2# algorithm - random sort

In [423…
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import math
import random
from functools import total_ordering
import bisect
from heapq import merge
from collections import Counter
from tqdm import tqdm
```

Helping Functions

In [424…
```python
#make random students
#make the students_df
#sort by gpa
def make_random_students(num_students):
    sat = np.random.randint(200, 801, size=num_students)
    gpa = np.random.randint(0, 101, size = num_students)

    data_dict = {"gpa" : gpa, "sat" : sat}
    students_df = pd.DataFrame.from_dict(data_dict)
    #left out the gpa sort
    return students_df
```

In [425…
```python
#makes the df_diamond
def diamond_branch(df):
    df_diamond = df.copy(deep=True)
    in_diamond = [0]*df.shape[0]
    for index, row in df.iterrows():
        sat = row["sat"]
        gpa = row["gpa"]
        if (((sat-5*gpa) < 500) & ((sat+5*gpa) < 1100) & ((sat-5*gpa) > -100) & ((sat+5*gpa
            in_diamond[index] = 1

    df_diamond['in_diamond'] = in_diamond #add col
    df_diamond = df_diamond[df_diamond['in_diamond']>0] #leave only thoser that have 1
    df_diamond = df_diamond.drop('in_diamond', axis=1) #erases in_diamond col
    df_diamond = df_diamond.reset_index(drop=True)

    n_d = df_diamond.shape[0]
    df_diamond = df_diamond[:n_d - n_d%10]

    return df_diamond
```

In [426…
```python
def random_sort(df, interactions, num_colors=10):
    nparr = np.array(df)
    n = len(nparr)
    for y in tqdm(range(interactions)):
        i = np.random.randint(0, n-1)
        if not((nparr[i,0] < nparr[i+1,0]) and (nparr[i,1] < nparr[i+1,1])):
            nparr[i+1], nparr[i] = nparr[i].copy(), nparr[i+1].copy()
    df_sorted = pd.DataFrame(nparr, columns = ['gpa','sat','color'])
    df_sorted['color'] = np.array([[i]*int(n/num_colors) for i in range(num_colors)]).flatt
    return df_sorted
```

Part 1 - Random Students and stacks by percentage analysis

In [427…
```python
num_students = 10000
students_df = make_random_students(num_students)
students_df
```

Out[427]:

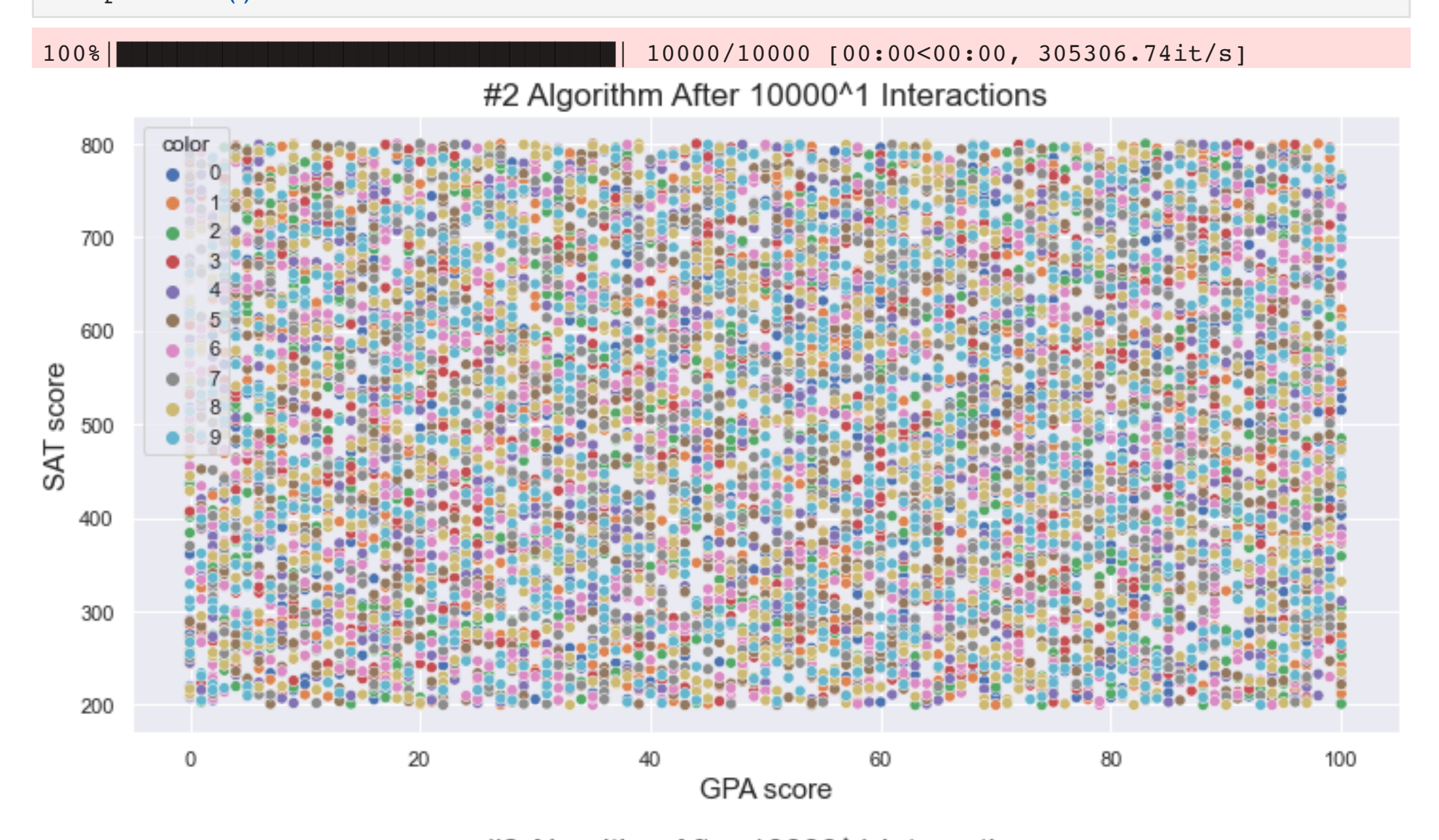
	gpa	sat
0	67	604
1	50	299
2	58	322
3	61	741
4	39	523
...	...	...
9995	65	673
9996	65	560
9997	10	629
9998	19	403
9999	46	660

10000 rows × 2 columns

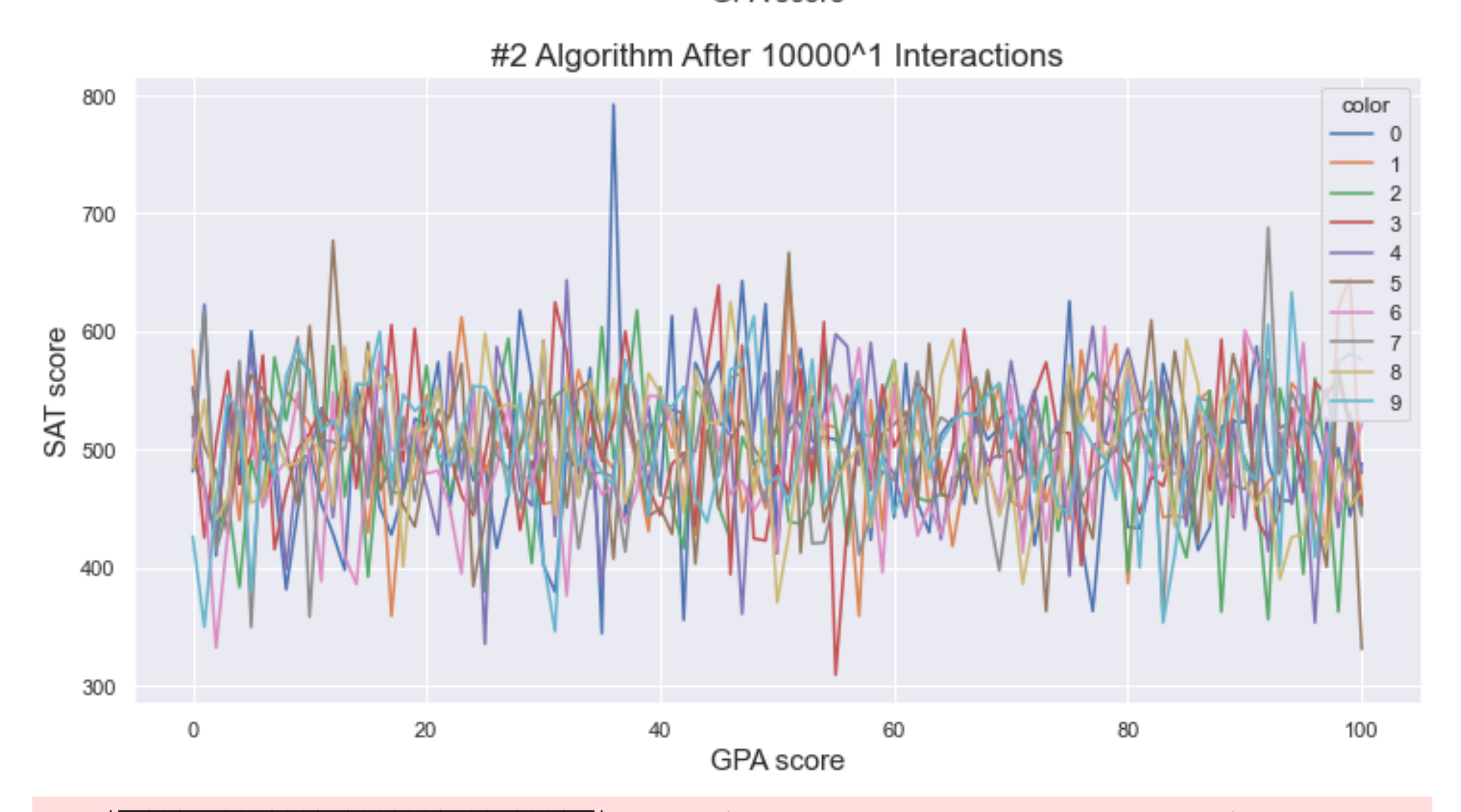
In [428…
```python
students_df_sort = students_df.copy()
students_df_sort['color'] = [0]*students_df_sort.shape[0]
for cycle in [1,1.3,1.6,1.9,2.2,2.5,2.8,3,'n3logn',3.3]:
    if cycle == 1:
        interactions = int(pow(num_students, cycle))
    elif cycle == 'n3logn':
        cycle = '3*log\({}\)'.format(num_students)
        interactions = int(pow(num_students,3)*math.log(num_students)-pow(num_students, 3))
    else:
        interactions = int(pow(num_students, cycle)-pow(num_students, cycle-0.5))
    students_df_sort = random_sort(students_df_sort, interactions, num_colors=10)

