Emre Çakmak Progress Journal

Table of contents

Introduction

This progress journal covers Emre Çakmak's work during their term at BDA 503 Fall 2022. Each section is an assignment or an individual work.

1 BDA-503 Assigment 1

A short brief of author and R use cases

Emre Çakmak
2022-10-09

Hi dear reader,

I'm Emre Çakmak from Istanbul/Turkey. I graduated from my bachelor at Istanbul Technical University, Industrial Engineering Department in 2018.

My current role is Data Scientist at E-commerce Department in LC Waikiki which is a Istanbul based global fashion retailer driving operations on more than 50 countries. I had different positions like Data Analyst, Business Intelligence Specialist in different companies during past 4 years. Especially in last 1 year, I dedicated to improve myself for application of ML Technics due to enrich customer&item based data. So, I'm a part of BDA Graduate Program in MEF University to wide my knowledge in audience management and marketing applications by the help of real-life use cases.

Here is my LinkedIn Profile

1.1 RStudio Global 2022 Conference - Quarto for the Curious

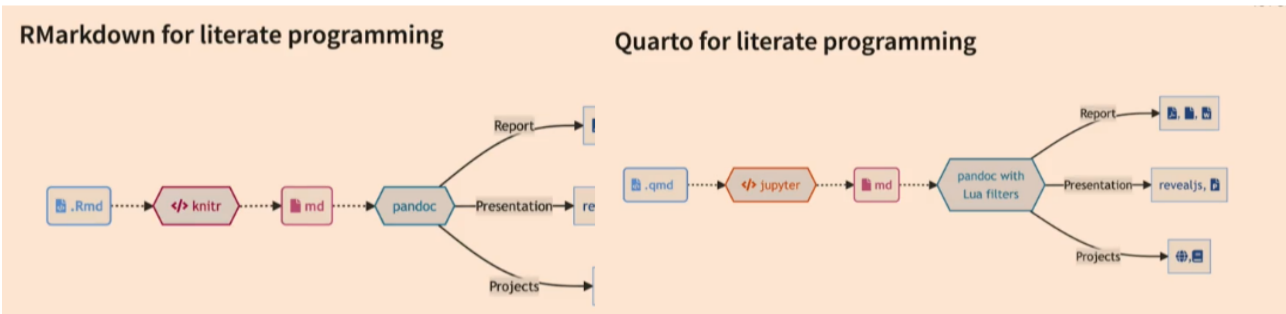
What's *Quarto* according to Tom Mock

In this paragraph, I aim to give you some main differences between *Quarto*, the brand new documentation system which has been released April 2022, and *RMarkdown* being used for almost a decade.

- Tom Mock says *Quarto* is Open source scientific and technical publishing system. Also he added that *Quarto* is the next generation of *RMarkdown*.

Here is some differences between them:

1.1.1 Preprocessing



(a) RMarkdown

(b) Quarto

Figure 1.1: RMarkdown vs Quarto Preprocessing Diagram

Altough it seems like they have almost same workflow behind the scenes; *Quarto* doesn't need to have R in the system to use it. It means that you can use *Quarto* in a fresh computer but *Rmarkdown* needs to have R in the system.

1.1.2 Language Support

The main purpose of releasing *Quarto* is improving the communication between data science communities whatever their language is. Because of this *Quarto* supports other languages as engine.

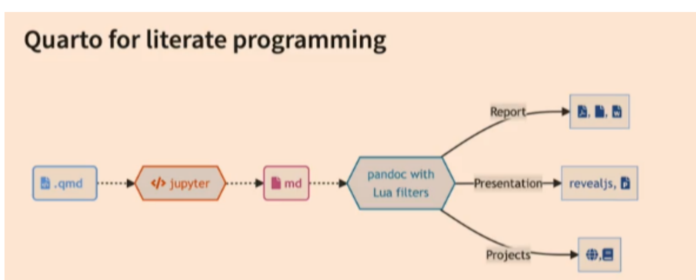


Figure 1.2: Jupyter as Quarto Engine

This availability in *Quarto* and not limiting with R allows people to collaborate as Python developer with others. Tom Mock figured this situation out like

- **Quarto: Comfortable baking in your own kitchen**
- **RMarkdown: Uncomfortable baking in corporate kitchen**

1.2 R Posts

This section includes 3 different R Programming use case

1.2.1 Web Scraping with R

It's very known fact that people have some struggle to access to a clean dataset. In these cases, we need to be a little bit creative to create our own dataset. And one way of the creating a new dataset is web scraping.

In this paragraph, I want to introduce how to scrape a web page by the help of R packages. The most common 2 packages are:

- {rvest}
- {RSelenium}

Note that: Some websites have strict policies against scraping. Be careful!

Step by step scraping of public IMDB Dataset

Step 1: Install Package

```
## Before you start, you need to execute once the code below.
##install.packages("rvest")
```

Step 2: Call the library and use html functions

```
## call the rvest library for required functions
library(rvest)

## define the website link you want to scrape
link = "https://www.imdb.com/search/title/?title_type=feature&num_votes=30000,&genres=come

## send a http get request to the link above and store it in a variable
page = read_html(link)

## filter and grab all elements in same class
titles = page %>% html_nodes(".lister-item-header a") %>% html_text()
```

```r
## preview the titles
titles[1:10]
```

```
 [1] "Bullet Train"                      "Hocus Pocus"
 [3] "Hocus Pocus 2"                     "Everything Everywhere All at Once"
 [5] "Thor: Love and Thunder"            "Beetle Juice"
 [7] "The Rocky Horror Picture Show"     "The Lost City"
 [9] "The Goonies"                       "Trick 'r Treat"
```

Step 3: Create other variables

```r
## apply same procedure to other variables
year= page %>% html_nodes(".text-muted.unbold") %>% html_text()
rating = page %>% html_nodes(".ratings-imdb-rating strong") %>% html_text()


## preview variables
year[1:10]
```

```
 [1] "(2022)" "(1993)" "(2022)" "(2022)" "(2022)" "(1988)" "(1975)" "(2022)"
 [9] "(1985)" "(2007)"
```

```r
rating[1:10]
```

```
 [1] "7.3" "6.9" "6.0" "8.1" "6.4" "7.5" "7.4" "6.1" "7.7" "6.7"
```

Step 4: Create data frame

```r
## create a dataset
movies = data.frame(titles, year, rating, stringsAsFactors = FALSE)
movies[1:10,]
```

```
                              titles   year rating
1                       Bullet Train (2022)    7.3
2                        Hocus Pocus (1993)    6.9
3                      Hocus Pocus 2 (2022)    6.0
4  Everything Everywhere All at Once (2022)    8.1
5             Thor: Love and Thunder (2022)    6.4
```

```
6                        Beetle Juice (1988)    7.5
7      The Rocky Horror Picture Show (1975)    7.4
8                       The Lost City (2022)    6.1
9                         The Goonies (1985)    7.7
10                      Trick 'r Treat (2007)    6.7
```

References of web scraping with R:

- Scraperapi
- Scrapingbee
- Appsilon

1.2.2 Simple Aggregations on Dataset

This part provides some basic aggregations and data manipulation methods in R via {dplyr} package.

Without leaving the concept in previous part, we can assume that we created our own dataset. So, what's next?

The process of extracting insightful information from datasets starts from understanding the data structure and manipulating them. R provides a package just for this: **{dplyr}**

Step by step aggregation & filtering & summarizing dataset

Step 1: Install Package

```r
## Before you start, you need to execute once the code below.
##install.packages("dplyr")
```

Step 2: Call the library

```r
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag
```

```
The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

Step 3: Select subset of data in different aspects

```r
## selecting specific columns
select(movies, titles, year)[1:10,]
```

```
                              titles   year
1                       Bullet Train (2022)
2                         Hocus Pocus (1993)
3                       Hocus Pocus 2 (2022)
4   Everything Everywhere All at Once (2022)
5              Thor: Love and Thunder (2022)
6                         Beetle Juice (1988)
7         The Rocky Horror Picture Show (1975)
8                       The Lost City (2022)
9                         The Goonies (1985)
10                      Trick 'r Treat (2007)
```

```r
## filter data according to specific condition
filter(movies, rating > 8)
```

```
                             titles       year rating
1 Everything Everywhere All at Once     (2022)    8.1
2             The Wolf of Wall Street     (2013)    8.2
3                  Back to the Future     (1985)    8.5
4                  Young Frankenstein     (1974)    8.0
5                            Coco (I) (2017)    8.4
```

```r
## sort rows
arrange(movies, desc(titles))[1:10,]
```

```
                                   titles       year rating
1                       Young Frankenstein     (1974)    8.0
2                                 Tusk (I) (2014)    5.3
3                           Trick 'r Treat     (2007)    6.7
4                   Thor: Love and Thunder     (2022)    6.4
```

```
5                    The Wolf of Wall Street     (2013)    8.2
6  The Unbearable Weight of Massive Talent     (2022)    7.0
7                          The Suicide Squad     (2021)    7.2
8              The Rocky Horror Picture Show     (1975)    7.4
9                              The Lost City     (2022)    6.1
10                             The Lost Boys     (1987)    7.2
```

```r
## select top n rows
top_n(movies, 3, titles)
```

```
             titles       year rating
1      Trick 'r Treat     (2007)    6.7
2 Young Frankenstein     (1974)    8.0
3               Tusk (I) (2014)    5.3
```

Step 4: Summarize Dataset

```r
## convert rating columns as numeric and calculate the average
summarise(movies, average_rating = mean(as.numeric(rating)))
```

```
  average_rating
1          6.976
```

```r
## group by and summarize
grouped_data = group_by(movies, year)
summarise(grouped_data, average_rating = mean(as.numeric(rating)))[1:5,]
```

```
# A tibble: 5 x 2
  year   average_rating
  <chr>           <dbl>
1 (1974)            8
2 (1975)            7.4
3 (1984)            7.8
4 (1985)            8.1
5 (1986)            7.1
```

Step 5: %>% Operator

This operator takes the object from the left and gives it as the first argument to the function on the right. It makes your code more readable.

```r
## same grouping and summarizing operation at step4

movies %>%
  group_by(year) %>%
  summarise(average_rating = mean(as.numeric(rating)))%>%
  top_n(5, desc(average_rating))
```

```
# A tibble: 5 x 2
  year         average_rating
  <chr>                 <dbl>
1 (2002)                  5.2
2 (2009)                  5.4
3 (2020)                  5.2
4 (I) (2014)              5.3
5 (I) (2022)              5.1
```

Reference of aggregations with R:

- courses.cs.ut.ee

1.2.3 Visualization with R

Step 1: Install Package

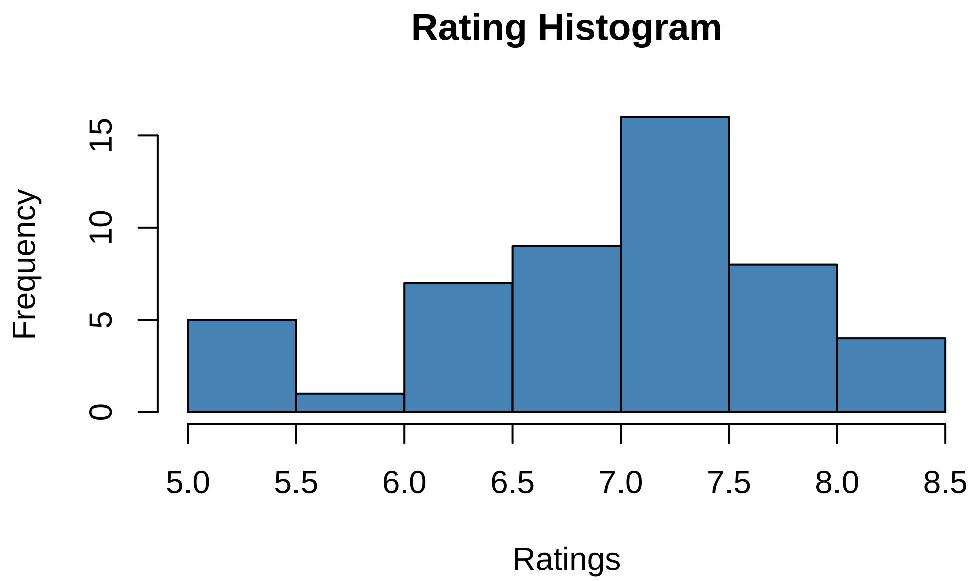
```r
## Before you start, you need to execute once the code below.
##install.packages("ggplot2")
```

Step 2: Call the library

```r
library(ggplot2)
```

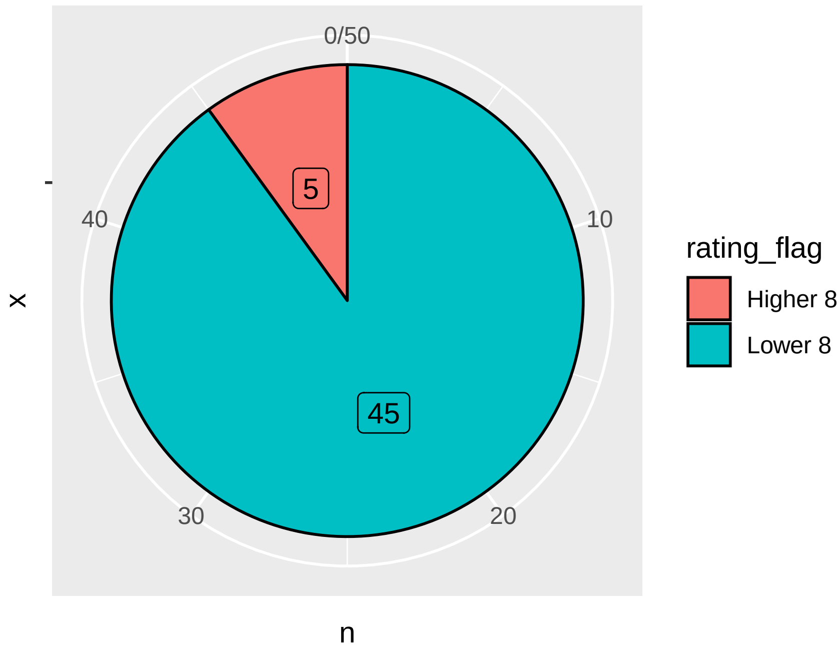
Step 3: Histogram with ggplot2

```r
## convert rating field as numeric and keep in original dataset
movies$rating=as.numeric(rating)

## histogram for ratings
hist(movies$rating,col='steelblue',main='Rating Histogram',
     xlab='Ratings')
```



Step 4: Pie Chart with ggplot2

```
## Set a new flag in dataset
movies=movies %>% mutate(rating_flag = case_when(rating>=8~ "Higher 8", TRUE ~ "Lower 8"))

## creating a new table for better visualization
count_movies=movies %>% count(rating_flag)

## pie chart according to rating of movies
ggplot(count_movies, aes(x = "", y = n, fill = rating_flag)) +
  geom_col(color = "black") +
  geom_label(aes(label = n),
             position = position_stack(vjust = 0.5),
             show.legend = FALSE) +
  coord_polar(theta = "y")
```



References of visualization with R:

- r-charts
- kdnuggets

2 Inclass Exercise-1

Emre Çakmak
2022-10-19

This exercise has been prepared for understanding {dplyr} package usage for functional EDA. Main data set in this exercise will be planes data set derived from FAA.

First of all we have to install our packages.

```
install.packages("tidyverse")
install.packages("nycflights13")
```

Then we are calling our libraries.

```
library(tidyverse)
library(nycflights13)
```

Let's check first 10 rows of the data set. Fields and their meanings are:

- **tailnum**: Tail number.
- **year**: Year manufactured.
- **type**: Type of plane.
- **manufacturer**: Manufacturer of the aircraft.
- **model**: Model of the aircraft.
- **engines**: Number of engines
- **seats**: Number of seats
- **speed**: Average cruising speed in mph.
- **engine**: Type of engine.

```
planes %>%
  slice(1:10)
```

```
# A tibble: 10 x 9
   tailnum  year type                     manuf~1 model engines seats speed engine
   <chr>   <int> <chr>                    <chr>   <chr>   <int> <int> <int> <chr>
 1 N10156   2004 Fixed wing multi engi~ EMBRAER EMB-~       2    55    NA Turbo~
 2 N102UW   1998 Fixed wing multi engi~ AIRBUS~ A320~       2   182    NA Turbo~
 3 N103US   1999 Fixed wing multi engi~ AIRBUS~ A320~       2   182    NA Turbo~
 4 N104UW   1999 Fixed wing multi engi~ AIRBUS~ A320~       2   182    NA Turbo~
 5 N10575   2002 Fixed wing multi engi~ EMBRAER EMB-~       2    55    NA Turbo~
 6 N105UW   1999 Fixed wing multi engi~ AIRBUS~ A320~       2   182    NA Turbo~
 7 N107US   1999 Fixed wing multi engi~ AIRBUS~ A320~       2   182    NA Turbo~
 8 N108UW   1999 Fixed wing multi engi~ AIRBUS~ A320~       2   182    NA Turbo~
 9 N109UW   1999 Fixed wing multi engi~ AIRBUS~ A320~       2   182    NA Turbo~
10 N110UW   1999 Fixed wing multi engi~ AIRBUS~ A320~       2   182    NA Turbo~
# ... with abbreviated variable name 1: manufacturer
```

2.1 EXERCISE 1

Now, how many aircraft does exists for each manufacturing company? Let's calculate..

```
planes %>%
  group_by(manufacturer) %>%
  summarise(aircraft_count = n()) %>%
  arrange(desc(aircraft_count)) %>%
  print(n=Inf)
```

```
# A tibble: 35 x 2
   manufacturer                  aircraft_count
   <chr>                                  <int>
 1 BOEING                                  1630
 2 AIRBUS INDUSTRIE                         400
 3 BOMBARDIER INC                           368
 4 AIRBUS                                   336
 5 EMBRAER                                  299
 6 MCDONNELL DOUGLAS                        120
 7 MCDONNELL DOUGLAS AIRCRAFT CO            103
 8 MCDONNELL DOUGLAS CORPORATION             14
 9 CANADAIR                                   9
10 CESSNA                                     9
11 PIPER                                      5
12 AMERICAN AIRCRAFT INC                      2
```

```
13 BEECH                                     2
14 BELL                                      2
15 GULFSTREAM AEROSPACE                      2
16 STEWART MACO                              2
17 AGUSTA SPA                                1
18 AVIAT AIRCRAFT INC                        1
19 AVIONS MARCEL DASSAULT                    1
20 BARKER JACK L                             1
21 CANADAIR LTD                              1
22 CIRRUS DESIGN CORP                        1
23 DEHAVILLAND                               1
24 DOUGLAS                                   1
25 FRIEDEMANN JON                            1
26 HURLEY JAMES LARRY                        1
27 JOHN G HESS                               1
28 KILDALL GARY                              1
29 LAMBERT RICHARD                           1
30 LEARJET INC                               1
31 LEBLANC GLENN T                           1
32 MARZ BARRY                                1
33 PAIR MIKE E                               1
34 ROBINSON HELICOPTER CO                    1
35 SIKORSKY                                  1
```

It seems like there is a conflict in manufacturer names. Some of them represent the same company but in different names like Airbus and Airbus Industrie.

We need to clean and rewrite these names. Then we can apply same process again.

```r
planes =
planes %>%
  mutate(manufacturer = gsub("AIRBUS INDUSTRIE", "AIRBUS", manufacturer), manufacturer=gsu
```

The last version of distribution of air crafts according to their manufacturer is here.

```r
planes %>%
  group_by(manufacturer) %>%
  summarise(aircraft_count = n()) %>%
  arrange(desc(aircraft_count)) %>%
  mutate(aircraft_count_distrubiton=round(aircraft_count/sum(aircraft_count),2)) %>%
  print(n=Inf)
```

```
# A tibble: 32 x 3
```

```
   manufacturer           aircraft_count aircraft_count_distrubiton
   <chr>                           <int>                      <dbl>
 1 BOEING                           1630                       0.49
 2 AIRBUS                            736                       0.22
 3 BOMBARDIER INC                    368                       0.11
 4 EMBRAER                           299                       0.09
 5 MCDONNELL DOUGLAS                 237                       0.07
 6 CANADAIR                            9                       0
 7 CESSNA                              9                       0
 8 PIPER                               5                       0
 9 AMERICAN AIRCRAFT INC               2                       0
10 BEECH                               2                       0
11 BELL                                2                       0
12 GULFSTREAM AEROSPACE                2                       0
13 STEWART MACO                        2                       0
14 AGUSTA SPA                          1                       0
15 AVIAT AIRCRAFT INC                  1                       0
16 AVIONS MARCEL DASSAULT              1                       0
17 BARKER JACK L                       1                       0
18 CANADAIR LTD                        1                       0
19 CIRRUS DESIGN CORP                  1                       0
20 DEHAVILLAND                         1                       0
21 DOUGLAS                             1                       0
22 FRIEDEMANN JON                      1                       0
23 HURLEY JAMES LARRY                  1                       0
24 JOHN G HESS                         1                       0
25 KILDALL GARY                        1                       0
26 LAMBERT RICHARD                     1                       0
27 LEARJET INC                         1                       0
28 LEBLANC GLENN T                     1                       0
29 MARZ BARRY                          1                       0
30 PAIR MIKE E                         1                       0
31 ROBINSON HELICOPTER CO              1                       0
32 SIKORSKY                            1                       0
```

2.2 EXERCISE 2

Let's check the difference on aircraft capacities year by year

First, get only air crafts which have more than 50 seats. Then clear the data by filtering rows which have no information in Year column.

```
planes %>%
  filter(seats>50,!is.na(year))%>%
  group_by(year) %>%
  summarise(seat_avg = round(mean(seats),2)) %>%
  arrange(year) %>%
  print(n=Inf)
```

```
# A tibble: 38 x 2
    year seat_avg
   <int>    <dbl>
 1  1956     102
 2  1965     149
 3  1975     139
 4  1976     139
 5  1977     139
 6  1978     139
 7  1979     139
 8  1980     139
 9  1984     178
10  1985     174.
11  1986     196.
12  1987     181.
13  1988     190.
14  1989     163.
15  1990     179.
16  1991     181.
17  1992     195.
18  1993     198.
19  1994     178.
20  1995     187.
21  1996     170.
22  1997     179.
23  1998     169.
24  1999     167.
25  2000     163.
26  2001     152.
27  2002     132.
28  2003     106.
29  2004     116.
30  2005     117.
31  2006     141.
32  2007     140.
```

```
33  2008      147.
34  2009      194.
35  2010      164.
36  2011      214.
37  2012      207.
38  2013      206.
```

Let's check the biggest air craft in our database with it's tailnumber.

```
planes %>%
  arrange(desc(seats)) %>%
  slice(1) %>%
  select(tailnum, manufacturer, model, year, seats)
```

```
# A tibble: 1 x 5
  tailnum manufacturer model    year seats
  <chr>   <chr>        <chr>   <int> <int>
1 N670US  BOEING       747-451  1990   450
```

Exciting..Here is some information about the biggest airplane's history

THANKS FOR READING

3 Web Scraping and Clustering in Python

Emre Çakmak
2022-10-20

We are importing the required libraries. Especially selenium for scraping, pandas for dataframes and sklearn for clustering processes.

Step by step;

- We are defining the scraping URL
- We need to download chromedriver.exe to connect Google Chrome
- This web page has infinitive scroll. So, we need to set scrolling depth which should be to the bottom.
- Then, we need to catch necessary fields, item name and price.
- In this project, item names includes the coordinates of the NFT and it will give us enough information about the location.
- We will use this coordinates to calculate distance and region of the NFT
- Distance/Price rate is a spectacular field to determine best affordable NFT in terms of it's location.
- Cluster number has been predefined in this example but it depends on user's own decision according to elbow method.
- After clustering, data has been grouped by cluster numbers and calculated their mean and std.
- NFTs have been filtered by their distance/price rate if they are lower then their cluster's "mean-1.5*std"

```
from selenium import webdriver
from bs4 import BeautifulSoup
import pandas as pd
import socket

import io
import shutil
import re
```

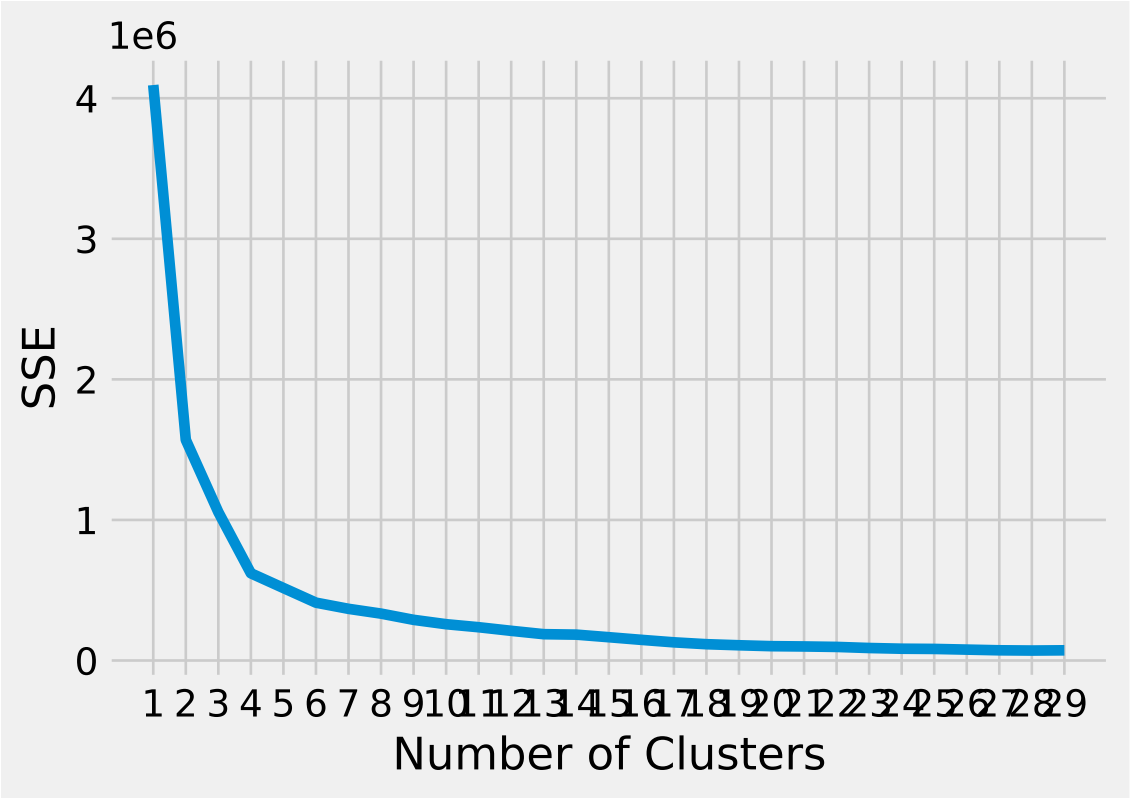
```python
import urllib.request
import os
import errno

import urllib.request
import xlsxwriter
import time

import pandas
import numpy

import matplotlib.pyplot as plt
from kneed import KneeLocator
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler

driver = webdriver.Chrome(r"C:\Users\EMRE\Documents\GitHub\mef06-cakmakem\ScrapingFiles\ch
driver.get("https://www.jpg.store/collection/pavia?minPrice=300000000&maxPrice=1000000000"

SCROLL_PAUSE_TIME = 0.8

# Get scroll height
last_height = driver.execute_script("return document.body.scrollHeight")
a=0
n=5
while a<n:
    # Scroll down to bottom
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")

    # Wait to load page
    time.sleep(SCROLL_PAUSE_TIME)

    # Calculate new scroll height and compare with last scroll height
    new_height = driver.execute_script("return document.body.scrollHeight")
    if new_height == last_height:
        break
    last_height = new_height
    a+=1
```

```
itemset = []
priceset=[]
content = driver.page_source
soup = BeautifulSoup(content)
for a in soup.find_all('div', attrs = {'class', 'styles_itemsGrid__J7c4P grid'}):
    children = a.findChildren("span", recursive=False)
    for child in children:
        sublings = child.find_all('div', attrs = {'class', 'NFTMarketplaceCard_nftMarketpl
        pricesclass = child.find_all('div', attrs = {'class', 'NFTMarketplaceCard_nftMarke
        for items in sublings:
            a=items.findChildren("h4",recursive=False)
            #.find_all('h4', attrs = {'id', 'asset-title'})
            #for i in a.contents:
            itemset.append(a[0])
        for price in pricesclass:
            b=price.findChildren("div", recursive=False)
            for c in b:
                d=c.findChildren("span",recursive=False)
                priceset.append(d[0])

df = pd.DataFrame(itemset, columns=["names"])
df_p = pd.DataFrame(priceset, columns=['price'])

df_final =  pd.concat([df,df_p],axis=1)

## df_final.to_excel('20102022_2_nft.xlsx', encoding='utf-16',engine='xlsxwriter')


df_final[['name','space','X','Y']]=df_final.names.str.split(' ',3,expand=True)
df_final=df_final[['X','Y','price']]
df_final=df_final[(df_final['X']!='-')&(df_final['Y']!='-')&(df_final['X']!='')&(df_final[

df_final['X']=pd.to_numeric(df_final['X'],downcast='integer')
df_final['Y']=pd.to_numeric(df_final['Y'],downcast='integer')

df_final['distance_to_origin']=(abs((df_final['X'])**2 + (df_final['Y']**2)))**0.5

kmeans_kwargs = {
    "init": "random",
    "n_init": 10,
    "max_iter": 300,
```

```
    "random_state": 42,
}

# # A list holds the SSE values for each k
sse = []
for k in range(1, 30):
    kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
    kmeans.fit(df_final[['X','Y','distance_to_origin']])
    sse.append(kmeans.inertia_)

plt.style.use("fivethirtyeight")
plt.plot(range(1, 30), sse)
plt.xticks(range(1, 30))
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.show()


##print('specify number of clusters')
##k=int(input())

k=10
kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
df_final['cluster']=kmeans.fit_predict(df_final[['X','Y','distance_to_origin']])
df_final['cluster'] = df_final['cluster'].astype('category')


results = df_final.groupby('cluster').agg({'price':['mean','std']})
results=results.reset_index()
results.columns=['cluster','mean','std']
df_final=df_final.merge(results,left_on='cluster',right_on='cluster')
df_final['low_limit']=df_final['mean']-1.5*df_final['std']
df_final['d_p_rate'] = df_final['distance_to_origin']/df_final['price'].astype(float)

results_dp = df_final.groupby('cluster').agg({'d_p_rate':['mean','std']})
results_dp=results_dp.reset_index()
results_dp.columns=['cluster','mean_dp','std_dp']
df_final=df_final.merge(results_dp,left_on='cluster',right_on='cluster')

df_final['low_limit_dp']=df_final['mean_dp']-1.5*df_final['std_dp']
```

```python
finalists=df_final[(df_final['price'].astype(float)<df_final['low_limit'])&(df_final['d_p_
finalists[1:10]
```

```
C:\Users\EMRE\AppData\Local\Temp\ipykernel_3344\1984115952.py:27: DeprecationWarning: executa
  driver = webdriver.Chrome(r"C:\Users\EMRE\Documents\GitHub\mef06-cakmakem\ScrapingFiles\chr
```




```
C:\Users\EMRE\AppData\Local\Programs\Python\Python310\lib\site-packages\IPython\core\formatte
  return method()
```

	X	Y	price	distance_to_origin	cluster	mean	std	low_limit	d_p_rate
8	-217	-44	302	128.495136	9	8.300342e+69	51.801443	8.300342e+69	0.425481
10	-197	-33	313	160.118706	9	8.300342e+69	51.801443	8.300342e+69	0.511561
12	-194	15	315	166.358048	9	8.300342e+69	51.801443	8.300342e+69	0.528121
13	-169	35	333	172.586210	9	8.300342e+69	51.801443	8.300342e+69	0.518277
14	-221	40	333	122.861711	9	8.300342e+69	51.801443	8.300342e+69	0.368954
15	-187	-63	333	163.088933	9	8.300342e+69	51.801443	8.300342e+69	0.489757
16	-223	-2	340	125.709984	9	8.300342e+69	51.801443	8.300342e+69	0.369735
17	-205	-10	341	153.006536	9	8.300342e+69	51.801443	8.300342e+69	0.448700
18	-177	-40	342	180.574085	9	8.300342e+69	51.801443	8.300342e+69	0.527994