technology at home and look f
orward to the time that is given in class to explore technology and all th
e fun and academic things that we can do with it Our goal is not only to l
earn but to have fun doing so Some of my students do not have access to te
chnology at home and we would like to enhance student learning by adding t
ablets to our classroom resources With these tablets the students will be
able to access educational games that are paired with our curriculum and w
ill expand their learning Students will be able to practice using technolo
gy while listening to a book online reading an e book playing math games l
ooking at pictures of famous presidents and so much more The possibilities
are endless by adding technology to our classroom and that is what we stri
ve to do to aim for the stars and then some nannan
```

In [19]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
# because although they are in this list but they matter a lot because
# they change the meaning of the entire sentence.
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [20]:

```
# Combining all the above preprocessing techniques for all the project essays
from tqdm import tqdm
preprocessed_essays_Train = []
# tqdm is for printing the status bar
for sentance in tqdm(X_train['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_essays_Train.append(sent.lower().strip())
```

```
100%|██████████| 49041/49041 [00:21<00:00, 2256.99it/s]
```

In [21]:

```
# after preprocesing of project essays
preprocessed_essays_Train[20000]
```

Out[21]:

```
'students brand new students kindergarten students eagerness learn located
military installation students country world students diverse bright stude
nts not access technology home look forward time given class explore techn
ology fun academic things goal not learn fun students not access technolog
y home would like enhance student learning adding tablets classroom resour
ces tablets students able access educational games paired curriculum expan
d learning students able practice using technology listening book online r
eading e book playing math games looking pictures famous presidents much p
ossibilities endless adding technology classroom strive aim stars nannan'
```

Preprocessing of project_title

Now we will simply apply the above preprocessing steps on the project title for the train data as well,as it is also a text feature

In [22]:

```
# printing some random titles.
print(X_train['project_title'].values[0])
print("="*50)
print(X_train['project_title'].values[150])
print("="*50)
print(X_train['project_title'].values[1000])
print("="*50)
print(X_train['project_title'].values[20000])
print("="*50)
print(X_train['project_title'].values[9999])
print("="*50)
```

```
Help Build Our Writing Center
==================================================
Growing minds with an iPad
==================================================
Gotta Get MORE Graphic Novels!
==================================================
Taking Off with Technology!
==================================================
What Did It Say? Can You Hear It Now?
==================================================
```

We have already written the preprocessing codes for different preprocessing approaches now we will simply use those codes on the project titles

In [23]:

```
preprocessed_project_titles_Train = []

for t in tqdm(X_train["project_title"]):
    title = decontracted(t)
    title = title.replace('\\r', ' ')
    title = title.replace('\\"', ' ')
    title = title.replace('\\n', ' ')
    title = re.sub('[^A-Za-z0-9]+', ' ', title)
    title = ' '.join(f for f in title.split() if f not in stopwords)
    preprocessed_project_titles_Train.append(title.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████████
████| 49041/49041 [00:01<00:00, 47763.45it/s]
```

In [24]:

```
# printing some random titles of train dataset after preprocessing

print(preprocessed_project_titles_Train[5000])
print("="*50)
print(preprocessed_project_titles_Train[7000])
print("="*50)
print(preprocessed_project_titles_Train[10000])
print("="*50)
print(preprocessed_project_titles_Train[45000])
print("="*50)
print(preprocessed_project_titles_Train[22000])
print("="*50)
```

```
force scooter motion
==================================================
stem kindergarten
==================================================
wiggling kindergarten
==================================================
new space
==================================================
help us fight childhood obesity hop fitness
==================================================
```

Test Data

Preprocessing of project_subject_categories

In [25]:

```
catogories = list(X_test['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
ng
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
on
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace
 it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"M
ath & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
ces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

X_test['clean_categories'] = cat_list
X_test.drop(['project_subject_categories'], axis=1, inplace=True)

from collections import Counter
my_counter = Counter()
for word in X_test['clean_categories'].values:
    my_counter.update(word.split())

cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))
```

Preprocessing of project_subject_subcategories

In [26]:

```
sub_catogories = list(X_test['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
ng
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
on

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace
 it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"M
ath & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spa
ces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

X_test['clean_subcategories'] = sub_cat_list
X_test.drop(['project_subject_subcategories'], axis=1, inplace=True)

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
for word in X_test['clean_subcategories'].values:
    my_counter.update(word.split())

sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))
```

Text Preprocessing

In [27]:

```
# merge two column text dataframe:
X_test["essay"] = X_test["project_essay_1"].map(str) +\
                        X_test["project_essay_2"].map(str) + \
                        X_test["project_essay_3"].map(str) + \
                        X_test["project_essay_4"].map(str)
```

In [28]:

```
X_test.head(5)
```

Out[28]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
36675	158529	p012531	d5418aad0ebd66f05263f9c42773b003	Mrs.	CA
88995	104451	p013643	72cd1d32d22b1d1e98b839e20aff93fb	Mr.	IL
71367	26079	p205185	eef03ef3528ea1ab9fc3b0d7c54c464a	Ms.	MI
53982	110544	p191726	52a0ec26e1010b35fa234518653328fb	Mrs.	AL
108487	128586	p053525	c92897ab11467e64f4ff6dd95897a46e	Ms.	NV

In [29]:

```
# printing some random reviews
print(X_test['essay'].values[0])
print("="*50)
print(X_test['essay'].values[150])
print("="*50)
print(X_test['essay'].values[1000])
print("="*50)
print(X_test['essay'].values[20000])
print("="*50)
print(X_test['essay'].values[9999])
print("="*50)
```

\"Look what I did on my own!\" Those are the magic words you love to hear from a 6 year old just learning to read and write. They take such pride an d joy in showing you what they can do with their own 2 hands. \r\n\r\nWhen learning becomes fun it is no longer at task- it is an accomplishment. My students are 6 and 7 years old. They love to touch, move, and interact wit h everything around them. They have a deep desire to be successful at what ever they attempt. While I have many students coming to me with a strong f oundation for reading and writing, I have many more that come to me strugg ling to do either. This huge gap between learning levels creates an extra layer of difficulty for me as a educator. I am constantly asking myself, \"How do I keep everyone challenged and moving forward when there are so m any different ability levels in one classroom?\" This challenge is why I t each and why I am trying to find ways to reach all of my learners.I am req uesting Boogie Boards, interactive Ipad letters, and Versatile kits to hel p my students learn to read, write, and build words with their hands. Thes e items will be placed in our word word and/or writing station during our language arts block. Students will have the freedom to choose the manner i n which they will practice their words and writing. These tools also allow me to differentiate for my students. The same tool can be used to practice words and skills at a variety of levels, helping me to reach all my unique learners at the same time.\r\n\r\nMy ultimate goal is to create a classroo m environment where my students become passionate and excited about learni ng. I have found that my first graders learn best when they are physicall y interacting with the material. A 6-year old desperately needs to love th e learning process. They are playful, curious, and love getting their hand s on anything that seems like a game. Unfortunately, using paper and penci l to learn often becomes more about the task, and less about the learning process. I believe that these tools will help me to improve the motivation and joy of learning in my classroom.\r\n\r\n\r\n\r\nnannan

==========

My first grade classroom is home to students who are enthusiastic and exci ted to learn. Our school is rich in diversity where over 75% of the studen ts are English Language Learners. The students have backgrounds coming fro m Guatemala, Mexico and West Africa. Our school is also poverty stricken w here more than 95% of our students are on free and reduced price lunch.\r \n\r\nMy goal is to provide my students with engaging, creative and meanin gful learning experiences so they can grow into confident, respectful and literate young people. We are a Title 1 school which means funds and resou rces are extremely limited at school and in my students' homes. Many of my students are not able to afford to buy crayons, markers or even glue in th eir homes. The opportunities I provide for them in the classroom are that much more meaningful and important to their overall growth. \r\nI am askin g for these materials to help increase and foster my students’ creativity and imagination. Their families are not able to afford crayons or markers so the use and exploration of art is very limited without these materials. The supplies I am requesting will help my students stay focused on the lea rning while having fun and engaging experiences.\r\n\r\nColoring improves students’ motors skills, stimulates creatively and provides stress relief. My students love to create and learn better through hands on activities. C rayons and markers will be used daily to add brightly colored illustration s to our writing and help us to work on our fine motor skills. We will use the dry erase markers to work on letter and number formation. We will also use the markers to work on writing our phonics sounds. \r\n\r\nIt is also important for my students to practice organization and responsibly for the ir materials. I am requesting small containers so the students will have a place to store their crayons. They will be able to keep their containers a t their seats allowing them to have their crayons within arms reach. \r\n \r\nThank you for taking the time to consider my project worth your donati on. My students will benefit from your generosity! \r\nnannan

==========

I teach an amazing group of 3rd graders at a high poverty inner city publi

c school in Indianapolis. \r\n\r\nAll of our students are on the free brea kfast and lunch program. Many of our students' parents work odd hours and they barely get to spend time with their children. I have some students be ing raised by a single parent or a grandparent. No matter what my students are going through at home, they always try their personal best in the clas sroom. My students know how high my expectations are for each one of them! My students work very hard in the classroom to earn awards. They earn awar ds for gaining fluency on Reflex Math and they also earn awards for growin g on Pivot and Raps180, both district mandated assessments. They have work ed so hard and deserve to get awards to show their hard work in beautiful color! I would also like to have color ink to print strategy posters and t ask cards for my students to use in the classroom. They are always referri ng to posters to help guide them on work. My students also love working in small groups during reading and math and need some new task cards! They al so need laminating pouches to keep their awards and task cards safe!\r\nHa ving colorful, vibrant materials for my classroom will help encourage all of my eager students to work on their tasks and enjoy their surrounding s.\r\nI will be able to readily create and print materials that will be us ed for this year and future ones as well.nannan

==========

My students are three, four and five year olds that attend either my a.m. class or my p.m. pre-kindergarten class. My students are considered at-ri sk in order to qualify for the pre-kindergarten program at my school. Man y of my students receive speech services. My students are bright, energet ic, eager to learn children who I know will go far with the right start.\r \n\r\nI teach at a title I school in the suburbs of Chicago. My district provides free breakfast and lunch to all students.My students enjoy listen ing to stories. Everyday I read at least two books to my students. This component to my listening station will enable my students to listen to eve n more books daily. My students will learn to independently listen to sto ries. Students can listen to a story alone or with a partner. They will learn the skills to follow along in the book, using audio and visual cues as to when to turn the pages.\r\n\r\nEarly exposure to reading can create a life long love of books! \r\nThis reading center will help do that. I h ave numerous books that are age appropriate that will foster independent r eading.nannan

==========

My co-teacher and I have 48 amazing first graders who are curious, eager a nd excited to be at school. Our classrooms are full of life, love, and la ughter. \r\nOur students love to collaborate and work together to solve p roblems and we are committed to providing a flexible and engaging environm ent that fits their energy level. As most first graders, our students lov e to move and are full of energy. We want to preserve the energy and wond er in our students so they will continue to be active learners.Do you reme mber your first grade classroom? If you were to walk into a first grade cl assroom today, it is likely you would see the same, or similar environmen t. \r\nWe want more for our students; we want to provide our students wit h an active learning environment in which they can fidget when needed and showcase their learning using dry erase tables.\r\nIn an article for The E ducation Facilities Clearinghouse, author Greg Smith explains, “American s chools, designed around a standard learning environment that supports a le cture style of teaching, have remained relatively unchanged for the last 5 0 years while we have evolved into a society of visual and tactile learner s. So what defines these 21st century learning environments? The words we most often used are flexible, agile, and adaptable, words that ultimately mean being able to adjust to new conditions, modify for a new use or purpo se, and allow flexibility to engage students in a variety of ways.” \r\nW e hope to give our students a learning environment such as the one Smith i llustrated in his article. In order to achieve this environment, we need t o provide flexible seating options.\r\nThis grant will provide dry erase f ilm for tables and alternative seating for two first grade classes. We en

```
vision students choosing how and where to learn to meet their individual n
eeds. For example, the wobble stools are designed to allow students who ne
ed tactile stimuli the freedom to move, so they can focus on learning. Oth
er options include standing, laying on mats, and comfortable couches.\r\nA
s teachers in the 21st century, we have the responsibility to teach and en
courage collaboration as we meet students' sensory needs. The dry erase su
rfaces and mobile furniture will provide unlimited opportunities for stude
nts to collaborate with others.nannan
==================================================
```

In [30]:

```
# creating a function named as decontracted which does the job of decontraction

# 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 [31]:

```
sent = decontracted(X_test['essay'].values[20000])
print(sent)
print("="*50)
```

```
My students are three, four and five year olds that attend either my a.m.
class or my p.m. pre-kindergarten class.  My students are considered at-ri
sk in order to qualify for the pre-kindergarten program at my school.  Man
y of my students receive speech services.  My students are bright, energet
ic, eager to learn children who I know will go far with the right start.\r
\n\r\nI teach at a title I school in the suburbs of Chicago.  My district
provides free breakfast and lunch to all students.My students enjoy listen
ing to stories.  Everyday I read at least two books to my students.  This
component to my listening station will enable my students to listen to eve
n more books daily.  My students will learn to independently listen to sto
ries.  Students can listen to a story alone or with a partner.  They will
learn the skills to follow along in the book, using audio and visual cues
as to when to turn the pages.\r\n\r\nEarly exposure to  reading can create
a life long love of books!  \r\nThis reading center will help do that. I h
ave numerous books that are age appropriate that will foster independent r
eading.nannan
==================================================
```

In [32]:

```
#\r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-pyt
hon/
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
My students are three, four and five year olds that attend either my a.m.
class or my p.m. pre-kindergarten class.  My students are considered at-ri
sk in order to qualify for the pre-kindergarten program at my school.  Man
y of my students receive speech services.  My students are bright, energet
ic, eager to learn children who I know will go far with the right start.
I teach at a title I school in the suburbs of Chicago.  My district provid
es free breakfast and lunch to all students.My students enjoy listening to
stories.  Everyday I read at least two books to my students.  This compone
nt to my listening station will enable my students to listen to even more
books daily.  My students will learn to independently listen to stories.
Students can listen to a story alone or with a partner.  They will learn t
he skills to follow along in the book, using audio and visual cues as to w
hen to turn the pages.    Early exposure to  reading can create a life lon
g love of books!    This reading center will help do that. I have numerous
books that are age appropriate that will foster independent reading.nannan
```

In [33]:

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

```
My students are three four and five year olds that attend either my a m cl
ass or my p m pre kindergarten class My students are considered at risk in
order to qualify for the pre kindergarten program at my school Many of my
students receive speech services My students are bright energetic eager to
learn children who I know will go far with the right start I teach at a ti
tle I school in the suburbs of Chicago My district provides free breakfast
and lunch to all students My students enjoy listening to stories Everyday
I read at least two books to my students This component to my listening st
ation will enable my students to listen to even more books daily My studen
ts will learn to independently listen to stories Students can listen to a
story alone or with a partner They will learn the skills to follow along i
n the book using audio and visual cues as to when to turn the pages Early
exposure to reading can create a life long love of books This reading cent
er will help do that I have numerous books that are age appropriate that w
ill foster independent reading nannan
```

In [34]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
# because although they are in this list but they matter a lot because
# they change the meaning of the entire sentence.
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [35]:

```
# Combining all the above preprocessing techniques for all the project essays
from tqdm import tqdm
preprocessed_essays_Test = []
# tqdm is for printing the status bar
for sentance in tqdm(X_test['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_essays_Test.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████████
█████| 36052/36052 [00:15<00:00, 2289.59it/s]
```

In [36]:

```
# after preprocesing of project essays
preprocessed_essays_Test[20000]
```

Out[36]:

```
'students three four five year olds attend either class p pre kindergarten
class students considered risk order qualify pre kindergarten program scho
ol many students receive speech services students bright energetic eager l
earn children know go far right start teach title school suburbs chicago d
istrict provides free breakfast lunch students students enjoy listening st
ories everyday read least two books students component listening station e
nable students listen even books daily students learn independently listen
stories students listen story alone partner learn skills follow along book
using audio visual cues turn pages early exposure reading create life long
love books reading center help numerous books age appropriate foster indep
endent reading nannan'
```

Preprocessing of project_title

Now we will simply apply the above preprocessing steps on the project title for the test data as well,as it is also a text feature

In [37]:

```
# printing some random titles.
print(X_test['project_title'].values[0])
print("="*50)
print(X_test['project_title'].values[150])
print("="*50)
print(X_test['project_title'].values[1000])
print("="*50)
print(X_test['project_title'].values[20000])
print("="*50)
print(X_test['project_title'].values[9999])
print("="*50)
```

```
Putting Our Hands-On Learning
==================================================
Let's Get Coloring!
==================================================
Brighten My Student's School Year! Part 3
==================================================
Independent Reading In Pre-Kindergarten!
==================================================
Flexible in First
==================================================
```

We have already written the preprocessing codes for different preprocessing approaches now we will simply use those codes on the project titles

In [38]:

```
preprocessed_project_titles_Test = []

for t in tqdm(X_test["project_title"]):
    title = decontracted(t)
    title = title.replace('\\r', ' ')
    title = title.replace('\\"', ' ')
    title = title.replace('\\n', ' ')
    title = re.sub('[^A-Za-z0-9]+', ' ', title)
    title = ' '.join(f for f in title.split() if f not in stopwords)
    preprocessed_project_titles_Test.append(title.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████████
████| 36052/36052 [00:00<00:00, 46547.36it/s]
```

In [39]:

```
# printing some random titles of train dataset after preprocessing

print(preprocessed_project_titles_Test[5000])
print("="*50)
print(preprocessed_project_titles_Test[7000])
print("="*50)
print(preprocessed_project_titles_Test[10000])
print("="*50)
print(preprocessed_project_titles_Test[4500])
print("="*50)
print(preprocessed_project_titles_Test[22000])
print("="*50)
```

```
turtle tank enrichment
==================================================
our museum live online too
==================================================
chapter one can you read us
==================================================
chromebook tutoring station
==================================================
horseshoe table
==================================================
```

Cross validation data

Preprocessing of project_subject_categories

In [40]:

```
catogories = list(X_cv['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
ng
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
on
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace
 it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"M
ath & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
ces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

X_cv['clean_categories'] = cat_list
X_cv.drop(['project_subject_categories'], axis=1, inplace=True)

from collections import Counter
my_counter = Counter()
for word in X_test['clean_categories'].values:
    my_counter.update(word.split())

cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))
```

Preprocessing of project_subject_subcategories

In [41]:

```
sub_catogories = list(X_cv['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
ng
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
on

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace
 it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"M
ath & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spa
ces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

X_cv['clean_subcategories'] = sub_cat_list
X_cv.drop(['project_subject_subcategories'], axis=1, inplace=True)

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
for word in X_cv['clean_subcategories'].values:
    my_counter.update(word.split())

sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))
```

Text Preprocessing

In [42]:

```
# merge two column text dataframe:
X_cv["essay"] = X_cv["project_essay_1"].map(str) +\
                        X_cv["project_essay_2"].map(str) + \
                        X_cv["project_essay_3"].map(str) + \
                        X_cv["project_essay_4"].map(str)
```

In [43]:

```
X_cv.head(5)
```

Out[43]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
83534	147714	p027347	12735fdf8d2d34bc18aed105a86f25f2	Mrs.	WA
74468	87925	p138058	6c80fb44be75778c35db983dd5adf707	Mrs.	NY
47678	31762	p139835	617af30ebdd0b231867955f6ee16d793	Mrs.	GA
16522	133707	p032906	2a518c2b4a6576c3b1367184c649d120	Ms.	CA
71102	69968	p176570	941044a67fae3ddda70fa20234f00e93	Mrs.	NY

In [44]:

```
# printing some random reviews
print(X_cv['essay'].values[0])
print("="*50)
print(X_cv['essay'].values[150])
print("="*50)
print(X_cv['essay'].values[1000])
print("="*50)
print(X_cv['essay'].values[20000])
print("="*50)
print(X_cv['essay'].values[9999])
print("="*50)
```

This is a highly diverse school, over 50% are African American, 5% white, and the remaining 45% are huge mix. This school is situated in one of the most linguistically diverse neighborhoods in the United States, and our sc hool is proof of that. The majority of our students are on the free and re duced lunch program, and many of our families have several adults working in the home (including many of our students). \r\n\r\nWhen describing our school, students and staff alike would say that we are a big family. We lo ve each other and take care of each other. The staff at our school are, in my opinion, the best in the district, we care about our students and we pu t them first over everything. Also in my opinion, the students here are th e BEST young adults in the entire universe. Although we have a strong acad emic program and a strong basketball program at our school, our arts progr am has a long way to go. We have some huge strides to make up in terms of equitable resources and it is my goal to grow the arts department and show case the enormous talent that fills our hallways.For the last decade, stud ents have mostly been exposed to painting with watercolor, drawing with pe ncil and marker and paper mache sculpture. I want to expose my students to even more exciting and awesome art materials. I plan to show them The Grea t Wave by Hokusai and have them experiment with woodblock printing with a modern twist. I want to show them how to play with batiking and tie dying Tshirts and fabrics. I want my students to learn to discover their identit ies though self portraits with paint markers and paper cutting. I'd like m y students to have real, lasting, memorable experiences in my classroom. I hope to inspire them to love art, to be amazed at what they can create, an d to provide them with the opportunity to dive head-first into the world o f fine art. I believe a good teacher not only sparks a fire in each studen t, but helps to keep that fire burning by showing students how to see the world in a new way. I hope that my lessons and the experiences my students gain from the projects in my room will inspire them to create art for the rest of their lives.nannan

==========

First grade is truly the age of discovery as school becomes less of a daun ting new challenge and more of a exploration into how much students can le arn. Our first grade classroom is full of students who are eager to learn. As a classroom that has close to 50% of students who use mostly Spanish (o r other languages!) at home, we will be working hard to read and write all while trying to stay true to the culture that makes their families proud. \r\n\r\nWe are a class \"on the move\" learning though embodied cognition and trying to stay as active as possible. We strive to incorporate movemen t into every lesson or routine inside of our classroom, because as our bod ies are moving, our brains are expanding. We are working hard and playing hard all school year long!Even in first grade, we are preparing students t o someday become independent citizens and healthy, enriching members of so ciety. In today's global and technologically-advancing world, even first g raders need to learn how to be productive, safe, and creative on the compu ter. Unfortunately, the need for greater access to technology isn't always met with the increase in school resources and funding. This project will a llow my students more time to use computers while not straining my schoo l's current technology. \r\nThese chromebooks will allow my students grea ter access to the internet, to the apps that help them learn, and to the w orld of knowledge and wonder that lies just beyond the desktop screen. St udents will no longer get frustrated by slow or out-dated technology, but instead will come into the class excited to use chromebooks that are new, up-to-date, and efficient.nannan

==========

Our Transitional Kindergarten Classroom has a morning class and an afterno on class. We have 21 students in each class, so we have 42 students sharin g our classroom and materials each day!\r\n\r\nOur elementary school is a Title 1 school where many of our students receive free or reduced lunch. \r\n\r\nMy students come from very diverse backgrounds as well as many stu dents in my class are English Language Learners. \r\n\r\nMy students are n

ow recognizing sight words in books and shout out to each other when they see something they know!We are in search of replacing our library foam mat with a fire resistant rug! Our foam mat did not pass fire code in the rece nt Fire Marshall visit. \r\n\r\nWith 42 students sharing our classroom and materials each day in our Transitional Kindergarten Classroom, our materia ls and our environment get lots of love! Other than our main letter rug fo r circle time, our entire classroom is covered with cold tile!\r\n\r\nIn o ur classroom library our students retell stories and songs for literacy ce nters, listen to stories with our listening center station, sit in our lib rary during projector screen time, present puppet shows together, create f elt board stories, participate in small group reading time with Ms. G-W or Mrs. Mew, and of course they enjoy getting cozy to read books of their cho ice during Free Choice time. \r\n\r\nPlease help us make our library a coz y and inviting place to be and read!\r\n\r\nThank you!\r\nMs. G-W, Mrs. Me w, AM TK and PM TKnannan

==========

I work for a Title I school with students from diverse backgrounds. We fac e many financial challenges as a Title I School, and we value every resour ce that we have. I do my best to challenge each and every one of my studen ts, because I have students with various academic considerations. Therefor e, I work diligently to differentiate and individualize instruction, so th at everyone can learn to the best of their ability. \r\n\r\nMy goal is for every student to grow and succeed - academically and socially - and come t o school with a curiosity for learning. My students are dynamic and sweet. Our school is community-oriented and supportive. We value partnerships wit h families, and we look for ways to help our students grow. Our school has created strong partnerships in the community this year to build conscious, young learners. My students value giving back to others in return. I love to create curious, confident learners.Technology integration is one of my greatest passions in life...next to teaching that is. Research shows that students learn best and actually retain more information when technology i s incorporated into their every day learning. In my classroom, I do a lot of small group instruction as well as instructional videos and digital por tfolios. As a completely portable learning tool, the iPad allows document ation and student sharing to be taken to a whole different level. The ipa d will also be linked up to our classroom projector to showcase student wo rk during our share out time as well as being hooked up to a TV near the s mall group table for quick and easy access to information during reading, writing, and math groups. This makes it even easier to cater my instructi on to individual students' needs.nannan

==========

My class is generally made up of overly \"active\" students, mostly boys. At this age, movement is very important. I encourage them to learn throug h song, dance, and movement. I am hoping to further their ability to have the freedom to move/learn through flexible seating. There is an abundance of research that shows students who are allowed to sit in various types of seating (with the ability to move and not be restrained to traditional sea ting styles), have increased ability to be academically successful. \r\n \r\nI hope to remove most of my traditional desk/chair type seating from m y classroom and replace them with rocker chairs, bean bag seats, bench sea ts, stability balls, wobble chairs, and floor pillows. \r\n\r\nReplacing traditional desk/chairs with flexible seating will allow my students to mo ve while learning. This type of environment allows my \"active\" students the freedom to move while still working.My class is generally made up of o verly \"active\" students, mostly boys. At this age, movement is very imp ortant. I encourage them to learn through song, dance, and movement. I a m hoping to further their ability to have the freedom to move/learn throug h flexible seating. There is an abundance of research that shows students who are allowed to sit in various types of seating (with the ability to mo ve and not be restrained to traditional seating styles), have increased ab ility to be academically successful. \r\n\r\nI hope to remove most of my

```
traditional desk/chair type seating from my classroom and replace them wit
h rocker chairs, bean bag seats, bench seats, stability balls, wobble chai
rs, and floor pillows.  \r\n\r\nReplacing traditional desk/chairs with fle
xible seating will allow my students to move while learning.  This type of
environment allows my \"active\" students the freedom to move while still
working.nannan
==================================================
```

In [45]:

```
# creating a function named as decontracted which does the job of decontraction

# 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 [46]:

```
sent = decontracted(X_cv['essay'].values[20000])
print(sent)
print("="*50)
```

```
I work for a Title I school with students from diverse backgrounds. We fac
e many financial challenges as a Title I School, and we value every resour
ce that we have. I do my best to challenge each and every one of my studen
ts, because I have students with various academic considerations. Therefor
e, I work diligently to differentiate and individualize instruction, so th
at everyone can learn to the best of their ability. \r\n\r\nMy goal is for
every student to grow and succeed - academically and socially - and come t
o school with a curiosity for learning. My students are dynamic and sweet.
Our school is community-oriented and supportive. We value partnerships wit
h families, and we look for ways to help our students grow. Our school has
created strong partnerships in the community this year to build conscious,
young learners. My students value giving back to others in return. I love
to create curious, confident learners.Technology integration is one of my
greatest passions in life...next to teaching that is.  Research shows that
students learn best and actually retain more information when technology i
s incorporated into their every day learning.  In my classroom, I do a lot
of small group instruction as well as instructional videos and digital por
tfolios.  As a completely portable learning tool, the iPad allows document
ation and student sharing to be taken to a whole different level.  The ipa
d will also be linked up to our classroom projector to showcase student wo
rk during our share out time as well as being hooked up to a TV near the s
mall group table for quick and easy access to information during reading,
writing, and math groups.  This makes it even easier to cater my instructi
on to individual students' needs.nannan
==================================================
```

In [47]:

```
#\r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-pyt
hon/
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
I work for a Title I school with students from diverse backgrounds. We fac
e many financial challenges as a Title I School, and we value every resour
ce that we have. I do my best to challenge each and every one of my studen
ts, because I have students with various academic considerations. Therefor
e, I work diligently to differentiate and individualize instruction, so th
at everyone can learn to the best of their ability.     My goal is for eve
ry student to grow and succeed - academically and socially - and come to s
chool with a curiosity for learning. My students are dynamic and sweet. Ou
r school is community-oriented and supportive. We value partnerships with
families, and we look for ways to help our students grow. Our school has c
reated strong partnerships in the community this year to build conscious,
young learners. My students value giving back to others in return. I love
to create curious, confident learners.Technology integration is one of my
greatest passions in life...next to teaching that is.  Research shows that
students learn best and actually retain more information when technology i
s incorporated into their every day learning.  In my classroom, I do a lot
of small group instruction as well as instructional videos and digital por
tfolios.  As a completely portable learning tool, the iPad allows document
ation and student sharing to be taken to a whole different level.  The ipa
d will also be linked up to our classroom projector to showcase student wo
rk during our share out time as well as being hooked up to a TV near the s
mall group table for quick and easy access to information during reading,
writing, and math groups.  This makes it even easier to cater my instructi
on to individual students' needs.nannan
```

In [48]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
# because although they are in this list but they matter a lot because
# they change the meaning of the entire sentence.
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [49]:

```
# Combining all the above preprocessing techniques for all the project essays
from tqdm import tqdm
preprocessed_essays_Cv = []
# tqdm is for printing the status bar
for sentance in tqdm(X_cv['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_essays_Cv.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████████
█████| 24155/24155 [00:10<00:00, 2218.92it/s]
```

In [50]:

```
# after preprocesing of project essays
preprocessed_essays_Test[20000]
```

Out[50]:

```
'students three four five year olds attend either class p pre kindergarten
class students considered risk order qualify pre kindergarten program scho
ol many students receive speech services students bright energetic eager l
earn children know go far right start teach title school suburbs chicago d
istrict provides free breakfast lunch students students enjoy listening st
ories everyday read least two books students component listening station e
nable students listen even books daily students learn independently listen
stories students listen story alone partner learn skills follow along book
using audio visual cues turn pages early exposure reading create life long
love books reading center help numerous books age appropriate foster indep
endent reading nannan'
```

Preprocessing of project_title

Now we will simply apply the above preprocessing steps on the project title for the Cross Validation data as well,as it is also a text feature

In [51]:

```
# printing some random titles.
print(X_cv['project_title'].values[0])
print("="*50)
print(X_cv['project_title'].values[150])
print("="*50)
print(X_cv['project_title'].values[1000])
print("="*50)
print(X_cv['project_title'].values[20000])
print("="*50)
print(X_cv['project_title'].values[9999])
print("="*50)
```

```
Help Fund Memorable Art Projects!
==================================================
Crazy for Chromebooks
==================================================
In Need of a Cozy Place to BE and READ!
==================================================
21st Century Guided Math and Literacy Rotations
==================================================
Creating a friendly learning environment for active learners....
==================================================
```

In [52]:

```
preprocessed_project_titles_Cv = []

for t in tqdm(X_cv["project_title"]):
    title = decontracted(t)
    title = title.replace('\\r', ' ')
    title = title.replace('\\"', ' ')
    title = title.replace('\\n', ' ')
    title = re.sub('[^A-Za-z0-9]+', ' ', title)
    title = ' '.join(f for f in title.split() if f not in stopwords)
    preprocessed_project_titles_Cv.append(title.lower().strip())
```

100%|██| 24155/24155 [00:00<00:00, 47119.82it/s]

In [53]:

```
# printing some random titles of crossvalidation dataset after preprocessing

print(preprocessed_project_titles_Cv[5000])
print("="*50)
print(preprocessed_project_titles_Cv[7000])
print("="*50)
print(preprocessed_project_titles_Cv[10000])
print("="*50)
print(preprocessed_project_titles_Cv[4500])
print("="*50)
print(preprocessed_project_titles_Cv[22000])
print("="*50)
```

```
stepping up our game
==================================================
taking flight
==================================================
steaming up
==================================================
hats off financial literacy
==================================================
all about that bass
==================================================
```

Preparing Data For Models

In [54]:

```
project_data.columns
```

Out[54]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', 'project_grade_category', 'project_subject_categories',
       'project_subject_subcategories', 'project_title', 'project_essay_
1',
       'project_essay_2', 'project_essay_3', 'project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approve
d'],
      dtype='object')
```

we are going to consider

- school_state : categorical data
- clean_categories : categorical data
- clean_subcategories : categorical data
- project_grade_category : categorical data
- teacher_prefix : categorical data
- project_title : text data
- text : text data
- project_resource_summary: text data (optinal)
- quantity : numerical (optinal)
- teacher_number_of_previously_posted_projects : numerical
- price : numerical

Now firstly we will be vectorizing the categorical data

For vectorizing the categorical data we will be using One Hot Encoding Technique

One Hot Encoding Of Project Clean Categories

In [55]:

```
# we use count vectorizer to convert the values into one hot encoded features


from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False,
binary=True)

# we will be using the X_train for fitting our model because that is the only data a us
er knows rest all are for testing purposes
vectorizer.fit(X_train['clean_categories'].values)

print(vectorizer.get_feature_names())

categories_one_hot_train = vectorizer.transform(X_train['clean_categories'].values)

categories_one_hot_test = vectorizer.transform(X_test['clean_categories'].values)

categories_one_hot_cv = vectorizer.transform(X_cv['clean_categories'].values)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearnin
g', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
```

In [56]:

```
print("Shape of  Train data matrix after one hot encoding ",categories_one_hot_train.sh
ape)
print("Shape of  Test data matrix after one hot encoding ",categories_one_hot_test.shap
e)
print("Shape of  CV data matrix after one hot encoding ",categories_one_hot_cv.shape)
```

```
Shape of  Train data matrix after one hot encoding  (49041, 9)
Shape of  Test data matrix after one hot encoding  (36052, 9)
Shape of  CV data matrix after one hot encoding  (24155, 9)
```

One Hot Encoding Of Cleaned Project Sub Category

In [57]:

```
# we use count vectorizer to convert the values into one hot encoded features


from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=Fal
se, binary=True)

# we will be using the X_train for fitting our model because that is the only data a us
er knows rest all are for testing purposes
vectorizer.fit(X_train['clean_subcategories'].values)

print(vectorizer.get_feature_names())

subcategories_one_hot_train = vectorizer.transform(X_train['clean_subcategories'].value
s)

subcategories_one_hot_test = vectorizer.transform(X_test['clean_subcategories'].values)

subcategories_one_hot_cv = vectorizer.transform(X_cv['clean_subcategories'].values)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'Extracurricular',
'ParentInvolvement', 'Civics_Government', 'ForeignLanguages', 'NutritionEd
ucation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'Ch
aracterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music',
'History_Geography', 'EarlyDevelopment', 'Health_LifeScience', 'ESL', 'Gym
_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'Appli
edSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literac
y']
```

In [58]:

```
print("Shape of  Train data matrix after one hot encoding ",subcategories_one_hot_train
.shape)
print("Shape of  Test data matrix after one hot encoding ",subcategories_one_hot_test.s
hape)
print("Shape of  CV data matrix after one hot encoding ",subcategories_one_hot_cv.shape
)
```

```
Shape of  Train data matrix after one hot encoding  (49041, 30)
Shape of  Test data matrix after one hot encoding  (36052, 30)
Shape of  CV data matrix after one hot encoding  (24155, 30)
```

One hot encoding of teacher prefix

In [59]:

```
mylist_teacher_prefix = list(X_train['teacher_prefix'])
```

In [60]:

```
# We are removing the duplicate values from our list of the teacher prefix

# Source :- https://www.w3schools.com/python/python_howto_remove_duplicates.asp

mylist_teacher_prefix_actual_Train = list(dict.fromkeys(mylist_teacher_prefix))
```

In [61]:

```
# removing the nan from the teacher prefix category as there is no such category of tea
cher which exists

mylist_teacher_prefix_actual_Train = [p for p in mylist_teacher_prefix_actual_Train if
str(p) != 'nan']
mylist_teacher_prefix_actual_Train
```

Out[61]:

```
['Ms.', 'Mrs.', 'Mr.', 'Teacher', 'Dr.']
```

In [62]:

```
# we use count vectorizer to convert the values into one hot encoded features

# now we are working on teacher prefix data

from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=mylist_teacher_prefix_actual_Train, lowercase=F
alse, binary=True)

# I was getting an error like "np.nan is an invalid document, expected byte or unicode
 string."
# below is the solution

# https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-valueerr
or-np-nan-is-an-invalid-document


vectorizer.fit(X_train['teacher_prefix'].values.astype('U'))
print(vectorizer.get_feature_names())

teacher_prefix_one_hot_train = vectorizer.transform(X_train['teacher_prefix'].values.as
type('U'))
teacher_prefix_one_hot_test = vectorizer.transform(X_test['teacher_prefix'].values.asty
pe('U'))
teacher_prefix_one_hot_cv = vectorizer.transform(X_cv['teacher_prefix'].values.astype(
'U'))
```

```
['Ms.', 'Mrs.', 'Mr.', 'Teacher', 'Dr.']
```

In [63]:

```
print("Shape of matrix of Train data after one hot encoding ",teacher_prefix_one_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",teacher_prefix_one_hot_test.shape)
print("Shape of matrix of Cross Validation data after one hot encoding ",teacher_prefix_one_hot_cv.shape)
```

```
Shape of matrix of Train data after one hot encoding  (49041, 5)
Shape of matrix of Test data after one hot encoding  (36052, 5)
Shape of matrix of Cross Validation data after one hot encoding  (24155,
5)
```

One Hot encoding of project grade category

In [64]:

```
mylist_project_grade_category = list(X_train['project_grade_category'])
```

In [65]:

```
# We are removing the duplicate values from our list of the project grade category

# Source :- https://www.w3schools.com/python/python_howto_remove_duplicates.asp

mylist_project_grade_category_actual = list(dict.fromkeys(mylist_project_grade_category))
```

In [66]:

```
type(mylist_project_grade_category_actual)
print(mylist_project_grade_category_actual[0])

n = len(mylist_project_grade_category_actual)
print(n)

# I already saw by running the code that the word Grades is unnecessarily present in the elements of list hence trying to remove that word
# how to remove a word from a sentence --> https://codescracker.com/python/program/python-program-remove-word-from-sentence.htm
for m in range(0,4,1):
    words = mylist_project_grade_category_actual[m].split()
    mylist_project_grade_category_actual[m] = ''.join([j for j in words if j not in "Grades"])
print(mylist_project_grade_category_actual)
```

```
Grades 3-5
4
['3-5', 'PreK-2', '6-8', '9-12']
```

In [67]:

```
# we use count vectorizer to convert the values into one hot encoded features

# now we are working on project grade category data

from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=mylist_project_grade_category_actual, lowercase
=False, binary=True)


vectorizer.fit(X_train['project_grade_category'].values)
print(vectorizer.get_feature_names())

project_grade_categories_one_hot_train = vectorizer.transform(X_train['project_grade_ca
tegory'].values)
project_grade_categories_one_hot_test = vectorizer.transform(X_test['project_grade_cate
gory'].values)
project_grade_categories_one_hot_cv = vectorizer.transform(X_cv['project_grade_categor
y'].values)
```

```
['3-5', 'PreK-2', '6-8', '9-12']
```

In [68]:

```
print("Shape of matrix of Train data after one hot encoding ",project_grade_categories_
one_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",project_grade_categories_o
ne_hot_test.shape)
print("Shape of matrix of Cross Validation data after one hot encoding ",project_grade_
categories_one_hot_cv.shape)
```

```
Shape of matrix of Train data after one hot encoding  (49041, 4)
Shape of matrix of Test data after one hot encoding  (36052, 4)
Shape of matrix of Cross Validation data after one hot encoding  (24155,
4)
```

One hot encoding of School States

In [69]:

```
type(list(project_data['school_state']))
```

Out[69]:

```
list
```

In [70]:

```
mylist = list(X_train['school_state'])
```

In [71]:

```
# We are removing the duplicate values from our list of the state codes

# Source :- https://www.w3schools.com/python/python_howto_remove_duplicates.asp

mylist_actual = list(dict.fromkeys(mylist))
```

In [72]:

```
# we use count vectorizer to convert the values into one hot encoded features

# now we are working on school state data

from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(vocabulary=mylist_actual, lowercase=False, binary=True)

vectorizer.fit(X_train['school_state'].values)

print(vectorizer.get_feature_names())

school_state_categories_one_hot_train = vectorizer.transform(X_train['school_state'].va
lues)
school_state_categories_one_hot_test = vectorizer.transform(X_test['school_state'].valu
es)
school_state_categories_one_hot_cv = vectorizer.transform(X_cv['school_state'].values)
```

```
['CA', 'MA', 'TX', 'GA', 'NJ', 'NC', 'AL', 'NY', 'VA', 'PA', 'OK', 'UT',
'KS', 'OR', 'SC', 'FL', 'AZ', 'CO', 'DE', 'MT', 'WA', 'MN', 'MI', 'NM', 'M
D', 'NV', 'CT', 'AR', 'IL', 'WI', 'IN', 'MS', 'MO', 'LA', 'OH', 'TN', 'I
D', 'WV', 'RI', 'KY', 'AK', 'SD', 'ME', 'WY', 'NH', 'IA', 'NE', 'DC', 'V
T', 'HI', 'ND']
```

In [73]:

```
print("Shape of matrix of Train data after one hot encoding ",school_state_categories_o
ne_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",school_state_categories_on
e_hot_test.shape)
print("Shape of matrix of Cross Validation data after one hot encoding ",school_state_c
ategories_one_hot_cv.shape)
```

```
Shape of matrix of Train data after one hot encoding  (49041, 51)
Shape of matrix of Test data after one hot encoding  (36052, 51)
Shape of matrix of Cross Validation data after one hot encoding  (24155, 5
1)
```

Now as we have done the vectorizing of categorical data now we will be vectorizing the text data using different techniques

Vectorizing the text data

Technique -1 Bag of words(BOW)

Essays

Essays Train Data

In [74]:

```
# I am not setting the value of min_df here because i read here
# https://stackoverflow.com/questions/27697766/understanding-min-df-and-max-df-in-scikit-countvectorizer
# that min_df helps in better performance but might also give poor clusters hence am trying without it this time



vectorizer = CountVectorizer()
vectorizer.fit(preprocessed_essays_Train)

essay_bow_train = vectorizer.transform(preprocessed_essays_Train)

print("Shape of matrix after bag of words ",essay_bow_train.shape)
```

```
Shape of matrix after bag of words  (49041, 41074)
```

Essay Test Data

In [75]:

```
# I am not setting the value of min_df here because i read here
# https://stackoverflow.com/questions/27697766/understanding-min-df-and-max-df-in-scikit-countvectorizer
# that min_df helps in better performance but might also give poor clusters hence am trying without it this time


# now note that the below two lines are wrong because we have already trained the
# model on the training data now using that model we should get the bow representation
# of test data. After all training from test data only and then cecking for its accuracy will ofcourse give
# good accuracy.

# lines not to be used (I used them but then going through the code realised the mistake)

#vectorizer = CountVectorizer()
#vectorizer.fit(preprocessed_essays_Test)

essay_bow_test = vectorizer.transform(preprocessed_essays_Test)

print("Shape of matrix after bag of words ",essay_bow_test.shape)
```

```
Shape of matrix after bag of words  (36052, 41074)
```

Essay Cross Validation data

In [76]:

```
# I am not setting the value of min_df here because i read here
# https://stackoverflow.com/questions/27697766/understanding-min-df-and-max-df-in-scikit-countvectorizer
# that min_df helps in better performance but might also give poor clusters hence am trying without it this time

# similarly below two lines should not be used

#vectorizer = CountVectorizer()
#vectorizer.fit(preprocessed_essays_Cv)

essay_bow_cv = vectorizer.transform(preprocessed_essays_Cv)

print("Shape of matrix after bag of words ",essay_bow_cv.shape)
```

```
Shape of matrix after bag of words  (24155, 41074)
```

Project Title

Bag of words on Project Title Train data

In [77]:

```
vectorizer.fit(preprocessed_project_titles_Train)
project_title_bow_train = vectorizer.transform(preprocessed_project_titles_Train)
print("Shape of matrix after bag of words ",project_title_bow_train.shape)
```

```
Shape of matrix after bag of words  (49041, 11659)
```

Bag of words on Project Title Test data

In [78]:

```
project_title_bow_test = vectorizer.transform(preprocessed_project_titles_Test)
print("Shape of matrix after bag of words ",project_title_bow_test.shape)
```

```
Shape of matrix after bag of words  (36052, 11659)
```

Bag of words on Project Title Cross Validation data

In [79]:

```
project_title_bow_cv = vectorizer.transform(preprocessed_project_titles_Cv)
print("Shape of matrix after bag of words ",project_title_bow_cv.shape)
```

```
Shape of matrix after bag of words  (24155, 11659)
```

Technique-2 TFIDF

Essay Data

Essay Train Data

In [156]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(max_features = 6000, min_df = 10)
vectorizer.fit(preprocessed_essays_Train)

text_tfidf_train = vectorizer.transform(preprocessed_essays_Train)
print("Shape of matrix after tfidf ",text_tfidf_train.shape)

print(type(text_tfidf_train))

# we are converting a dictionary with word as a key, and the idf as a value

dictionary = dict(zip(vectorizer.get_feature_names(), list(vectorizer.idf_)))
tfidf_words = set(dictionary.keys())
```

```
Shape of matrix after tfidf  (49041, 6000)
<class 'scipy.sparse.csr.csr_matrix'>
```

Essay Test data

In [157]:

```
text_tfidf_test = vectorizer.transform(preprocessed_essays_Test)
print("Shape of matrix after tfidf ",text_tfidf_test.shape)
```

```
Shape of matrix after tfidf  (36052, 6000)
```

Essay Cross Validation Data

In [158]:

```
text_tfidf_cv = vectorizer.transform(preprocessed_essays_Cv)
print("Shape of matrix after tfidf ",text_tfidf_cv.shape)
```

```
Shape of matrix after tfidf  (24155, 6000)
```

Project Title

Project title train data

In [159]:

```
vectorizer = TfidfVectorizer(max_features = 2000, min_df = 10)

vectorizer.fit(preprocessed_project_titles_Train)

project_title_tfidf_train = vectorizer.transform(preprocessed_project_titles_Train)

print("Shape of matrix after tfidf ",project_title_tfidf_train.shape)

# we are converting a dictionary with word as a key, and the idf as a value

dictionary = dict(zip(vectorizer.get_feature_names(), list(vectorizer.idf_)))
tfidf_project_title_words = set(dictionary.keys())
```

```
Shape of matrix after tfidf  (49041, 2000)
```

Project title test data

In [160]:

```
project_title_tfidf_test = vectorizer.transform(preprocessed_project_titles_Test)
print("Shape of matrix after tfidf ",project_title_tfidf_test.shape)
```

```
Shape of matrix after tfidf  (36052, 2000)
```

Project title cross validation data

In [161]:

```
title_tfidf_cv = vectorizer.transform(preprocessed_project_titles_Cv)
print("Shape of matrix after tfidf ",title_tfidf_cv.shape)
```

```
Shape of matrix after tfidf  (24155, 2000)
```

Technique-3 Average Word to Vector

Using pretrained w2v model in the file glove

In [86]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-p
ickle-to-save-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

Essay Train data

In [87]:

```
# average Word2Vec
# compute average word2vec for each preprocessed essay.

avg_w2v_vectors_train = []; # the avg-w2v for each essay is stored in this list
for sentence in tqdm(preprocessed_essays_Train): # for each essay
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a essay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_train.append(vector)

print(len(avg_w2v_vectors_train))
print(len(avg_w2v_vectors_train[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████████
██████| 49041/49041 [00:10<00:00, 4615.36it/s]

49041
300
```

Essay Test data

In [88]:

```
# average Word2Vec
# compute average word2vec for each preprocessed essay.

avg_w2v_vectors_test = []; # the avg-w2v for each essay is stored in this list
for sentence in tqdm(preprocessed_essays_Test): # for each essay
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a essay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_test.append(vector)

print(len(avg_w2v_vectors_test))
print(len(avg_w2v_vectors_test[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████████
██████| 36052/36052 [00:07<00:00, 4621.37it/s]

36052
300
```

Essay Cross Validation data

In [89]:

```
# average Word2Vec
# compute average word2vec for each preprocessed essay.

avg_w2v_vectors_cv = []; # the avg-w2v for each essay is stored in this list
for sentence in tqdm(preprocessed_essays_Cv): # for each essay
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a essay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_cv.append(vector)

print(len(avg_w2v_vectors_cv))
print(len(avg_w2v_vectors_cv[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████████
██████| 24155/24155 [00:05<00:00, 4598.35it/s]

24155
300
```

Project Title train data

In [90]:

```
# average Word2Vec
# compute average word2vec for each preprocessed essay.

avg_w2v_vectors_project_title_train = []; # the avg-w2v for each essay is stored in thi
s list
for sentence in tqdm(preprocessed_project_titles_Train): # for each essay
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a essay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_project_title_train.append(vector)

print(len(avg_w2v_vectors_project_title_train))
print(len(avg_w2v_vectors_project_title_train[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████████
████| 49041/49041 [00:00<00:00, 84872.79it/s]

49041
300
```

Project Title test data

In [91]:

```
# average Word2Vec
# compute average word2vec for each preprocessed essay.

avg_w2v_vectors_project_title_test = []; # the avg-w2v for each essay is stored in this
list
for sentence in tqdm(preprocessed_project_titles_Test): # for each essay
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a essay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_project_title_test.append(vector)

print(len(avg_w2v_vectors_project_title_test))
print(len(avg_w2v_vectors_project_title_test[0]))
```

```
100%|████████████████████████████████████████████████████████████████████
████| 36052/36052 [00:00<00:00, 83635.58it/s]

36052
300
```

Project title Cross Validation data

In [92]:

```
# average Word2Vec
# compute average word2vec for each preprocessed essay.

avg_w2v_vectors_project_title_cv = []; # the avg-w2v for each essay is stored in this l
ist
for sentence in tqdm(preprocessed_project_titles_Cv): # for each essay
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a essay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_project_title_cv.append(vector)

print(len(avg_w2v_vectors_project_title_cv))
print(len(avg_w2v_vectors_project_title_cv[0]))
```

```
100%|████████████████████████████████████████████████████████████████████
████| 24155/24155 [00:00<00:00, 84096.02it/s]

24155
300
```

Technique-4 TFIDF weighted Word To Vector

Essay train data

In [162]:

```
# tfidf Word2Vec
# computing average word2vec for each Project Title is stored in this list

tfidf_w2v_vectors_text_train = []; # the tfidf-w2v for each essay
for sentence in tqdm(preprocessed_essays_Train): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the title
    for word in sentence.split(): # for each word in a title
        if (word in glove_words) and (word in dictionary):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_text_train.append(vector)

print(len(tfidf_w2v_vectors_text_train))
print(len(tfidf_w2v_vectors_text_train[0]))
```

```
  0%|
| 0/49041 [00:00<?, ?it/s]

  0%||
| 84/49041 [00:00<00:58, 833.63it/s]

  0%|▌
| 163/49041 [00:00<00:59, 819.48it/s]

  1%|▊
| 247/49041 [00:00<00:59, 823.83it/s]

  1%|▉
| 341/49041 [00:00<00:57, 848.86it/s]

  1%|█
| 432/49041 [00:00<00:56, 865.18it/s]

  1%|█▎
| 518/49041 [00:00<00:56, 860.94it/s]

  1%|█▍
| 596/49041 [00:00<00:58, 833.01it/s]

  1%|█▋
| 674/49041 [00:00<00:59, 815.43it/s]

  2%|█▊
| 759/49041 [00:00<00:58, 823.94it/s]

  2%|█▉
| 849/49041 [00:01<00:57, 843.23it/s]

  2%|██
| 932/49041 [00:01<00:57, 834.48it/s]

  2%|██▏
| 1017/49041 [00:01<00:57, 836.50it/s]

  2%|██▍
| 1102/49041 [00:01<00:57, 840.43it/s]

  2%|██▌
| 1186/49041 [00:01<00:57, 836.24it/s]

  3%|██▊
| 1270/49041 [00:01<00:58, 823.22it/s]

  3%|██▉
| 1353/49041 [00:01<00:58, 811.30it/s]

  3%|███
| 1434/49041 [00:01<00:59, 794.09it/s]

  3%|███▏
| 1516/49041 [00:01<00:59, 800.04it/s]

  3%|███▍
| 1599/49041 [00:01<00:58, 806.84it/s]
```

```
  3%|██
| 1684/49041 [00:02<00:57, 817.60it/s]

  4%|██
| 1766/49041 [00:02<00:58, 814.11it/s]

  4%|██
| 1853/49041 [00:02<00:56, 828.80it/s]

  4%|██
| 1940/49041 [00:02<00:56, 836.62it/s]

  4%|██
| 2024/49041 [00:02<00:57, 823.45it/s]

  4%|██
| 2107/49041 [00:02<00:58, 808.38it/s]

  4%|██
| 2188/49041 [00:02<00:58, 807.02it/s]

  5%|██
| 2272/49041 [00:02<00:57, 816.07it/s]

  5%|██
| 2359/49041 [00:02<00:56, 830.66it/s]

  5%|██
| 2445/49041 [00:02<00:55, 837.46it/s]

  5%|██
| 2529/49041 [00:03<00:56, 824.29it/s]

  5%|██
| 2621/49041 [00:03<00:54, 847.26it/s]

  6%|██
| 2706/49041 [00:03<00:55, 841.29it/s]

  6%|██
| 2791/49041 [00:03<00:55, 830.45it/s]

  6%|██
| 2880/49041 [00:03<00:54, 842.88it/s]

  6%|██
| 2965/49041 [00:03<00:55, 835.73it/s]

  6%|██
| 3049/49041 [00:03<00:55, 825.82it/s]

  6%|██
| 3135/49041 [00:03<00:54, 834.72it/s]

  7%|██
| 3221/49041 [00:03<00:54, 840.89it/s]

  7%|██
| 3311/49041 [00:03<00:53, 852.98it/s]

  7%|██
```

```
| 3404/49041 [00:04<00:52, 872.87it/s]

  7%|▌
| 3493/49041 [00:04<00:51, 877.88it/s]

  7%|▌
| 3581/49041 [00:04<00:52, 867.12it/s]

  7%|▌
| 3668/49041 [00:04<00:52, 865.83it/s]

  8%|▋
| 3755/49041 [00:04<00:53, 850.08it/s]

  8%|▋
| 3843/49041 [00:04<00:52, 855.92it/s]

  8%|▋
| 3930/49041 [00:04<00:52, 859.35it/s]

  8%|▋
| 4017/49041 [00:04<00:52, 860.59it/s]

  8%|▋
| 4104/49041 [00:04<00:52, 858.98it/s]

  9%|▋
| 4190/49041 [00:05<00:52, 850.85it/s]

  9%|▋
| 4279/49041 [00:05<00:52, 860.41it/s]

  9%|▋
| 4366/49041 [00:05<00:52, 850.04it/s]

  9%|▋
| 4456/49041 [00:05<00:51, 862.68it/s]

  9%|▊
| 4543/49041 [00:05<00:52, 841.42it/s]

  9%|▊
| 4628/49041 [00:05<00:53, 837.11it/s]

 10%|▊
| 4715/49041 [00:05<00:52, 840.02it/s]

 10%|▊
| 4800/49041 [00:05<00:53, 833.83it/s]

 10%|▊
| 4884/49041 [00:05<00:53, 831.38it/s]

 10%|▊
| 4968/49041 [00:05<00:54, 810.29it/s]

 10%|▊
| 5057/49041 [00:06<00:52, 831.53it/s]

 10%|▊
| 5142/49041 [00:06<00:52, 835.11it/s]
```

```
 11%|▉▉▉▉▉▉▉
| 5232/49041 [00:06<00:51, 851.72it/s]

 11%|▉▉▉▉▉▉▉▏
| 5318/49041 [00:06<00:51, 843.63it/s]

 11%|▉▉▉▉▉▉▉▍
| 5405/49041 [00:06<00:51, 844.14it/s]

 11%|▉▉▉▉▉▉▉▍
| 5490/49041 [00:06<00:52, 828.74it/s]

 11%|▉▉▉▉▉▉▉▊
| 5575/49041 [00:06<00:52, 832.16it/s]

 12%|▉▉▉▉▉▉▉▊
| 5659/49041 [00:06<00:53, 806.36it/s]

 12%|▉▉▉▉▉▉▉▊
| 5748/49041 [00:06<00:52, 823.48it/s]

 12%|▉▉▉▉▉▉▉▉
| 5835/49041 [00:06<00:51, 835.15it/s]

 12%|▉▉▉▉▉▉▉▉▏
| 5934/49041 [00:07<00:49, 874.76it/s]

 12%|▉▉▉▉▉▉▉▉▏
| 6023/49041 [00:07<00:50, 844.75it/s]

 12%|▉▉▉▉▉▉▉▉▍
| 6109/49041 [00:07<00:51, 832.64it/s]

 13%|▉▉▉▉▉▉▉▉▍
| 6200/49041 [00:07<00:50, 849.79it/s]

 13%|▉▉▉▉▉▉▉▉▊
| 6286/49041 [00:07<00:51, 836.13it/s]

 13%|▉▉▉▉▉▉▉▉▊
| 6370/49041 [00:07<00:51, 831.54it/s]

 13%|▉▉▉▉▉▉▉▉▉
| 6454/49041 [00:07<00:51, 832.84it/s]

 13%|▉▉▉▉▉▉▉▉▉▏
| 6538/49041 [00:07<00:52, 807.28it/s]

 14%|▉▉▉▉▉▉▉▉▉▏
| 6623/49041 [00:07<00:52, 815.43it/s]

 14%|▉▉▉▉▉▉▉▉▉▍
| 6707/49041 [00:08<00:51, 820.89it/s]

 14%|▉▉▉▉▉▉▉▉▉▍
| 6795/49041 [00:08<00:50, 836.65it/s]

 14%|▉▉▉▉▉▉▉▉▉▊
| 6879/49041 [00:08<00:51, 814.66it/s]
```

14%|
| 6968/49041 [00:08<00:50, 834.57it/s]

14%|
| 7052/49041 [00:08<00:50, 824.64it/s]

15%|
| 7135/49041 [00:08<00:51, 821.33it/s]

15%|
| 7222/49041 [00:08<00:50, 834.46it/s]

15%|
| 7306/49041 [00:08<00:50, 827.42it/s]

15%|
| 7389/49041 [00:08<00:51, 814.25it/s]

15%|
| 7478/49041 [00:08<00:49, 833.56it/s]

15%|
| 7564/49041 [00:09<00:49, 838.93it/s]

16%|
| 7652/49041 [00:09<00:48, 849.35it/s]

16%|
| 7739/49041 [00:09<00:48, 855.19it/s]

16%|
| 7833/49041 [00:09<00:46, 877.17it/s]

16%|
| 7921/49041 [00:09<00:47, 872.86it/s]

16%|
| 8009/49041 [00:09<00:48, 840.21it/s]

17%|
| 8095/49041 [00:09<00:48, 844.35it/s]

17%|
| 8180/49041 [00:09<00:49, 831.62it/s]

17%|
| 8264/49041 [00:09<00:49, 817.08it/s]

17%|
| 8346/49041 [00:09<00:50, 811.40it/s]

17%|
| 8428/49041 [00:10<00:53, 765.56it/s]

17%|
| 8510/49041 [00:10<00:52, 775.11it/s]

18%|
| 8591/49041 [00:10<00:51, 782.38it/s]

18%|

```
| 8680/49041 [00:10<00:49, 811.36it/s]

 18%|
| 8762/49041 [00:10<00:50, 794.53it/s]

 18%|
| 8842/49041 [00:10<00:51, 778.19it/s]

 18%|
| 8921/49041 [00:10<00:52, 766.11it/s]

 18%|
| 9001/49041 [00:10<00:51, 775.90it/s]

 19%|
| 9079/49041 [00:10<00:52, 766.31it/s]

 19%|
| 9165/49041 [00:11<00:50, 790.61it/s]

 19%|
| 9250/49041 [00:11<00:49, 806.75it/s]

 19%|
| 9334/49041 [00:11<00:48, 814.64it/s]

 19%|
| 9421/49041 [00:11<00:47, 826.41it/s]

 19%|
| 9511/49041 [00:11<00:46, 846.26it/s]

 20%|
| 9596/49041 [00:11<00:47, 828.27it/s]

 20%|
| 9690/49041 [00:11<00:45, 857.25it/s]

 20%|
| 9777/49041 [00:11<00:46, 849.13it/s]

 20%|
| 9863/49041 [00:11<00:47, 816.61it/s]

 20%|
| 9946/49041 [00:11<00:49, 795.40it/s]

 20%|
| 10027/49041 [00:12<00:50, 777.38it/s]

 21%|
| 10108/49041 [00:12<00:49, 785.21it/s]

 21%|
| 10187/49041 [00:12<00:49, 778.84it/s]

 21%|
| 10266/49041 [00:12<00:50, 768.74it/s]

 21%|
| 10348/49041 [00:12<00:49, 778.66it/s]
```

```
 21%|██████████████▏
| 10427/49041 [00:12<00:49, 777.92it/s]

 21%|██████████████▍
| 10510/49041 [00:12<00:48, 788.60it/s]

 22%|██████████████▌
| 10591/49041 [00:12<00:48, 790.96it/s]

 22%|██████████████▋
| 10671/49041 [00:12<00:50, 756.03it/s]

 22%|██████████████▊
| 10748/49041 [00:13<00:50, 756.30it/s]

 22%|██████████████▉
| 10824/49041 [00:13<00:50, 751.68it/s]

 22%|███████████████▏
| 10906/49041 [00:13<00:49, 770.16it/s]

 22%|███████████████▎
| 10988/49041 [00:13<00:48, 783.77it/s]

 23%|███████████████▍
| 11077/49041 [00:13<00:46, 811.25it/s]

 23%|███████████████▌
| 11159/49041 [00:13<00:47, 800.23it/s]

 23%|███████████████▋
| 11240/49041 [00:13<00:47, 798.97it/s]

 23%|███████████████▊
| 11326/49041 [00:13<00:46, 814.65it/s]

 23%|███████████████▉
| 11419/49041 [00:13<00:44, 844.45it/s]

 23%|████████████████
| 11504/49041 [00:13<00:45, 829.48it/s]

 24%|████████████████▏
| 11592/49041 [00:14<00:44, 837.45it/s]

 24%|████████████████▎
| 11677/49041 [00:14<00:45, 829.55it/s]

 24%|████████████████▍
| 11761/49041 [00:14<00:45, 823.61it/s]

 24%|████████████████▌
| 11849/49041 [00:14<00:44, 839.10it/s]

 24%|████████████████▋
| 11935/49041 [00:14<00:44, 840.84it/s]

 25%|████████████████▊
| 12026/49041 [00:14<00:43, 856.27it/s]
```

25%|
| 12112/49041 [00:14<00:44, 833.26it/s]

25%|
| 12196/49041 [00:14<00:44, 826.21it/s]

25%|
| 12284/49041 [00:14<00:43, 839.96it/s]

25%|
| 12373/49041 [00:14<00:43, 852.50it/s]

25%|
| 12461/49041 [00:15<00:42, 856.30it/s]

26%|
| 12547/49041 [00:15<00:43, 842.93it/s]

26%|
| 12632/49041 [00:15<00:44, 826.10it/s]

26%|
| 12716/49041 [00:15<00:43, 828.36it/s]

26%|
| 12813/49041 [00:15<00:41, 864.62it/s]

26%|
| 12900/49041 [00:15<00:41, 864.41it/s]

26%|
| 12987/49041 [00:15<00:41, 859.02it/s]

27%|
| 13074/49041 [00:15<00:42, 845.47it/s]

27%|
| 13159/49041 [00:15<00:42, 839.92it/s]

27%|
| 13248/49041 [00:16<00:42, 850.10it/s]

27%|
| 13334/49041 [00:16<00:42, 848.61it/s]

27%|
| 13419/49041 [00:16<00:42, 840.96it/s]

28%|
| 13504/49041 [00:16<00:42, 842.90it/s]

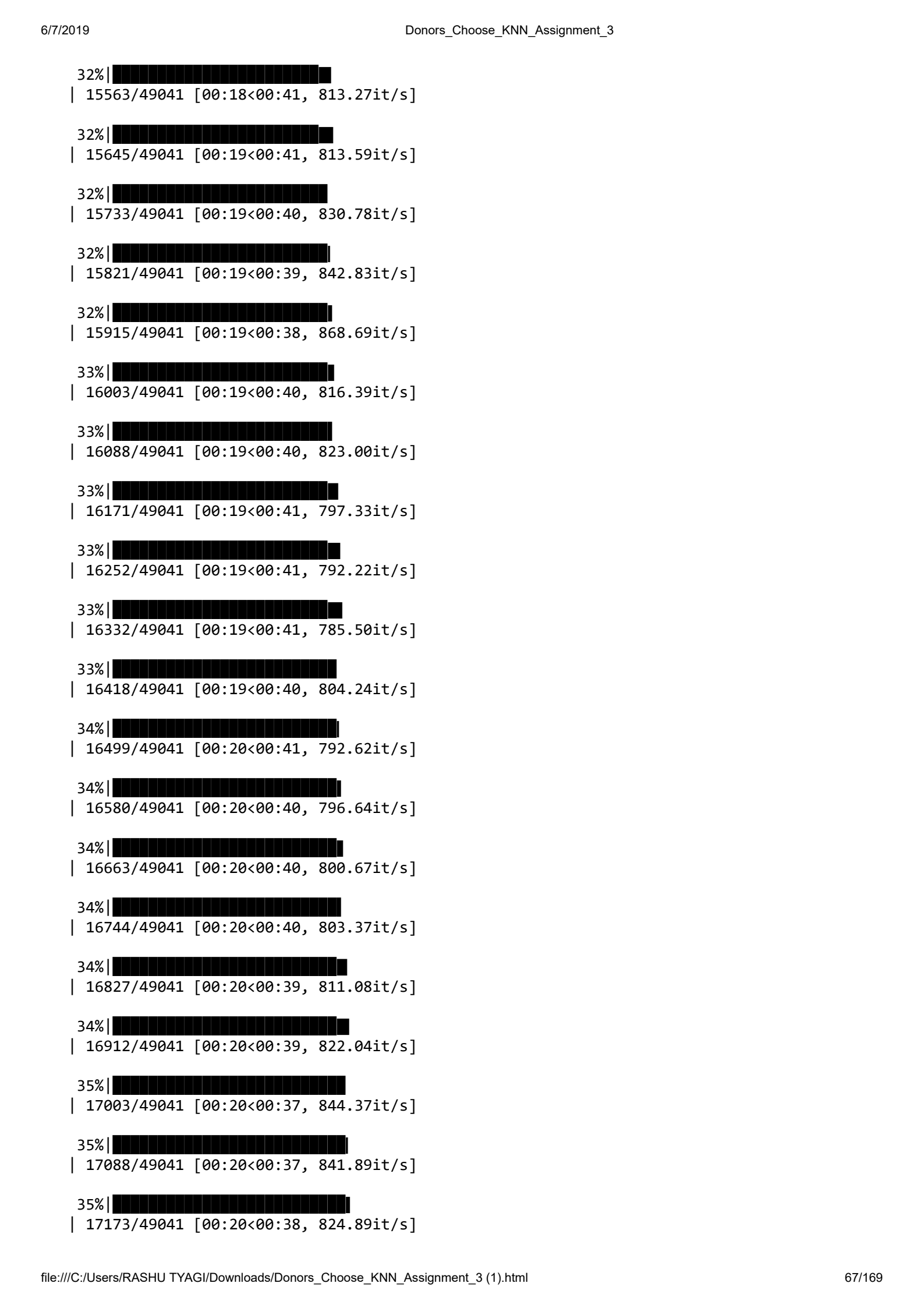
28%|
| 13589/49041 [00:16<00:42, 830.85it/s]

28%|
| 13673/49041 [00:16<00:42, 831.19it/s]

28%|
| 13757/49041 [00:16<00:43, 818.73it/s]

28%|

```
| 13839/49041 [00:16<00:44, 792.76it/s]

 28%|
| 13919/49041 [00:16<00:44, 792.83it/s]

 29%|
| 13999/49041 [00:16<00:50, 692.04it/s]

 29%|
| 14071/49041 [00:17<00:51, 684.75it/s]

 29%|
| 14148/49041 [00:17<00:49, 704.91it/s]

 29%|
| 14231/49041 [00:17<00:47, 734.60it/s]

 29%|
| 14311/49041 [00:17<00:46, 751.91it/s]

 29%|
| 14397/49041 [00:17<00:44, 776.88it/s]

 30%|
| 14476/49041 [00:17<00:44, 772.77it/s]

 30%|
| 14559/49041 [00:17<00:43, 785.29it/s]

 30%|
| 14640/49041 [00:17<00:43, 791.65it/s]

 30%|
| 14721/49041 [00:17<00:43, 795.22it/s]

 30%|
| 14801/49041 [00:18<00:44, 773.66it/s]

 30%|
| 14883/49041 [00:18<00:43, 783.24it/s]

 31%|
| 14970/49041 [00:18<00:42, 805.47it/s]

 31%|
| 15061/49041 [00:18<00:40, 832.66it/s]

 31%|
| 15145/49041 [00:18<00:41, 808.34it/s]

 31%|
| 15227/49041 [00:18<00:42, 787.11it/s]

 31%|
| 15311/49041 [00:18<00:42, 801.36it/s]

 31%|
| 15398/49041 [00:18<00:41, 819.77it/s]

 32%|
| 15481/49041 [00:18<00:41, 811.90it/s]
```


32%| | 15563/49041 [00:18<00:41, 813.27it/s]
32%| | 15645/49041 [00:19<00:41, 813.59it/s]
32%| | 15733/49041 [00:19<00:40, 830.78it/s]
32%| | 15821/49041 [00:19<00:39, 842.83it/s]
32%| | 15915/49041 [00:19<00:38, 868.69it/s]
33%| | 16003/49041 [00:19<00:40, 816.39it/s]
33%| | 16088/49041 [00:19<00:40, 823.00it/s]
33%| | 16171/49041 [00:19<00:41, 797.33it/s]
33%| | 16252/49041 [00:19<00:41, 792.22it/s]
33%| | 16332/49041 [00:19<00:41, 785.50it/s]
33%| | 16418/49041 [00:19<00:40, 804.24it/s]
34%| | 16499/49041 [00:20<00:41, 792.62it/s]
34%| | 16580/49041 [00:20<00:40, 796.64it/s]
34%| | 16663/49041 [00:20<00:40, 800.67it/s]
34%| | 16744/49041 [00:20<00:40, 803.37it/s]
34%| | 16827/49041 [00:20<00:39, 811.08it/s]
34%| | 16912/49041 [00:20<00:39, 822.04it/s]
35%| | 17003/49041 [00:20<00:37, 844.37it/s]
35%| | 17088/49041 [00:20<00:37, 841.89it/s]
35%| | 17173/49041 [00:20<00:38, 824.89it/s]

35%|
| 17265/49041 [00:21<00:37, 848.96it/s]

35%|
| 17351/49041 [00:21<00:38, 823.20it/s]

36%|
| 17434/49041 [00:21<00:38, 822.92it/s]

36%|
| 17517/49041 [00:21<00:38, 820.80it/s]

36%|
| 17604/49041 [00:21<00:37, 833.64it/s]

36%|
| 17688/49041 [00:21<00:38, 823.53it/s]

36%|
| 17775/49041 [00:21<00:37, 830.25it/s]

36%|
| 17860/49041 [00:21<00:37, 833.79it/s]

37%|
| 17944/49041 [00:21<00:38, 801.19it/s]

37%|
| 18026/49041 [00:21<00:38, 805.00it/s]

37%|
| 18107/49041 [00:22<00:38, 801.70it/s]

37%|
| 18194/49041 [00:22<00:37, 819.30it/s]

37%|
| 18277/49041 [00:22<00:37, 812.32it/s]

37%|
| 18359/49041 [00:22<00:39, 777.26it/s]

38%|
| 18438/49041 [00:22<00:39, 769.12it/s]

38%|
| 18528/49041 [00:22<00:38, 802.57it/s]

38%|
| 18609/49041 [00:22<00:37, 803.78it/s]

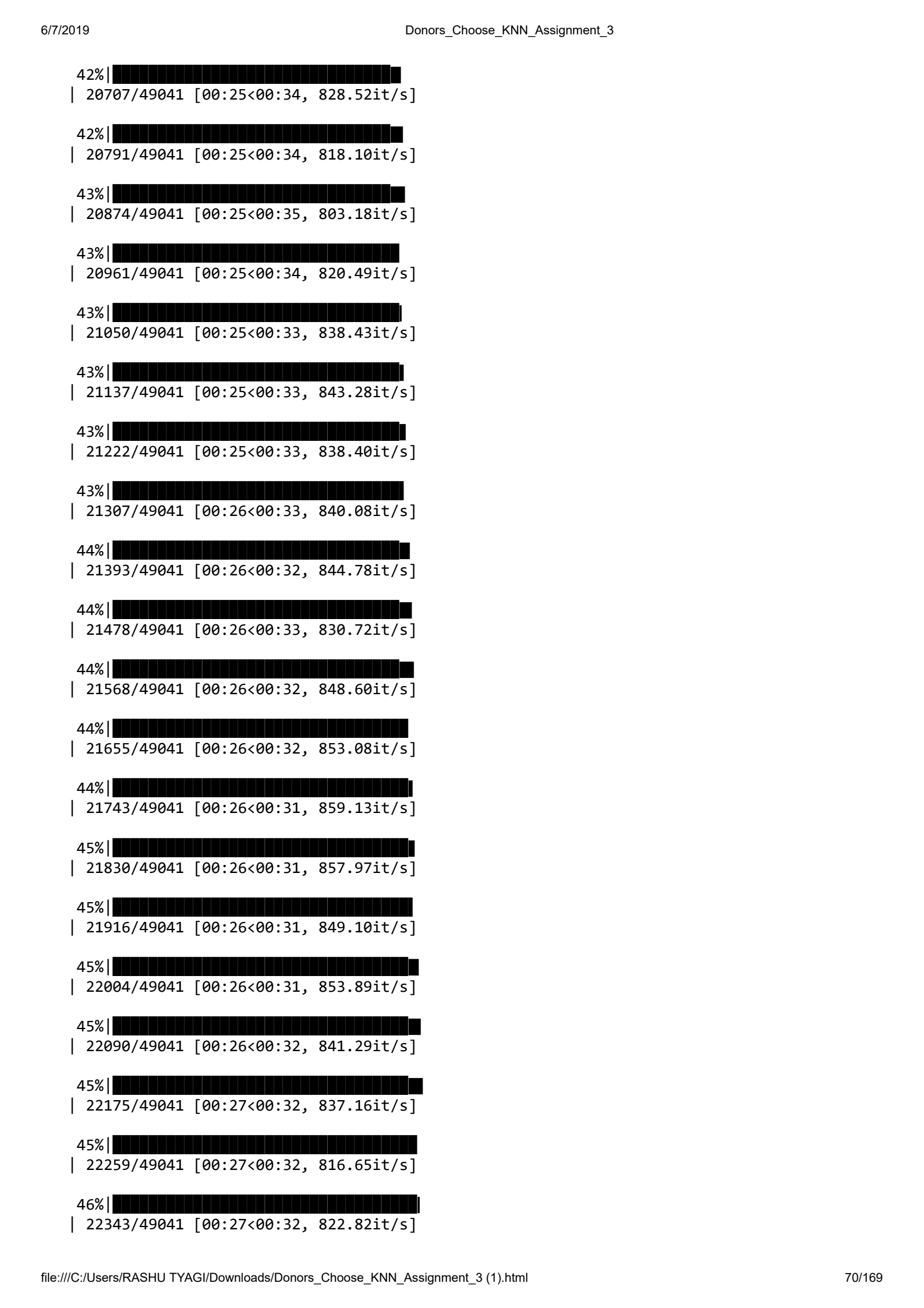
38%|
| 18697/49041 [00:22<00:36, 823.46it/s]

38%|
| 18780/49041 [00:22<00:36, 821.12it/s]

38%|
| 18863/49041 [00:22<00:37, 814.74it/s]

39%|

```
| 18945/49041 [00:23<00:37, 803.76it/s]

 39%|
| 19026/49041 [00:23<00:38, 774.05it/s]

 39%|
| 19109/49041 [00:23<00:38, 783.93it/s]

 39%|
| 19188/49041 [00:23<00:38, 779.40it/s]

 39%|
| 19267/49041 [00:23<00:38, 773.99it/s]

 39%|
| 19358/49041 [00:23<00:36, 808.75it/s]

 40%|
| 19441/49041 [00:23<00:36, 813.18it/s]

 40%|
| 19528/49041 [00:23<00:35, 827.71it/s]

 40%|
| 19618/49041 [00:23<00:34, 846.46it/s]

 40%|
| 19703/49041 [00:24<00:36, 814.17it/s]

 40%|
| 19785/49041 [00:24<00:36, 804.57it/s]

 41%|
| 19866/49041 [00:24<00:37, 778.98it/s]

 41%|
| 19945/49041 [00:24<00:38, 748.53it/s]

 41%|
| 20028/49041 [00:24<00:37, 769.38it/s]

 41%|
| 20113/49041 [00:24<00:36, 786.55it/s]

 41%|
| 20193/49041 [00:24<00:37, 778.30it/s]

 41%|
| 20273/49041 [00:24<00:36, 780.64it/s]

 42%|
| 20357/49041 [00:24<00:35, 797.02it/s]

 42%|
| 20444/49041 [00:24<00:35, 815.98it/s]

 42%|
| 20533/49041 [00:25<00:34, 835.07it/s]

 42%|
| 20622/49041 [00:25<00:33, 846.65it/s]
```


42%| | 20707/49041 [00:25<00:34, 828.52it/s]
42%| | 20791/49041 [00:25<00:34, 818.10it/s]
43%| | 20874/49041 [00:25<00:35, 803.18it/s]
43%| | 20961/49041 [00:25<00:34, 820.49it/s]
43%| | 21050/49041 [00:25<00:33, 838.43it/s]
43%| | 21137/49041 [00:25<00:33, 843.28it/s]
43%| | 21222/49041 [00:25<00:33, 838.40it/s]
43%| | 21307/49041 [00:26<00:33, 840.08it/s]
44%| | 21393/49041 [00:26<00:32, 844.78it/s]
44%| | 21478/49041 [00:26<00:33, 830.72it/s]
44%| | 21568/49041 [00:26<00:32, 848.60it/s]
44%| | 21655/49041 [00:26<00:32, 853.08it/s]
44%| | 21743/49041 [00:26<00:31, 859.13it/s]
45%| | 21830/49041 [00:26<00:31, 857.97it/s]
45%| | 21916/49041 [00:26<00:31, 849.10it/s]
45%| | 22004/49041 [00:26<00:31, 853.89it/s]
45%| | 22090/49041 [00:26<00:32, 841.29it/s]
45%| | 22175/49041 [00:27<00:32, 837.16it/s]
45%| | 22259/49041 [00:27<00:32, 816.65it/s]
46%| | 22343/49041 [00:27<00:32, 822.82it/s]

46%|
| 22429/49041 [00:27<00:31, 831.78it/s]

46%|
| 22513/49041 [00:27<00:31, 833.08it/s]

46%|
| 22597/49041 [00:27<00:32, 825.97it/s]

46%|
| 22682/49041 [00:27<00:31, 832.06it/s]

46%|
| 22768/49041 [00:27<00:31, 838.64it/s]

47%|
| 22854/49041 [00:27<00:31, 843.12it/s]

47%|
| 22943/49041 [00:27<00:30, 854.78it/s]

47%|
| 23029/49041 [00:28<00:31, 815.68it/s]

47%|
| 23111/49041 [00:28<00:32, 793.93it/s]

47%|
| 23191/49041 [00:28<00:33, 760.99it/s]

47%|
| 23273/49041 [00:28<00:33, 776.10it/s]

48%|
| 23356/49041 [00:28<00:32, 789.92it/s]

48%|
| 23436/49041 [00:28<00:32, 784.18it/s]

48%|
| 23529/49041 [00:28<00:31, 821.35it/s]

48%|
| 23612/49041 [00:28<00:31, 819.71it/s]

48%|
| 23701/49041 [00:28<00:30, 835.52it/s]

49%|
| 23791/49041 [00:28<00:29, 852.10it/s]

49%|
| 23877/49041 [00:29<00:30, 834.73it/s]

49%|
| 23970/49041 [00:29<00:29, 857.08it/s]

49%|
| 24061/49041 [00:29<00:28, 870.54it/s]

49%|

```
| 24149/49041 [00:29<00:29, 848.75it/s]

 49%|
| 24240/49041 [00:29<00:28, 866.13it/s]

 50%|
| 24327/49041 [00:29<00:28, 855.28it/s]

 50%|
| 24413/49041 [00:29<00:29, 844.78it/s]

 50%|
| 24500/49041 [00:29<00:28, 850.30it/s]

 50%|
| 24588/49041 [00:29<00:28, 857.16it/s]

 50%|
| 24679/49041 [00:30<00:27, 870.53it/s]

 51%|
| 24767/49041 [00:30<00:27, 868.88it/s]

 51%|
| 24854/49041 [00:30<00:28, 863.36it/s]

 51%|
| 24941/49041 [00:30<00:28, 831.22it/s]

 51%|
| 25025/49041 [00:30<00:29, 827.05it/s]

 51%|
| 25108/49041 [00:30<00:29, 806.97it/s]

 51%|
| 25191/49041 [00:30<00:29, 811.92it/s]

 52%|
| 25275/49041 [00:30<00:29, 817.77it/s]

 52%|
| 25366/49041 [00:30<00:28, 841.70it/s]

 52%|
| 25452/49041 [00:30<00:27, 845.28it/s]

 52%|
| 25537/49041 [00:31<00:28, 837.44it/s]

 52%|
| 25621/49041 [00:31<00:28, 824.03it/s]

 52%|
| 25704/49041 [00:31<00:28, 819.15it/s]

 53%|
| 25792/49041 [00:31<00:27, 834.83it/s]

 53%|
| 25877/49041 [00:31<00:27, 837.52it/s]
```

53%|
| 25961/49041 [00:31<00:27, 836.43it/s]

53%|
| 26048/49041 [00:31<00:27, 844.36it/s]

53%|
| 26138/49041 [00:31<00:26, 856.07it/s]

53%|
| 26224/49041 [00:31<00:27, 832.03it/s]

54%|
| 26312/49041 [00:31<00:26, 844.08it/s]

54%|
| 26397/49041 [00:32<00:27, 829.22it/s]

54%|
| 26481/49041 [00:32<00:27, 810.72it/s]

54%|
| 26570/49041 [00:32<00:27, 831.04it/s]

54%|
| 26656/49041 [00:32<00:26, 838.29it/s]

55%|
| 26741/49041 [00:32<00:26, 837.54it/s]

55%|
| 26826/49041 [00:32<00:26, 839.42it/s]

55%|
| 26911/49041 [00:32<00:26, 832.33it/s]

55%|
| 26995/49041 [00:32<00:26, 828.62it/s]

55%|
| 27088/49041 [00:32<00:25, 852.50it/s]

55%|
| 27174/49041 [00:33<00:25, 842.88it/s]

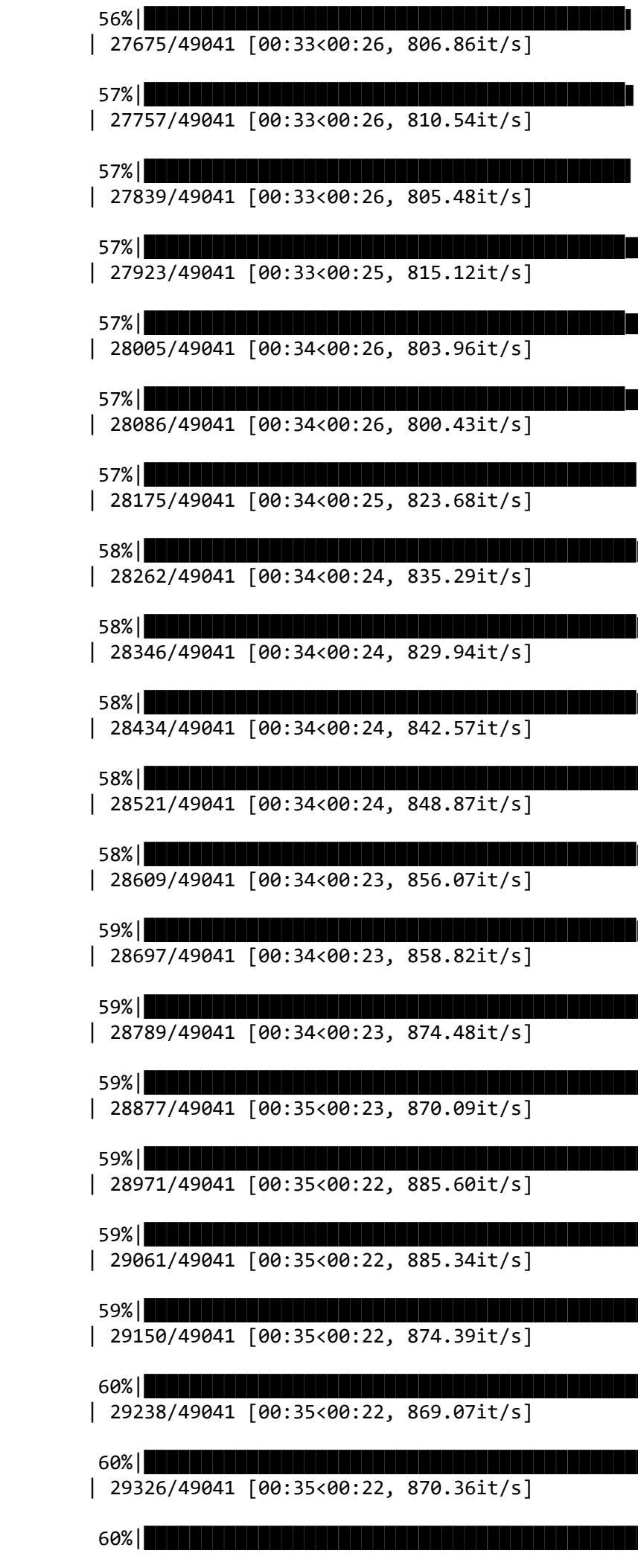
56%|
| 27260/49041 [00:33<00:25, 846.18it/s]

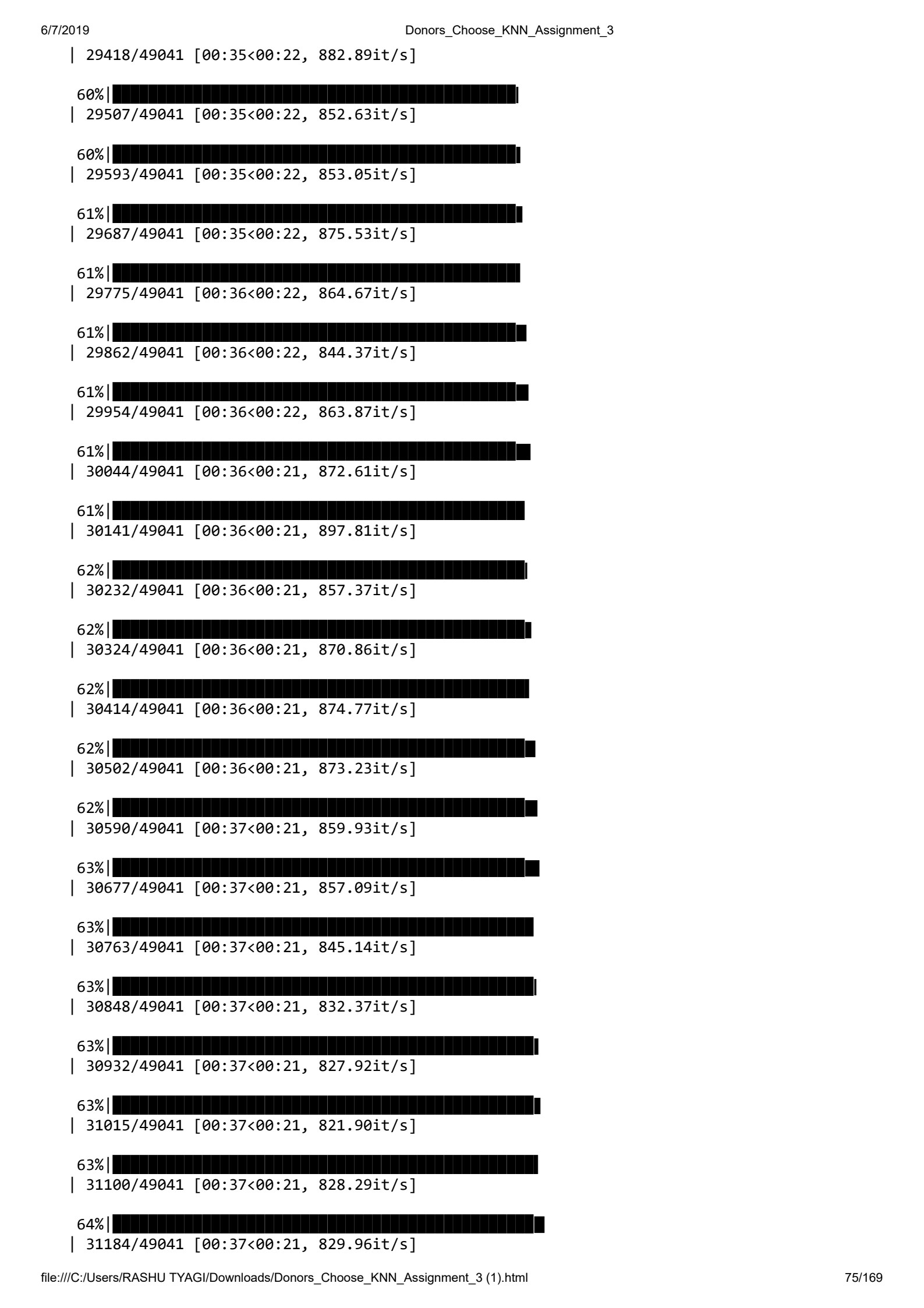
56%|
| 27345/49041 [00:33<00:26, 823.42it/s]

56%|
| 27428/49041 [00:33<00:27, 799.84it/s]

56%|
| 27509/49041 [00:33<00:27, 791.69it/s]

56%|
| 27593/49041 [00:33<00:26, 803.84it/s]


56%| | 27675/49041 [00:33<00:26, 806.86it/s]
57%| | 27757/49041 [00:33<00:26, 810.54it/s]
57%| | 27839/49041 [00:33<00:26, 805.48it/s]
57%| | 27923/49041 [00:33<00:25, 815.12it/s]
57%| | 28005/49041 [00:34<00:26, 803.96it/s]
57%| | 28086/49041 [00:34<00:26, 800.43it/s]
57%| | 28175/49041 [00:34<00:25, 823.68it/s]
58%| | 28262/49041 [00:34<00:24, 835.29it/s]
58%| | 28346/49041 [00:34<00:24, 829.94it/s]
58%| | 28434/49041 [00:34<00:24, 842.57it/s]
58%| | 28521/49041 [00:34<00:24, 848.87it/s]
58%| | 28609/49041 [00:34<00:23, 856.07it/s]
59%| | 28697/49041 [00:34<00:23, 858.82it/s]
59%| | 28789/49041 [00:34<00:23, 874.48it/s]
59%| | 28877/49041 [00:35<00:23, 870.09it/s]
59%| | 28971/49041 [00:35<00:22, 885.60it/s]
59%| | 29061/49041 [00:35<00:22, 885.34it/s]
59%| | 29150/49041 [00:35<00:22, 874.39it/s]
60%| | 29238/49041 [00:35<00:22, 869.07it/s]
60%| | 29326/49041 [00:35<00:22, 870.36it/s]
60%|


| 29418/49041 [00:35<00:22, 882.89it/s]
60%|
| 29507/49041 [00:35<00:22, 852.63it/s]
60%|
| 29593/49041 [00:35<00:22, 853.05it/s]
61%|
| 29687/49041 [00:35<00:22, 875.53it/s]
61%|
| 29775/49041 [00:36<00:22, 864.67it/s]
61%|
| 29862/49041 [00:36<00:22, 844.37it/s]
61%|
| 29954/49041 [00:36<00:22, 863.87it/s]
61%|
| 30044/49041 [00:36<00:21, 872.61it/s]
61%|
| 30141/49041 [00:36<00:21, 897.81it/s]
62%|
| 30232/49041 [00:36<00:21, 857.37it/s]
62%|
| 30324/49041 [00:36<00:21, 870.86it/s]
62%|
| 30414/49041 [00:36<00:21, 874.77it/s]
62%|
| 30502/49041 [00:36<00:21, 873.23it/s]
62%|
| 30590/49041 [00:37<00:21, 859.93it/s]
63%|
| 30677/49041 [00:37<00:21, 857.09it/s]
63%|
| 30763/49041 [00:37<00:21, 845.14it/s]
63%|
| 30848/49041 [00:37<00:21, 832.37it/s]
63%|
| 30932/49041 [00:37<00:21, 827.92it/s]
63%|
| 31015/49041 [00:37<00:21, 821.90it/s]
63%|
| 31100/49041 [00:37<00:21, 828.29it/s]
64%|
| 31184/49041 [00:37<00:21, 829.96it/s]

64%|
| 31268/49041 [00:37<00:21, 823.90it/s]

64%|
| 31352/49041 [00:37<00:21, 826.88it/s]

64%|
| 31444/49041 [00:38<00:20, 850.99it/s]

64%|
| 31530/49041 [00:38<00:20, 837.99it/s]

64%|
| 31615/49041 [00:38<00:20, 838.48it/s]

65%|
| 31701/49041 [00:38<00:20, 840.54it/s]

65%|
| 31786/49041 [00:38<00:20, 838.97it/s]

65%|
| 31870/49041 [00:38<00:20, 834.96it/s]

65%|
| 31954/49041 [00:38<00:21, 800.89it/s]

65%|
| 32035/49041 [00:38<00:21, 798.35it/s]

65%|
| 32116/49041 [00:38<00:21, 791.45it/s]

66%|
| 32200/49041 [00:38<00:20, 803.65it/s]

66%|
| 32281/49041 [00:39<00:21, 764.89it/s]

66%|
| 32364/49041 [00:39<00:21, 782.79it/s]

66%|
| 32443/49041 [00:39<00:21, 768.33it/s]

66%|
| 32521/49041 [00:39<00:21, 751.30it/s]

66%|
| 32606/49041 [00:39<00:21, 776.86it/s]

67%|
| 32689/49041 [00:39<00:20, 791.37it/s]

67%|
| 32769/49041 [00:39<00:20, 789.87it/s]

67%|
| 32855/49041 [00:39<00:20, 805.75it/s]

67%|
| 32938/49041 [00:39<00:19, 811.07it/s]

67%|
| 33020/49041 [00:40<00:19, 804.87it/s]

68%|
| 33106/49041 [00:40<00:19, 818.88it/s]

68%|
| 33193/49041 [00:40<00:19, 831.82it/s]

68%|
| 33281/49041 [00:40<00:18, 844.01it/s]

68%|
| 33379/49041 [00:40<00:17, 879.59it/s]

68%|
| 33468/49041 [00:40<00:18, 855.50it/s]

68%|
| 33555/49041 [00:40<00:18, 845.40it/s]

69%|
| 33640/49041 [00:40<00:18, 837.53it/s]

69%|
| 33725/49041 [00:40<00:19, 797.02it/s]

69%|
| 33806/49041 [00:40<00:19, 796.79it/s]

69%|
| 33887/49041 [00:41<00:18, 798.92it/s]

69%|
| 33972/49041 [00:41<00:18, 811.87it/s]

69%|
| 34054/49041 [00:41<00:18, 809.84it/s]

70%|
| 34137/49041 [00:41<00:18, 813.96it/s]

70%|
| 34219/49041 [00:41<00:18, 805.39it/s]

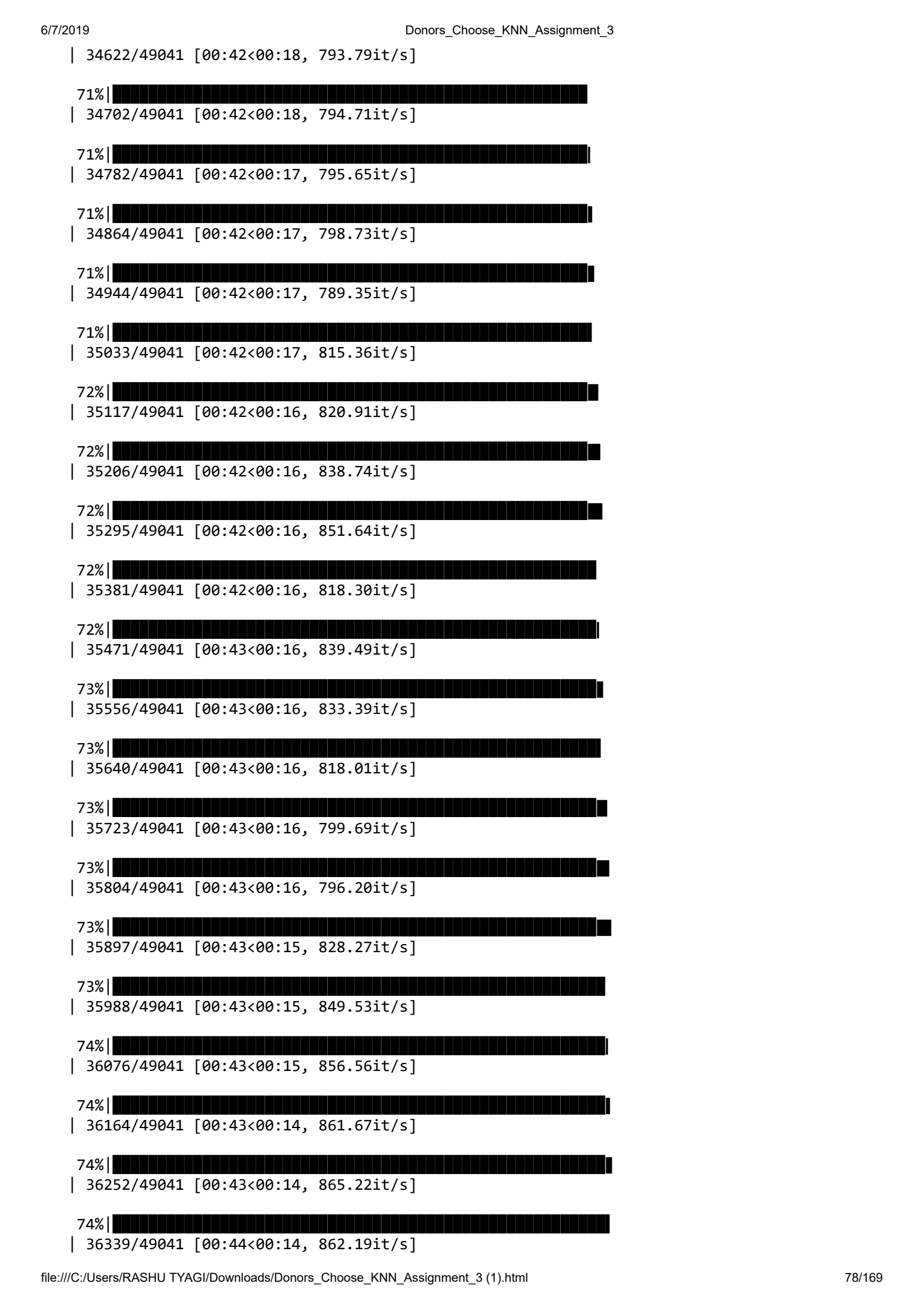
70%|
| 34300/49041 [00:41<00:18, 785.15it/s]

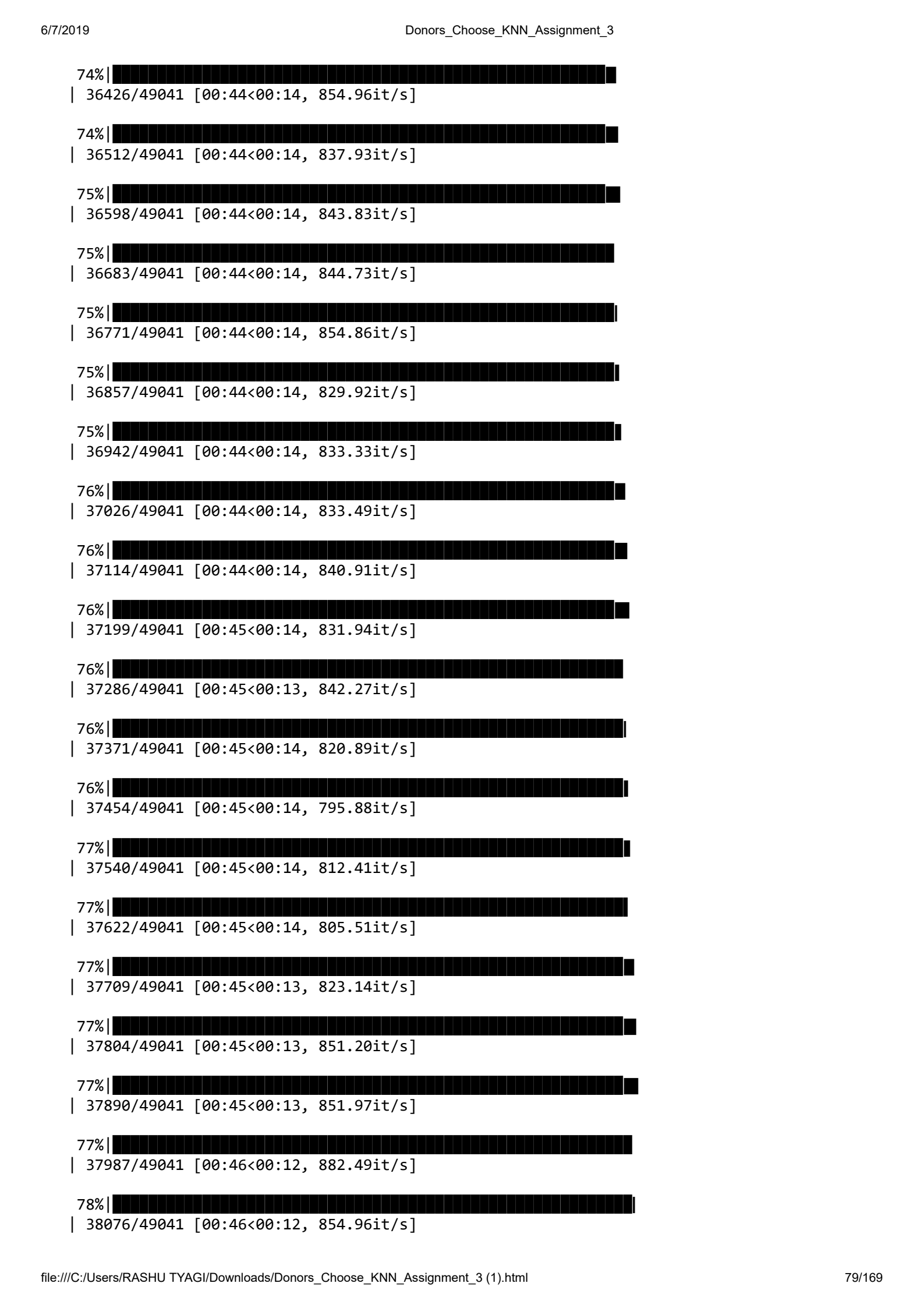
70%|
| 34379/49041 [00:41<00:18, 777.88it/s]

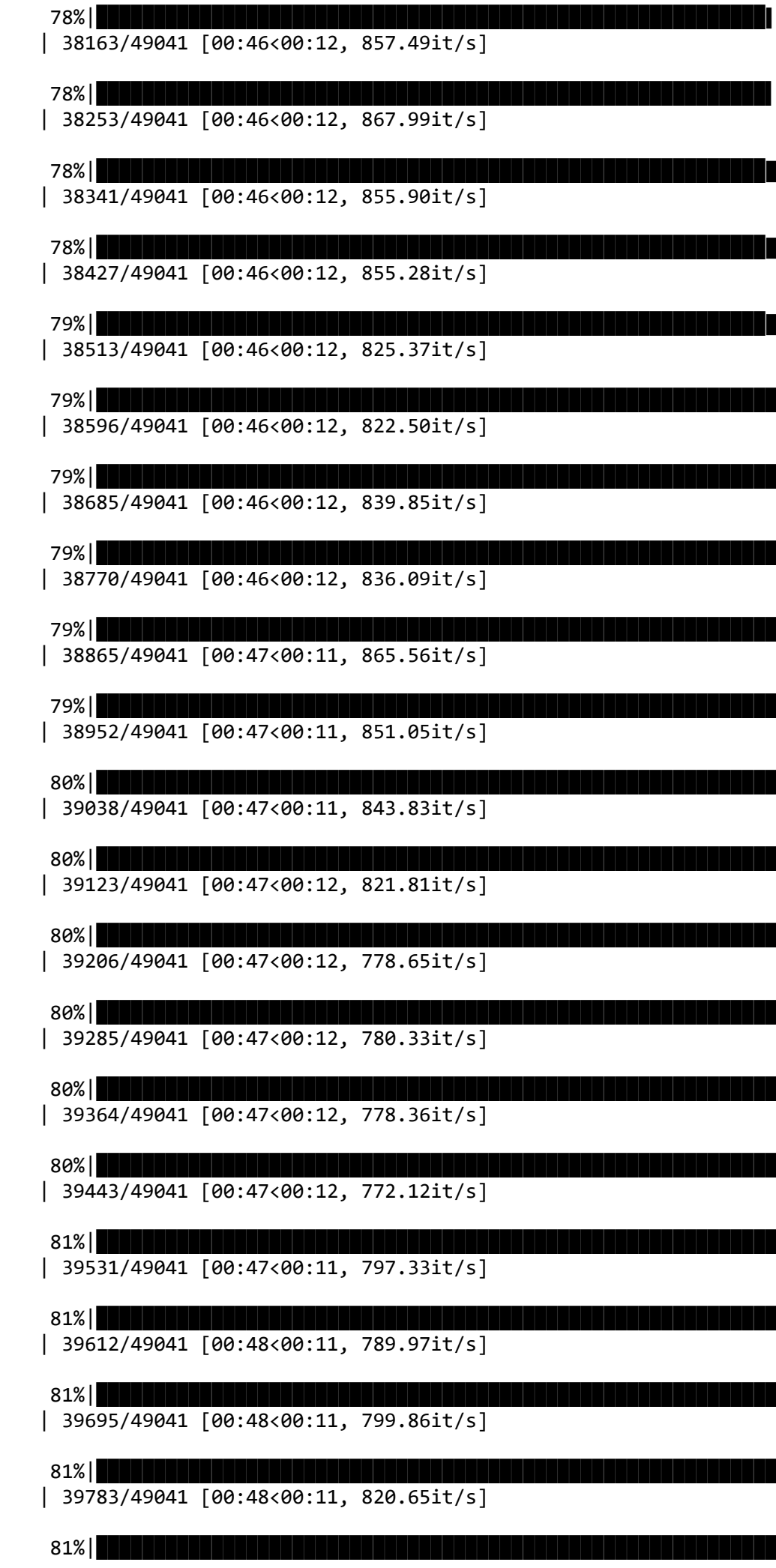
70%|
| 34459/49041 [00:41<00:18, 781.46it/s]

70%|
| 34538/49041 [00:41<00:18, 777.76it/s]

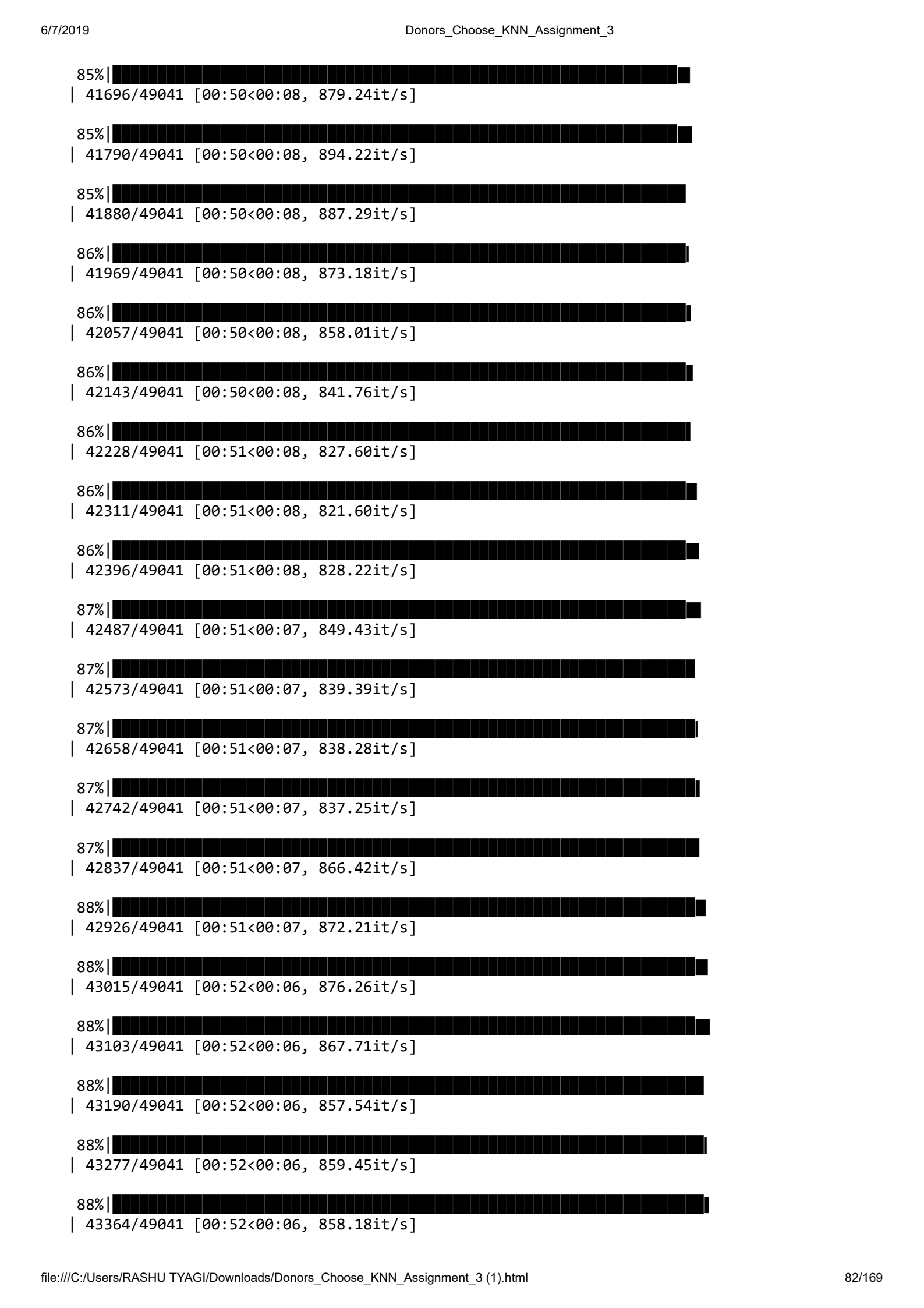
71%|


| 34622/49041 [00:42<00:18, 793.79it/s]
71%|
| 34702/49041 [00:42<00:18, 794.71it/s]
71%|
| 34782/49041 [00:42<00:17, 795.65it/s]
71%|
| 34864/49041 [00:42<00:17, 798.73it/s]
71%|
| 34944/49041 [00:42<00:17, 789.35it/s]
71%|
| 35033/49041 [00:42<00:17, 815.36it/s]
72%|
| 35117/49041 [00:42<00:16, 820.91it/s]
72%|
| 35206/49041 [00:42<00:16, 838.74it/s]
72%|
| 35295/49041 [00:42<00:16, 851.64it/s]
72%|
| 35381/49041 [00:42<00:16, 818.30it/s]
72%|
| 35471/49041 [00:43<00:16, 839.49it/s]
73%|
| 35556/49041 [00:43<00:16, 833.39it/s]
73%|
| 35640/49041 [00:43<00:16, 818.01it/s]
73%|
| 35723/49041 [00:43<00:16, 799.69it/s]
73%|
| 35804/49041 [00:43<00:16, 796.20it/s]
73%|
| 35897/49041 [00:43<00:15, 828.27it/s]
73%|
| 35988/49041 [00:43<00:15, 849.53it/s]
74%|
| 36076/49041 [00:43<00:15, 856.56it/s]
74%|
| 36164/49041 [00:43<00:14, 861.67it/s]
74%|
| 36252/49041 [00:43<00:14, 865.22it/s]
74%|
| 36339/49041 [00:44<00:14, 862.19it/s]


74%| | 36426/49041 [00:44<00:14, 854.96it/s]
74%| | 36512/49041 [00:44<00:14, 837.93it/s]
75%| | 36598/49041 [00:44<00:14, 843.83it/s]
75%| | 36683/49041 [00:44<00:14, 844.73it/s]
75%| | 36771/49041 [00:44<00:14, 854.86it/s]
75%| | 36857/49041 [00:44<00:14, 829.92it/s]
75%| | 36942/49041 [00:44<00:14, 833.33it/s]
76%| | 37026/49041 [00:44<00:14, 833.49it/s]
76%| | 37114/49041 [00:44<00:14, 840.91it/s]
76%| | 37199/49041 [00:45<00:14, 831.94it/s]
76%| | 37286/49041 [00:45<00:13, 842.27it/s]
76%| | 37371/49041 [00:45<00:14, 820.89it/s]
76%| | 37454/49041 [00:45<00:14, 795.88it/s]
77%| | 37540/49041 [00:45<00:14, 812.41it/s]
77%| | 37622/49041 [00:45<00:14, 805.51it/s]
77%| | 37709/49041 [00:45<00:13, 823.14it/s]
77%| | 37804/49041 [00:45<00:13, 851.20it/s]
77%| | 37890/49041 [00:45<00:13, 851.97it/s]
77%| | 37987/49041 [00:46<00:12, 882.49it/s]
78%| | 38076/49041 [00:46<00:12, 854.96it/s]


78%| | 38163/49041 [00:46<00:12, 857.49it/s]
78%| | 38253/49041 [00:46<00:12, 867.99it/s]
78%| | 38341/49041 [00:46<00:12, 855.90it/s]
78%| | 38427/49041 [00:46<00:12, 855.28it/s]
79%| | 38513/49041 [00:46<00:12, 825.37it/s]
79%| | 38596/49041 [00:46<00:12, 822.50it/s]
79%| | 38685/49041 [00:46<00:12, 839.85it/s]
79%| | 38770/49041 [00:46<00:12, 836.09it/s]
79%| | 38865/49041 [00:47<00:11, 865.56it/s]
79%| | 38952/49041 [00:47<00:11, 851.05it/s]
80%| | 39038/49041 [00:47<00:11, 843.83it/s]
80%| | 39123/49041 [00:47<00:12, 821.81it/s]
80%| | 39206/49041 [00:47<00:12, 778.65it/s]
80%| | 39285/49041 [00:47<00:12, 780.33it/s]
80%| | 39364/49041 [00:47<00:12, 778.36it/s]
80%| | 39443/49041 [00:47<00:12, 772.12it/s]
81%| | 39531/49041 [00:47<00:11, 797.33it/s]
81%| | 39612/49041 [00:48<00:11, 789.97it/s]
81%| | 39695/49041 [00:48<00:11, 799.86it/s]
81%| | 39783/49041 [00:48<00:11, 820.65it/s]
81%|

```
| 39866/49041 [00:48<00:11, 821.65it/s]

 81%|
| 39949/49041 [00:48<00:11, 792.94it/s]

 82%|
| 40034/49041 [00:48<00:11, 806.34it/s]

 82%|
| 40115/49041 [00:48<00:11, 796.26it/s]

 82%|
| 40203/49041 [00:48<00:10, 815.66it/s]

 82%|
| 40287/49041 [00:48<00:10, 821.05it/s]

 82%|
| 40378/49041 [00:48<00:10, 843.80it/s]

 83%|
| 40463/49041 [00:49<00:10, 832.73it/s]

 83%|
| 40547/49041 [00:49<00:10, 823.24it/s]

 83%|
| 40631/49041 [00:49<00:10, 826.48it/s]

 83%|
| 40720/49041 [00:49<00:09, 842.75it/s]

 83%|
| 40806/49041 [00:49<00:09, 846.02it/s]

 83%|
| 40895/49041 [00:49<00:09, 856.94it/s]

 84%|
| 40982/49041 [00:49<00:09, 858.95it/s]

 84%|
| 41073/49041 [00:49<00:09, 871.82it/s]

 84%|
| 41162/49041 [00:49<00:09, 875.31it/s]

 84%|
| 41250/49041 [00:49<00:08, 869.61it/s]

 84%|
| 41338/49041 [00:50<00:08, 870.73it/s]

 84%|
| 41426/49041 [00:50<00:08, 866.47it/s]

 85%|
| 41515/49041 [00:50<00:08, 871.52it/s]

 85%|
| 41607/49041 [00:50<00:08, 881.13it/s]
```


85%| | 41696/49041 [00:50<00:08, 879.24it/s]
85%| | 41790/49041 [00:50<00:08, 894.22it/s]
85%| | 41880/49041 [00:50<00:08, 887.29it/s]
86%| | 41969/49041 [00:50<00:08, 873.18it/s]
86%| | 42057/49041 [00:50<00:08, 858.01it/s]
86%| | 42143/49041 [00:50<00:08, 841.76it/s]
86%| | 42228/49041 [00:51<00:08, 827.60it/s]
86%| | 42311/49041 [00:51<00:08, 821.60it/s]
86%| | 42396/49041 [00:51<00:08, 828.22it/s]
87%| | 42487/49041 [00:51<00:07, 849.43it/s]
87%| | 42573/49041 [00:51<00:07, 839.39it/s]
87%| | 42658/49041 [00:51<00:07, 838.28it/s]
87%| | 42742/49041 [00:51<00:07, 837.25it/s]
87%| | 42837/49041 [00:51<00:07, 866.42it/s]
88%| | 42926/49041 [00:51<00:07, 872.21it/s]
88%| | 43015/49041 [00:52<00:06, 876.26it/s]
88%| | 43103/49041 [00:52<00:06, 867.71it/s]
88%| | 43190/49041 [00:52<00:06, 857.54it/s]
88%| | 43277/49041 [00:52<00:06, 859.45it/s]
88%| | 43364/49041 [00:52<00:06, 858.18it/s]

89%| | 43450/49041 [00:52<00:06, 831.98it/s]
89%| | 43538/49041 [00:52<00:06, 844.05it/s]
89%| | 43634/49041 [00:52<00:06, 874.04it/s]
89%| | 43722/49041 [00:52<00:06, 873.91it/s]
89%| | 43810/49041 [00:52<00:06, 861.10it/s]
90%| | 43897/49041 [00:53<00:05, 859.26it/s]
90%| | 43984/49041 [00:53<00:05, 853.01it/s]
90%| | 44070/49041 [00:53<00:05, 854.29it/s]
90%| | 44156/49041 [00:53<00:05, 846.51it/s]
90%| | 44241/49041 [00:53<00:05, 835.76it/s]
90%| | 44326/49041 [00:53<00:05, 838.27it/s]
91%| | 44411/49041 [00:53<00:05, 841.16it/s]
91%| | 44504/49041 [00:53<00:05, 865.39it/s]
91%| | 44591/49041 [00:53<00:05, 835.15it/s]
91%| | 44678/49041 [00:53<00:05, 843.44it/s]
91%| | 44765/49041 [00:54<00:05, 849.49it/s]
91%| | 44851/49041 [00:54<00:04, 845.69it/s]
92%| | 44940/49041 [00:54<00:04, 856.70it/s]
92%| | 45026/49041 [00:54<00:04, 831.07it/s]
92%| | 45117/49041 [00:54<00:04, 849.07it/s]
92%|

```
|       | 45204/49041 [00:54<00:04, 853.40it/s]

 92%|
|       | 45290/49041 [00:54<00:04, 832.40it/s]

 93%|
|       | 45374/49041 [00:54<00:04, 815.77it/s]

 93%|
|       | 45457/49041 [00:54<00:04, 818.28it/s]

 93%|
|       | 45540/49041 [00:55<00:04, 805.57it/s]

 93%|
|       | 45622/49041 [00:55<00:04, 806.49it/s]

 93%|
|       | 45709/49041 [00:55<00:04, 823.87it/s]

 93%|
|       | 45794/49041 [00:55<00:03, 827.34it/s]

 94%|
|       | 45890/49041 [00:55<00:03, 861.49it/s]

 94%|
|       | 45977/49041 [00:55<00:03, 818.40it/s]

 94%|
|       | 46063/49041 [00:55<00:03, 828.69it/s]

 94%|
|       | 46157/49041 [00:55<00:03, 857.50it/s]

 94%|
|       | 46244/49041 [00:55<00:03, 851.87it/s]

 94%|
|       | 46330/49041 [00:55<00:03, 849.85it/s]

 95%|
|       | 46418/49041 [00:56<00:03, 856.53it/s]

 95%|
|       | 46510/49041 [00:56<00:02, 872.82it/s]

 95%|
|       | 46598/49041 [00:56<00:02, 845.52it/s]

 95%|
|       | 46688/49041 [00:56<00:02, 858.92it/s]

 95%|
|       | 46775/49041 [00:56<00:02, 834.27it/s]

 96%|
|       | 46862/49041 [00:56<00:02, 840.45it/s]

 96%|
|       | 46947/49041 [00:56<00:02, 831.68it/s]
```

96%|
| 47034/49041 [00:56<00:02, 840.96it/s]

96%|
| 47122/49041 [00:56<00:02, 851.34it/s]

96%|
| 47208/49041 [00:56<00:02, 826.99it/s]

96%|
| 47293/49041 [00:57<00:02, 833.11it/s]

97%|
| 47381/49041 [00:57<00:01, 846.06it/s]

97%|
| 47474/49041 [00:57<00:01, 865.42it/s]

97%|
| 47561/49041 [00:57<00:01, 857.25it/s]

97%|
| 47654/49041 [00:57<00:01, 874.69it/s]

97%|
| 47743/49041 [00:57<00:01, 878.46it/s]

98%|
| 47832/49041 [00:57<00:01, 869.65it/s]

98%|
| 47920/49041 [00:57<00:01, 850.72it/s]

98%|
| 48012/49041 [00:57<00:01, 868.58it/s]

98%|
| 48100/49041 [00:58<00:01, 859.91it/s]

98%|
| 48189/49041 [00:58<00:00, 866.87it/s]

98%|
| 48276/49041 [00:58<00:00, 861.88it/s]

99%|
| 48363/49041 [00:58<00:00, 832.85it/s]

99%|
| 48456/49041 [00:58<00:00, 858.05it/s]

99%|
| 48543/49041 [00:58<00:00, 842.31it/s]

99%|
| 48631/49041 [00:58<00:00, 852.63it/s]

99%|
| 48717/49041 [00:58<00:00, 854.00it/s]

```
100%|████████████████████████████████████████████████████████████
██████ | 48804/49041 [00:58<00:00, 858.01it/s]

100%|████████████████████████████████████████████████████████████
██████ | 48894/49041 [00:58<00:00, 865.84it/s]

100%|████████████████████████████████████████████████████████████
██████ | 48981/49041 [00:59<00:00, 852.51it/s]

100%|████████████████████████████████████████████████████████████
██████| 49041/49041 [00:59<00:00, 829.64it/s]

49041
300
```

Essay test data

In [163]:

```
# tfidf Word2Vec
# computing average word2vec for each Project Title is stored in this list

tfidf_w2v_vectors_text_test = []; # the tfidf-w2v for each essay
for sentence in tqdm(preprocessed_essays_Test): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the title
    for word in sentence.split(): # for each word in a title
        if (word in glove_words) and (word in dictionary):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_text_test.append(vector)

print(len(tfidf_w2v_vectors_text_test))
print(len(tfidf_w2v_vectors_text_test[0]))
```

```
  0%|
| 0/36052 [00:00<?, ?it/s]

  0%|
| 94/36052 [00:00<00:38, 933.02it/s]

  1%|▏
| 181/36052 [00:00<00:39, 911.08it/s]

  1%|▏
| 273/36052 [00:00<00:39, 911.75it/s]

  1%|▏
| 368/36052 [00:00<00:38, 920.94it/s]

  1%|▏
| 461/36052 [00:00<00:38, 921.63it/s]

  2%|▏
| 545/36052 [00:00<00:39, 893.43it/s]

  2%|▏
| 636/36052 [00:00<00:39, 896.48it/s]

  2%|▏
| 723/36052 [00:00<00:39, 883.62it/s]

  2%|▏
| 808/36052 [00:00<00:40, 872.46it/s]

  2%|▏
| 892/36052 [00:01<00:40, 860.45it/s]

  3%|▏
| 981/36052 [00:01<00:40, 867.17it/s]

  3%|▏
| 1066/36052 [00:01<00:41, 849.75it/s]

  3%|▏
| 1158/36052 [00:01<00:40, 867.28it/s]

  3%|▏
| 1246/36052 [00:01<00:39, 870.31it/s]

  4%|▏
| 1333/36052 [00:01<00:40, 853.11it/s]

  4%|▏
| 1424/36052 [00:01<00:39, 867.53it/s]

  4%|▏
| 1511/36052 [00:01<00:40, 853.65it/s]

  4%|▏
| 1597/36052 [00:01<00:40, 843.66it/s]

  5%|▏
| 1687/36052 [00:01<00:40, 857.81it/s]
```

```
  5%|███▉
| 1773/36052 [00:02<00:41, 834.22it/s]

  5%|███▉
| 1857/36052 [00:02<00:41, 831.72it/s]

  5%|████
| 1948/36052 [00:02<00:40, 852.00it/s]

  6%|████▏
| 2034/36052 [00:02<00:39, 852.53it/s]

  6%|████▏
| 2127/36052 [00:02<00:38, 872.53it/s]

  6%|████▍
| 2215/36052 [00:02<00:38, 870.20it/s]

  6%|████▍
| 2303/36052 [00:02<00:39, 856.00it/s]

  7%|████▋
| 2389/36052 [00:02<00:40, 823.46it/s]

  7%|████▊
| 2475/36052 [00:02<00:40, 833.80it/s]

  7%|████▉
| 2566/36052 [00:02<00:39, 853.54it/s]

  7%|█████
| 2658/36052 [00:03<00:38, 868.19it/s]

  8%|█████▏
| 2746/36052 [00:03<00:39, 849.71it/s]

  8%|█████▎
| 2832/36052 [00:03<00:39, 848.48it/s]

  8%|█████▍
| 2918/36052 [00:03<00:39, 845.06it/s]

  8%|█████▌
| 3003/36052 [00:03<00:40, 822.67it/s]

  9%|█████▋
| 3088/36052 [00:03<00:39, 828.86it/s]

  9%|█████▊
| 3173/36052 [00:03<00:39, 833.28it/s]

  9%|█████▉
| 3261/36052 [00:03<00:38, 845.05it/s]

  9%|██████
| 3346/36052 [00:03<00:38, 840.71it/s]

 10%|██████▏
| 3431/36052 [00:04<00:38, 841.65it/s]

 10%|██████▏
```

```
| 3517/36052 [00:04<00:38, 844.77it/s]

 10%|
| 3602/36052 [00:04<00:38, 834.50it/s]

 10%|
| 3695/36052 [00:04<00:37, 856.90it/s]

 10%|
| 3785/36052 [00:04<00:37, 867.49it/s]

 11%|
| 3872/36052 [00:04<00:37, 852.24it/s]

 11%|
| 3958/36052 [00:04<00:37, 854.55it/s]

 11%|
| 4045/36052 [00:04<00:37, 858.45it/s]

 11%|
| 4137/36052 [00:04<00:36, 872.96it/s]

 12%|
| 4225/36052 [00:04<00:36, 870.57it/s]

 12%|
| 4313/36052 [00:05<00:36, 863.80it/s]

 12%|
| 4404/36052 [00:05<00:36, 875.38it/s]

 12%|
| 4492/36052 [00:05<00:36, 856.95it/s]

 13%|
| 4578/36052 [00:05<00:37, 845.97it/s]

 13%|
| 4663/36052 [00:05<00:37, 835.39it/s]

 13%|
| 4747/36052 [00:05<00:37, 832.47it/s]

 13%|
| 4831/36052 [00:05<00:37, 827.92it/s]

 14%|
| 4923/36052 [00:05<00:36, 851.81it/s]

 14%|
| 5017/36052 [00:05<00:35, 874.69it/s]

 14%|
| 5108/36052 [00:05<00:35, 883.19it/s]

 14%|
| 5197/36052 [00:06<00:36, 845.63it/s]

 15%|
| 5283/36052 [00:06<00:36, 846.71it/s]
```

```
 15%|
| 5370/36052 [00:06<00:36, 851.69it/s]

 15%|
| 5458/36052 [00:06<00:35, 858.12it/s]

 15%|
| 5545/36052 [00:06<00:35, 857.25it/s]

 16%|
| 5632/36052 [00:06<00:35, 859.25it/s]

 16%|
| 5719/36052 [00:06<00:35, 853.00it/s]

 16%|
| 5809/36052 [00:06<00:34, 864.75it/s]

 16%|
| 5897/36052 [00:06<00:34, 867.32it/s]

 17%|
| 5984/36052 [00:06<00:35, 858.55it/s]

 17%|
| 6070/36052 [00:07<00:35, 852.10it/s]

 17%|
| 6156/36052 [00:07<00:35, 850.08it/s]

 17%|
| 6243/36052 [00:07<00:34, 854.02it/s]

 18%|
| 6329/36052 [00:07<00:34, 851.42it/s]

 18%|
| 6419/36052 [00:07<00:34, 863.31it/s]

 18%|
| 6506/36052 [00:07<00:34, 859.57it/s]

 18%|
| 6593/36052 [00:07<00:34, 849.56it/s]

 19%|
| 6679/36052 [00:07<00:34, 843.32it/s]

 19%|
| 6769/36052 [00:07<00:34, 855.40it/s]

 19%|
| 6863/36052 [00:07<00:33, 877.27it/s]

 19%|
| 6955/36052 [00:08<00:32, 887.77it/s]

 20%|
| 7044/36052 [00:08<00:33, 860.86it/s]
```

20%|
| 7135/36052 [00:08<00:33, 873.19it/s]

20%|
| 7223/36052 [00:08<00:33, 870.73it/s]

20%|
| 7313/36052 [00:08<00:32, 877.45it/s]

21%|
| 7405/36052 [00:08<00:32, 885.35it/s]

21%|
| 7495/36052 [00:08<00:32, 885.22it/s]

21%|
| 7584/36052 [00:08<00:32, 877.53it/s]

21%|
| 7672/36052 [00:08<00:32, 873.75it/s]

22%|
| 7761/36052 [00:09<00:32, 875.13it/s]

22%|
| 7849/36052 [00:09<00:33, 851.80it/s]

22%|
| 7940/36052 [00:09<00:32, 867.74it/s]

22%|
| 8027/36052 [00:09<00:32, 867.66it/s]

23%|
| 8114/36052 [00:09<00:32, 865.16it/s]

23%|
| 8201/36052 [00:09<00:32, 844.69it/s]

23%|
| 8286/36052 [00:09<00:32, 841.86it/s]

23%|
| 8371/36052 [00:09<00:33, 835.09it/s]

23%|
| 8458/36052 [00:09<00:32, 841.03it/s]

24%|
| 8543/36052 [00:09<00:33, 829.51it/s]

24%|
| 8627/36052 [00:10<00:33, 828.37it/s]

24%|
| 8710/36052 [00:10<00:33, 822.16it/s]

24%|
| 8796/36052 [00:10<00:32, 826.60it/s]

25%|

```
| 8879/36052 [00:10<00:33, 812.33it/s]

 25%|
| 8967/36052 [00:10<00:32, 827.46it/s]

 25%|
| 9054/36052 [00:10<00:32, 835.67it/s]

 25%|
| 9141/36052 [00:10<00:31, 841.44it/s]

 26%|
| 9232/36052 [00:10<00:31, 860.52it/s]

 26%|
| 9320/36052 [00:10<00:30, 863.78it/s]

 26%|
| 9407/36052 [00:10<00:31, 858.63it/s]

 26%|
| 9501/36052 [00:11<00:30, 877.33it/s]

 27%|
| 9589/36052 [00:11<00:30, 855.82it/s]

 27%|
| 9675/36052 [00:11<00:31, 850.07it/s]

 27%|
| 9763/36052 [00:11<00:30, 857.00it/s]

 27%|
| 9849/36052 [00:11<00:31, 828.94it/s]

 28%|
| 9933/36052 [00:11<00:31, 830.36it/s]

 28%|
| 10020/36052 [00:11<00:30, 840.08it/s]

 28%|
| 10112/36052 [00:11<00:30, 860.72it/s]

 28%|
| 10204/36052 [00:11<00:29, 875.85it/s]

 29%|
| 10292/36052 [00:12<00:29, 862.35it/s]

 29%|
| 10383/36052 [00:12<00:29, 872.65it/s]

 29%|
| 10471/36052 [00:12<00:29, 860.17it/s]

 29%|
| 10558/36052 [00:12<00:29, 859.74it/s]

 30%|
| 10645/36052 [00:12<00:30, 830.80it/s]
```

30%|
| 10734/36052 [00:12<00:29, 847.21it/s]

30%|
| 10820/36052 [00:12<00:30, 831.91it/s]

30%|
| 10904/36052 [00:12<00:30, 832.49it/s]

30%|
| 10992/36052 [00:12<00:29, 844.43it/s]

31%|
| 11077/36052 [00:12<00:29, 834.39it/s]

31%|
| 11161/36052 [00:13<00:29, 834.18it/s]

31%|
| 11245/36052 [00:13<00:30, 819.49it/s]

31%|
| 11336/36052 [00:13<00:29, 842.97it/s]

32%|
| 11421/36052 [00:13<00:29, 821.36it/s]

32%|
| 11513/36052 [00:13<00:29, 843.38it/s]

32%|
| 11603/36052 [00:13<00:28, 857.81it/s]

32%|
| 11690/36052 [00:13<00:28, 859.57it/s]

33%|
| 11782/36052 [00:13<00:27, 875.09it/s]

33%|
| 11870/36052 [00:13<00:27, 871.32it/s]

33%|
| 11958/36052 [00:13<00:28, 860.45it/s]

33%|
| 12047/36052 [00:14<00:27, 864.74it/s]

34%|
| 12146/36052 [00:14<00:26, 894.71it/s]

34%|
| 12236/36052 [00:14<00:26, 896.18it/s]

34%|
| 12326/36052 [00:14<00:26, 882.24it/s]

34%|
| 12415/36052 [00:14<00:27, 872.35it/s]

35%|
| 12503/36052 [00:14<00:27, 865.03it/s]

35%|
| 12592/36052 [00:14<00:26, 870.43it/s]

35%|
| 12680/36052 [00:14<00:26, 866.20it/s]

35%|
| 12768/36052 [00:14<00:26, 868.46it/s]

36%|
| 12857/36052 [00:15<00:26, 870.39it/s]

36%|
| 12945/36052 [00:15<00:26, 858.54it/s]

36%|
| 13031/36052 [00:15<00:27, 842.02it/s]

36%|
| 13119/36052 [00:15<00:27, 848.72it/s]

37%|
| 13208/36052 [00:15<00:26, 856.43it/s]

37%|
| 13304/36052 [00:15<00:25, 884.34it/s]

37%|
| 13395/36052 [00:15<00:25, 889.98it/s]

37%|
| 13485/36052 [00:15<00:26, 853.13it/s]

38%|
| 13572/36052 [00:15<00:26, 856.26it/s]

38%|
| 13658/36052 [00:15<00:26, 847.94it/s]

38%|
| 13750/36052 [00:16<00:25, 864.19it/s]

38%|
| 13838/36052 [00:16<00:25, 866.93it/s]

39%|
| 13930/36052 [00:16<00:25, 880.34it/s]

39%|
| 14019/36052 [00:16<00:25, 868.43it/s]

39%|
| 14107/36052 [00:16<00:25, 868.75it/s]

39%|
| 14195/36052 [00:16<00:25, 867.49it/s]

40%|

```
| 14284/36052 [00:16<00:24, 872.25it/s]

 40%|
| 14372/36052 [00:16<00:24, 872.59it/s]

 40%|
| 14460/36052 [00:16<00:24, 866.27it/s]

 40%|
| 14547/36052 [00:16<00:24, 866.38it/s]

 41%|
| 14634/36052 [00:17<00:25, 850.32it/s]

 41%|
| 14720/36052 [00:17<00:25, 843.83it/s]

 41%|
| 14810/36052 [00:17<00:24, 858.22it/s]

 41%|
| 14896/36052 [00:17<00:24, 854.26it/s]

 42%|
| 14983/36052 [00:17<00:24, 857.14it/s]

 42%|
| 15069/36052 [00:17<00:25, 821.85it/s]

 42%|
| 15158/36052 [00:17<00:24, 839.37it/s]

 42%|
| 15243/36052 [00:17<00:25, 830.94it/s]

 43%|
| 15327/36052 [00:17<00:24, 829.31it/s]

 43%|
| 15411/36052 [00:17<00:25, 816.25it/s]

 43%|
| 15496/36052 [00:18<00:25, 819.50it/s]

 43%|
| 15586/36052 [00:18<00:24, 840.31it/s]

 43%|
| 15671/36052 [00:18<00:24, 830.66it/s]

 44%|
| 15755/36052 [00:18<00:24, 829.17it/s]

 44%|
| 15839/36052 [00:18<00:24, 831.59it/s]

 44%|
| 15923/36052 [00:18<00:24, 828.44it/s]

 44%|
| 16009/36052 [00:18<00:24, 831.04it/s]
```

45%|
| 16096/36052 [00:18<00:23, 838.22it/s]

45%|
| 16180/36052 [00:18<00:23, 834.44it/s]

45%|
| 16270/36052 [00:19<00:23, 851.25it/s]

45%|
| 16356/36052 [00:19<00:23, 852.01it/s]

46%|
| 16443/36052 [00:19<00:22, 855.47it/s]

46%|
| 16534/36052 [00:19<00:22, 866.82it/s]

46%|
| 16621/36052 [00:19<00:23, 829.04it/s]

46%|
| 16708/36052 [00:19<00:23, 840.10it/s]

47%|
| 16793/36052 [00:19<00:22, 840.40it/s]

47%|
| 16878/36052 [00:19<00:22, 841.43it/s]

47%|
| 16963/36052 [00:19<00:22, 840.74it/s]

47%|
| 17050/36052 [00:19<00:22, 847.57it/s]

48%|
| 17140/36052 [00:20<00:22, 858.31it/s]

48%|
| 17226/36052 [00:20<00:22, 839.39it/s]

48%|
| 17319/36052 [00:20<00:21, 862.39it/s]

48%|
| 17406/36052 [00:20<00:22, 842.03it/s]

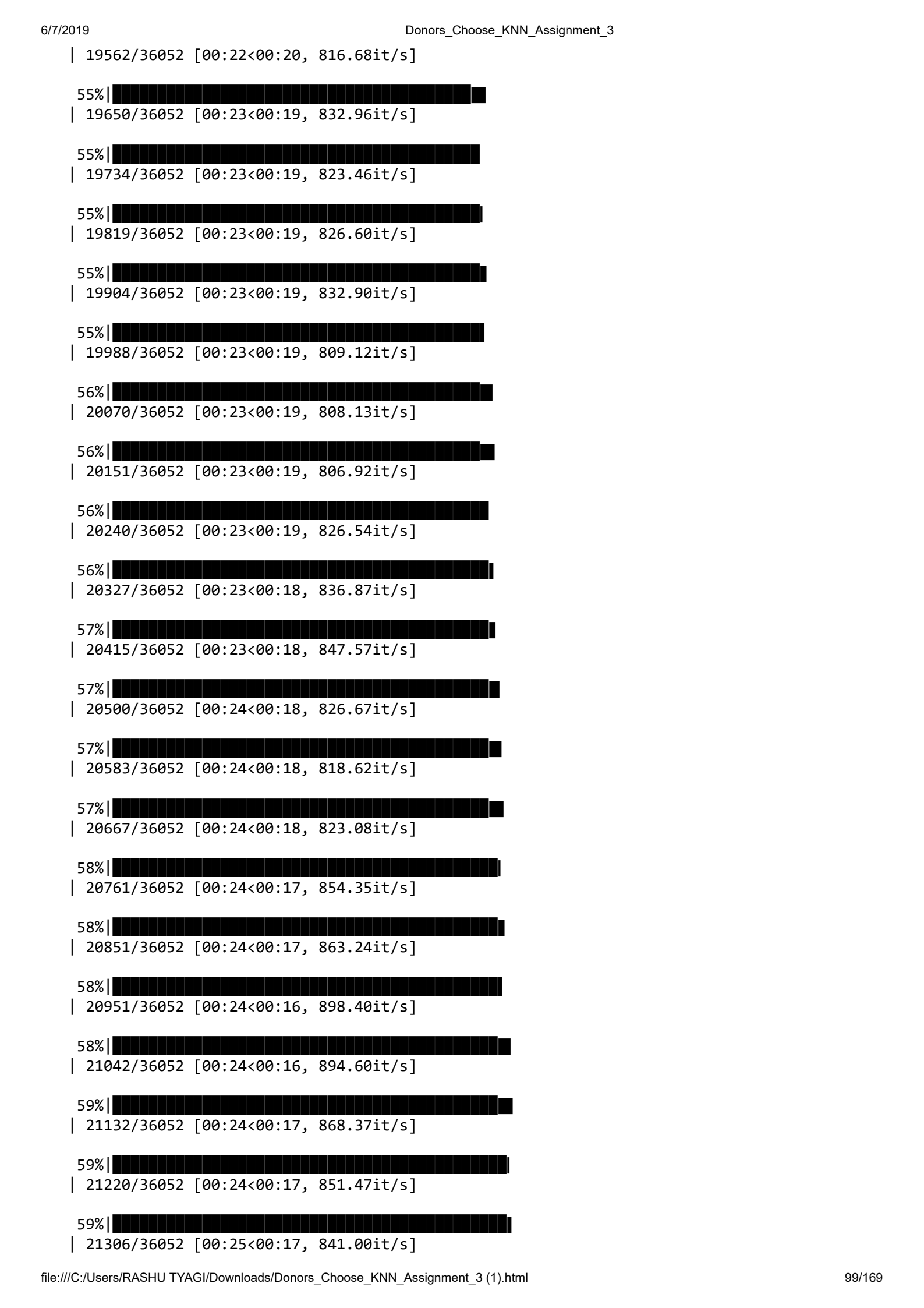
49%|
| 17491/36052 [00:20<00:22, 840.07it/s]

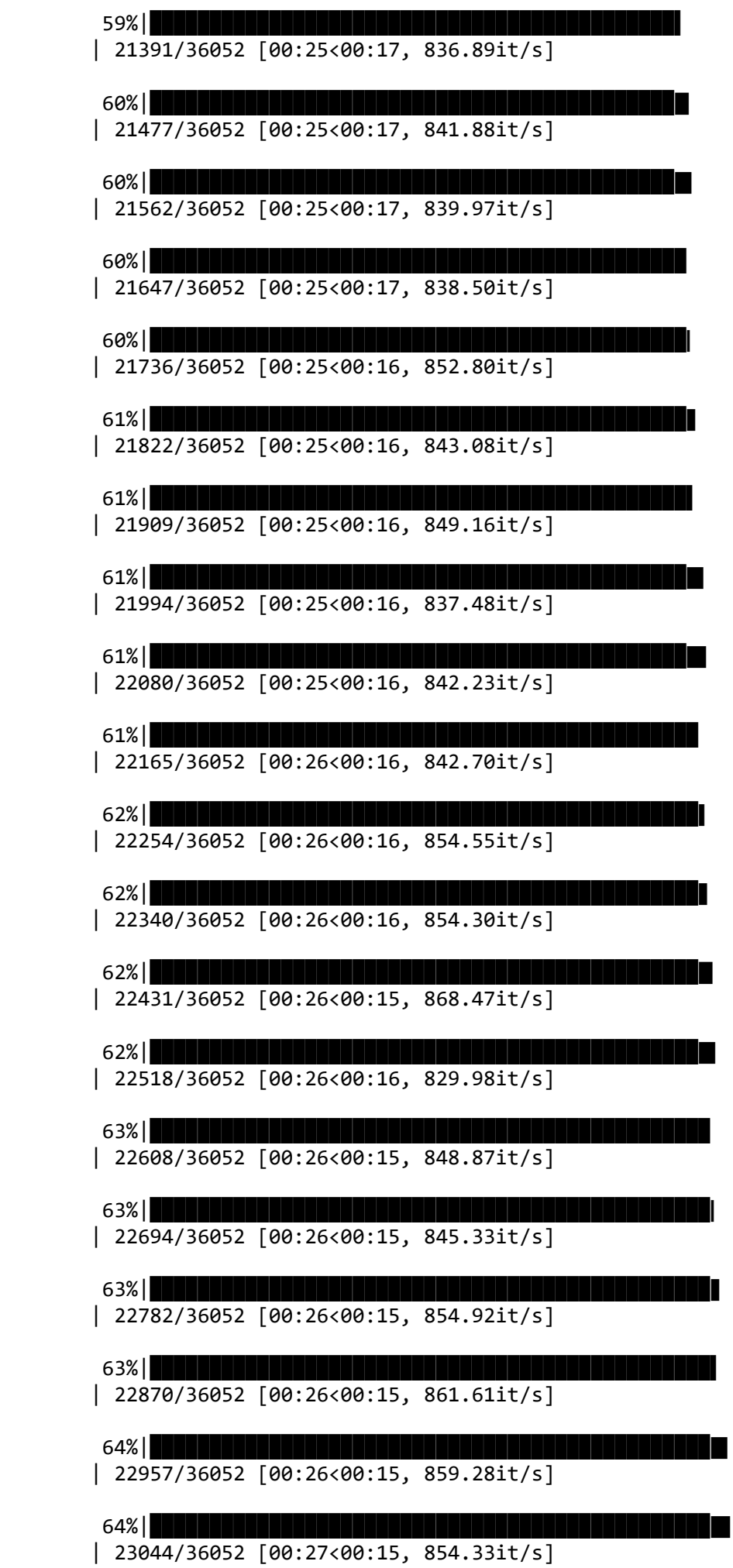
49%|
| 17581/36052 [00:20<00:21, 855.48it/s]

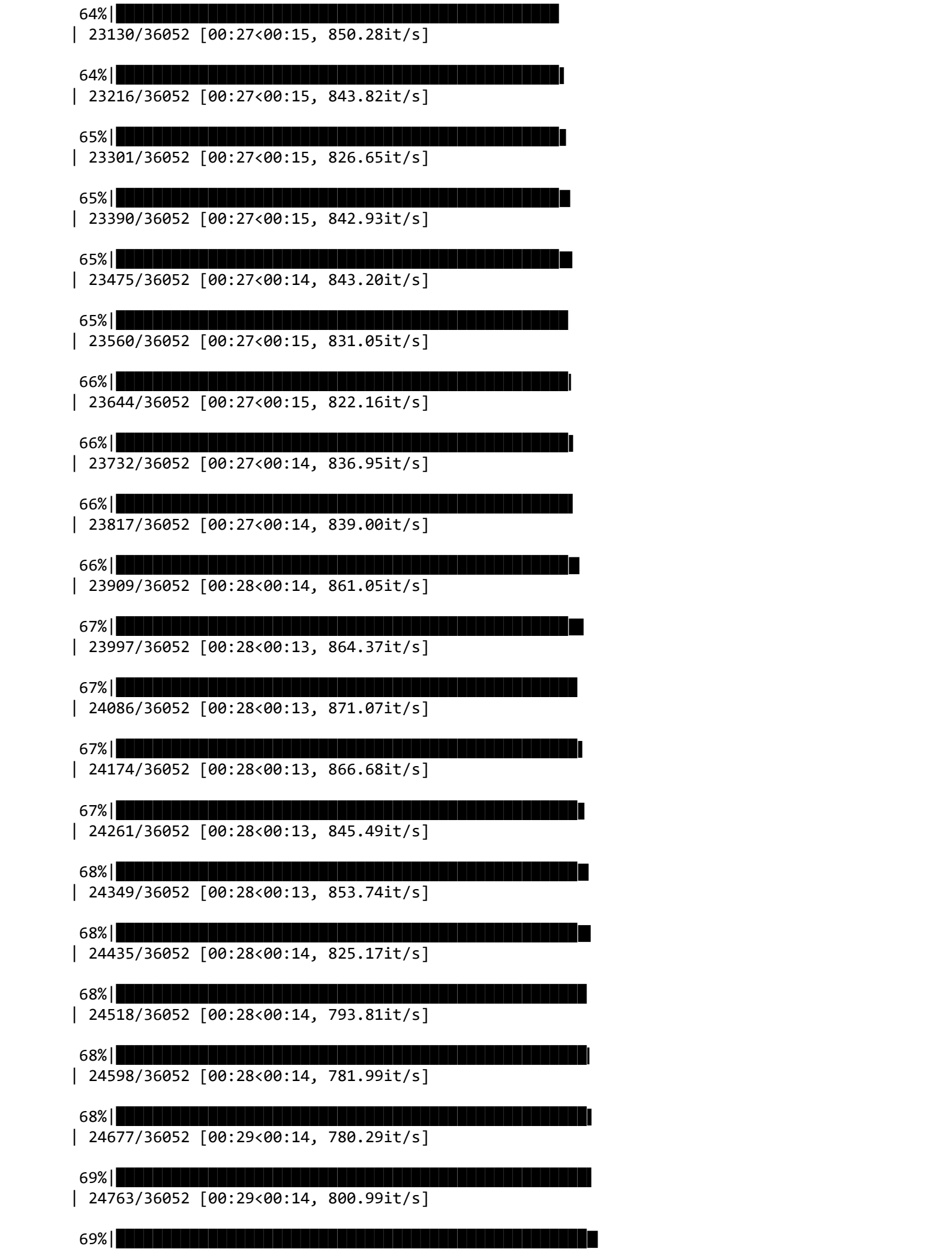
49%|
| 17667/36052 [00:20<00:21, 852.42it/s]

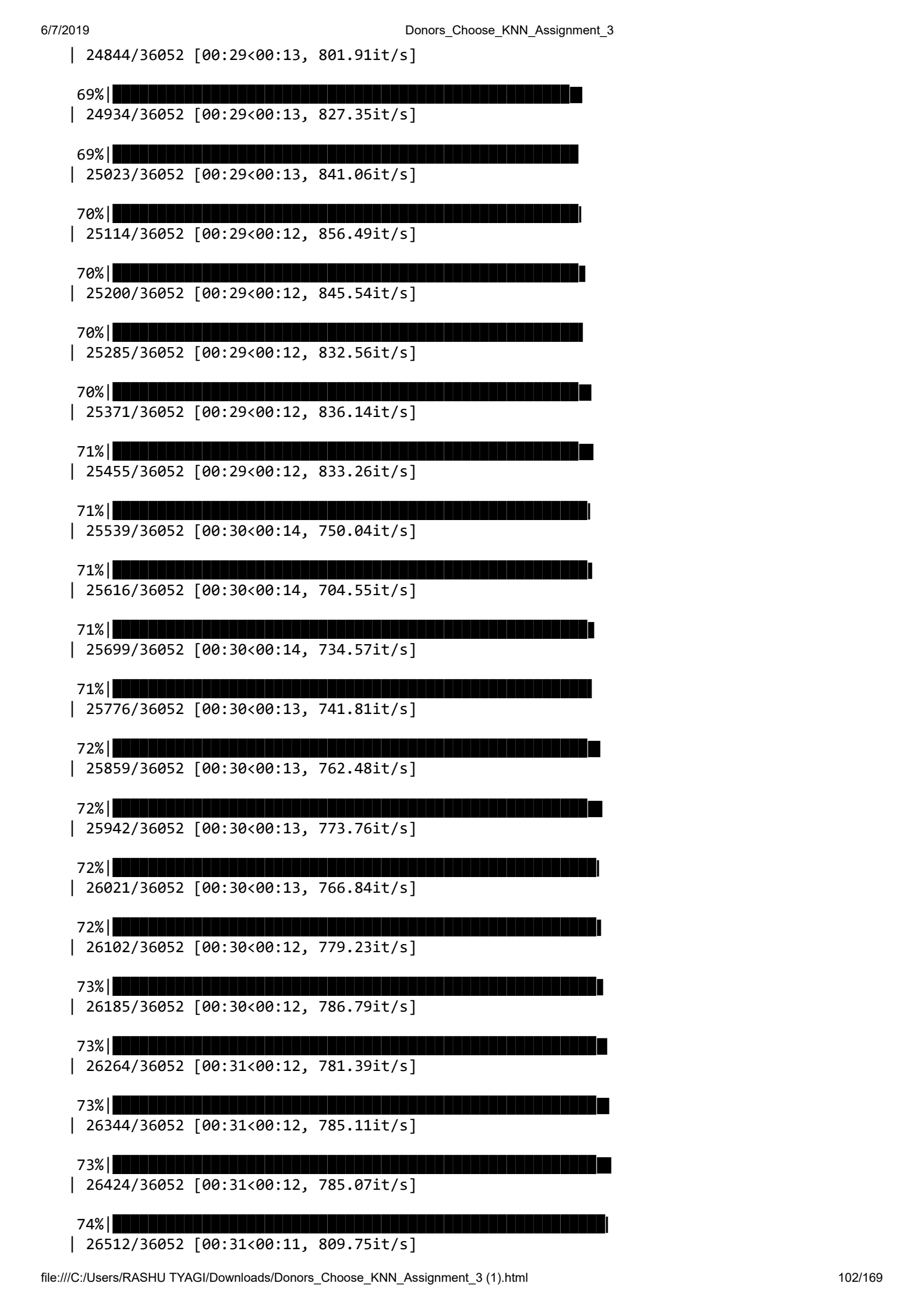
49%|
| 17753/36052 [00:20<00:21, 854.44it/s]

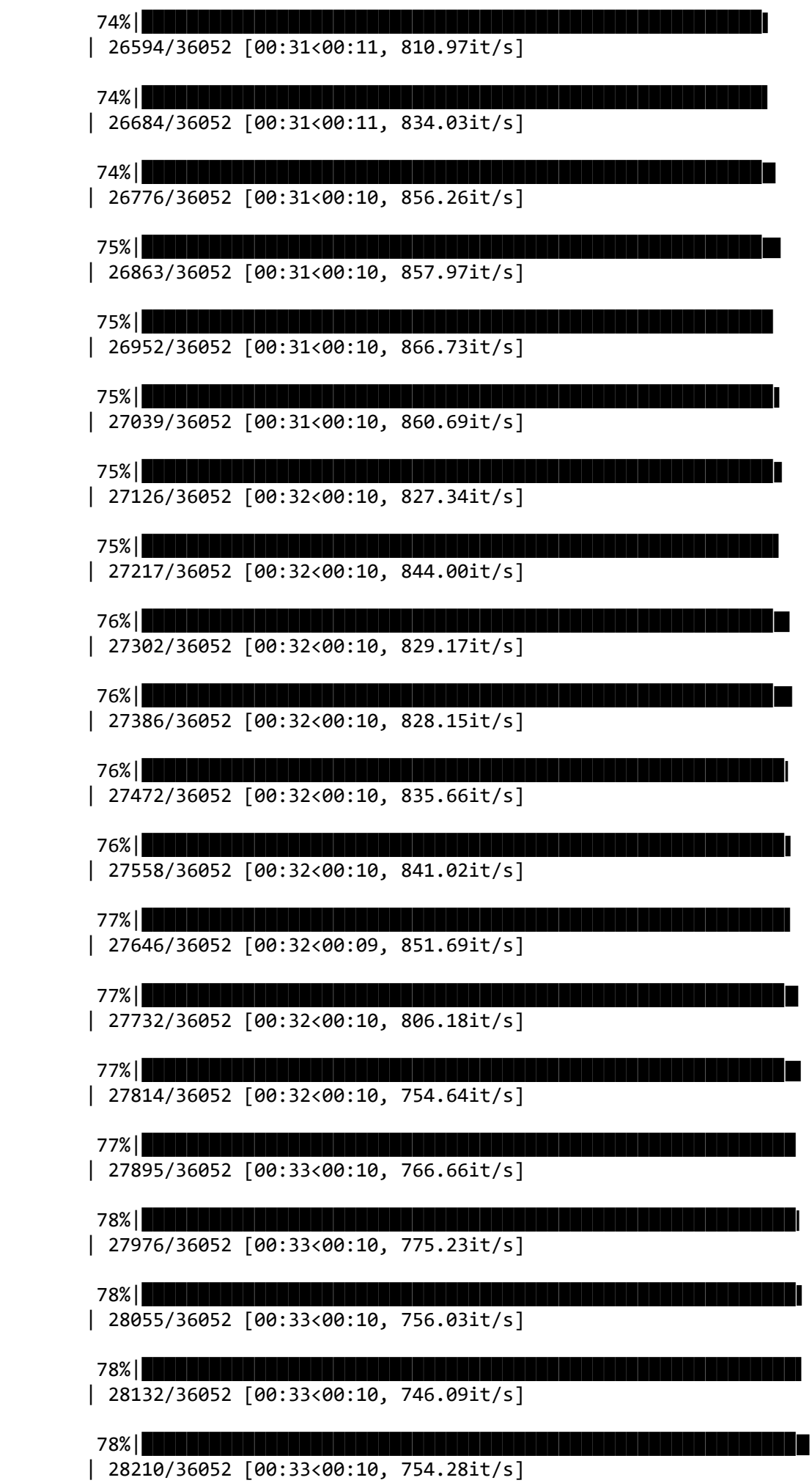
```
 49%|
| 17839/36052 [00:20<00:21, 846.63it/s]

 50%|
| 17926/36052 [00:20<00:21, 851.66it/s]

 50%|
| 18012/36052 [00:21<00:21, 844.64it/s]

 50%|
| 18097/36052 [00:21<00:21, 829.61it/s]

 50%|
| 18181/36052 [00:21<00:21, 828.44it/s]

 51%|
| 18268/36052 [00:21<00:21, 837.49it/s]

 51%|
| 18357/36052 [00:21<00:20, 852.47it/s]

 51%|
| 18443/36052 [00:21<00:20, 848.91it/s]

 51%|
| 18528/36052 [00:21<00:20, 839.86it/s]

 52%|
| 18616/36052 [00:21<00:20, 849.65it/s]

 52%|
| 18702/36052 [00:21<00:20, 848.37it/s]

 52%|
| 18787/36052 [00:22<00:21, 820.17it/s]

 52%|
| 18870/36052 [00:22<00:20, 818.88it/s]

 53%|
| 18957/36052 [00:22<00:20, 831.62it/s]

 53%|
| 19043/36052 [00:22<00:20, 839.39it/s]

 53%|
| 19128/36052 [00:22<00:20, 822.65it/s]

 53%|
| 19220/36052 [00:22<00:19, 847.84it/s]

 54%|
| 19306/36052 [00:22<00:20, 813.61it/s]

 54%|
| 19392/36052 [00:22<00:20, 825.25it/s]

 54%|
| 19478/36052 [00:22<00:19, 833.53it/s]

 54%|
```


| 19562/36052 [00:22<00:20, 816.68it/s]
55%|
| 19650/36052 [00:23<00:19, 832.96it/s]
55%|
| 19734/36052 [00:23<00:19, 823.46it/s]
55%|
| 19819/36052 [00:23<00:19, 826.60it/s]
55%|
| 19904/36052 [00:23<00:19, 832.90it/s]
55%|
| 19988/36052 [00:23<00:19, 809.12it/s]
56%|
| 20070/36052 [00:23<00:19, 808.13it/s]
56%|
| 20151/36052 [00:23<00:19, 806.92it/s]
56%|
| 20240/36052 [00:23<00:19, 826.54it/s]
56%|
| 20327/36052 [00:23<00:18, 836.87it/s]
57%|
| 20415/36052 [00:23<00:18, 847.57it/s]
57%|
| 20500/36052 [00:24<00:18, 826.67it/s]
57%|
| 20583/36052 [00:24<00:18, 818.62it/s]
57%|
| 20667/36052 [00:24<00:18, 823.08it/s]
58%|
| 20761/36052 [00:24<00:17, 854.35it/s]
58%|
| 20851/36052 [00:24<00:17, 863.24it/s]
58%|
| 20951/36052 [00:24<00:16, 898.40it/s]
58%|
| 21042/36052 [00:24<00:16, 894.60it/s]
59%|
| 21132/36052 [00:24<00:17, 868.37it/s]
59%|
| 21220/36052 [00:24<00:17, 851.47it/s]
59%|
| 21306/36052 [00:25<00:17, 841.00it/s]


59%|
| 21391/36052 [00:25<00:17, 836.89it/s]
60%|
| 21477/36052 [00:25<00:17, 841.88it/s]
60%|
| 21562/36052 [00:25<00:17, 839.97it/s]
60%|
| 21647/36052 [00:25<00:17, 838.50it/s]
60%|
| 21736/36052 [00:25<00:16, 852.80it/s]
61%|
| 21822/36052 [00:25<00:16, 843.08it/s]
61%|
| 21909/36052 [00:25<00:16, 849.16it/s]
61%|
| 21994/36052 [00:25<00:16, 837.48it/s]
61%|
| 22080/36052 [00:25<00:16, 842.23it/s]
61%|
| 22165/36052 [00:26<00:16, 842.70it/s]
62%|
| 22254/36052 [00:26<00:16, 854.55it/s]
62%|
| 22340/36052 [00:26<00:16, 854.30it/s]
62%|
| 22431/36052 [00:26<00:15, 868.47it/s]
62%|
| 22518/36052 [00:26<00:16, 829.98it/s]
63%|
| 22608/36052 [00:26<00:15, 848.87it/s]
63%|
| 22694/36052 [00:26<00:15, 845.33it/s]
63%|
| 22782/36052 [00:26<00:15, 854.92it/s]
63%|
| 22870/36052 [00:26<00:15, 861.61it/s]
64%|
| 22957/36052 [00:26<00:15, 859.28it/s]
64%|
| 23044/36052 [00:27<00:15, 854.33it/s]


64%| | 23130/36052 [00:27<00:15, 850.28it/s]
64%| | 23216/36052 [00:27<00:15, 843.82it/s]
65%| | 23301/36052 [00:27<00:15, 826.65it/s]
65%| | 23390/36052 [00:27<00:15, 842.93it/s]
65%| | 23475/36052 [00:27<00:14, 843.20it/s]
65%| | 23560/36052 [00:27<00:15, 831.05it/s]
66%| | 23644/36052 [00:27<00:15, 822.16it/s]
66%| | 23732/36052 [00:27<00:14, 836.95it/s]
66%| | 23817/36052 [00:27<00:14, 839.00it/s]
66%| | 23909/36052 [00:28<00:14, 861.05it/s]
67%| | 23997/36052 [00:28<00:13, 864.37it/s]
67%| | 24086/36052 [00:28<00:13, 871.07it/s]
67%| | 24174/36052 [00:28<00:13, 866.68it/s]
67%| | 24261/36052 [00:28<00:13, 845.49it/s]
68%| | 24349/36052 [00:28<00:13, 853.74it/s]
68%| | 24435/36052 [00:28<00:14, 825.17it/s]
68%| | 24518/36052 [00:28<00:14, 793.81it/s]
68%| | 24598/36052 [00:28<00:14, 781.99it/s]
68%| | 24677/36052 [00:29<00:14, 780.29it/s]
69%| | 24763/36052 [00:29<00:14, 800.99it/s]
69%|


| 24844/36052 [00:29<00:13, 801.91it/s]
69%|
| 24934/36052 [00:29<00:13, 827.35it/s]
69%|
| 25023/36052 [00:29<00:13, 841.06it/s]
70%|
| 25114/36052 [00:29<00:12, 856.49it/s]
70%|
| 25200/36052 [00:29<00:12, 845.54it/s]
70%|
| 25285/36052 [00:29<00:12, 832.56it/s]
70%|
| 25371/36052 [00:29<00:12, 836.14it/s]
71%|
| 25455/36052 [00:29<00:12, 833.26it/s]
71%|
| 25539/36052 [00:30<00:14, 750.04it/s]
71%|
| 25616/36052 [00:30<00:14, 704.55it/s]
71%|
| 25699/36052 [00:30<00:14, 734.57it/s]
71%|
| 25776/36052 [00:30<00:13, 741.81it/s]
72%|
| 25859/36052 [00:30<00:13, 762.48it/s]
72%|
| 25942/36052 [00:30<00:13, 773.76it/s]
72%|
| 26021/36052 [00:30<00:13, 766.84it/s]
72%|
| 26102/36052 [00:30<00:12, 779.23it/s]
73%|
| 26185/36052 [00:30<00:12, 786.79it/s]
73%|
| 26264/36052 [00:31<00:12, 781.39it/s]
73%|
| 26344/36052 [00:31<00:12, 785.11it/s]
73%|
| 26424/36052 [00:31<00:12, 785.07it/s]
74%|
| 26512/36052 [00:31<00:11, 809.75it/s]


74%| | 26594/36052 [00:31<00:11, 810.97it/s]
74%| | 26684/36052 [00:31<00:11, 834.03it/s]
74%| | 26776/36052 [00:31<00:10, 856.26it/s]
75%| | 26863/36052 [00:31<00:10, 857.97it/s]
75%| | 26952/36052 [00:31<00:10, 866.73it/s]
75%| | 27039/36052 [00:31<00:10, 860.69it/s]
75%| | 27126/36052 [00:32<00:10, 827.34it/s]
75%| | 27217/36052 [00:32<00:10, 844.00it/s]
76%| | 27302/36052 [00:32<00:10, 829.17it/s]
76%| | 27386/36052 [00:32<00:10, 828.15it/s]
76%| | 27472/36052 [00:32<00:10, 835.66it/s]
76%| | 27558/36052 [00:32<00:10, 841.02it/s]
77%| | 27646/36052 [00:32<00:09, 851.69it/s]
77%| | 27732/36052 [00:32<00:10, 806.18it/s]
77%| | 27814/36052 [00:32<00:10, 754.64it/s]
77%| | 27895/36052 [00:33<00:10, 766.66it/s]
78%| | 27976/36052 [00:33<00:10, 775.23it/s]
78%| | 28055/36052 [00:33<00:10, 756.03it/s]
78%| | 28132/36052 [00:33<00:10, 746.09it/s]
78%| | 28210/36052 [00:33<00:10, 754.28it/s]

78%|
| 28286/36052 [00:33<00:10, 753.29it/s]

79%|
| 28374/36052 [00:33<00:09, 783.59it/s]

79%|
| 28461/36052 [00:33<00:09, 803.72it/s]

79%|
| 28558/36052 [00:33<00:08, 845.55it/s]

79%|
| 28644/36052 [00:33<00:08, 833.31it/s]

80%|
| 28728/36052 [00:34<00:08, 828.50it/s]

80%|
| 28816/36052 [00:34<00:08, 842.74it/s]

80%|
| 28901/36052 [00:34<00:08, 843.06it/s]

80%|
| 28988/36052 [00:34<00:08, 848.88it/s]

81%|
| 29074/36052 [00:34<00:08, 837.95it/s]

81%|
| 29158/36052 [00:34<00:08, 824.25it/s]

81%|
| 29243/36052 [00:34<00:08, 828.60it/s]

81%|
| 29326/36052 [00:34<00:08, 822.30it/s]

82%|
| 29409/36052 [00:34<00:08, 817.95it/s]

82%|
| 29491/36052 [00:34<00:08, 793.50it/s]

82%|
| 29579/36052 [00:35<00:07, 816.52it/s]

82%|
| 29661/36052 [00:35<00:07, 805.04it/s]

83%|
| 29747/36052 [00:35<00:07, 816.69it/s]

83%|
| 29831/36052 [00:35<00:07, 816.99it/s]

83%|
| 29917/36052 [00:35<00:07, 827.69it/s]

83%|

| 30004/36052 [00:35<00:07, 838.23it/s]

83%|
| 30088/36052 [00:35<00:07, 815.06it/s]

84%|
| 30175/36052 [00:35<00:07, 829.06it/s]

84%|
| 30259/36052 [00:35<00:07, 821.00it/s]

84%|
| 30345/36052 [00:36<00:06, 830.56it/s]

84%|
| 30429/36052 [00:36<00:06, 805.33it/s]

85%|
| 30514/36052 [00:36<00:06, 814.10it/s]

85%|
| 30596/36052 [00:36<00:06, 812.74it/s]

85%|
| 30681/36052 [00:36<00:06, 821.56it/s]

85%|
| 30765/36052 [00:36<00:06, 826.11it/s]

86%|
| 30856/36052 [00:36<00:06, 847.80it/s]

86%|
| 30941/36052 [00:36<00:06, 810.39it/s]

86%|
| 31029/36052 [00:36<00:06, 828.36it/s]

86%|
| 31114/36052 [00:36<00:05, 832.81it/s]

87%|
| 31200/36052 [00:37<00:05, 838.84it/s]

87%|
| 31288/36052 [00:37<00:05, 849.80it/s]

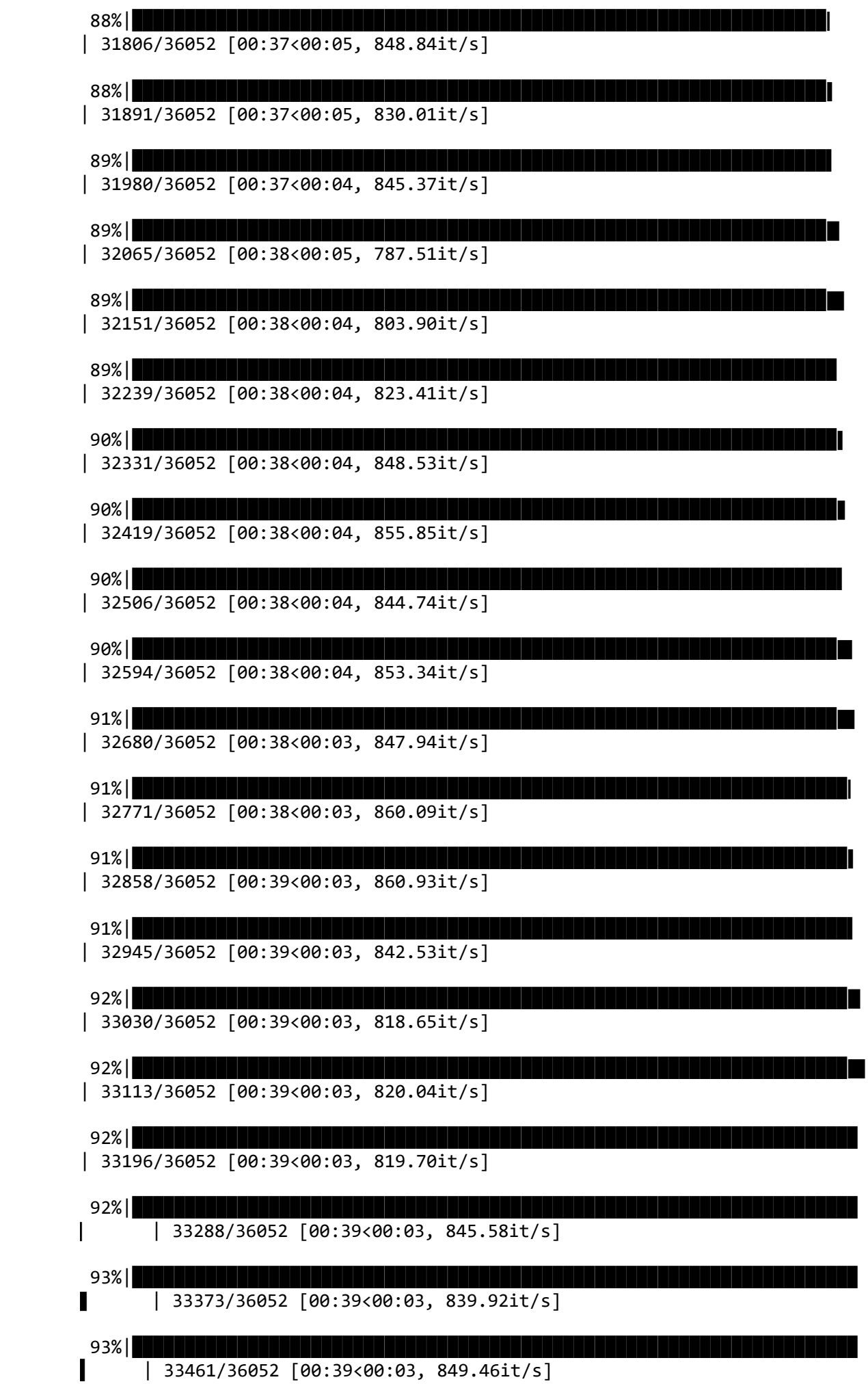
87%|
| 31374/36052 [00:37<00:05, 830.15it/s]

87%|
| 31458/36052 [00:37<00:05, 813.09it/s]

87%|
| 31544/36052 [00:37<00:05, 825.91it/s]

88%|
| 31632/36052 [00:37<00:05, 839.51it/s]

88%|
| 31720/36052 [00:37<00:05, 846.75it/s]


88%| | 31806/36052 [00:37<00:05, 848.84it/s]
88%| | 31891/36052 [00:37<00:05, 830.01it/s]
89%| | 31980/36052 [00:37<00:04, 845.37it/s]
89%| | 32065/36052 [00:38<00:05, 787.51it/s]
89%| | 32151/36052 [00:38<00:04, 803.90it/s]
89%| | 32239/36052 [00:38<00:04, 823.41it/s]
90%| | 32331/36052 [00:38<00:04, 848.53it/s]
90%| | 32419/36052 [00:38<00:04, 855.85it/s]
90%| | 32506/36052 [00:38<00:04, 844.74it/s]
90%| | 32594/36052 [00:38<00:04, 853.34it/s]
91%| | 32680/36052 [00:38<00:03, 847.94it/s]
91%| | 32771/36052 [00:38<00:03, 860.09it/s]
91%| | 32858/36052 [00:39<00:03, 860.93it/s]
91%| | 32945/36052 [00:39<00:03, 842.53it/s]
92%| | 33030/36052 [00:39<00:03, 818.65it/s]
92%| | 33113/36052 [00:39<00:03, 820.04it/s]
92%| | 33196/36052 [00:39<00:03, 819.70it/s]
92%| | 33288/36052 [00:39<00:03, 845.58it/s]
93%| | 33373/36052 [00:39<00:03, 839.92it/s]
93%| | 33461/36052 [00:39<00:03, 849.46it/s]

93%|▉ | 33553/36052 [00:39<00:02, 868.14it/s]

93%|▉ | 33642/36052 [00:39<00:02, 873.30it/s]

94%|▉ | 33731/36052 [00:40<00:02, 873.19it/s]

94%|▉ | 33821/36052 [00:40<00:02, 876.32it/s]

94%|▉ | 33909/36052 [00:40<00:02, 859.92it/s]

94%|▉ | 33996/36052 [00:40<00:02, 841.12it/s]

95%|▉ | 34081/36052 [00:40<00:02, 837.44it/s]

95%|▉ | 34170/36052 [00:40<00:02, 850.35it/s]

95%|▉ | 34256/36052 [00:40<00:02, 815.30it/s]

95%|▉ | 34347/36052 [00:40<00:02, 841.00it/s]

96%|▉ | 34432/36052 [00:40<00:01, 818.66it/s]

96%|▉ | 34517/36052 [00:40<00:01, 827.10it/s]

96%|▉ | 34601/36052 [00:41<00:01, 829.14it/s]

96%|▉ | 34689/36052 [00:41<00:01, 840.65it/s]

96%|▉ | 34776/36052 [00:41<00:01, 845.71it/s]

97%|▉ | 34862/36052 [00:41<00:01, 847.46it/s]

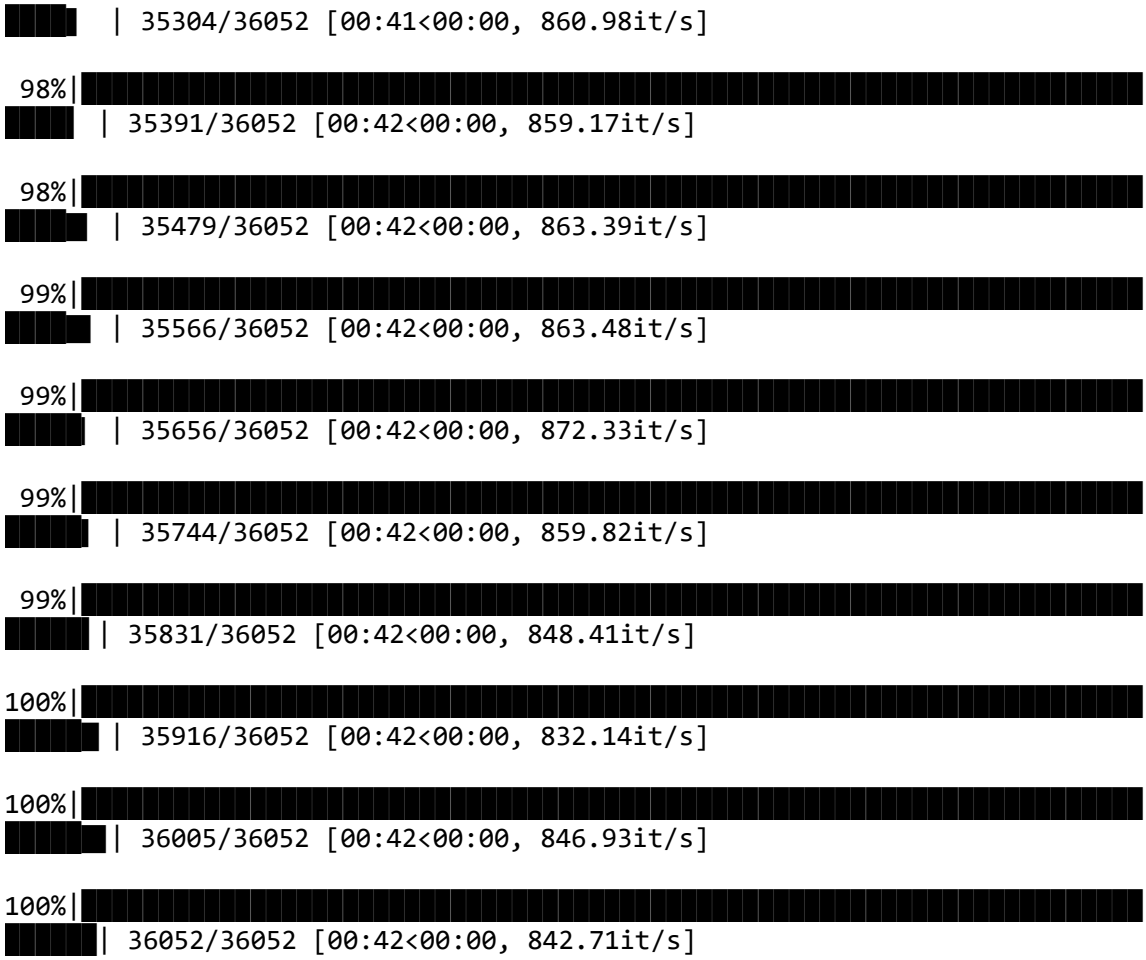
97%|▉ | 34952/36052 [00:41<00:01, 860.55it/s]

97%|▉ | 35040/36052 [00:41<00:01, 866.06it/s]

97%|▉ | 35128/36052 [00:41<00:01, 868.31it/s]

98%|▉ | 35215/36052 [00:41<00:00, 850.41it/s]

98%|




```
36052
300
```

Essay cross validation

In [164]:

```
# tfidf Word2Vec
# computing average word2vec for each Project Title is stored in this list

tfidf_w2v_vectors_text_cv = []; # the tfidf-w2v for each essay
for sentence in tqdm(preprocessed_essays_Cv): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the title
    for word in sentence.split(): # for each word in a title
        if (word in glove_words) and (word in dictionary):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_text_cv.append(vector)

print(len(tfidf_w2v_vectors_text_cv))
print(len(tfidf_w2v_vectors_text_cv[0]))
```

```
  0%|
| 0/24155 [00:00<?, ?it/s]

  0%|▌
| 82/24155 [00:00<00:29, 814.07it/s]

  1%|▌
| 171/24155 [00:00<00:28, 833.73it/s]

  1%|▉
| 260/24155 [00:00<00:28, 848.08it/s]

  1%|▉
| 342/24155 [00:00<00:28, 837.49it/s]

  2%|▉
| 429/24155 [00:00<00:28, 845.26it/s]

  2%|▉
| 509/24155 [00:00<00:28, 829.21it/s]

  2%|▉
| 583/24155 [00:00<00:29, 795.86it/s]

  3%|▉
| 657/24155 [00:00<00:30, 771.55it/s]

  3%|▉
| 743/24155 [00:00<00:29, 794.49it/s]

  3%|▉
| 828/24155 [00:01<00:28, 808.69it/s]

  4%|▉
| 907/24155 [00:01<00:29, 798.84it/s]

  4%|▉
| 988/24155 [00:01<00:28, 800.42it/s]

  4%|▉
| 1080/24155 [00:01<00:27, 831.24it/s]

  5%|▉
| 1163/24155 [00:01<00:28, 812.06it/s]

  5%|▉
| 1256/24155 [00:01<00:27, 842.44it/s]

  6%|▉
| 1341/24155 [00:01<00:28, 795.65it/s]

  6%|▉
| 1436/24155 [00:01<00:27, 834.80it/s]

  6%|▉
| 1521/24155 [00:01<00:27, 830.15it/s]

  7%|▉
| 1610/24155 [00:01<00:26, 845.55it/s]
```

```
  7%|▋         | 1696/24155 [00:02<00:26, 838.11it/s]

  7%|▋         | 1786/24155 [00:02<00:26, 851.49it/s]

  8%|▊         | 1874/24155 [00:02<00:25, 858.02it/s]

  8%|▊         | 1964/24155 [00:02<00:25, 865.92it/s]

  8%|▊         | 2051/24155 [00:02<00:26, 849.96it/s]

  9%|▊         | 2137/24155 [00:02<00:25, 851.15it/s]

  9%|▉         | 2223/24155 [00:02<00:26, 841.95it/s]

 10%|▉         | 2308/24155 [00:02<00:26, 830.20it/s]

 10%|▉         | 2395/24155 [00:02<00:26, 835.21it/s]

 10%|▉         | 2484/24155 [00:02<00:25, 849.08it/s]

 11%|▉         | 2570/24155 [00:03<00:25, 835.71it/s]

 11%|▉         | 2654/24155 [00:03<00:25, 827.71it/s]

 11%|▉         | 2743/24155 [00:03<00:25, 841.39it/s]

 12%|▉         | 2828/24155 [00:03<00:25, 834.70it/s]

 12%|▉         | 2912/24155 [00:03<00:25, 831.92it/s]

 12%|▉         | 2996/24155 [00:03<00:25, 832.52it/s]

 13%|▉         | 3080/24155 [00:03<00:26, 806.60it/s]

 13%|▉         | 3168/24155 [00:03<00:25, 823.29it/s]

 13%|▉         | 3254/24155 [00:03<00:25, 832.21it/s]

 14%|▉         | 3338/24155 [00:04<00:25, 818.15it/s]

 14%|▉         |
```

```
| 3420/24155 [00:04<00:25, 816.93it/s]

 15%|
| 3503/24155 [00:04<00:25, 819.09it/s]

 15%|
| 3588/24155 [00:04<00:24, 823.96it/s]

 15%|
| 3671/24155 [00:04<00:24, 823.90it/s]

 16%|
| 3754/24155 [00:04<00:24, 821.50it/s]

 16%|
| 3841/24155 [00:04<00:24, 833.72it/s]

 16%|
| 3925/24155 [00:04<00:24, 833.77it/s]

 17%|
| 4010/24155 [00:04<00:24, 836.76it/s]

 17%|
| 4099/24155 [00:04<00:23, 850.26it/s]

 17%|
| 4187/24155 [00:05<00:23, 857.14it/s]

 18%|
| 4280/24155 [00:05<00:22, 875.97it/s]

 18%|
| 4370/24155 [00:05<00:22, 878.57it/s]

 18%|
| 4458/24155 [00:05<00:22, 859.20it/s]

 19%|
| 4545/24155 [00:05<00:23, 847.93it/s]

 19%|
| 4630/24155 [00:05<00:23, 829.46it/s]

 20%|
| 4719/24155 [00:05<00:23, 844.98it/s]

 20%|
| 4806/24155 [00:05<00:22, 850.48it/s]

 20%|
| 4896/24155 [00:05<00:22, 862.96it/s]

 21%|
| 4988/24155 [00:05<00:21, 874.98it/s]

 21%|
| 5076/24155 [00:06<00:22, 846.87it/s]

 21%|
| 5165/24155 [00:06<00:22, 857.49it/s]
```

22%|
| 5251/24155 [00:06<00:22, 843.80it/s]

22%|
| 5336/24155 [00:06<00:22, 838.82it/s]

22%|
| 5423/24155 [00:06<00:22, 846.13it/s]

23%|
| 5513/24155 [00:06<00:21, 857.34it/s]

23%|
| 5605/24155 [00:06<00:21, 873.40it/s]

24%|
| 5702/24155 [00:06<00:20, 895.99it/s]

24%|
| 5792/24155 [00:06<00:21, 869.36it/s]

24%|
| 5880/24155 [00:07<00:21, 848.11it/s]

25%|
| 5966/24155 [00:07<00:22, 825.33it/s]

25%|
| 6054/24155 [00:07<00:21, 839.25it/s]

25%|
| 6141/24155 [00:07<00:21, 846.50it/s]

26%|
| 6231/24155 [00:07<00:20, 859.99it/s]

26%|
| 6318/24155 [00:07<00:21, 843.62it/s]

27%|
| 6407/24155 [00:07<00:20, 855.28it/s]

27%|
| 6493/24155 [00:07<00:20, 849.70it/s]

27%|
| 6583/24155 [00:07<00:20, 859.91it/s]

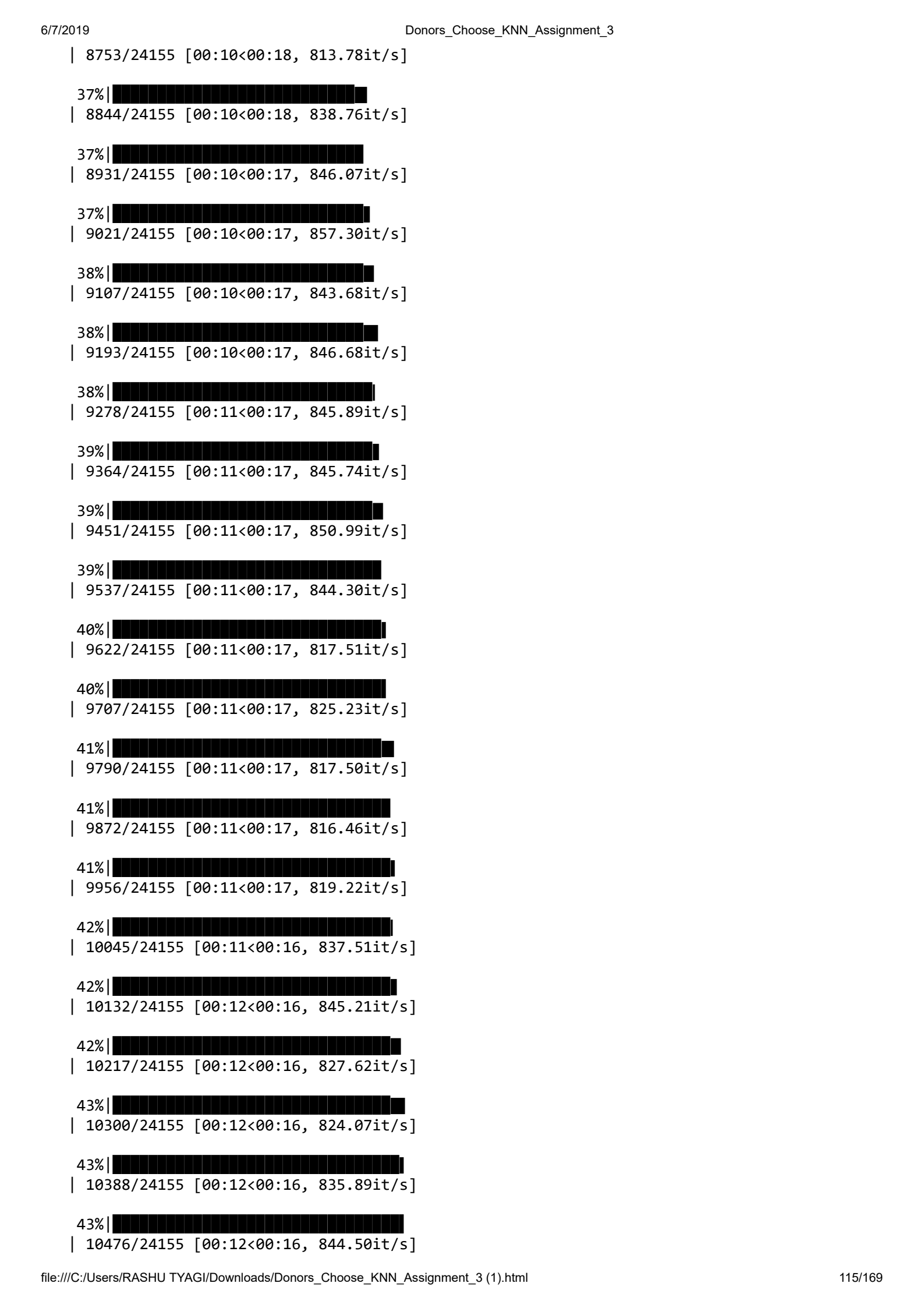
28%|
| 6670/24155 [00:07<00:20, 846.01it/s]

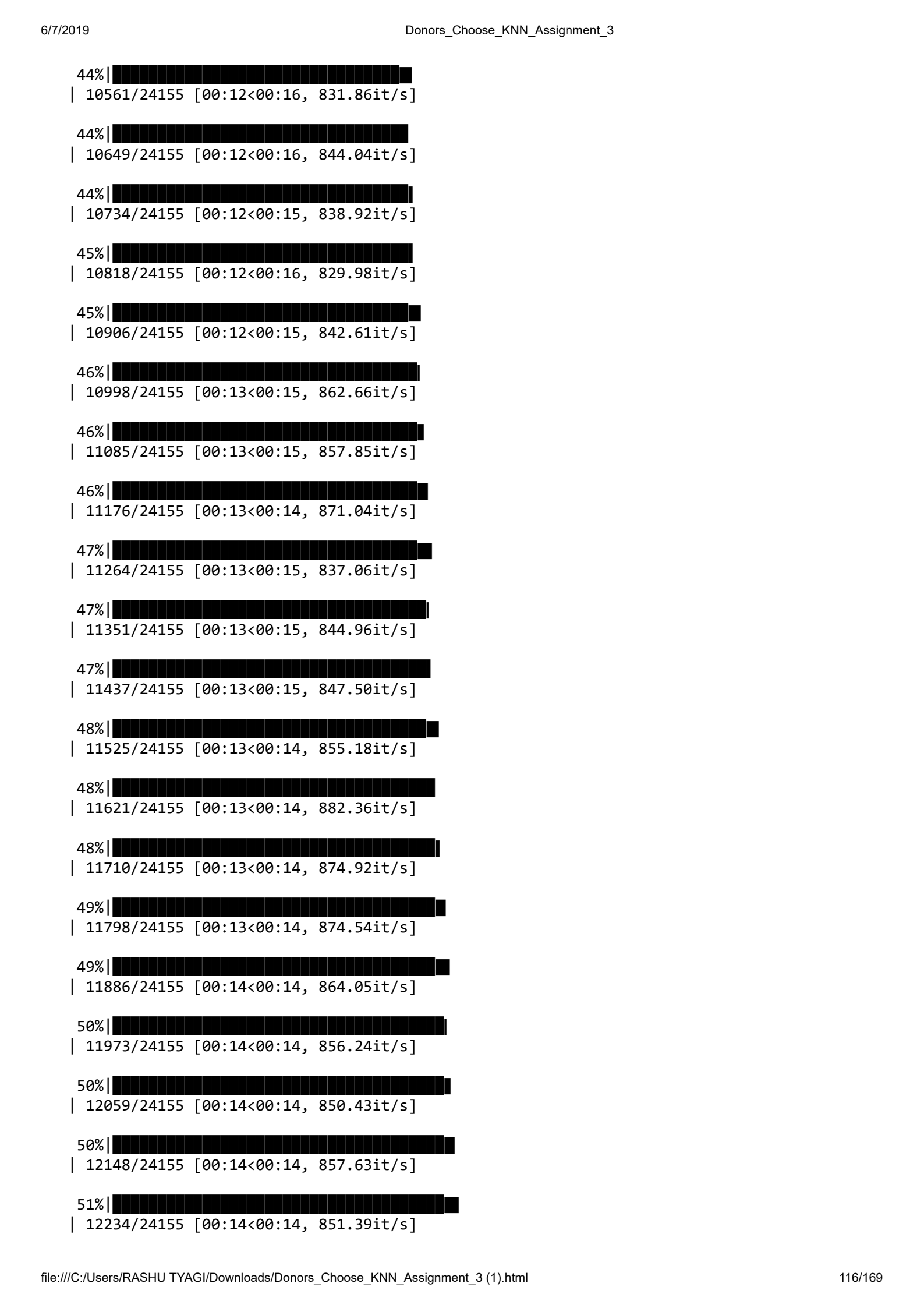
28%|
| 6758/24155 [00:08<00:20, 854.10it/s]

28%|
| 6844/24155 [00:08<00:20, 853.99it/s]

29%|
| 6933/24155 [00:08<00:20, 857.69it/s]

```
 29%|
| 7022/24155 [00:08<00:19, 865.27it/s]

 29%|
| 7110/24155 [00:08<00:19, 867.76it/s]

 30%|
| 7198/24155 [00:08<00:19, 869.58it/s]

 30%|
| 7285/24155 [00:08<00:19, 865.15it/s]

 31%|
| 7372/24155 [00:08<00:19, 839.75it/s]

 31%|
| 7457/24155 [00:08<00:20, 828.69it/s]

 31%|
| 7541/24155 [00:08<00:20, 820.62it/s]

 32%|
| 7624/24155 [00:09<00:20, 821.55it/s]

 32%|
| 7710/24155 [00:09<00:19, 826.18it/s]

 32%|
| 7793/24155 [00:09<00:20, 813.42it/s]

 33%|
| 7876/24155 [00:09<00:19, 816.56it/s]

 33%|
| 7969/24155 [00:09<00:19, 845.89it/s]

 33%|
| 8057/24155 [00:09<00:18, 854.02it/s]

 34%|
| 8148/24155 [00:09<00:18, 868.27it/s]

 34%|
| 8236/24155 [00:09<00:18, 857.18it/s]

 34%|
| 8324/24155 [00:09<00:18, 859.59it/s]

 35%|
| 8411/24155 [00:09<00:18, 858.21it/s]

 35%|
| 8499/24155 [00:10<00:18, 862.76it/s]

 36%|
| 8586/24155 [00:10<00:18, 833.37it/s]

 36%|
| 8670/24155 [00:10<00:19, 809.50it/s]

 36%|
```


| 8753/24155 [00:10<00:18, 813.78it/s]
37%|
| 8844/24155 [00:10<00:18, 838.76it/s]
37%|
| 8931/24155 [00:10<00:17, 846.07it/s]
37%|
| 9021/24155 [00:10<00:17, 857.30it/s]
38%|
| 9107/24155 [00:10<00:17, 843.68it/s]
38%|
| 9193/24155 [00:10<00:17, 846.68it/s]
38%|
| 9278/24155 [00:11<00:17, 845.89it/s]
39%|
| 9364/24155 [00:11<00:17, 845.74it/s]
39%|
| 9451/24155 [00:11<00:17, 850.99it/s]
39%|
| 9537/24155 [00:11<00:17, 844.30it/s]
40%|
| 9622/24155 [00:11<00:17, 817.51it/s]
40%|
| 9707/24155 [00:11<00:17, 825.23it/s]
41%|
| 9790/24155 [00:11<00:17, 817.50it/s]
41%|
| 9872/24155 [00:11<00:17, 816.46it/s]
41%|
| 9956/24155 [00:11<00:17, 819.22it/s]
42%|
| 10045/24155 [00:11<00:16, 837.51it/s]
42%|
| 10132/24155 [00:12<00:16, 845.21it/s]
42%|
| 10217/24155 [00:12<00:16, 827.62it/s]
43%|
| 10300/24155 [00:12<00:16, 824.07it/s]
43%|
| 10388/24155 [00:12<00:16, 835.89it/s]
43%|
| 10476/24155 [00:12<00:16, 844.50it/s]


44%| | 10561/24155 [00:12<00:16, 831.86it/s]
44%| | 10649/24155 [00:12<00:16, 844.04it/s]
44%| | 10734/24155 [00:12<00:15, 838.92it/s]
45%| | 10818/24155 [00:12<00:16, 829.98it/s]
45%| | 10906/24155 [00:12<00:15, 842.61it/s]
46%| | 10998/24155 [00:13<00:15, 862.66it/s]
46%| | 11085/24155 [00:13<00:15, 857.85it/s]
46%| | 11176/24155 [00:13<00:14, 871.04it/s]
47%| | 11264/24155 [00:13<00:15, 837.06it/s]
47%| | 11351/24155 [00:13<00:15, 844.96it/s]
47%| | 11437/24155 [00:13<00:15, 847.50it/s]
48%| | 11525/24155 [00:13<00:14, 855.18it/s]
48%| | 11621/24155 [00:13<00:14, 882.36it/s]
48%| | 11710/24155 [00:13<00:14, 874.92it/s]
49%| | 11798/24155 [00:13<00:14, 874.54it/s]
49%| | 11886/24155 [00:14<00:14, 864.05it/s]
50%| | 11973/24155 [00:14<00:14, 856.24it/s]
50%| | 12059/24155 [00:14<00:14, 850.43it/s]
50%| | 12148/24155 [00:14<00:14, 857.63it/s]
51%| | 12234/24155 [00:14<00:14, 851.39it/s]

51%|
| 12320/24155 [00:14<00:14, 832.36it/s]

51%|
| 12405/24155 [00:14<00:14, 835.78it/s]

52%|
| 12489/24155 [00:14<00:13, 835.28it/s]

52%|
| 12580/24155 [00:14<00:13, 852.16it/s]

52%|
| 12669/24155 [00:15<00:13, 858.84it/s]

53%|
| 12756/24155 [00:15<00:13, 860.29it/s]

53%|
| 12843/24155 [00:15<00:13, 841.43it/s]

54%|
| 12928/24155 [00:15<00:13, 834.65it/s]

54%|
| 13013/24155 [00:15<00:13, 837.40it/s]

54%|
| 13099/24155 [00:15<00:13, 842.25it/s]

55%|
| 13184/24155 [00:15<00:13, 827.98it/s]

55%|
| 13276/24155 [00:15<00:12, 851.86it/s]

55%|
| 13362/24155 [00:15<00:12, 844.89it/s]

56%|
| 13454/24155 [00:15<00:12, 864.41it/s]

56%|
| 13541/24155 [00:16<00:12, 856.50it/s]

56%|
| 13627/24155 [00:16<00:12, 850.60it/s]

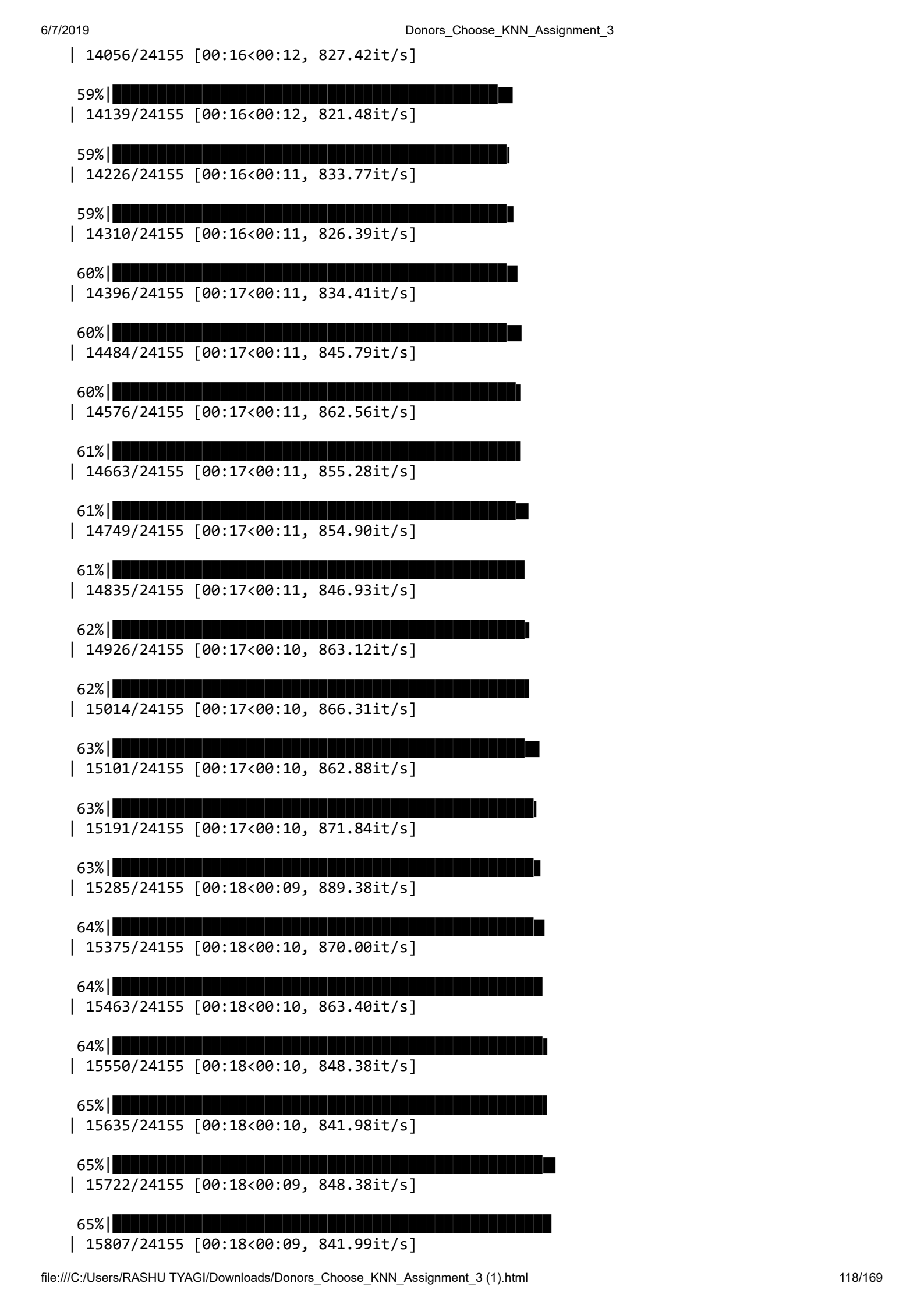
57%|
| 13713/24155 [00:16<00:12, 849.03it/s]

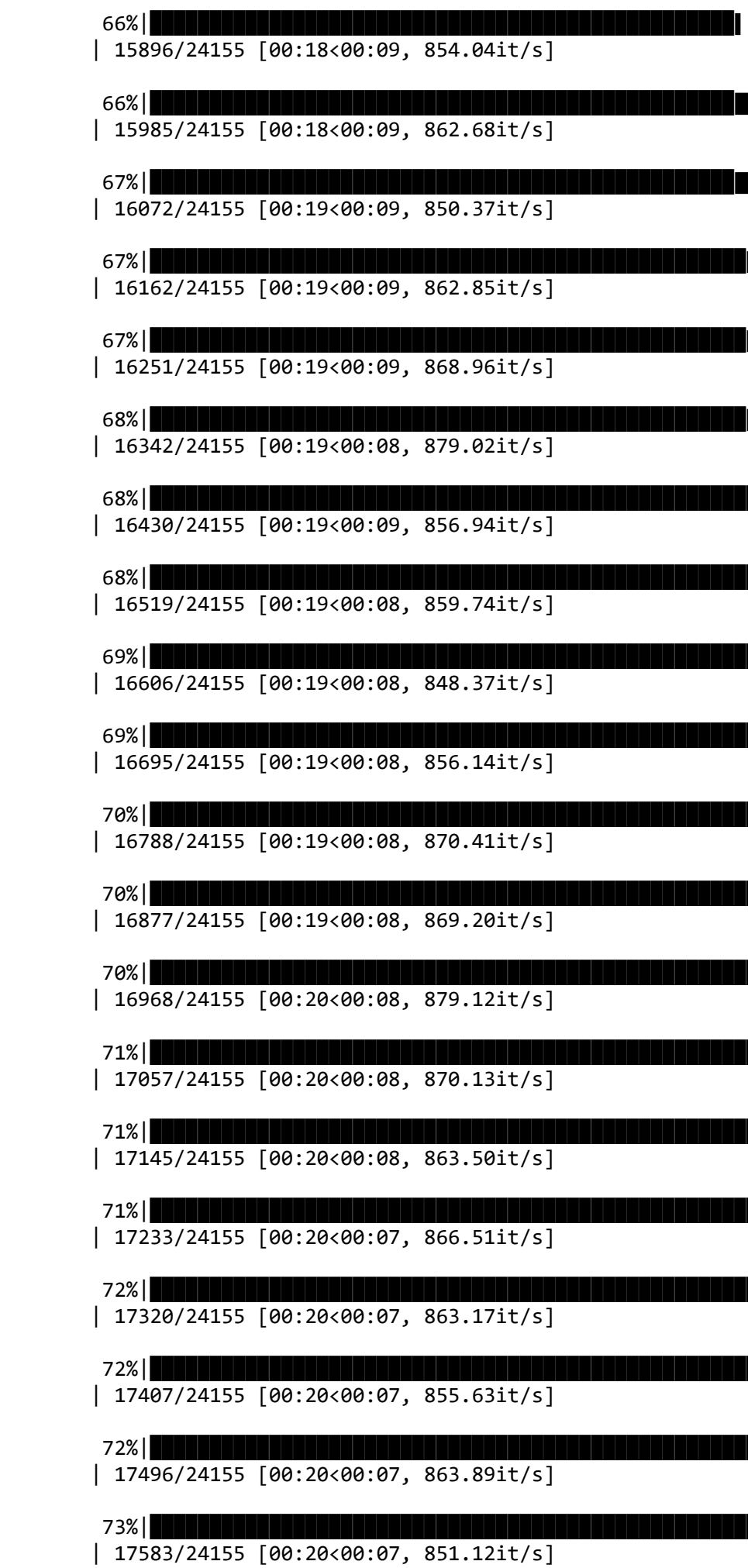
57%|
| 13799/24155 [00:16<00:12, 850.45it/s]

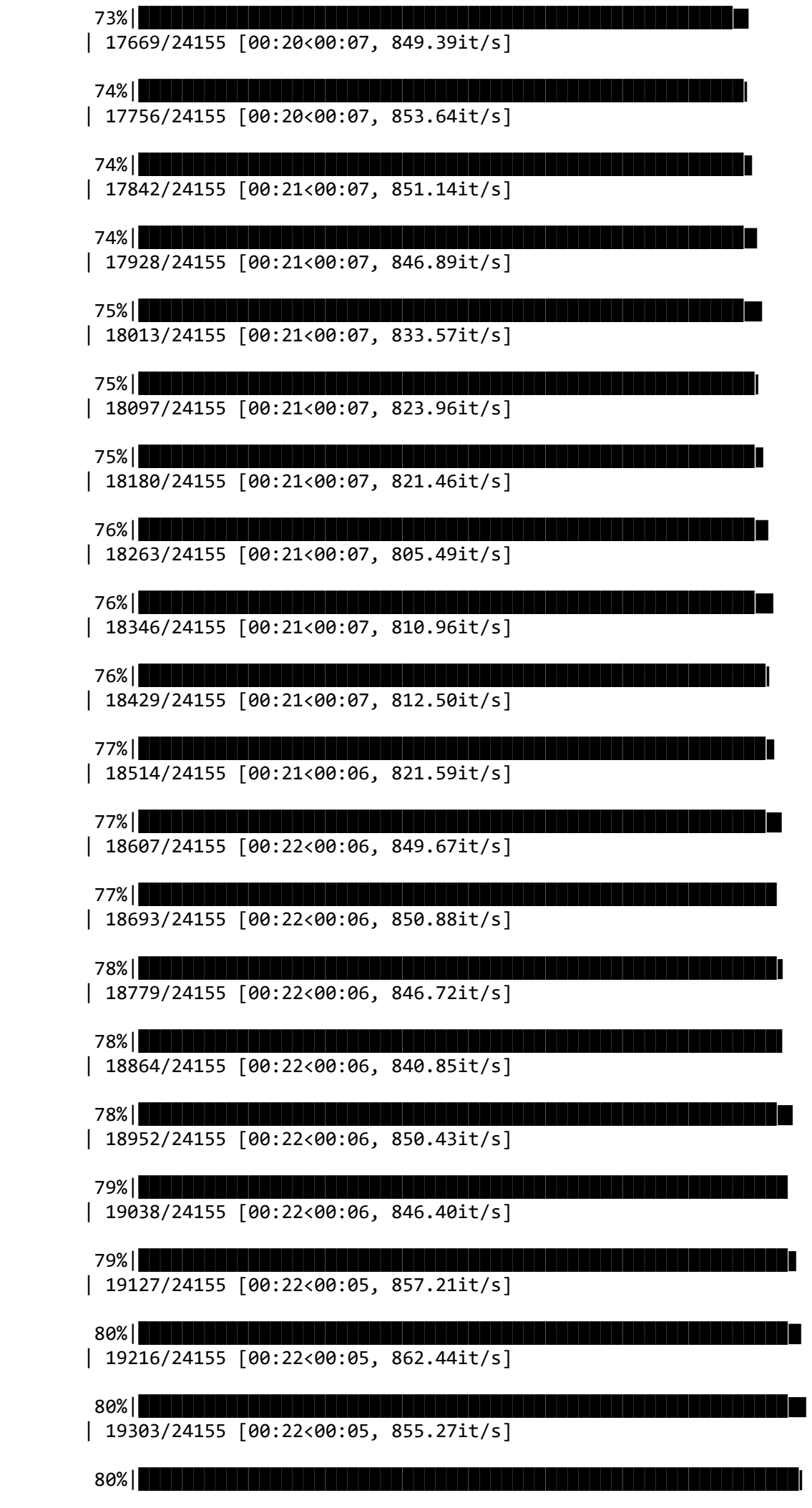
57%|
| 13885/24155 [00:16<00:12, 846.42it/s]

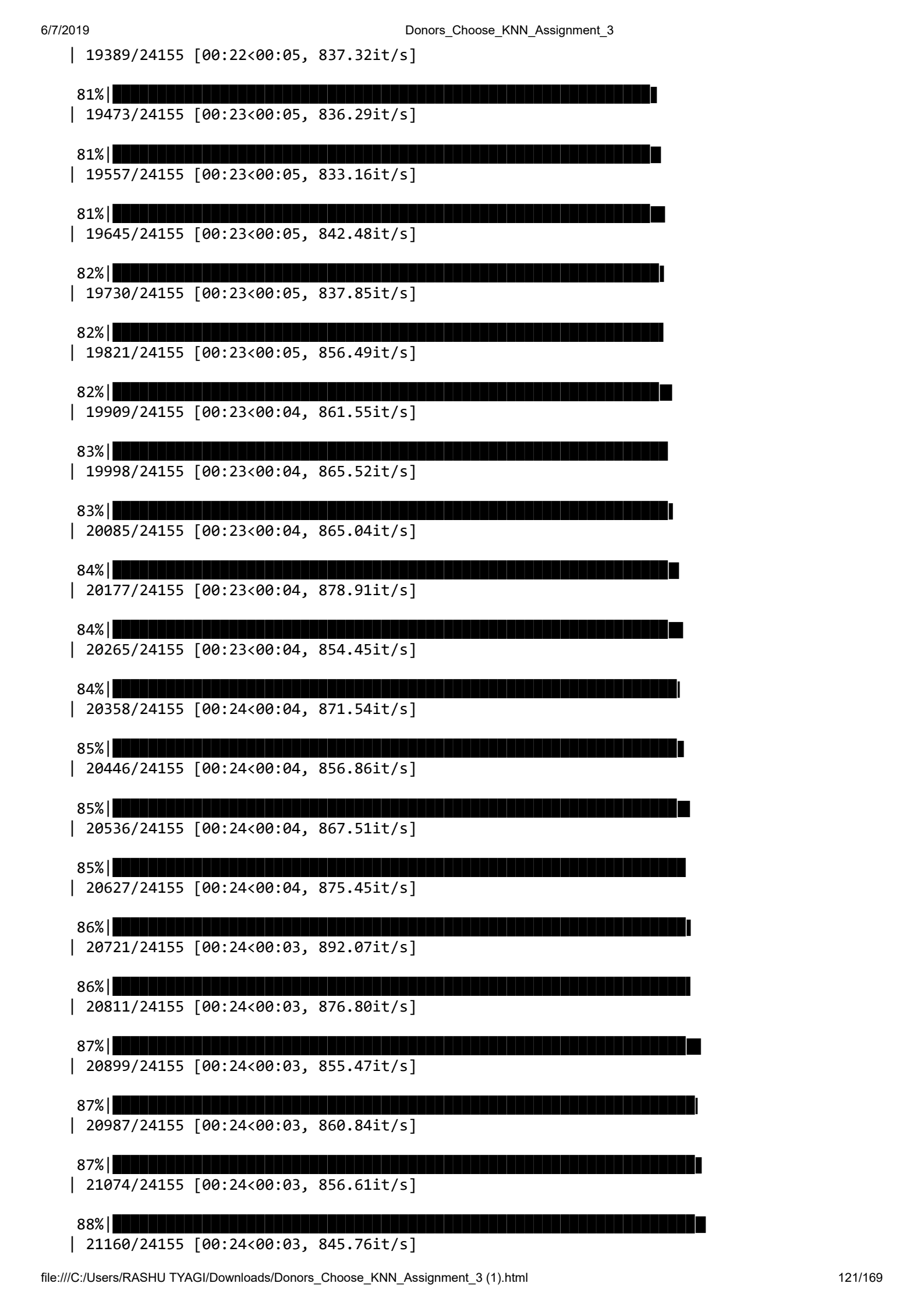
58%|
| 13971/24155 [00:16<00:11, 848.68it/s]

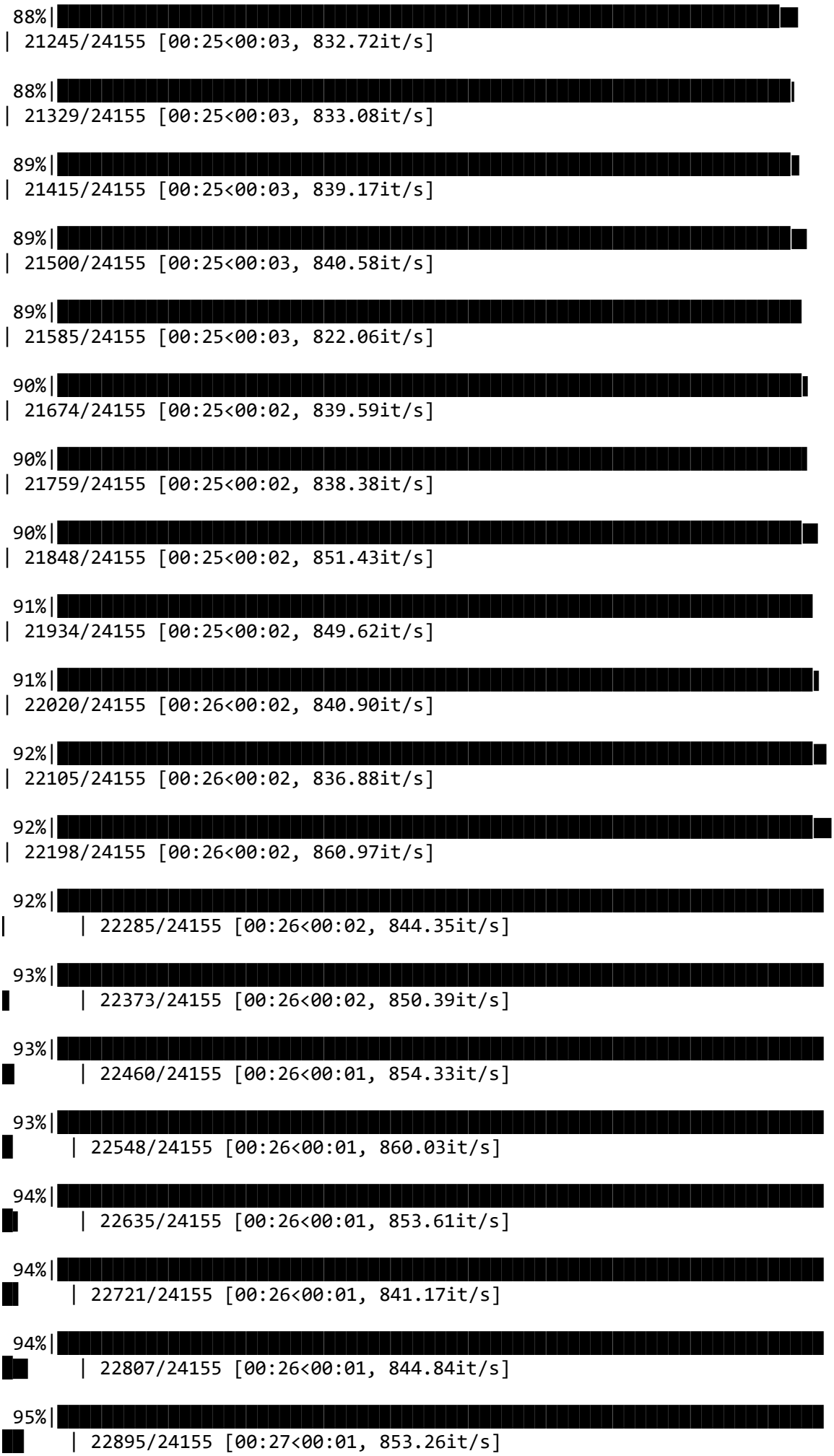
58%|


| 14056/24155 [00:16<00:12, 827.42it/s]
59%|
| 14139/24155 [00:16<00:12, 821.48it/s]
59%|
| 14226/24155 [00:16<00:11, 833.77it/s]
59%|
| 14310/24155 [00:16<00:11, 826.39it/s]
60%|
| 14396/24155 [00:17<00:11, 834.41it/s]
60%|
| 14484/24155 [00:17<00:11, 845.79it/s]
60%|
| 14576/24155 [00:17<00:11, 862.56it/s]
61%|
| 14663/24155 [00:17<00:11, 855.28it/s]
61%|
| 14749/24155 [00:17<00:11, 854.90it/s]
61%|
| 14835/24155 [00:17<00:11, 846.93it/s]
62%|
| 14926/24155 [00:17<00:10, 863.12it/s]
62%|
| 15014/24155 [00:17<00:10, 866.31it/s]
63%|
| 15101/24155 [00:17<00:10, 862.88it/s]
63%|
| 15191/24155 [00:17<00:10, 871.84it/s]
63%|
| 15285/24155 [00:18<00:09, 889.38it/s]
64%|
| 15375/24155 [00:18<00:10, 870.00it/s]
64%|
| 15463/24155 [00:18<00:10, 863.40it/s]
64%|
| 15550/24155 [00:18<00:10, 848.38it/s]
65%|
| 15635/24155 [00:18<00:10, 841.98it/s]
65%|
| 15722/24155 [00:18<00:09, 848.38it/s]
65%|
| 15807/24155 [00:18<00:09, 841.99it/s]


66%| | 15896/24155 [00:18<00:09, 854.04it/s]
66%| | 15985/24155 [00:18<00:09, 862.68it/s]
67%| | 16072/24155 [00:19<00:09, 850.37it/s]
67%| | 16162/24155 [00:19<00:09, 862.85it/s]
67%| | 16251/24155 [00:19<00:09, 868.96it/s]
68%| | 16342/24155 [00:19<00:08, 879.02it/s]
68%| | 16430/24155 [00:19<00:09, 856.94it/s]
68%| | 16519/24155 [00:19<00:08, 859.74it/s]
69%| | 16606/24155 [00:19<00:08, 848.37it/s]
69%| | 16695/24155 [00:19<00:08, 856.14it/s]
70%| | 16788/24155 [00:19<00:08, 870.41it/s]
70%| | 16877/24155 [00:19<00:08, 869.20it/s]
70%| | 16968/24155 [00:20<00:08, 879.12it/s]
71%| | 17057/24155 [00:20<00:08, 870.13it/s]
71%| | 17145/24155 [00:20<00:08, 863.50it/s]
71%| | 17233/24155 [00:20<00:07, 866.51it/s]
72%| | 17320/24155 [00:20<00:07, 863.17it/s]
72%| | 17407/24155 [00:20<00:07, 855.63it/s]
72%| | 17496/24155 [00:20<00:07, 863.89it/s]
73%| | 17583/24155 [00:20<00:07, 851.12it/s]


73%| | 17669/24155 [00:20<00:07, 849.39it/s]
74%| | 17756/24155 [00:20<00:07, 853.64it/s]
74%| | 17842/24155 [00:21<00:07, 851.14it/s]
74%| | 17928/24155 [00:21<00:07, 846.89it/s]
75%| | 18013/24155 [00:21<00:07, 833.57it/s]
75%| | 18097/24155 [00:21<00:07, 823.96it/s]
75%| | 18180/24155 [00:21<00:07, 821.46it/s]
76%| | 18263/24155 [00:21<00:07, 805.49it/s]
76%| | 18346/24155 [00:21<00:07, 810.96it/s]
76%| | 18429/24155 [00:21<00:07, 812.50it/s]
77%| | 18514/24155 [00:21<00:06, 821.59it/s]
77%| | 18607/24155 [00:22<00:06, 849.67it/s]
77%| | 18693/24155 [00:22<00:06, 850.88it/s]
78%| | 18779/24155 [00:22<00:06, 846.72it/s]
78%| | 18864/24155 [00:22<00:06, 840.85it/s]
78%| | 18952/24155 [00:22<00:06, 850.43it/s]
79%| | 19038/24155 [00:22<00:06, 846.40it/s]
79%| | 19127/24155 [00:22<00:05, 857.21it/s]
80%| | 19216/24155 [00:22<00:05, 862.44it/s]
80%| | 19303/24155 [00:22<00:05, 855.27it/s]
80%|


| 19389/24155 [00:22<00:05, 837.32it/s]
81%|
| 19473/24155 [00:23<00:05, 836.29it/s]
81%|
| 19557/24155 [00:23<00:05, 833.16it/s]
81%|
| 19645/24155 [00:23<00:05, 842.48it/s]
82%|
| 19730/24155 [00:23<00:05, 837.85it/s]
82%|
| 19821/24155 [00:23<00:05, 856.49it/s]
82%|
| 19909/24155 [00:23<00:04, 861.55it/s]
83%|
| 19998/24155 [00:23<00:04, 865.52it/s]
83%|
| 20085/24155 [00:23<00:04, 865.04it/s]
84%|
| 20177/24155 [00:23<00:04, 878.91it/s]
84%|
| 20265/24155 [00:23<00:04, 854.45it/s]
84%|
| 20358/24155 [00:24<00:04, 871.54it/s]
85%|
| 20446/24155 [00:24<00:04, 856.86it/s]
85%|
| 20536/24155 [00:24<00:04, 867.51it/s]
85%|
| 20627/24155 [00:24<00:04, 875.45it/s]
86%|
| 20721/24155 [00:24<00:03, 892.07it/s]
86%|
| 20811/24155 [00:24<00:03, 876.80it/s]
87%|
| 20899/24155 [00:24<00:03, 855.47it/s]
87%|
| 20987/24155 [00:24<00:03, 860.84it/s]
87%|
| 21074/24155 [00:24<00:03, 856.61it/s]
88%|
| 21160/24155 [00:24<00:03, 845.76it/s]


88%| | 21245/24155 [00:25<00:03, 832.72it/s]
88%| | 21329/24155 [00:25<00:03, 833.08it/s]
89%| | 21415/24155 [00:25<00:03, 839.17it/s]
89%| | 21500/24155 [00:25<00:03, 840.58it/s]
89%| | 21585/24155 [00:25<00:03, 822.06it/s]
90%| | 21674/24155 [00:25<00:02, 839.59it/s]
90%| | 21759/24155 [00:25<00:02, 838.38it/s]
90%| | 21848/24155 [00:25<00:02, 851.43it/s]
91%| | 21934/24155 [00:25<00:02, 849.62it/s]
91%| | 22020/24155 [00:26<00:02, 840.90it/s]
92%| | 22105/24155 [00:26<00:02, 836.88it/s]
92%| | 22198/24155 [00:26<00:02, 860.97it/s]
92%| | 22285/24155 [00:26<00:02, 844.35it/s]
93%| | 22373/24155 [00:26<00:02, 850.39it/s]
93%| | 22460/24155 [00:26<00:01, 854.33it/s]
93%| | 22548/24155 [00:26<00:01, 860.03it/s]
94%| | 22635/24155 [00:26<00:01, 853.61it/s]
94%| | 22721/24155 [00:26<00:01, 841.17it/s]
94%| | 22807/24155 [00:26<00:01, 844.84it/s]
95%| | 22895/24155 [00:27<00:01, 853.26it/s]

```
 95%|████████████████████████████████████████████████████████████████
██▍     | 22984/24155 [00:27<00:01, 862.14it/s]

 96%|████████████████████████████████████████████████████████████████
██▋     | 23071/24155 [00:27<00:01, 852.55it/s]

 96%|████████████████████████████████████████████████████████████████
██▋     | 23160/24155 [00:27<00:01, 861.56it/s]

 96%|████████████████████████████████████████████████████████████████
██▉     | 23247/24155 [00:27<00:01, 859.71it/s]

 97%|████████████████████████████████████████████████████████████████
███     | 23334/24155 [00:27<00:00, 843.42it/s]

 97%|████████████████████████████████████████████████████████████████
███▏    | 23434/24155 [00:27<00:00, 876.33it/s]

 97%|████████████████████████████████████████████████████████████████
███▍    | 23525/24155 [00:27<00:00, 881.65it/s]

 98%|████████████████████████████████████████████████████████████████
███▋    | 23618/24155 [00:27<00:00, 893.72it/s]

 98%|████████████████████████████████████████████████████████████████
███▉    | 23708/24155 [00:27<00:00, 893.67it/s]

 99%|████████████████████████████████████████████████████████████████
████▏   | 23798/24155 [00:28<00:00, 883.10it/s]

 99%|████████████████████████████████████████████████████████████████
████▍   | 23888/24155 [00:28<00:00, 886.26it/s]

 99%|████████████████████████████████████████████████████████████████
████▋   | 23982/24155 [00:28<00:00, 899.84it/s]

100%|████████████████████████████████████████████████████████████████
████▉   | 24076/24155 [00:28<00:00, 909.59it/s]

100%|████████████████████████████████████████████████████████████████
| 24155/24155 [00:28<00:00, 848.65it/s]


24155
300
```

Project Title Train data

In [171]:

```
vectorizer = TfidfVectorizer(max_features = 2000, min_df = 10)

vectorizer.fit(preprocessed_project_titles_Train)

project_title_tfidf_train = vectorizer.transform(preprocessed_project_titles_Train)

print("Shape of matrix after tfidf ",project_title_tfidf_train.shape)

# we are converting a dictionary with word as a key, and the idf as a value

dictionary = dict(zip(vectorizer.get_feature_names(), list(vectorizer.idf_)))
tfidf_project_title_words = set(dictionary.keys())
```

```
Shape of matrix after tfidf  (49041, 2000)
```

In [172]:

```
# tfidf Word2Vec
# computing average word2vec for each Project Title is stored in this list

tfidf_w2v_vectors_project_title_train = []; # the tfidf-w2v for each essay
for sentence in tqdm(preprocessed_project_titles_Train): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the title
    for word in sentence.split(): # for each word in a title
        if (word in glove_words) and (word in dictionary):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_project_title_train.append(vector)

print(len(tfidf_w2v_vectors_project_title_train))
print(len(tfidf_w2v_vectors_project_title_train[0]))
```

```
  0%|
| 0/49041 [00:00<?, ?it/s]

  9%|▉
| 4432/49041 [00:00<00:01, 43998.31it/s]

 18%|█▊
| 8843/49041 [00:00<00:00, 43935.66it/s]

 27%|██▋
| 13324/49041 [00:00<00:00, 44099.05it/s]

 36%|███▌
| 17753/49041 [00:00<00:00, 44059.88it/s]

 45%|████▍
| 22297/49041 [00:00<00:00, 44369.86it/s]

 55%|█████▍
| 26802/49041 [00:00<00:00, 44471.67it/s]

 64%|██████▍
| 31322/49041 [00:00<00:00, 44594.77it/s]

 73%|███████▎
| 35850/49041 [00:00<00:00, 44701.21it/s]

 82%|████████▏
| 40335/49041 [00:00<00:00, 44648.04it/s]

 92%|█████████▏
| 44901/49041 [00:01<00:00, 44850.18it/s]

100%|██████████
| 49041/49041 [00:01<00:00, 44579.38it/s]

49041
300
```

Project title test data

In [173]:

```
# tfidf Word2Vec
# computing average word2vec for each Project Title is stored in this list

tfidf_w2v_vectors_project_title_test = []; # the tfidf-w2v for each essay
for sentence in tqdm(preprocessed_project_titles_Test): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the title
    for word in sentence.split(): # for each word in a title
        if (word in glove_words) and (word in dictionary):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_project_title_test.append(vector)

print(len(tfidf_w2v_vectors_project_title_test))
print(len(tfidf_w2v_vectors_project_title_test[0]))
```

```
  0%|
| 0/36052 [00:00<?, ?it/s]

 12%|████████
| 4361/36052 [00:00<00:00, 43282.10it/s]

 25%|████████████████
| 8858/36052 [00:00<00:00, 43681.59it/s]

 37%|████████████████████████
| 13161/36052 [00:00<00:00, 43391.86it/s]

 50%|████████████████████████████████
| 17850/36052 [00:00<00:00, 44293.40it/s]

 62%|████████████████████████████████████████
| 22426/36052 [00:00<00:00, 44611.28it/s]

 75%|████████████████████████████████████████████████
| 27020/36052 [00:00<00:00, 44902.01it/s]

 88%|████████████████████████████████████████████████████████
| 31645/36052 [00:00<00:00, 45204.30it/s]

100%|████████████████████████████████████████████████████████████
| 36052/36052 [00:00<00:00, 44897.84it/s]

36052
300
```

project title cross validation data

In [174]:

```
# tfidf Word2Vec
# computing average word2vec for each Project Title is stored in this list

tfidf_w2v_vectors_project_title_cv = []; # the tfidf-w2v for each essay
for sentence in tqdm(preprocessed_project_titles_Cv): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the title
    for word in sentence.split(): # for each word in a title
        if (word in glove_words) and (word in dictionary):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_project_title_cv.append(vector)

print(len(tfidf_w2v_vectors_project_title_cv))
print(len(tfidf_w2v_vectors_project_title_cv[0]))
```

```
  0%|
| 0/24155 [00:00<?, ?it/s]

 18%|
| 4459/24155 [00:00<00:00, 44266.46it/s]

 37%|
| 8981/24155 [00:00<00:00, 44452.31it/s]

 56%|
| 13583/24155 [00:00<00:00, 44815.48it/s]

 75%|
| 18224/24155 [00:00<00:00, 45185.54it/s]

 94%|
| 22813/24155 [00:00<00:00, 45293.28it/s]

100%|
| 24155/24155 [00:00<00:00, 45101.61it/s]

24155
300
```

Numerical Features

vectorizing numerical features

Price for projects

In [100]:

```
# now firstly we will try to add the price and the quantity of the items required from
 the resource dataframe

price_quantity_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'
}).reset_index()

price_quantity_data.head(2)
```

Out[100]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

now we need to join the above dataframe with our train,test,cv data that we already have

In [101]:

```
X_train = pd.merge(X_train, price_quantity_data, on='id', how='left')
X_test = pd.merge(X_test, price_quantity_data, on='id', how='left')
X_cv = pd.merge(X_cv, price_quantity_data, on='id', how='left')
```

we will be performing the normalization of the numerical data here

Normalizing the price data

In [102]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.

normalizer.fit(X_train['price'].values.reshape(-1,1))

price_train = normalizer.transform(X_train['price'].values.reshape(-1,1))
price_cv = normalizer.transform(X_cv['price'].values.reshape(-1,1))
price_test = normalizer.transform(X_test['price'].values.reshape(-1,1))

print("After vectorizations")
print(price_train.shape, y_train.shape)
print(price_cv.shape, y_cv.shape)
print(price_test.shape, y_test.shape)
```

```
After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

Normalizing the quantity data

In [103]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

# normalizer.fit(X_train['quantity'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.

normalizer.fit(X_train['quantity'].values.reshape(-1,1))

quantity_train = normalizer.transform(X_train['quantity'].values.reshape(-1,1))
quantity_test = normalizer.transform(X_test['quantity'].values.reshape(-1,1))
quantity_cv = normalizer.transform(X_cv['quantity'].values.reshape(-1,1))

print("After vectorizations")
print(quantity_train.shape, y_train.shape)
print(quantity_test.shape, y_test.shape)
print(quantity_cv.shape, y_cv.shape)
```

```
After vectorizations
(49041, 1) (49041,)
(36052, 1) (36052,)
(24155, 1) (24155,)
```

Normalizing the number of previously posted projects by a teacher

In [104]:

```
normalizer = Normalizer()

# normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.

normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

prev_projects_train = normalizer.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
prev_projects_cv = normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
prev_projects_test = normalizer.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

print("After converting into vectors form")
print(prev_projects_train.shape, y_train.shape)
print(prev_projects_cv.shape, y_cv.shape)
print(prev_projects_test.shape, y_test.shape)
```

```
After converting into vectors form
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

Set 1: categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW)

Now we need to merge all the numerical vectors(categorical features,text features,numerical features) given above for set-1 which we created using different methods

In [105]:

```
from scipy.sparse import hstack

X_train_merge = hstack((categories_one_hot_train,subcategories_one_hot_train,teacher_prefix_one_hot_train,project_grade_categories_one_hot_train,school_state_categories_one_hot_train,essay_bow_train,project_title_bow_train,price_train,quantity_train,prev_projects_train)).tocsr()
X_test_merge = hstack((categories_one_hot_test,subcategories_one_hot_test,teacher_prefix_one_hot_test,project_grade_categories_one_hot_test,school_state_categories_one_hot_test,essay_bow_test,project_title_bow_test,price_test,quantity_test,prev_projects_test)).tocsr()
X_cv_merge = hstack((categories_one_hot_cv,subcategories_one_hot_cv,teacher_prefix_one_hot_cv,project_grade_categories_one_hot_cv,school_state_categories_one_hot_cv,essay_bow_cv,project_title_bow_cv,price_cv,quantity_cv,prev_projects_cv)).tocsr()
```

In [106]:

```
# this will be our finally created data matrix dimensions

print(X_train_merge.shape, y_train.shape)
print(X_test_merge.shape, y_test.shape)
print(X_cv_merge.shape, y_cv.shape)
```

```
(49041, 52835) (49041,)
(36052, 52835) (36052,)
(24155, 52835) (24155,)
```

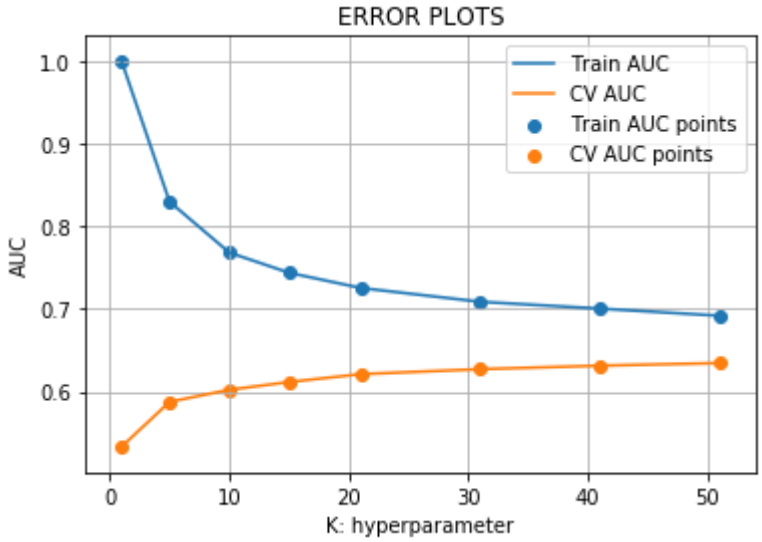
Applying KNN

Hyper parameter Tuning (USING THE BASIC FOR LOOP METHOD GRIDSEARCH-CV CAN ALSO BE USED HERE)

In [108]:

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score

"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence va
lues, or non-thresholded measure of
decisions (as returned by “decision_function” on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

"""

train_auc = []
cv_auc = []
a = []
b = []

K = [1, 5, 10, 15, 21, 31, 41, 51,]

for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_train_merge, y_train)

    y_train_pred =  neigh.predict_proba(X_train_merge)[:,1]
    y_cv_pred =  neigh.predict_proba(X_cv_merge)[:,1]


    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
 of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
    a.append(y_train_pred)
    b.append(y_cv_pred)

plt.plot(K, train_auc, label='Train AUC')
plt.plot(K, cv_auc, label='CV AUC')

plt.scatter(K, train_auc, label='Train AUC points')
plt.scatter(K, cv_auc, label='CV AUC points')

plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```

```
  0%|
| 0/8 [00:00<?, ?it/s]
 12%|
| 1/8 [04:18<30:08, 258.32s/it]
 25%|
| 2/8 [09:00<26:33, 265.62s/it]
 38%|
| 3/8 [13:41<22:30, 270.11s/it]
 50%|
| 4/8 [18:24<18:15, 273.83s/it]
 62%|
| 5/8 [23:06<13:49, 276.45s/it]
 75%|
| 6/8 [27:43<09:13, 276.68s/it]
 88%|
| 7/8 [32:31<04:39, 279.87s/it]
100%|
| 8/8 [37:15<00:00, 281.31s/it]
```



FROM ABOVE PLOT WE CAN SAY THAT THE BEST K WILL BE AROUND 28 BECAUSE AFTER THAT THE GRAPH OF CV_AUC IS ALMOST CONSTANT HENCE LET USE CONSIDER BEST K = 29

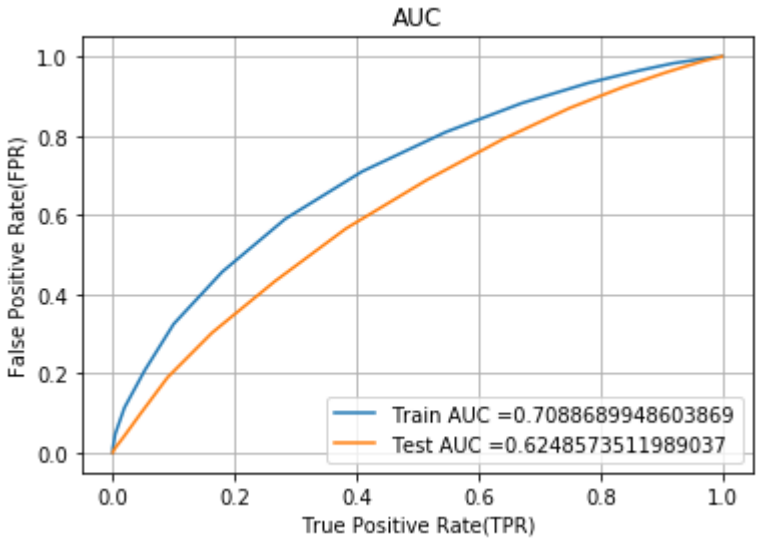
In [109]:

```
best_k_set_1 = 31
```

NOW USING THE BEST K VALUE RECEIVED FROM ABOVE WE WILL BE TRAINING OUR MODEL

In [110]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


neigh = KNeighborsClassifier(n_neighbors=best_k_set_1,n_jobs = -4)
neigh.fit(X_train_merge, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred =  neigh.predict_proba(X_train_merge)[:,1]
y_test_pred =  neigh.predict_proba(X_test_merge)[:,1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



In [ ]:

Creating the confusion matrix for the above train data results

In [111]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

    t = threshould[np.argmax(tpr*(1-fpr))]

    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.rou
nd(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

Creating the confusion matrix for train and test data

In [112]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_f
pr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr
)))
```

```
====================================================================================
==========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24791603938644033 for threshold 0.742
[[ 3374  4052]
 [ 7946 33669]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24975469520310625 for threshold 0.774
[[ 2644  2815]
 [ 9518 21075]]
```

Showing the confusion matrix for train data visually

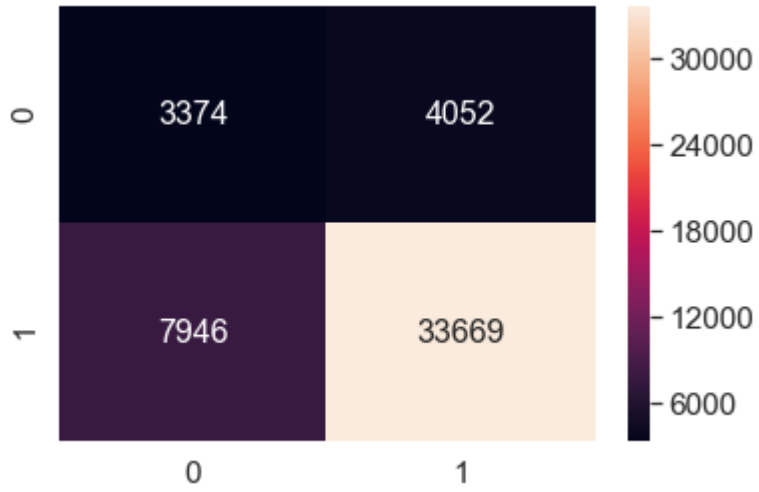
In [113]:

```
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how
-can-i-plot-a-confusion-matrix

import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt

df_cm = pd.DataFrame(confusion_matrix(y_train,predict(y_train_pred, tr_thresholds, trai
n_fpr, train_fpr)), range(2),
                  range(2))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm, annot=True,annot_kws={"size": 16},fmt='g')# font size
```

```
the maximum value of tpr*(1-fpr) 0.24791603938644033 for threshold 0.742
```

Out[113]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22b949326d8>
```



We can see that the True Positive values is the highest which is a good sign also.But the True Negative value is less which is not a good sign

Showing the confusion matrix for the test data visually

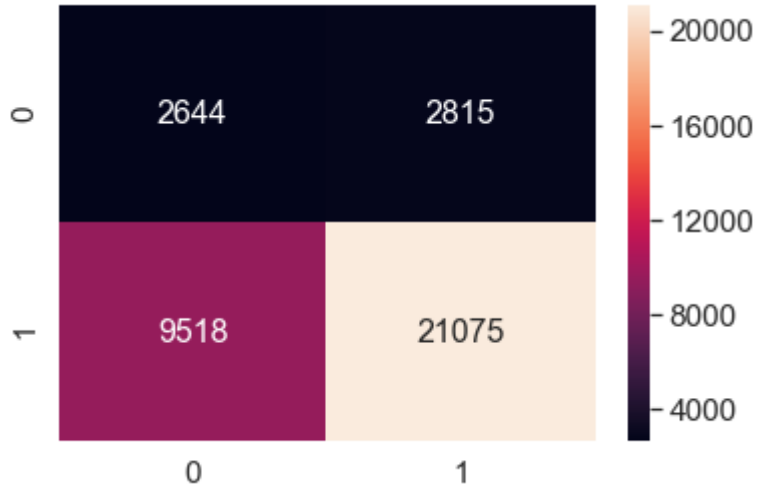
In [114]:

```
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how
-can-i-plot-a-confusion-matrix

import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt

df_cm_test = pd.DataFrame(confusion_matrix(y_test,predict(y_test_pred, tr_thresholds, t
est_fpr, test_fpr)), range(2),
                  range(2))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm_test, annot=True,annot_kws={"size": 16},fmt='g')# font size
```

```
the maximum value of tpr*(1-fpr) 0.24975469520310625 for threshold 0.774
```

Out[114]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22b94d93080>
```



From above we can say that we have large number of True positive but we also have a large number of False positive hence we can say that our model is a bit biased towards the positive class thats why it is predicting the positive class so many times

Set 2 : categorical, numerical features + project_title(TFIDF) + preprocessed_essay (TFIDF)

Now we need to merge all the numerical vectors(categorical features,text features,numerical features) given above for set-2 which we created using different methods

In [123]:

```
from scipy.sparse import hstack

X_train_merge_set_2 = hstack((categories_one_hot_train,subcategories_one_hot_train,teac
her_prefix_one_hot_train,project_grade_categories_one_hot_train,school_state_categories
_one_hot_train,text_tfidf_train,project_title_tfidf_train,price_train,quantity_train,pr
ev_projects_train)).tocsr()
X_test_merge_set_2 = hstack((categories_one_hot_test,subcategories_one_hot_test,teacher
_prefix_one_hot_test,project_grade_categories_one_hot_test,school_state_categories_one_
hot_test,text_tfidf_test,project_title_tfidf_test,price_test,quantity_test,prev_project
s_test)).tocsr()
X_cv_merge_set_2 = hstack((categories_one_hot_cv,subcategories_one_hot_cv,teacher_prefi
x_one_hot_cv,project_grade_categories_one_hot_cv,school_state_categories_one_hot_cv,tex
t_tfidf_cv,title_tfidf_cv,price_cv,quantity_cv,prev_projects_cv)).tocsr()
```

In [124]:

```
# this will be our finally created data matrix dimensions

print(X_train_merge_set_2.shape, y_train.shape)
print(X_test_merge_set_2.shape, y_test.shape)
print(X_cv_merge_set_2.shape, y_cv.shape)
```

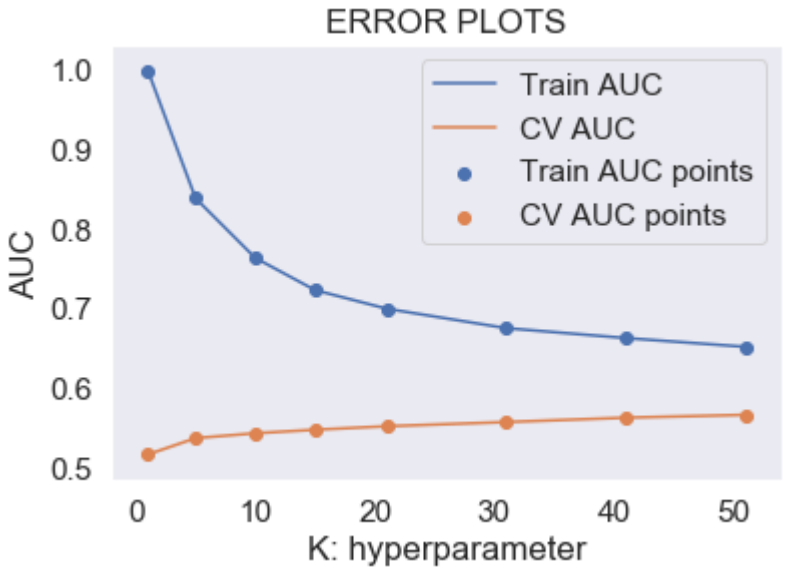
```
(49041, 8102) (49041,)
(36052, 8102) (36052,)
(24155, 8102) (24155,)
```

Hyper parameter Tuning (USING THE BASIC FOR LOOP METHOD GRIDSEARCH-CV CAN ALSO BE USED HERE)

In [125]:

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score

"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence va
lues, or non-thresholded measure of
decisions (as returned by “decision_function” on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

"""

train_auc = []
cv_auc = []
a = []
b = []

K = [1, 5, 10, 15, 21, 31, 41, 51,]

for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_train_merge_set_2, y_train)

    y_train_pred =  neigh.predict_proba(X_train_merge_set_2)[:,1]
    y_cv_pred =  neigh.predict_proba(X_cv_merge_set_2)[:,1]


    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
 of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
    a.append(y_train_pred)
    b.append(y_cv_pred)

plt.plot(K, train_auc, label='Train AUC')
plt.plot(K, cv_auc, label='CV AUC')

plt.scatter(K, train_auc, label='Train AUC points')
plt.scatter(K, cv_auc, label='CV AUC points')

plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```

```
  0%|
| 0/8 [00:00<?, ?it/s]

 12%|
| 1/8 [04:18<30:07, 258.16s/it]

 25%|
| 2/8 [08:57<26:26, 264.47s/it]

 38%|
| 3/8 [13:40<22:29, 269.96s/it]

 50%|
| 4/8 [18:17<18:08, 272.11s/it]

 62%|
| 5/8 [22:50<13:36, 272.33s/it]

 75%|
| 6/8 [27:20<09:03, 271.69s/it]

 88%|
           | 7/8 [31:52<04:31, 271.70s/it]

100%|
          | 8/8 [36:22<00:00, 271.38s/it]
```



FROM ABOVE PLOT WE CAN SAY THAT THE BEST K WILL BE AROUND 35 BECAUSE AFTER THAT THE GRAPH OF CV_AUC IS ALMOST CONSTANT HENCE LET USE CONSIDER BEST K = 35

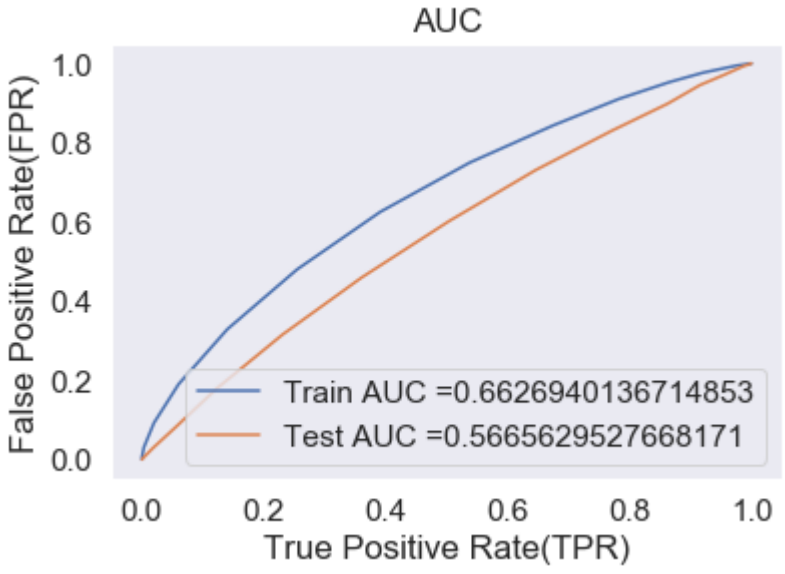
In [126]:

```
best_k_set_2 = 41
```

NOW USING THE BEST K VALUE RECEIVED FROM ABOVE WE WILL BE TRAINING OUR MODEL

In [127]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


neigh = KNeighborsClassifier(n_neighbors=best_k_set_2)
neigh.fit(X_train_merge_set_2, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred =  neigh.predict_proba(X_train_merge_set_2)[:,1]
y_test_pred =  neigh.predict_proba(X_test_merge_set_2)[:,1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



Creating the confusion matrix for the above data results

In [128]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_f
pr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr
)))
```

```
====================================================================================
==========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24856813276940434 for threshold 0.829
[[ 3432  3994]
 [10419 31196]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24998589797327114 for threshold 0.854
[[ 2709  2750]
 [12186 18407]]
```

Showing the train confusion matrix visually

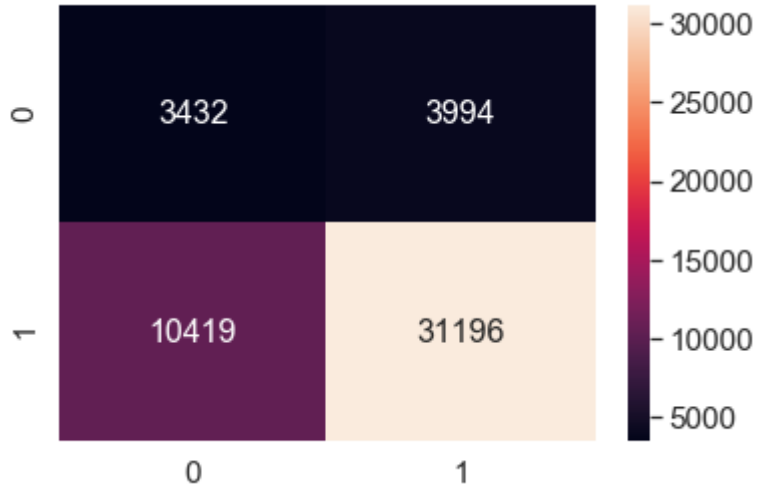
In [129]:

```
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how
-can-i-plot-a-confusion-matrix

import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt

df_cm_train = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_threshold
s, train_fpr, train_fpr)), range(2),
                  range(2))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm_train, annot=True,annot_kws={"size": 16},fmt = 'g')# font size
```

```
the maximum value of tpr*(1-fpr) 0.24856813276940434 for threshold 0.829
```

Out[129]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22b949fa320>
```



Showing the confusion matrix for the test data visually

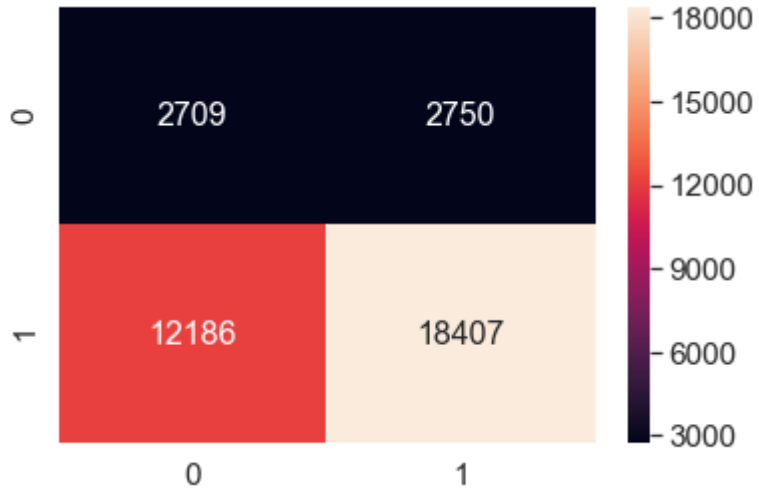
In [130]:

```
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how
-can-i-plot-a-confusion-matrix

import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt

df_cm_train = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds,
test_fpr, test_fpr)))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm_train, annot=True,annot_kws={"size": 16},fmt = 'g')# font size
```

```
the maximum value of tpr*(1-fpr) 0.24998589797327114 for threshold 0.854
```

Out[130]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22b9454ad68>
```



Here we can see that false positives and true positives are high in number it might also link towards the fact that our model is biased towards the positive class.

In [ ]:

```

```

Set 3 : categorical, numerical features + project_title(AVG W2V) + preprocessed_essay (AVG W2V)

In [131]:

```
from scipy.sparse import hstack

X_train_merge_set_3 = hstack((categories_one_hot_train,subcategories_one_hot_train,teac
her_prefix_one_hot_train,project_grade_categories_one_hot_train,school_state_categories
_one_hot_train,avg_w2v_vectors_train,avg_w2v_vectors_project_title_train,price_train,qu
antity_train,prev_projects_train)).tocsr()
X_test_merge_set_3 = hstack((categories_one_hot_test,subcategories_one_hot_test,teacher
_prefix_one_hot_test,project_grade_categories_one_hot_test,school_state_categories_one_
hot_test,avg_w2v_vectors_test,avg_w2v_vectors_project_title_test,price_test,quantity_te
st,prev_projects_test)).tocsr()
X_cv_merge_set_3 = hstack((categories_one_hot_cv,subcategories_one_hot_cv,teacher_prefi
x_one_hot_cv,project_grade_categories_one_hot_cv,school_state_categories_one_hot_cv,avg
_w2v_vectors_cv,avg_w2v_vectors_project_title_cv,price_cv,quantity_cv,prev_projects_cv
)).tocsr()
```

In [132]:

```
# this will be our finally created data matrix dimensions

print(X_train_merge_set_3.shape, y_train.shape)
print(X_test_merge_set_3.shape, y_test.shape)
print(X_cv_merge_set_3.shape, y_cv.shape)
```

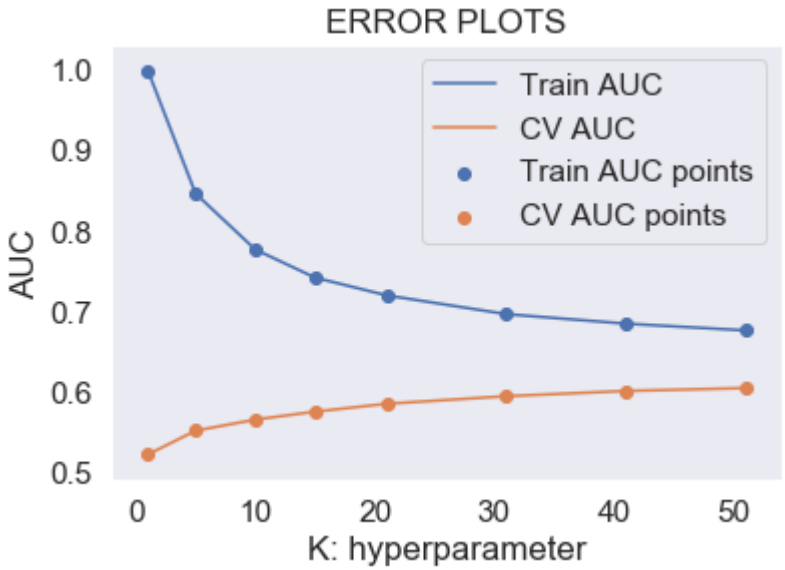
```
(49041, 702) (49041,)
(36052, 702) (36052,)
(24155, 702) (24155,)
```

Hyper parameter Tuning (USING THE BASIC FOR LOOP METHOD GRIDSEARCH-CV CAN ALSO BE USED HERE)

In [133]:

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score

"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence va
lues, or non-thresholded measure of
decisions (as returned by “decision_function” on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

"""

train_auc = []
cv_auc = []
a = []
b = []

K = [1, 5, 10, 15, 21, 31, 41, 51,]

for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_train_merge_set_3, y_train)

    y_train_pred =  neigh.predict_proba(X_train_merge_set_3)[:,1]
    y_cv_pred =  neigh.predict_proba(X_cv_merge_set_3)[:,1]


    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
 of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
    a.append(y_train_pred)
    b.append(y_cv_pred)

plt.plot(K, train_auc, label='Train AUC')
plt.plot(K, cv_auc, label='CV AUC')

plt.scatter(K, train_auc, label='Train AUC points')
plt.scatter(K, cv_auc, label='CV AUC points')

plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```

```
  0%|
| 0/8 [00:00<?, ?it/s]

 12%|
| 1/8 [1:02:21<7:16:27, 3741.02s/it]

 25%|
| 2/8 [2:05:53<6:16:15, 3762.60s/it]

 38%|
| 3/8 [3:10:13<5:15:57, 3791.53s/it]

 50%|
| 4/8 [4:13:08<4:12:27, 3786.78s/it]

 62%|
| 5/8 [10:16:03<7:39:09, 9183.31s/it]

 75%|
| 6/8 [16:44:54<7:27:34, 13427.49s/it]

 88%|
| 7/8 [18:23:57<3:06:21, 11182.00s/it]

100%|
| 8/8 [19:36:54<00:00, 9140.60s/it]
```



FROM ABOVE PLOT WE CAN SAY THAT THE BEST K WILL BE AROUND 35 BECAUSE AFTER THAT THE GRAPH OF CV_AUC IS ALMOST CONSTANT HENCE LET USE CONSIDER BEST K = 35

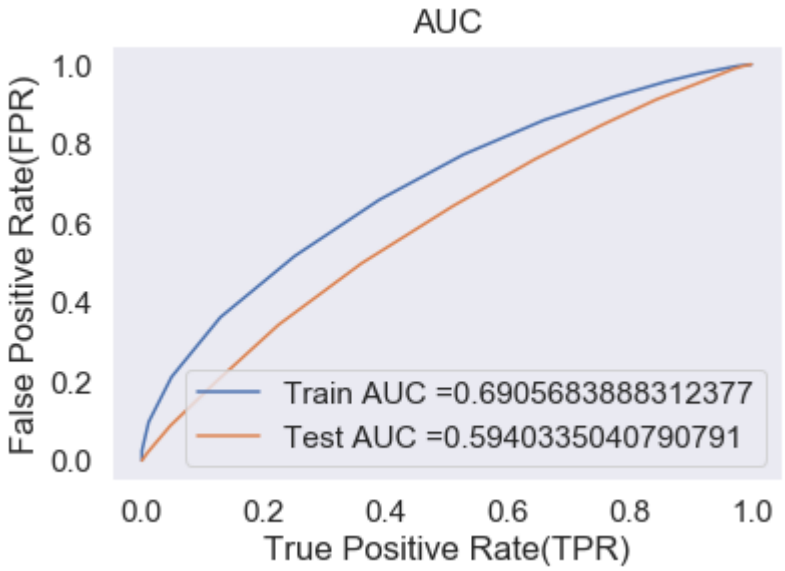
In [134]:

```
best_k_set_3 = 35
```

NOW USING THE BEST K VALUE RECEIVED FROM ABOVE WE WILL BE TRAINING OUR MODEL

In [135]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


neigh = KNeighborsClassifier(n_neighbors=best_k_set_3)
neigh.fit(X_train_merge_set_3, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred =  neigh.predict_proba(X_train_merge_set_3)[:,1]
y_test_pred =  neigh.predict_proba(X_test_merge_set_3)[:,1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



Creating the confusion matrix for the above data results

In [136]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_f
pr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr
)))
```

```
====================================================================================
==========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24925272201839369 for threshold 0.829
[[ 3510  3916]
 [ 9472 32143]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24990032945891405 for threshold 0.857
[[ 2675  2784]
 [10949 19644]]
```

Showing the train confusion matrix visually

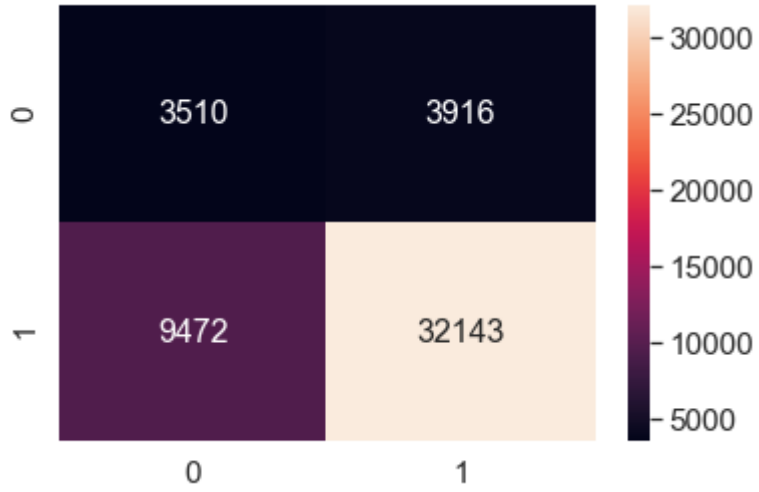
In [140]:

```
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how
-can-i-plot-a-confusion-matrix

import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt

df_cm_train = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_threshold
s, train_fpr, train_fpr)), range(2),
                  range(2))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm_train, annot=True,annot_kws={"size": 16},fmt = 'g')# font size
```

```
the maximum value of tpr*(1-fpr) 0.24925272201839369 for threshold 0.829
```

Out[140]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22b945ce5f8>
```



We can see above clearly that the number of False positives are more than the number of False Negatives

Showing the test confusion matrix visually

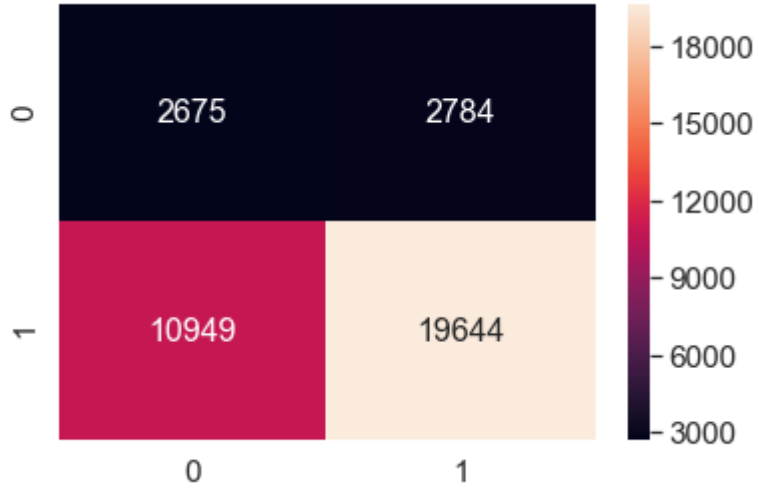
In [141]:

```
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how
-can-i-plot-a-confusion-matrix

import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt

df_cm_train = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds,
test_fpr, test_fpr)))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm_train, annot=True,annot_kws={"size": 16},fmt = 'g')# font size
```

```
the maximum value of tpr*(1-fpr) 0.24990032945891405 for threshold 0.857
```

Out[141]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22be57c8860>
```



Here we can see that we are getting a lot of False positives and true positives which directly indicate that our model is biased towards the positive class.

SET-4 Applying KNN brute force on TFIDF W2V

categorical, numerical features + project_title(TFIDF W2V) + preprocessed_essay (TFIDF W2V)

In [185]:

```
from scipy.sparse import hstack

X_train_merge_set_4 = hstack((categories_one_hot_train,subcategories_one_hot_train,teac
her_prefix_one_hot_train,project_grade_categories_one_hot_train,school_state_categories
_one_hot_train,tfidf_w2v_vectors_text_train,tfidf_w2v_vectors_project_title_train,price
_train,quantity_train,prev_projects_train)).tocsr()
X_test_merge_set_4 = hstack((categories_one_hot_test,subcategories_one_hot_test,teacher
_prefix_one_hot_test,project_grade_categories_one_hot_test,school_state_categories_one_
hot_test,tfidf_w2v_vectors_text_test,tfidf_w2v_vectors_project_title_test,price_test,qu
antity_test,prev_projects_test)).tocsr()
X_cv_merge_set_4 = hstack((categories_one_hot_cv,subcategories_one_hot_cv,teacher_prefi
x_one_hot_cv,project_grade_categories_one_hot_cv,school_state_categories_one_hot_cv,tfi
df_w2v_vectors_text_cv,tfidf_w2v_vectors_project_title_cv,price_cv,quantity_cv,prev_pro
jects_cv)).tocsr()
```

In [186]:

```
# this will be our finally created data matrix dimensions

print(X_train_merge_set_4.shape, y_train.shape)
print(X_test_merge_set_4.shape, y_test.shape)
print(X_cv_merge_set_4.shape,y_cv.shape)
```

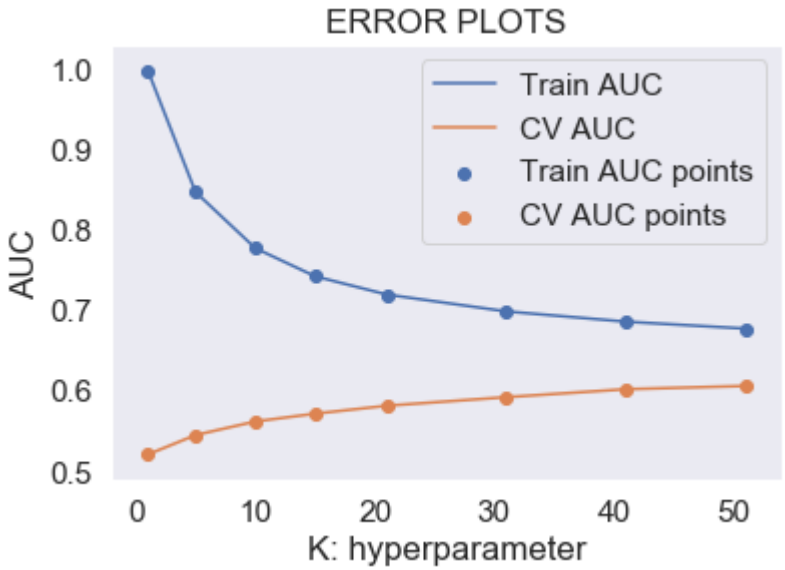
```
(49041, 702) (49041,)
(36052, 702) (36052,)
(24155, 702) (24155,)
```

Hyper parameter Tuning (USING THE BASIC FOR LOOP METHOD GRIDSEARCH-CV CAN ALSO BE USED HERE)

In [187]:

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score

"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence va
lues, or non-thresholded measure of
decisions (as returned by "decision_function" on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

"""

train_auc = []
cv_auc = []
a = []
b = []

K = [1, 5, 10, 15, 21, 31, 41, 51,]

for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_train_merge_set_4, y_train)

    y_train_pred =  neigh.predict_proba(X_train_merge_set_4)[:,1]
    y_cv_pred =  neigh.predict_proba(X_cv_merge_set_4)[:,1]


    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
 of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
    a.append(y_train_pred)
    b.append(y_cv_pred)

plt.plot(K, train_auc, label='Train AUC')
plt.plot(K, cv_auc, label='CV AUC')

plt.scatter(K, train_auc, label='Train AUC points')
plt.scatter(K, cv_auc, label='CV AUC points')

plt.legend()

plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```

```
  0%|
| 0/8 [00:00<?, ?it/s]

 12%|
| 1/8 [1:02:49<7:19:45, 3769.32s/it]

 25%|
| 2/8 [2:06:29<6:18:28, 3784.70s/it]

 38%|
| 3/8 [3:10:13<5:16:22, 3796.41s/it]

 50%|
| 4/8 [4:11:43<4:10:57, 3764.38s/it]

 62%|
| 5/8 [5:12:47<3:06:42, 3734.19s/it]

 75%|
| 6/8 [6:16:34<2:05:24, 3762.06s/it]

 88%|
| 7/8 [7:20:15<1:02:59, 3779.81s/it]

100%|
| 8/8 [8:22:27<00:00, 3765.44s/it]
```



FROM ABOVE PLOT WE CAN SAY THAT THE BEST K WILL BE AROUND 35 BECAUSE AFTER THAT THE GRAPH OF CV_AUC IS ALMOST CONSTANT HENCE LET USE CONSIDER BEST K = 35

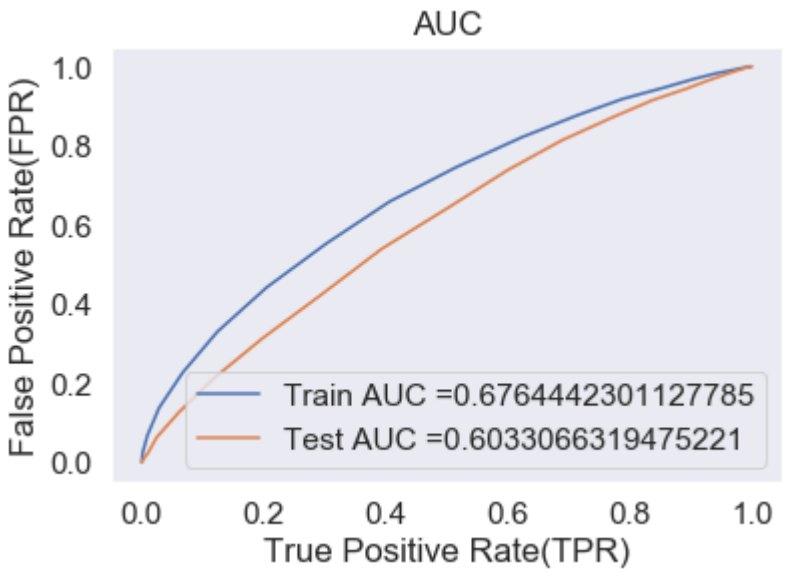
In [189]:

```
best_k_set_4 = 53
```

NOW USING THE BEST K VALUE RECEIVED FROM ABOVE WE WILL BE TRAINING OUR MODEL

In [190]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


neigh = KNeighborsClassifier(n_neighbors=best_k_set_4,n_jobs = -1)
neigh.fit(X_train_merge_set_4, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred =  neigh.predict_proba(X_train_merge_set_4)[:,1]
y_test_pred =  neigh.predict_proba(X_test_merge_set_4)[:,1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



Creating confusion matrix for the above results

In [191]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_f
pr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr
)))
```

```
====================================================================================
==========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.2497121069369317 for threshold 0.83
[[ 3587  3839]
 [10545 31070]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2499830121331791 for threshold 0.849
[[ 2707  2752]
 [10864 19729]]
```

Showing the train confusion matrix visually

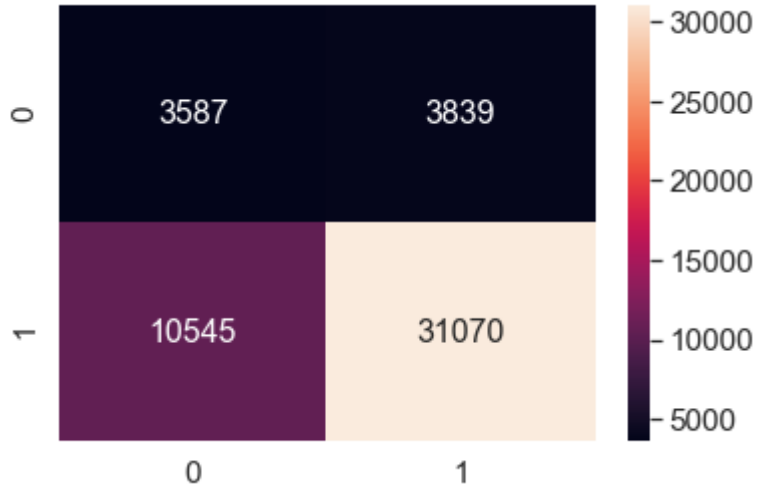
In [192]:

```
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how
-can-i-plot-a-confusion-matrix

import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt

df_cm_train = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_threshold
s, train_fpr, train_fpr)), range(2),
                  range(2))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm_train, annot=True,annot_kws={"size": 16},fmt = 'g')# font size
```

```
the maximum value of tpr*(1-fpr) 0.2497121069369317 for threshold 0.83
```

Out[192]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22b94440c50>
```



Showing the test confusion matrix visually

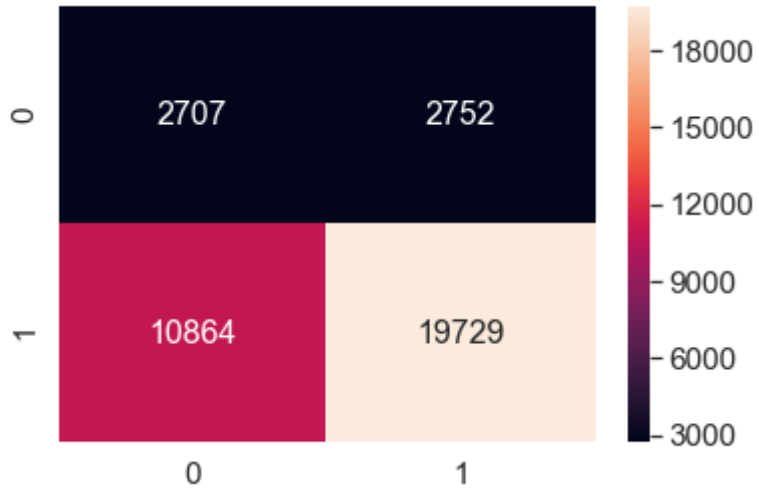
In [193]:

```
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how
-can-i-plot-a-confusion-matrix

import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt

df_cm_train = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds,
test_fpr, test_fpr)))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm_train, annot=True,annot_kws={"size": 16},fmt = 'g')# font size
```

```
the maximum value of tpr*(1-fpr) 0.2499830121331791 for threshold 0.849
```

Out[193]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22be57017b8>
```



The model has good amount of true positives which is a good sign.But a lot of negative class points are classified as positive class which is not a good sign.

Select Best K features On Set_2

In [195]:

```
print(X_train_merge_set_2.shape, y_train.shape)
print(X_test_merge_set_2.shape, y_test.shape)
print(X_cv_merge_set_2.shape, y_cv.shape)
```

```
(49041, 8102) (49041,)
(36052, 8102) (36052,)
(24155, 8102) (24155,)
```

Now above we have 8102 features out of which we want to select the best 2000 features

In [196]:

```
from sklearn.datasets import load_digits
from sklearn.feature_selection import SelectKBest, chi2
```

In [199]:

```
X_train_merge_set_2_best = SelectKBest(chi2, k=2000).fit_transform(X_train_merge_set_2,
y_train)
X_test_merge_set_2_best = SelectKBest(chi2, k=2000).fit_transform(X_test_merge_set_2,y_
test)
X_cv_merge_set_2_best = SelectKBest(chi2, k=2000).fit_transform(X_cv_merge_set_2,y_cv)
```

In [200]:

```
print(X_train_merge_set_2_best.shape)
print(X_test_merge_set_2_best.shape)
print(X_cv_merge_set_2_best.shape)
```

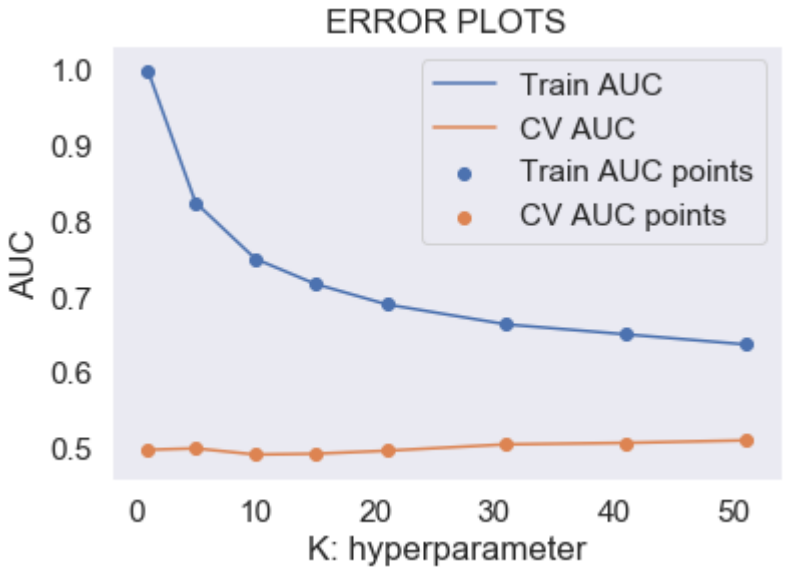
```
(49041, 2000)
(36052, 2000)
(24155, 2000)
```

Hyper parameter Tuning for the best 2000 features (USING THE BASIC FOR LOOP METHOD GRIDSEARCH-CV CAN ALSO BE USED HERE)

In [201]:

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score

"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence va
lues, or non-thresholded measure of
decisions (as returned by “decision_function” on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

"""

train_auc = []
cv_auc = []
a = []
b = []

K = [1, 5, 10, 15, 21, 31, 41, 51]

for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_train_merge_set_2_best, y_train)

    y_train_pred =  neigh.predict_proba(X_train_merge_set_2_best)[:,1]
    y_cv_pred =  neigh.predict_proba(X_cv_merge_set_2_best)[:,1]


    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
 of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
    a.append(y_train_pred)
    b.append(y_cv_pred)

plt.plot(K, train_auc, label='Train AUC')
plt.plot(K, cv_auc, label='CV AUC')

plt.scatter(K, train_auc, label='Train AUC points')
plt.scatter(K, cv_auc, label='CV AUC points')

plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```

```
  0%|
| 0/8 [00:00<?, ?it/s]

 12%|
| 1/8 [02:13<15:32, 133.28s/it]

 25%|
| 2/8 [04:45<13:53, 138.96s/it]

 38%|
| 3/8 [07:17<11:54, 142.89s/it]

 50%|
| 4/8 [09:50<09:43, 145.87s/it]

 62%|
| 5/8 [12:23<07:24, 148.08s/it]

 75%|
| 6/8 [14:58<05:00, 150.02s/it]

 88%|
           | 7/8 [17:30<02:30, 150.85s/it]

100%|
          | 8/8 [20:03<00:00, 151.25s/it]
```



From above we can easily see that for lower values of k we have large AUC on training data while we have Low AUC of the test data hence we can say that our model falls under the category of overfitting for those lower values of k.

Choosing the best k from above plot

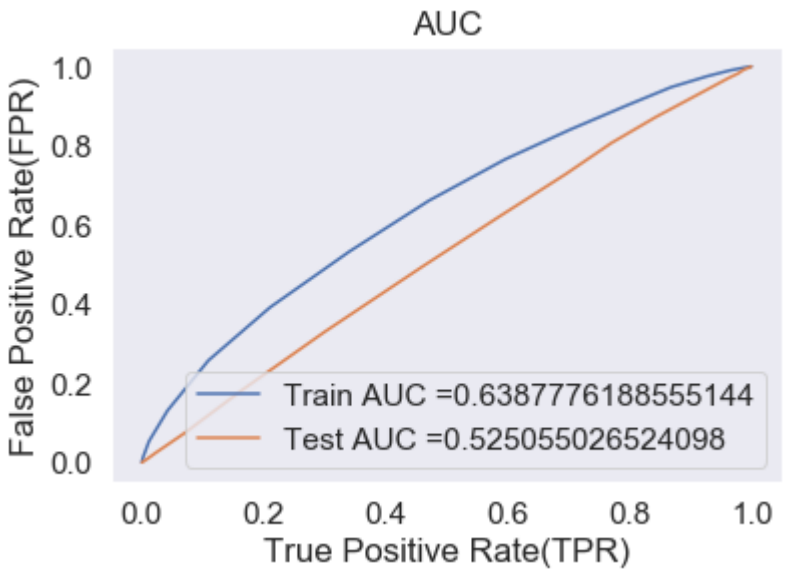
In [202]:

```
best_k_selecKBest = 49
```

NOW USING THE BEST K VALUE RECEIVED FROM ABOVE WE WILL BE TRAINING OUR MODEL

In [203]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


neigh = KNeighborsClassifier(n_neighbors=best_k_selecKBest)
neigh.fit(X_train_merge_set_2_best, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred =  neigh.predict_proba(X_train_merge_set_2_best)[:,1]
y_test_pred =  neigh.predict_proba(X_test_merge_set_2_best)[:,1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



Auc values received here for the test data are the worst it means that working on only 2000 features was not a good idea.

Now getting the confusion matrix for the train and test data based on above plots

In [204]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_f
pr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr
)))
```

```
====================================================================================
==========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.2491539469166972 for threshold 0.837
[[ 3929  3497]
 [14100 27515]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2483979476844841 for threshold 0.878
[[ 3822  1637]
 [20501 10092]]
```

visually plotting the confusion matrix for train data

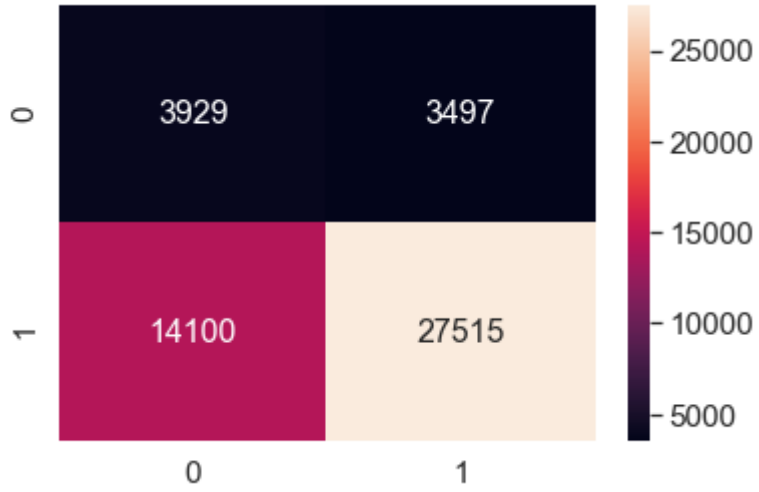
In [205]:

```
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how
-can-i-plot-a-confusion-matrix

import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt

df_cm_train = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_threshold
s, train_fpr, train_fpr)), range(2),
                  range(2))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm_train, annot=True,annot_kws={"size": 16},fmt = 'g')# font size
```

```
the maximum value of tpr*(1-fpr) 0.2491539469166972 for threshold 0.837
```

Out[205]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22be59e9208>
```



Visually plotting the confusion matrix for test data

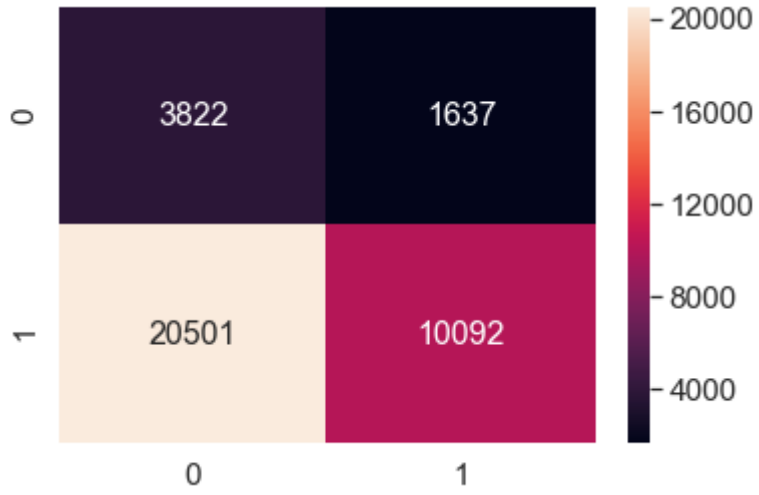
In [206]:

```
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how
-can-i-plot-a-confusion-matrix

import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt

df_cm_train = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds,
test_fpr, test_fpr)))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm_train, annot=True,annot_kws={"size": 16},fmt = 'g')# font size
```

```
the maximum value of tpr*(1-fpr) 0.2483979476844841 for threshold 0.878
```

Out[206]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22b9490b5c0>
```



Cannot find much of good predictions and also the model is giving the worse AUC as compared to previous models which we did in different sets.

Conclusions

In [207]:

```
# Compare all your models using Prettytable library
# http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

#If you get a ModuleNotFoundError error , install prettytable using: pip3 install prett
ytable

x = PrettyTable()
x.field_names = ["Vectorizer Technique", "Model", "Hyper Parameter K Value", "AUC"]

x.add_row(["BOW", "Brute", 31, 0.62])
x.add_row(["TFIDF", "Brute", 41, 0.56])
x.add_row(["AVG W2V", "Brute", 35, 0.59])
x.add_row(["TFIDF W2V", "Brute", 53, 0.60])

print(x)
```

```
+----------------------+-------+-------------------------+------+
| Vectorizer Technique | Model | Hyper Parameter K Value | AUC  |
+----------------------+-------+-------------------------+------+
|         BOW          | Brute |            31           | 0.62 |
|        TFIDF         | Brute |            41           | 0.56 |
|       AVG W2V        | Brute |            35           | 0.59 |
|      TFIDF W2V       | Brute |            53           | 0.6  |
+----------------------+-------+-------------------------+------+
```

Looks like k in the range of 31-41 will work good based on the trainings for different vectorizations techniques.

In [ ]:

```

```

In [ ]:

```

```