    sns.scatterplot(data=students_df_sort, x="gpa", y="sat", hue="color", palette='deep', m
    sns.set(rc = {'figure.figsize':(12,6)})
    plt.title('#2 Algorithm After {}^{} Interactions'.format(num_students, cycle), size = 1
    plt.ylabel('SAT score', size = 15)
    plt.xlabel('GPA score', size = 15)
    plt.show()
    if cycle
    sns.lineplot(data=students_df_sort, x="gpa", y="sat", hue="color", palette='deep', mark
    sns.set(rc = {'figure.figsize':(12,6)})
    plt.title('#2 Algorithm After {}^{} Interactions'.format(num_students, cycle), size = 1
    plt.ylabel('SAT score', size = 15)
    plt.xlabel('GPA score', size = 15)
    plt.show()
```

```
100%|██████████| 10000/10000 [00:00<00:00, 305306.74it/s]
```

#2 Algorithm After 10000^1 Interactions

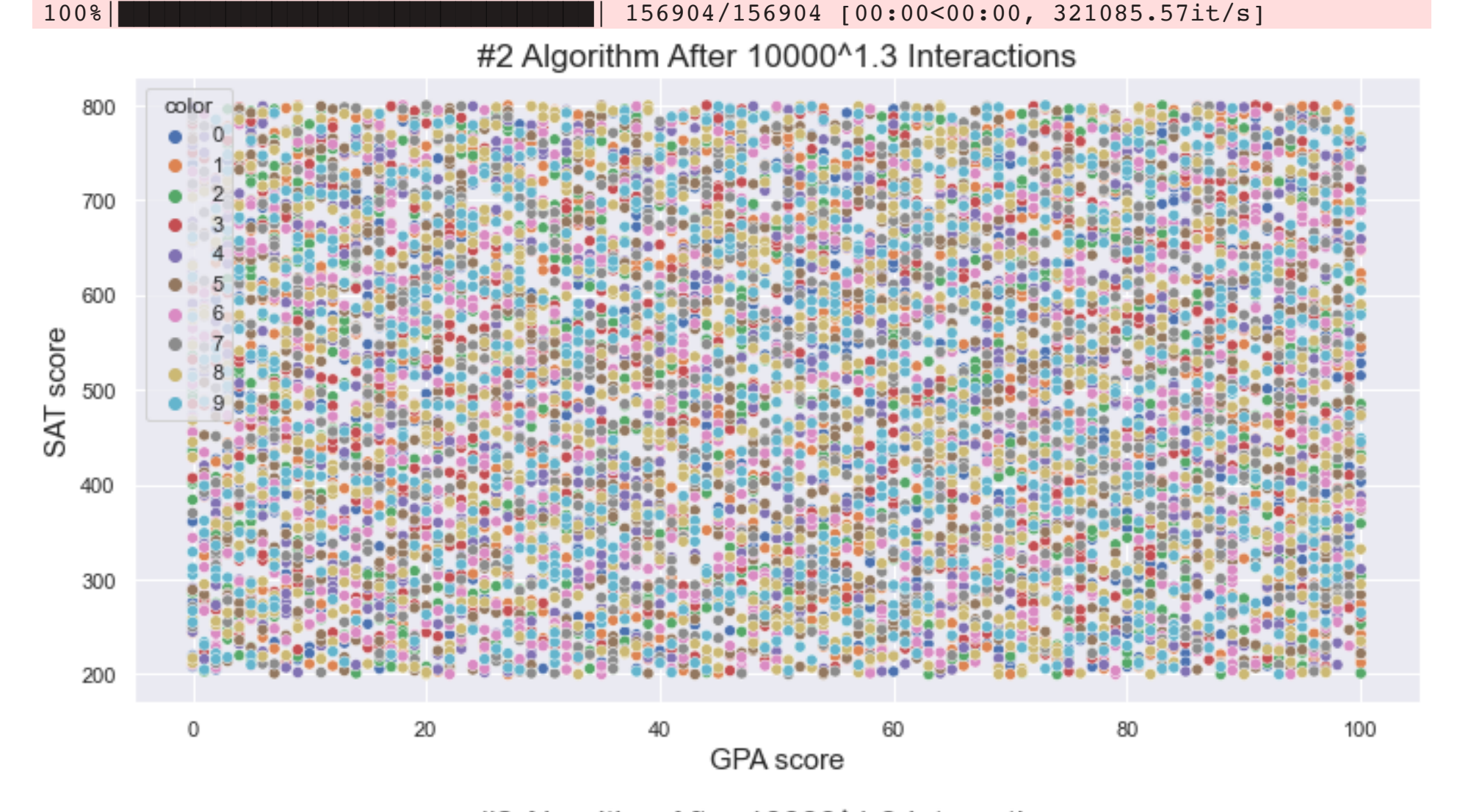


#2 Algorithm After 10000^1 Interactions

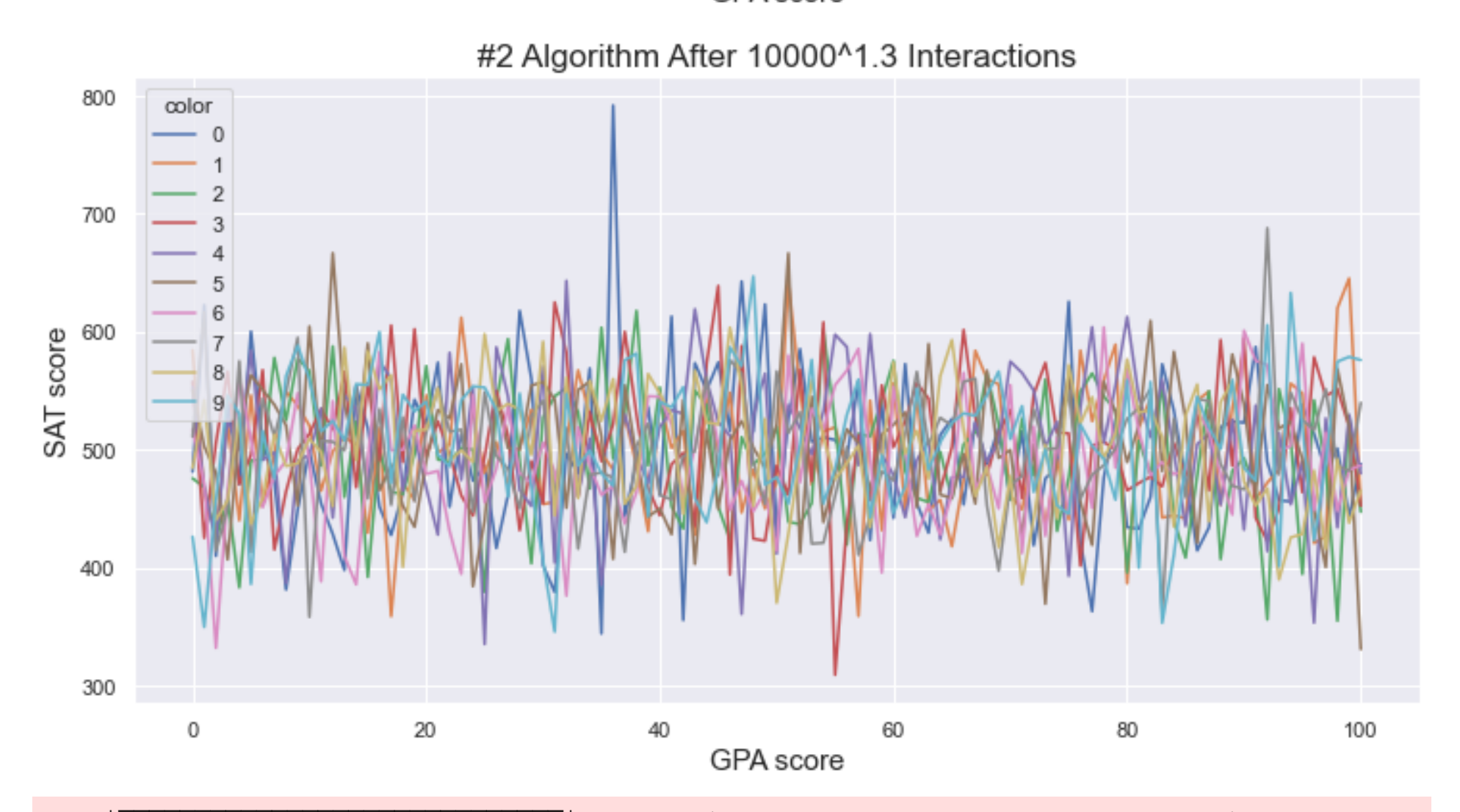


```
100%|██████████| 156904/156904 [00:00<00:00, 321085.57it/s]
```

#2 Algorithm After 10000^1.3 Interactions



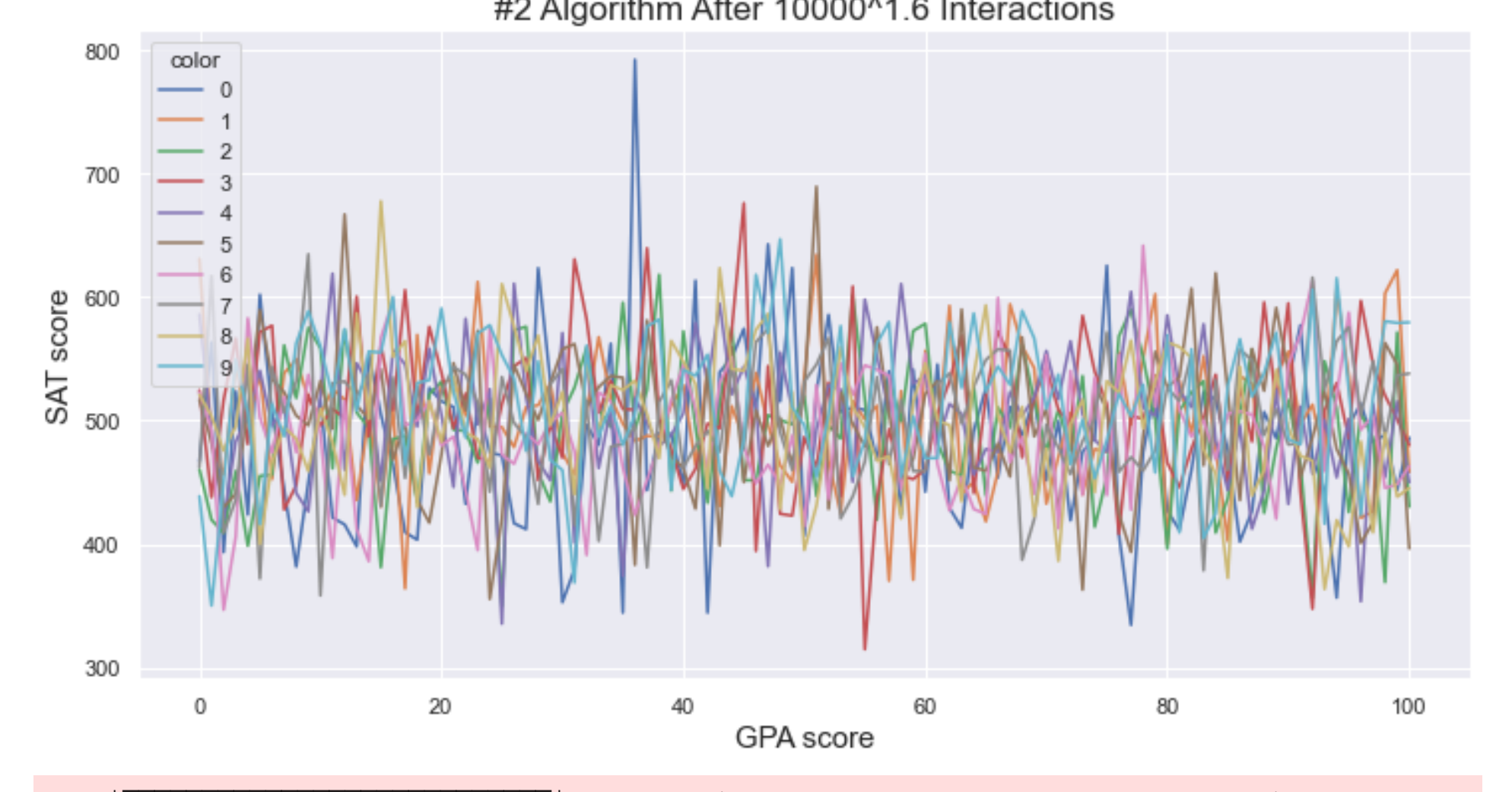
#2 Algorithm After 10000^1.3 Interactions



```
100%|██████████| 2486767/2486767 [00:08<00:00, 285045.98it/s]
```

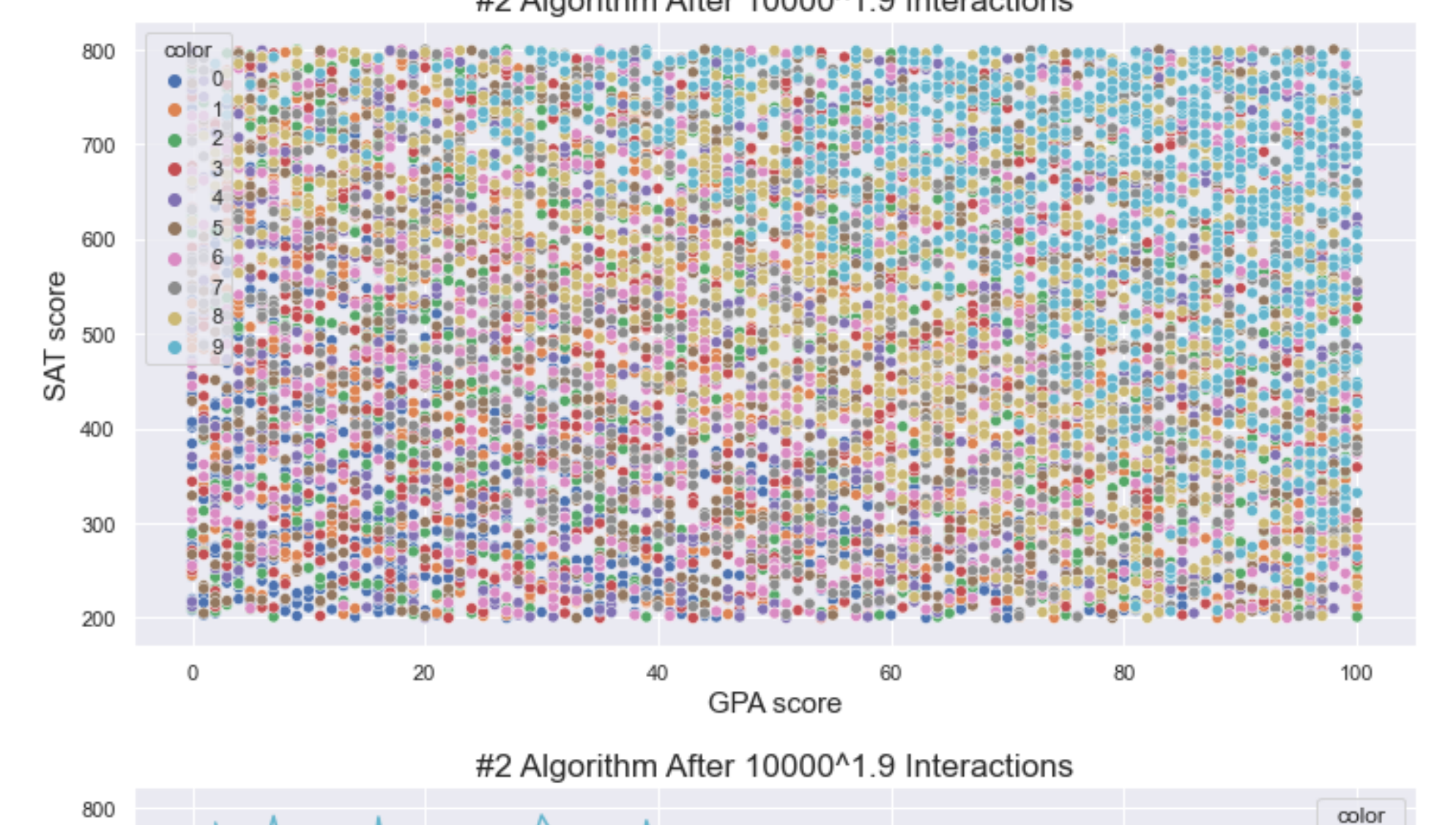
#2 Algorithm After 10000^1.6 Interactions

#2 Algorithm After 10000^1.6 Interactions

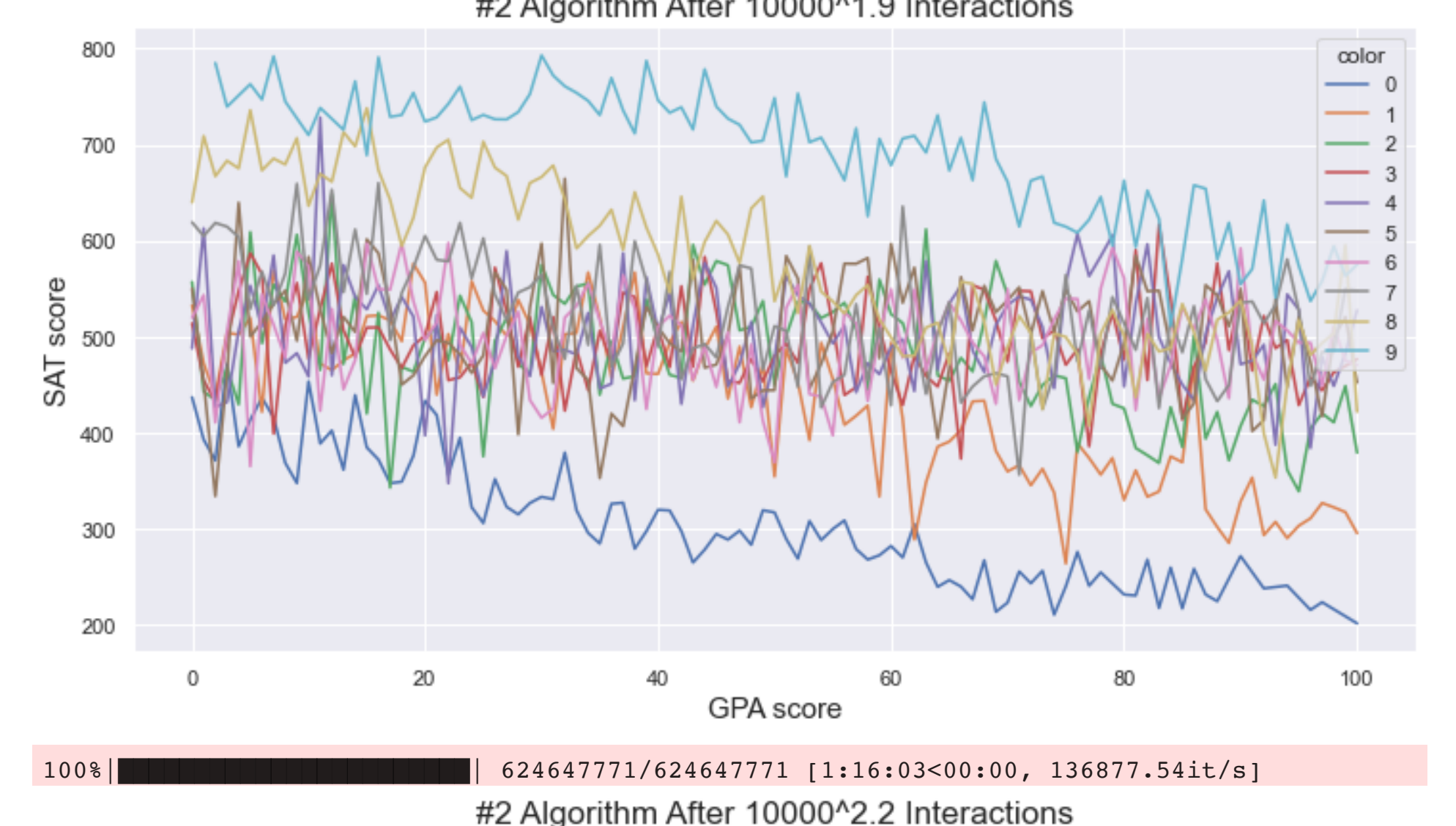


```
100%|██████████| 39412609/39412609 [02:16<00:00, 288999.35it/s]
```

#2 Algorithm After 10000^1.9 Interactions

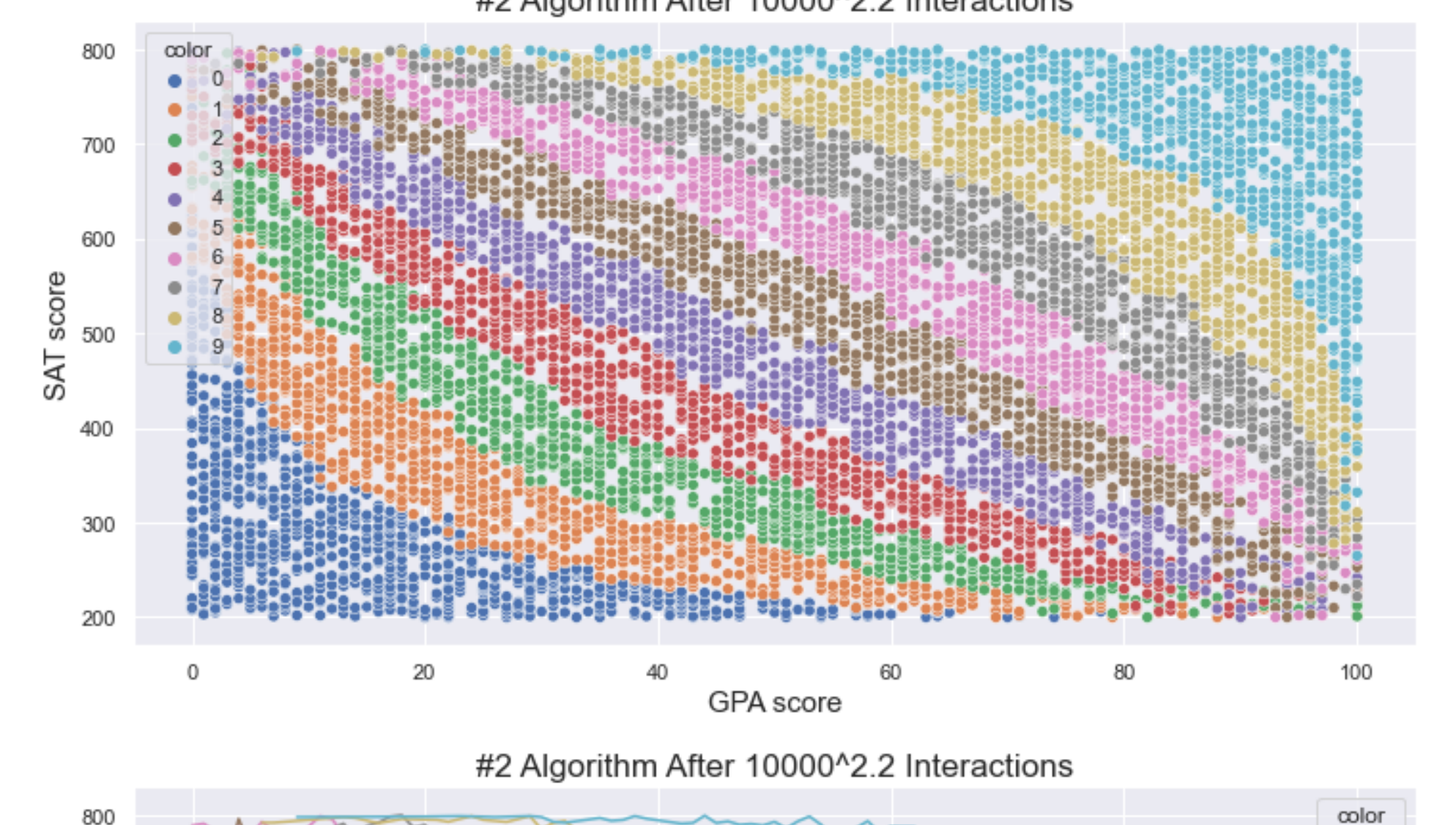


#2 Algorithm After 10000^1.9 Interactions

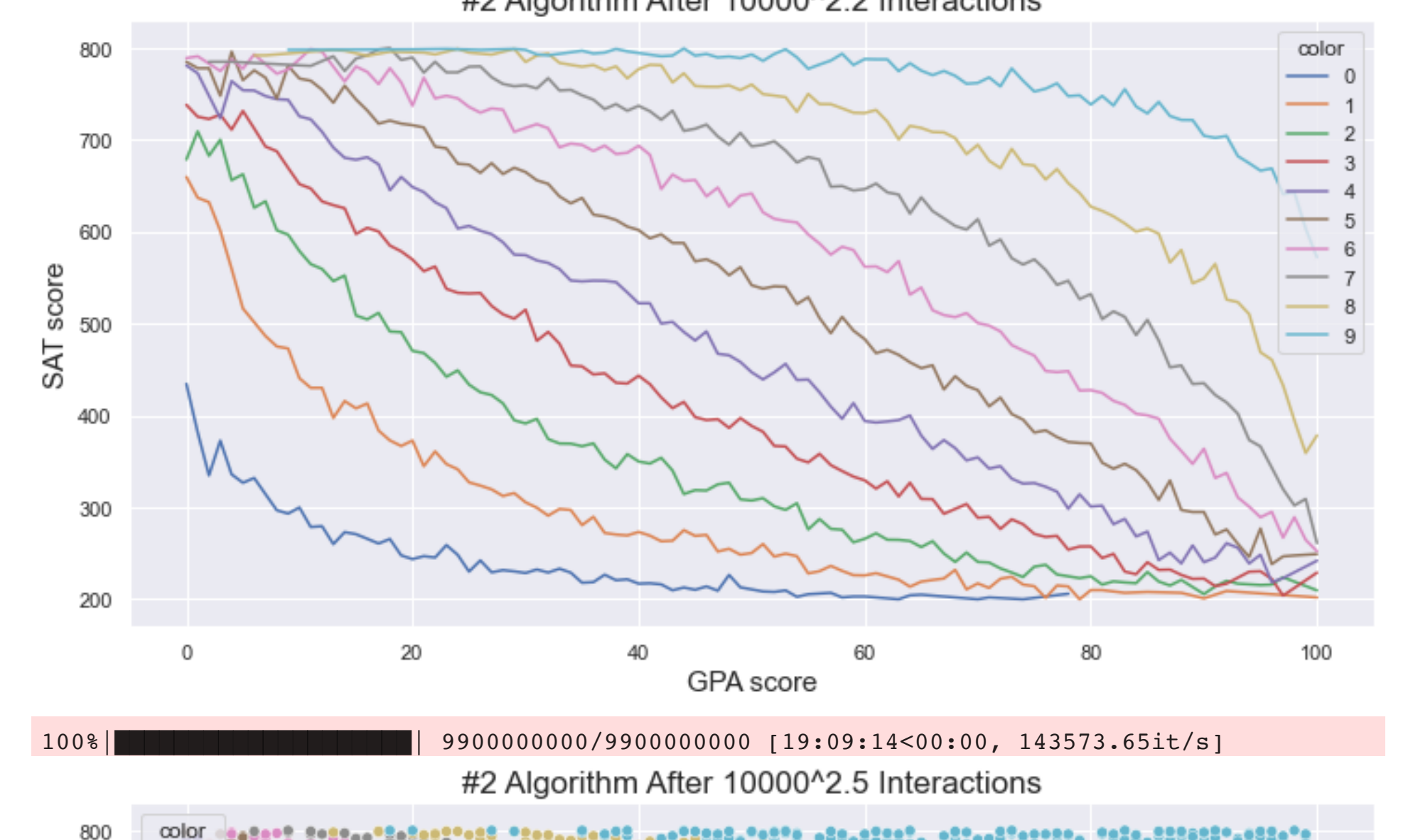


```
100%|██████████| 624647771/624647771 [1:16:03<00:00, 136877.54it/s]
```

#2 Algorithm After 10000^2.2 Interactions

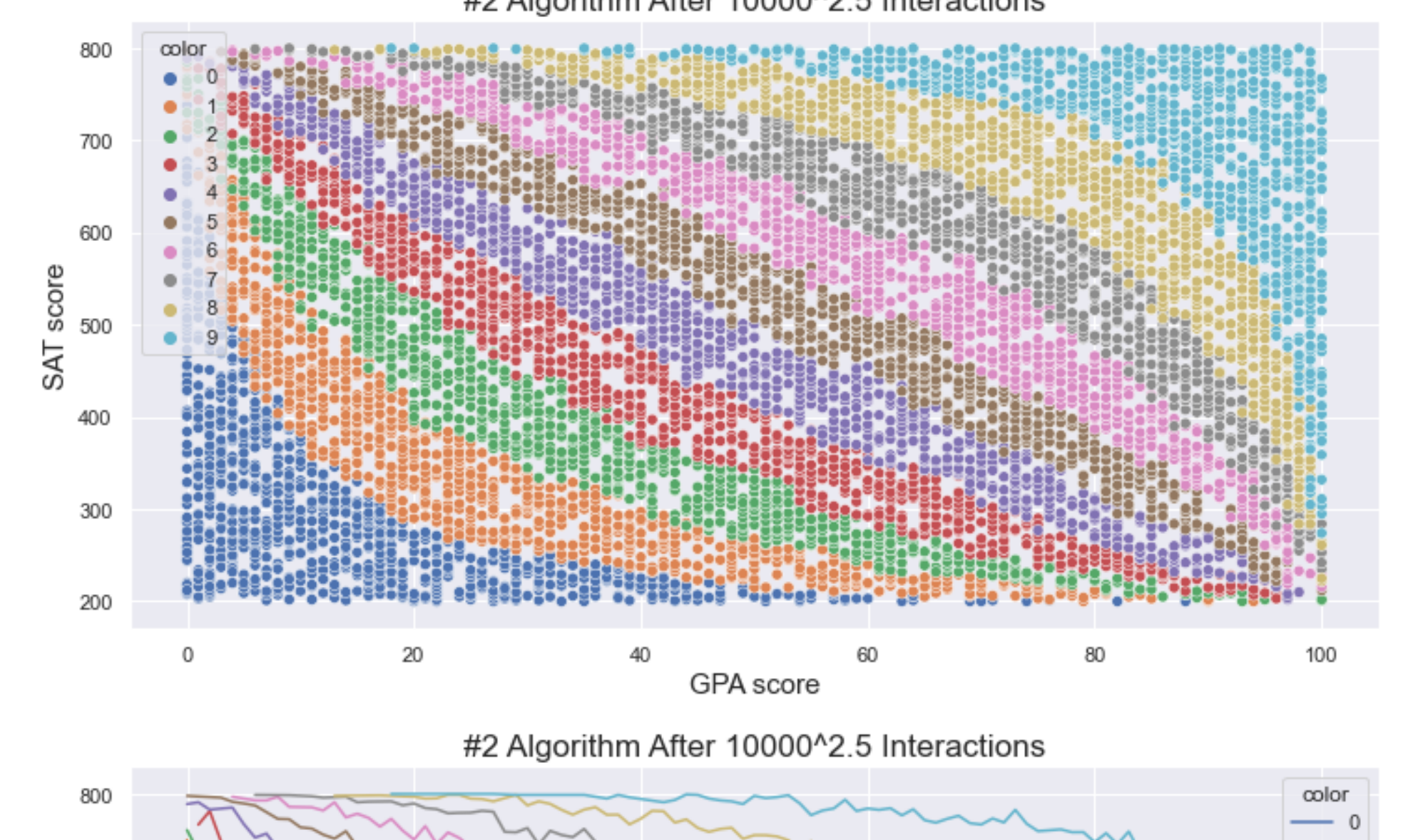


#2 Algorithm After 10000^2.2 Interactions

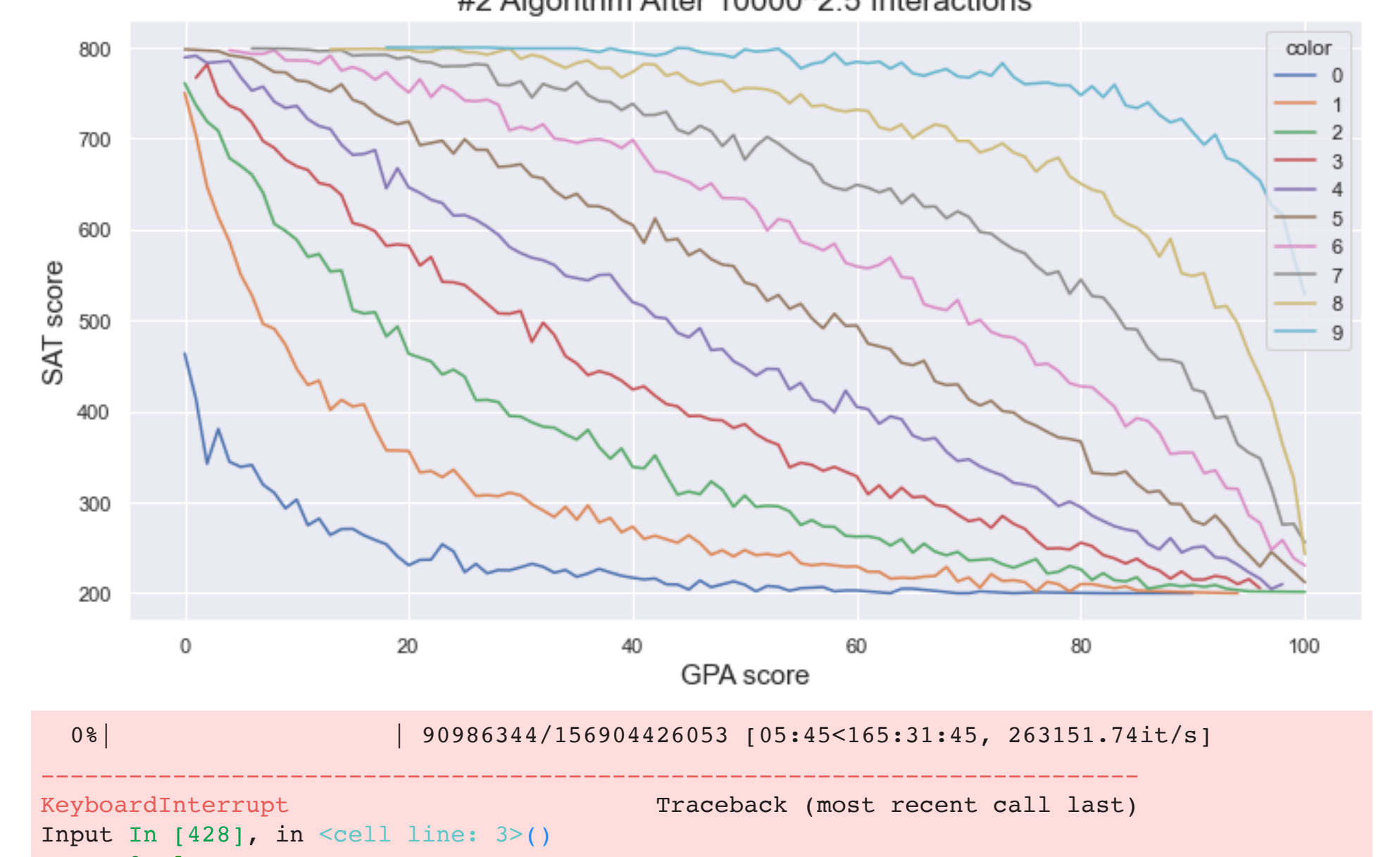


```
100%|██████████| 9900000000/9900000000 [19:09:14<00:00, 143573.65it/s]
```

#2 Algorithm After 10000^2.5 Interactions



#2 Algorithm After 10000^2.5 Interactions



```
  0%|          | 90986344/156904426053 [05:45<165:31:45, 263151.74it/s]
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Input In [428], in <cell line: 3>()
      9 else:
     10     interactions = int(pow(num_students, cycle)-pow(num_students, cycle-0.5))
---> 11 students_df_sort = random_sort(students_df_sort, interactions, num_colors=10)
     13 sns.scatterplot(data=students_df_sort, x="gpa", y="sat", hue="color", palette='dee
p', markers = True, ci= None)
     14 sns.set(rc = {'figure.figsize':(12,6)})

Input In [426], in random_sort(df, interactions, num_colors)
      3 n = len(nparr)
      4 for y in tqdm(range(interactions)):
----> 5     i = np.random.randint(0, n-1)
      6     if not((nparr[i,0] < nparr[i+1,0]) and (nparr[i,1] < nparr[i+1,1])):
      7         nparr[i+1], nparr[i] = nparr[i].copy(), nparr[i+1].copy()

KeyboardInterrupt:
```

In [365…
```python
diamond_df = diamond_branch(students_df)
print("Students in the Diamond Shape: {}".format(diamond_df.shape[0]))
diamond_df.head()
```

Students in the Diamond Shape: 130

Out[365]:

	gpa	sat
0	83	350
1	98	550
2	26	485
3	71	670
4	53	404

In [366…
```python
num_students = diamond_df.shape[0]
students_df_sort = diamond_df.copy()
for cycle in [1,1.3,1.6,1.9,2.2,2.5,2.8,3,'3 * log250',3.3]:
    if cycle == 1:
        interactions = int(pow(num_students, cycle))
    elif cycle == '3 * log250':
        interactions = int(pow(num_students,3)*math.log(num_students)-pow(num_students, 3))
    else:
        interactions = int(pow(num_students, cycle)-pow(num_students, cycle-0.5))
    students_df_sort = algorithm_2(students_df_sort, interactions, num_colors=10)

    sns.lineplot(data=students_df_sort, x="gpa", y="sat", hue="color", palette='deep', mark
    sns.set(rc = {'figure.figsize':(12,6)})
    plt.title('#2 Algorithm After {}^{} Interactions'.format(num_students, cycle), size = 1
    plt.ylabel('SAT score', size = 15)
    plt.xlabel('GPA score', size = 15)
    plt.show()
```

```
100%|██████████| 130/130 [00:00<00:00, 4653.06it/s]
```

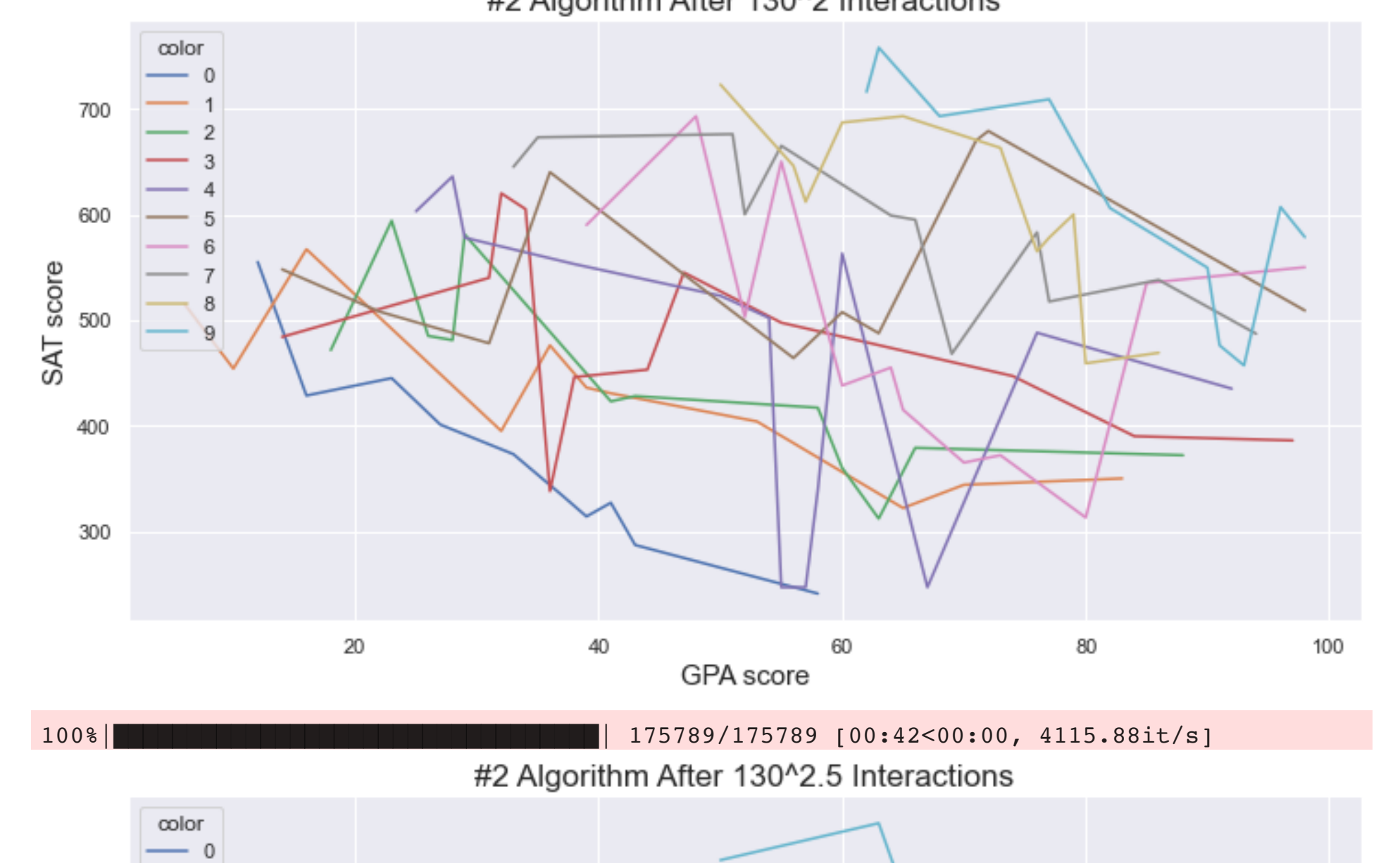
#2 Algorithm After 130^1 Interactions

```
100%|██████████| 1352/1352 [00:00<00:00, 5212.69it/s]
```

#2 Algorithm After 130^1.5 Interactions

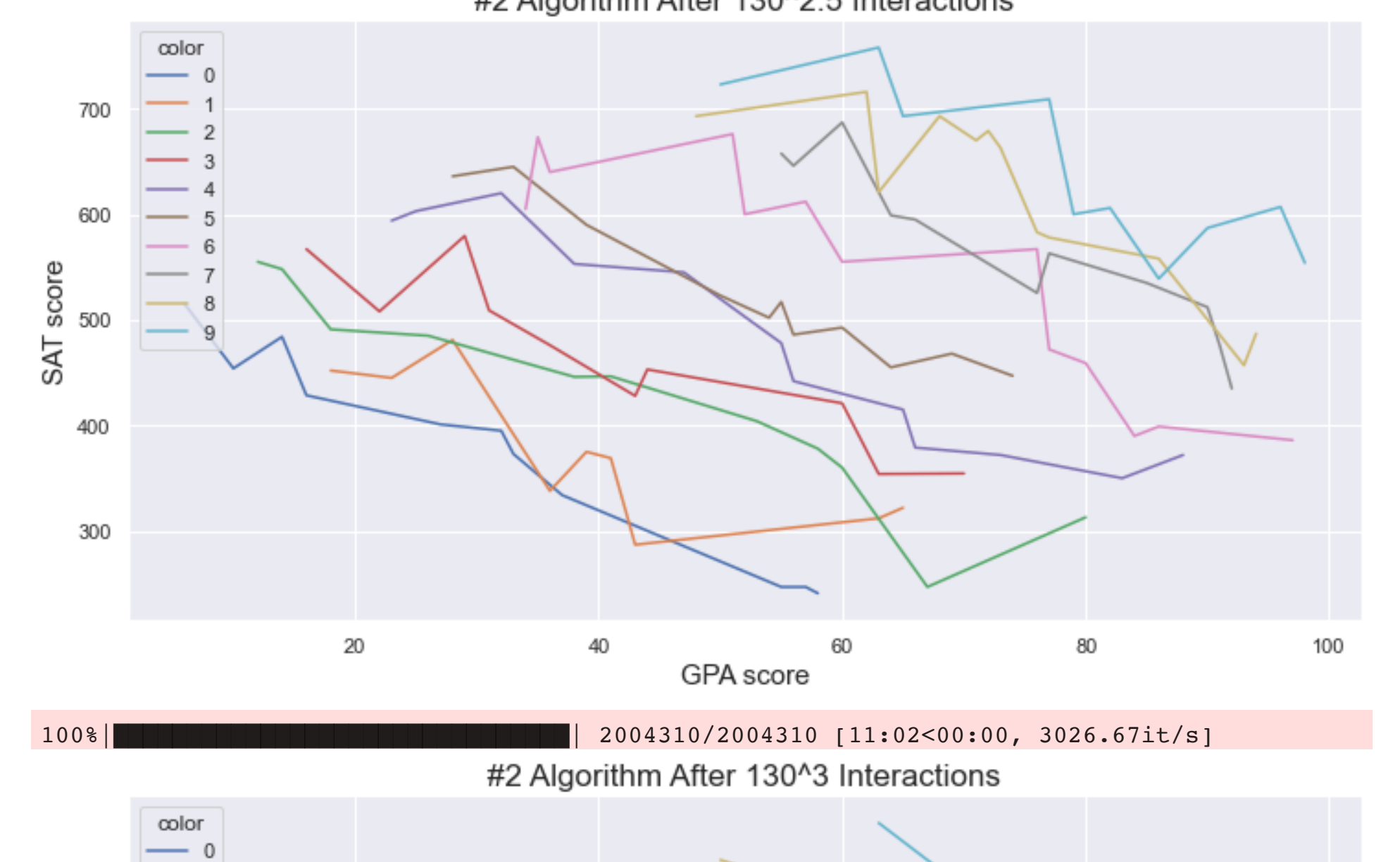
```
100%|██████████| 15417/15417 [00:03<00:00, 4684.16it/s]
```

#2 Algorithm After 130^2 Interactions



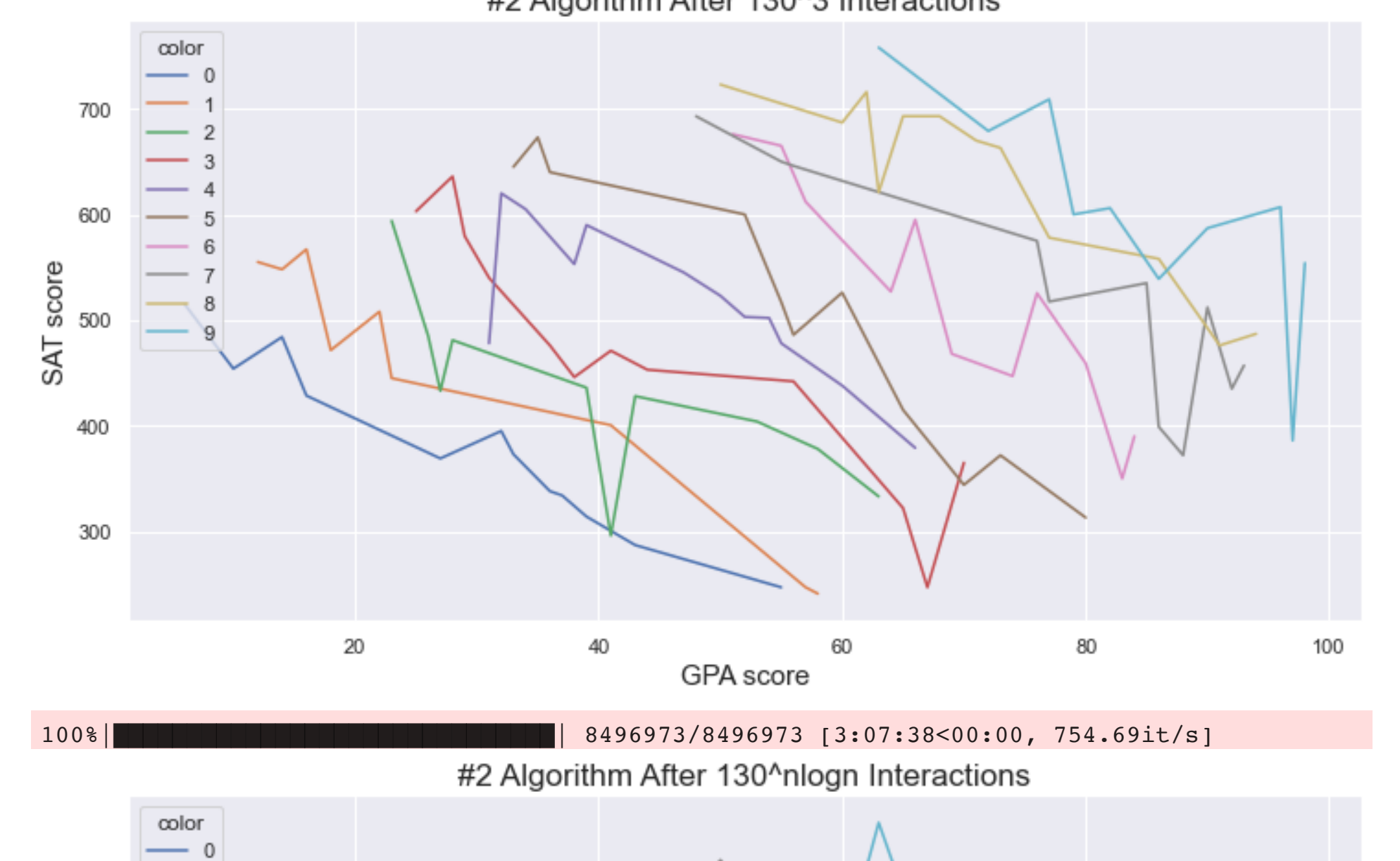
```
100%|██████████| 175789/175789 [00:42<00:00, 4115.88it/s]
```

#2 Algorithm After 130^2.5 Interactions



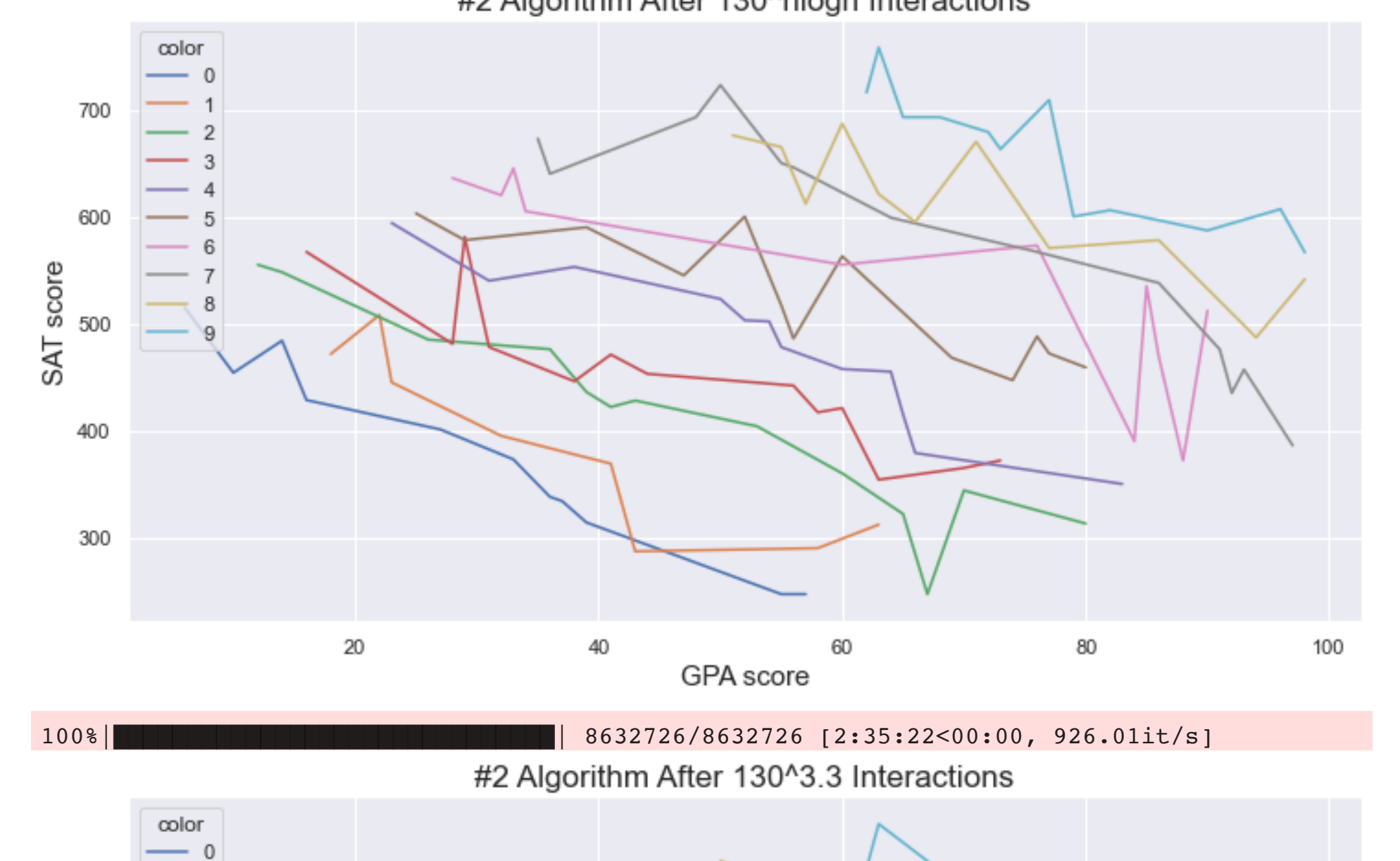
```
100%|██████████| 2004310/2004310 [11:02<00:00, 3026.67it/s]
```

#2 Algorithm After 130^3 Interactions



```
100%|██████████| 8496973/8496973 [3:07:38<00:00, 754.69it/s]
```

#2 Algorithm After 130^nlogn Interactions



```
100%|██████████| 8632726/8632726 [2:35:22<00:00, 926.01it/s]
```

#2 Algorithm After 130^3.3 Interactions

In [ ]: