תרגיל 2

מועד הגשה: 31.12.2022. הגשה במודל. עבודה עצמאית בלבד.

יש להתמקד בתשובות עם הסברים, טבלאות וגרפים ברורים וקריאים (כותרות, צירים והסברים). את הקוד יש לצרף כנספח בסוף העבודה.

נא להגיש כקובץ PDF יחיד.

א. מצאו דאטה שאתם חושדים שהוא מתפלג לפי חוק חזקה. יש להסביר מדוע בחרתם את מה שבחרתם.

ב. שרטטו היסטוגרמה, את ה CCDF האמפירית והתאימו קו ישר לזנב.

ג. חשבו את α ואת $x_{\min}$.

ד. חשבו את ה goodness-of-fit והסבירו האם התוצאות שקיבלתם תומכות בהנחה שהנתונים הם חוק חזקה. אם לא, מדוע?

תרגיל בית בנומריות

סעיף א:

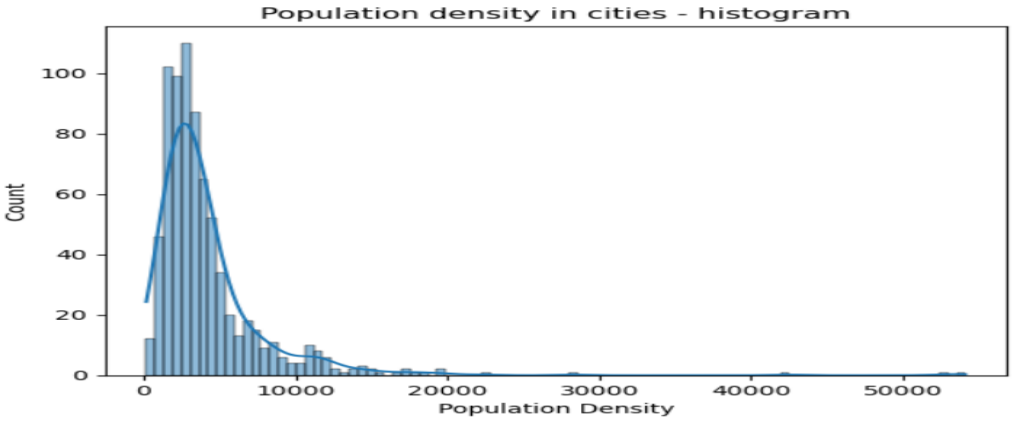
בחרתי dataset המתאר צפיפות אוכלוסיה לפי 754 ערים. הקובץ כלל: שם מדינה. שם עיר, גודל העיר(Square Miles), כמות האוכלוסיה בעיר **וצפיפות האוכלוסיה בעיר**(כמות לחלק לשטח).

המשתנה צפיפות האוכלוסיה לפי עיר הוא המשתנה שחשדתי שמתפלג לפי חוק חזקה.

זאת הייתה אחת הדוגמאות שהובאו בהרצאה ובויקיפדיה לחוק חזקה, כיוון שבאופן הגיוני יש ערים שצפיפות האוכלוסיה בהן גבוהה או בינונית (רוב הערים) ויש מעט ערים שצפיפות האוכלוסיה בהן נמוכה.

סעיף ב:

היסטוגרמה: שרטטתי הסטוגרמה בעזרת histplot של seaborn.



CDF:

חישבתי אותו לצורך החישוב של הCCDF.
את ה CDF חישבתי כמו שמחשבים שכיחות יחסית מצטברת ל 1 (התפלגות מצטברת חלקי גודל המדגם)עבור משתנה:
הפונקציה histplot מחזירה שני מערכים. המערך הראשון (קראתי לו hist) מחזיר את הצפיפות לכל מחלקה (גובה המלבן בהיסטוגרמה) והמערך השני(קראתי לו confines) מחזיר את גבולות המחלקה (ניתן להגדיר לכמה מחלקות רוצים לחלק, חילקתי ל100).
הCDF הוא השכיחות היחסית המצטברת בכל טווח חלקי גודל המדגם- אצלי 754.

CCDF:

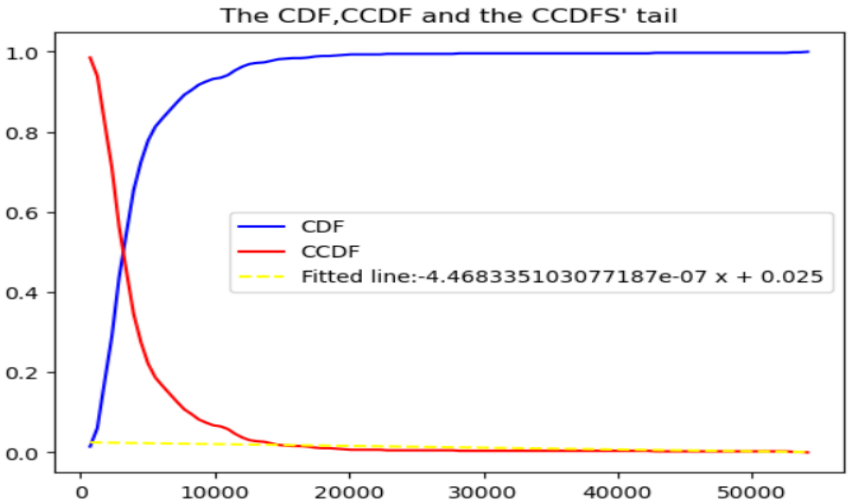
ה CCDF הוא פונקציה משלימה ל CDF ולכן היא 1 פחות CDF.

התאמת קו לזנב:

בחרתי להשתמש בפונקציה polyfit שמקבלת נקודות ומתאימה להם קו(וגם אפשר לבחור את רמת הפולינום). הגדרתי את הנקודות להיות מה שנראה לי הזנב באופן הבא:
כמו שאפשר לראות בגרף -
ציר הX הוא הטווחים של המחלקות וציר הY הוא הCDF/CCDF(הם אותם ערכים רק בסדר עולה ויורד בהתאמה). הכנסתי את הנקודות שאני חושבת שמהוות זנב (פחות או יותר) לתוך משתנים X ו Y כך ש X הוא עשרת האיברים האחרונים במערך שמייצג את טווחי המחלקות, כלומר תשעה מחלקות סך הכל ו Y הוא ערכי ה CCDF (שחושבו קודם) באותן המחלקות. הכנסתי את הנקודות ל polyfit והגדרתי את רמת הפולינום להיות 1 (כדי שיהיה קו ישר). הפונקציה מחזירה חותך ושיפוע והתקבל הישר הבא:

```
]: print(slope, "x +", intercept)
-4.468335103077187e-07 x + 0.02522756672138167
```

הגרף:



סעיף ג:

אלפא ו Xmin:

השתמשתי בשיטה למציאת הפרמטרים שראינו בהרצאה Clauset 7 :
הגדרתי את אלפא בעזרת הנוסחה שמתקבלת לאחר max likelihood.

$$\alpha(X_{min}) = 1 + \frac{1}{\langle \ln x_i \rangle - \ln x_{min}}$$

$$\frac{1}{N}\sum \ln x_i$$

לאחר מכן יישמתי את קולמוגורנוב-סמירנוב.
KS מחזיר את השגיאה המקסימלית בין ההתפלגות של הדאטה לבין פונקציית התפלגות מבוקשת(מודל). השגיאה המקסימלית הזו תלויה בפרמטרים המרכיבים את ההתפלגות(אם מדובר על התפלגות נורמלית אז תוחלת ושונות ואם מדובר בחוק חזקה אז Xmin ואלפא) ולכן, על מנת לשחזר את הפרמטר, מצאתי את הפרמטר שמביא את השגיאה הזו למינימום.

היישום בפיתון:
הגדרתי שתי פונקציות, אחת נקראת KS והשניה find Xmin.

הפונקציה **KS** מקבלת Xmin ודאטה ומחזירה מקסימום שגיאה, כאשר Xmin לא ידוע, היא מחזירה למעשה שגיאה מקסימלית כפונקציה של Xmin.
KS ממיינת את הדאטה ומגדירה את אלפא להיות הנוסחה שהוצגה קודם (תלויה ב Xmin). הגדרתי את N להיות האורך של המערך המכיל רק את התצפיות מתוך הדאטה שהתקבל הגדולות מXmin, מאחר שקולמוגורנוב סמירנוב מחשב שגיאה רק עבור תצפיות שגדולות מ Xmin.
הפונקציה מחשבת את השגיאה עבור כל התצפיות האלו ומכניסה אותה למערך ומחזירה את השגיאה המקסימלית.
השגיאה מחושבת להיות ההפרש בין ה CDF האימפירי (i/N) ל CDF האמיתי (פונקצית התפלגות של חוק חזקה) שתלויה באלפא וXmin.

הפונקציה **find Xmin** מוצאת מינימום לפונקציה המתקבלת מ KS. השתמשתי ב optimize.minimize פונקציה לאופטימיזציה של ספריית Scipy. הפרמטרים העיקריים שהפונקציה מקבלת הם: פונקציה שלה היא מוצאת קיצון וניחוש התחלתי.

כאמור הפונקציה שיש למצוא לה מינימום היא הפונקציה שחוזרת מ KS . את הניחוש ההתחלתי הגדרתי להיות התצפית המינימלית בדאטה. באחד הנסיונות הגדרתי אותו להיות דווקא אמצע הטווח של המלבן הגבוה ביותר בהסטוגרמה כיוון שכפי שניתן לראות, הוא לא השמאלי ביותר אך זה לא שינה את התוצאה.

כך קיבלתי את Xmin והצבתי אותו בנוסחה לאלפא וקיבלתי את הפרמטרים המבוקשים:

```
print("the estimated pharameters: Xmin-",xmin,"alpha-",a)
```

```
the estimated pharameters: Xmin- 1266.9999603271472 alpha- 1.9680658090044765
```

לאחר ששלחתי את ה Xmin שהתקבל לפונקציה KS, קיבלתי את ה DN ששווה ל-0.14523 וישמש אותי בסעיף הבא.

סעיף ד:

כדי לחשב goodness of fit הגרלתי 2,000 מערכים של תצפיות חדשות(המחשב לא עמד ביותר). כל מערך בגודל של הדאטה המקורי והתצפיות בו הן משתנים שהוגרלו מהתפלגות חוק חזקה עם הפרמטרים שמצאתי בסעיף הקודם.

עבור כל מערך כזה חישבתי KS ושמרתי אותם במערך חדש. לאחר מכן בדקתי כמה פעמים ה KS היה קטן יותר מה DN.

קיבלתי שכל ה KSים קטנים יותר מה DN ולכן ההנחה שהנחתי שהדאטה מתפלג חוק חזקה לא נכונה. ההנחה נדחית מאחר שהגרלתי דאטה שבוודאות מתפלג חוק חזקה ועבורו קיבלתי תמיד שגיאה קטנה יותר מה DN, לכן אפשר להסיק שהDN שמצאתי לא קטן מספיק כדי לקבוע שהנתונים באמת מתפלגים לפי חוק חזקה עם הפרמטרים שמצאתי.

In [1]:

```
import numpy as np
import pandas as pd
import math
import random
import scipy as sc
import seaborn as sns
from matplotlib import pyplot as plt
from scipy import integrate
from scipy.special import logsumexp
```

A:

In [2]:

```
data=pd.read_csv("uscitypopdensity.csv")
data.head(5)
```

Out[2]:

	Index	City	State	Population Density (Persons/Square Mile)	2016 Population	Land Area (Square Miles)
0	1	New York	New York	28211	8537673	303
1	2	Los Angeles	California	8484	3976322	469
2	3	Chicago	Illinois	11883	2704958	228
3	4	Houston	Texas	3842	2303482	600
4	5	Phoenix	Arizona	3126	1615017	517

In [3]:

```
data=data.rename(columns={"Population Density (Persons/Square Mile)":"Population Density"})
```

In [4]:

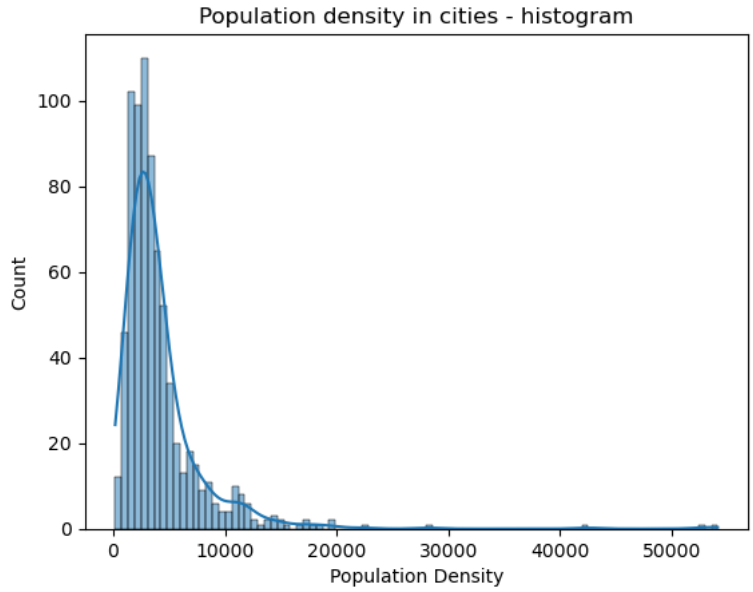
```
data['Population Density'].describe()
```

Out[4]:

```
count      754.000000
mean      4242.729443
std       4323.792554
min        172.000000
25%       2076.000000
50%       3128.500000
75%       4720.000000
max      54138.000000
Name: Population Density, dtype: float64
```

In [5]:

```
plt.title("Population density in cities - histogram")
sns.histplot(data=data,x="Population Density",kde=True)
plt.show()
```



In [6]:

```
data = data['Population Density'].values
```

B:

calculate the cdf and the ccdf:

In [7]:

```
hist,confines = np.histogram(data, bins=100)
cdf=np.cumsum(hist)
cdf=cdf/cdf[-1]
ccdf=1-cdf
```

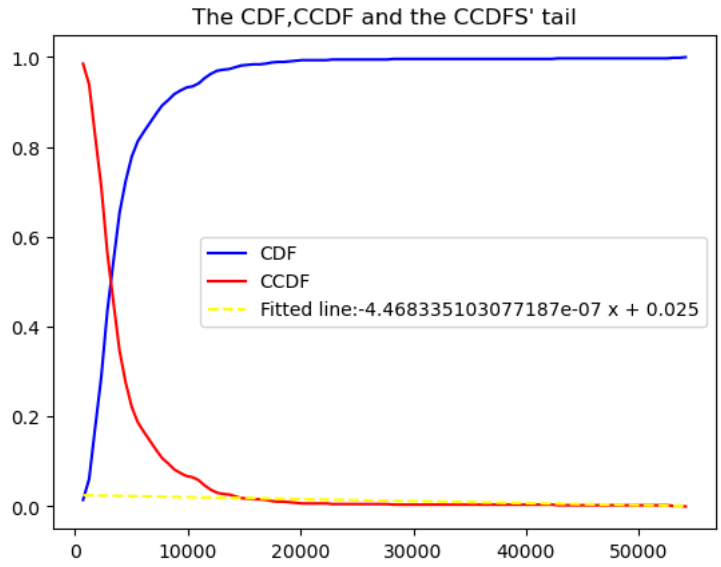
In [8]:

```
# Fit a straight line to the tail of the CCDF
x = confines[1:]
y = ccdf
coefficients = np.polyfit(x[-10:], y[-10:], 1)
slope, intercept = coefficients

# Plot the CDF and CCDF
plt.title("The CDF,CCDF and the CCDFS' tail")
plt.plot(confines[1:],cdf,color='blue', label='CDF')
plt.plot(confines[1:],ccdf,color='red', label='CCDF')

# Plot the fitted line on top of the CCDF plot
fit_y = slope * x + intercept
plt.plot(confines[1:], fit_y, '--',color='yellow', label='Fitted line:-4.468335103077187e-07 x + 0.025')

plt.legend()
plt.show()
```



In [9]:

```
print(slope, "x +", intercept)
```

```
-4.468335103077187e-07 x + 0.02522756672138167
```

C:

The KS function get xmin and data and return the maximum error

In [11]:

```
def ks(Xmin,data):
    data.sort()
    a=1+1/(np.average(np.log(data[data>=Xmin]))-np.log(Xmin))
    ks_residuals=[]
    N=len(data[data>=Xmin])
    for i in range(len(data)):
        if data[i]>=Xmin:
            ks_res=np.abs((i/N)-(1-(data[i]/Xmin)**(1-a)))
            ks_residuals.append(ks_res)
    return(max(ks_residuals))
```

Thise function get data and return the XMin by minimize the ks(the max error)

In [12]:

```
def findXmin(d):
    xmin=sc.optimize.minimize(ks,x0=172,method='Nelder-Mead',args=(d))
    Xmin=xmin.x[0]
    return(Xmin)
```

In [13]:

```
xmin=findXmin(data)
xmin
```

Out[13]:

```
1266.9999603271472
```

In [14]:

```
a=1+1/(np.average(np.log(data[data>=xmin]))-np.log(xmin))
print("the estimated pharameters: Xmin-",xmin,"alpha-",a)
```

```
the estimated pharameters: Xmin- 1266.9999603271472 alpha- 1.9680658090044765
```

D:

In [15]:

```
Dn=ks(1266.1,data)
print ("The Dn",Dn)
```

```
The Dn 0.1452337271515275
```

In [16]:

```
ks_test=[]
count_bigger=0
count_smaller=0

for i in range(2000):
    new=[]
    for i in range(0,753):
        u=random.uniform(0, 1)
        Xi=xmin*(1-u)**(1/1-a)
        new.append(Xi)
    new=np.array(new)
    #new.sort()
    ks_newsample=ks(xmin,new)
    ks_test.append(ks_newsample)
    if(ks_newsample<=Dn):
        count_smaller=count_smaller+1
    else:
        count_bigger=count_bigger+1

print("The DN:",Dn)
print("The percentage of times the KS is smaller than the DN:", (count_smaller/2000)*100,'%')
print("The percentage of times the KS is bigger than the DN:", (count_bigger/2000)*100,'%')
```

```
The DN: 0.1452337271515275
The percentage of times the KS is smaller than the DN: 100.0 %
The percentage of times the KS is bigger than the DN: 0.0 %
```

In [ ]: