```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import re
import time
from datetime import datetime
import matplotlib.dates as mdates
import matplotlib.ticker as ticker
from urllib.request import urlopen
from bs4 import BeautifulSoup
import requests
import warnings
warnings.filterwarnings('ignore')
print('Setup Complete!')
```

```
Setup Complete!
```

Web Scraping

```
no_pages = 2

def get_data(pageNo):
    headers = {"User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64; 
rv:66.0) Gecko/20100101 Firefox/66.0", "Accept-Encoding":"gzip, 
deflate", 
"Accept":"text/html,application/xhtml+xml,application/xml;q=0.9,*/
*;q=0.8", "DNT":"1","Connection":"close", "Upgrade-Insecure-
Requests":"1"}

    r = 
requests.get('https://www.amazon.com/gp/bestsellers/books/ref=zg_bs_pg
_'+str(pageNo)+'?ie=UTF8&pg='+str(pageNo), headers=headers)#, 
proxies=proxies)
    content = r.content
    soup = BeautifulSoup(content)
    #print(soup)

    alls = []
    for d in soup.findAll('div', attrs={'class':'a-section a-spacing-
none aok-relative'}):
        #print(d)
        name = d.find('span', attrs={'class':'zg-text-center-align'})
        n = name.find_all('img', alt=True)
        #print(n[0]['alt'])
        author = d.find('a', attrs={'class':'a-size-small a-link-
child'})
```

```
        rating = d.find('span', attrs={'class':'a-icon-alt'})
        users_rated = d.find('a', attrs={'class':'a-size-small a-link-
normal'})
        price = d.find('span', attrs={'class':'p13n-sc-price'})

        all1=[]

        if name is not None:
            #print(n[0]['alt'])
            all1.append(n[0]['alt'])
        else:
            all1.append("unknown-product")

        if author is not None:
            #print(author.text)
            all1.append(author.text)
        elif author is None:
            author = d.find('span', attrs={'class':'a-size-small a-
color-base'})
            if author is not None:
                all1.append(author.text)
            else:
                all1.append('0')

        if rating is not None:
            #print(rating.text)
            all1.append(rating.text)
        else:
            all1.append('-1')

        if users_rated is not None:
            #print(price.text)
            all1.append(users_rated.text)
        else:
            all1.append('0')

        if price is not None:
            #print(price.text)
            all1.append(price.text)
        else:
            all1.append('0')
        alls.append(all1)
    return alls

results = []
for i in range(1, no_pages+1):
    results.append(get_data(i))
flatten = lambda l: [item for sublist in l for item in sublist]
df = pd.DataFrame(flatten(results),columns=['Book
```

```
Name','Author','Rating','Customers_Rated', 'Price'])
df.to_csv('amazon_products.csv', index=False, encoding='utf-8')

df = pd.read_csv("amazon_products.csv")

df.shape

(0, 5)

df.head()

Empty DataFrame
Columns: [Book Name, Author, Rating, Customers_Rated, Price]
Index: []

df.tail()

Empty DataFrame
Columns: [Book Name, Author, Rating, Customers_Rated, Price]
Index: []

df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 0 entries
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Book Name        0 non-null      object
 1   Author           0 non-null      object
 2   Rating           0 non-null      object
 3   Customers_Rated  0 non-null      object
 4   Price            0 non-null      object
dtypes: object(5)
memory usage: 0.0+ bytes

df['Rating'] = df['Rating'].apply(lambda x: x.split()[0])
df['Rating'] = pd.to_numeric(df['Rating'])
df["Price"] = df["Price"].str.replace('$', '')
df["Price"] = df["Price"].str.replace(',', '')
df['Price'] = df['Price'].apply(lambda x: x.split('.')[0])
df['Price'] = df['Price'].astype(int)
df["Customers_Rated"] = df["Customers_Rated"].str.replace(',', '')
df['Customers_Rated'] = pd.to_numeric(df['Customers_Rated'],
errors='ignore')
df.head()

Empty DataFrame
Columns: [Book Name, Author, Rating, Customers_Rated, Price]
Index: []

df.dtypes
```

```
Book Name           object
Author              object
Rating               int64
Customers_Rated      int64
Price                int32
dtype: object
```

```
## Replace the zero values in the DataFrame to NaN.
df.replace(str(0), np.nan, inplace=True)
df.replace(0, np.nan, inplace=True)

## Counting the Number of NaNs in the DataFrame
count_nan = len(df) - df.count()
count_nan
```

```
Book Name           0
Author              0
Rating              0
Customers_Rated     0
Price               0
dtype: int64
```

```
## Let's drop these NaNs.
df = df.dropna()

data = df.sort_values(["Price"], axis=0, ascending=False)[:15]
data
```

```
Empty DataFrame
Columns: [Book Name, Author, Rating, Customers_Rated, Price]
Index: []
```

```
from bokeh.models import ColumnDataSource
from bokeh.transform import dodge
import math
from bokeh.io import curdoc
curdoc().clear()
from bokeh.io import push_notebook, show, output_notebook
from bokeh.layouts import row
from bokeh.plotting import figure
from bokeh.transform import factor_cmap
from bokeh.models import Legend
output_notebook()
```

```
"(function(root) {\n  function now() {\n    return new Date();\n  }\n\
n  const force = true;\n\n  if (typeof root._bokeh_onload_callbacks
=== \"undefined\" || force === true) {\n
root._bokeh_onload_callbacks = [];\n    root._bokeh_is_loading =
undefined;\n  }\n\n\n  if (typeof (root._bokeh_timeout) ===
\"undefined\" || force === true) {\n    root._bokeh_timeout =
Date.now() + 5000;\n    root._bokeh_failed_load = false;\n  }\n\n
```

```
const NB_LOAD_WARNING = {'data': {'text/html':\n     \"<div
style='background-color: #fdd'>\\n\"+\n     \"<p>\\n\"+\n
\"BokehJS does not appear to have successfully loaded. If loading
BokehJS from CDN, this \\n\"+\n     \"may be due to a slow or bad
network connection. Possible fixes:\\n\"+\n     \"</p>\\n\"+\n
\"<ul>\\n\"+\n     \"<li>re-rerun `output_notebook()` to attempt to
load from CDN again, or</li>\\n\"+\n     \"<li>use INLINE resources
instead, as so:</li>\\n\"+\n     \"</ul>\\n\"+\n     \"<code>\\n\"+\n
\"from bokeh.resources import INLINE\\n\"+\n
\"output_notebook(resources=INLINE)\\n\"+\n     \"</code>\\n\"+\n
\"</div>\"}};\n\n  function display_loaded() {\n    const el =
document.getElementById(\"1002\");\n    if (el != null) {\n
el.textContent = \"BokehJS is loading...\";\n    }\n    if (root.Bokeh
!== undefined) {\n      if (el != null) {\n        el.textContent =
\"BokehJS \" + root.Bokeh.version + \" successfully loaded.\";\
n      }\n    } else if (Date.now() < root._bokeh_timeout) {\n
setTimeout(display_loaded, 100)\n    }\n  }\n\n  function
run_callbacks() {\n    try {\n
root._bokeh_onload_callbacks.forEach(function(callback) {\n        if
(callback != null)\n          callback();\n      });\n    } finally {\
n      delete root._bokeh_onload_callbacks\n    }\n
console.debug(\"Bokeh: all callbacks have finished\");\n  }\n\n
function load_libs(css_urls, js_urls, callback) {\n    if (css_urls ==
null) css_urls = [];\n    if (js_urls == null) js_urls = [];\n\n
root._bokeh_onload_callbacks.push(callback);\n    if
(root._bokeh_is_loading > 0) {\n      console.debug(\"Bokeh: BokehJS
is being loaded, scheduling callback at\", now());\n      return
null;\n    }\n    if (js_urls == null || js_urls.length === 0) {\n
run_callbacks();\n      return null;\n    }\n
console.debug(\"Bokeh: BokehJS not loaded, scheduling load and
callback at\", now());\n    root._bokeh_is_loading = css_urls.length +
js_urls.length;\n\n    function on_load() {\n
root._bokeh_is_loading--;\n      if (root._bokeh_is_loading === 0) {\n
console.debug(\"Bokeh: all BokehJS libraries/stylesheets loaded\");\n
run_callbacks()\n      }\n    }\n\n    function on_error(url) {\n
console.error(\"failed to load \" + url);\n    }\n\n    for (let i =
0; i < css_urls.length; i++) {\n      const url = css_urls[i];\n
const element = document.createElement(\"link\");\n
element.onload = on_load;\n      element.onerror = on_error.bind(null,
url);\n      element.rel = \"stylesheet\";\n      element.type =
\"text/css\";\n      element.href = url;\n      console.debug(\"Bokeh:
injecting link tag for BokehJS stylesheet: \", url);\n
document.body.appendChild(element);\n    }\n\n    for (let i = 0; i <
js_urls.length; i++) {\n      const url = js_urls[i];\n      const
element = document.createElement('script');\n      element.onload =
on_load;\n      element.onerror = on_error.bind(null, url);\n
element.async = false;\n      element.src = url;\n
console.debug(\"Bokeh: injecting script tag for BokehJS library: \",
url);\n      document.head.appendChild(element);\n    }\n  };\n\n
```

```
function inject_raw_css(css) {\n    const element =
document.createElement(\"style\");\n
element.appendChild(document.createTextNode(css));\n
document.body.appendChild(element);\n  }\n\n  const js_urls =
[\"https://cdn.bokeh.org/bokeh/release/bokeh-2.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-gl-2.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-widgets-2.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-tables-2.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-mathjax-2.4.3.min.js\"];\n
const css_urls = [];\n\n  const inline_js = [    function(Bokeh) {\n
Bokeh.set_log_level(\"info\");\n    },\nfunction(Bokeh) {\n    }\
n  ];\n\n  function run_inline_js() {\n    if (root.Bokeh !==
undefined || force === true) {\n          for (let i = 0; i <
inline_js.length; i++) {\n      inline_js[i].call(root, root.Bokeh);\n
}\nif (force === true) {\n        display_loaded();\n      }} else if
(Date.now() < root._bokeh_timeout) {\n      setTimeout(run_inline_js,
100);\n    } else if (!root._bokeh_failed_load) {\n
console.log(\"Bokeh: BokehJS failed to load within specified
timeout.\");\n      root._bokeh_failed_load = true;\n    } else if
(force !== true) {\n      const cell = $
(document.getElementById(\"1002\")).parents('.cell').data().cell;\n
cell.output_area.append_execute_result(NB_LOAD_WARNING)\n    }\n  }\n\
n  if (root._bokeh_is_loading === 0) {\n    console.debug(\"Bokeh:
BokehJS loaded, going straight to plotting\");\n    run_inline_js();\n
} else {\n    load_libs(css_urls, js_urls, function() {\n
console.debug(\"Bokeh: BokehJS plotting callback run at\", now());\n
run_inline_js();\n    });\n  }\n}(window));"
```

```
p = figure(x_range=data.iloc[:,1], plot_width=800, plot_height=550,
title="Authors Highest Priced Book", toolbar_location=None, tools="")

p.vbar(x=data.iloc[:,1], top=data.iloc[:,4], width=0.9)

p.xgrid.grid_line_color = None
p.y_range.start = 0
p.xaxis.major_label_orientation = math.pi/2

show(p)

""

data = df[df['Customers_Rated'] > 1000]
data = data.sort_values(['Rating'],axis=0, ascending=False)[:15]
data
```

```
Empty DataFrame
Columns: [Book Name, Author, Rating, Customers_Rated, Price]
Index: []
```

```
p = figure(x_range=data.iloc[:,0], plot_width=800, plot_height=600,
title="Top Rated Books with more than 1000 Customers Rating",
```

```
toolbar_location=None, tools="")

p.vbar(x=data.iloc[:,0], top=data.iloc[:,2], width=0.9)

p.xgrid.grid_line_color = None
p.y_range.start = 0
p.xaxis.major_label_orientation = math.pi/2
show(p)

""

p = figure(x_range=data.iloc[:,1], plot_width=800, plot_height=600,
title="Top Rated Books with more than 1000 Customers Rating",
toolbar_location=None, tools="")

p.vbar(x=data.iloc[:,1], top=data.iloc[:,2], width=0.9)

p.xgrid.grid_line_color = None
p.y_range.start = 0
p.xaxis.major_label_orientation = math.pi/2
show(p)

""

data = df.sort_values(["Customers_Rated"], axis=0, ascending=False)
[:20]

data
```

```
Empty DataFrame
Columns: [Book Name, Author, Rating, Customers_Rated, Price]
Index: []
```

```
from bokeh.transform import factor_cmap
from bokeh.models import Legend
from bokeh.palettes import Dark2_5 as palette
import itertools
from bokeh.palettes import d3
#colors has a list of colors which can be used in plots
colors = itertools.cycle(palette)

palette = d3['Category20'][20]

index_cmap = factor_cmap('Author', palette=palette,
                         factors=data["Author"])

p = figure(plot_width=700, plot_height=700, title = "Top Authors:
Rating vs. Customers Rated")
p.scatter('Rating','Customers_Rated',source=data,fill_alpha=0.6,
fill_color=index_cmap,size=20,legend='Author')
p.xaxis.axis_label = 'RATING'
```

```
p.yaxis.axis_label = 'CUSTOMERS RATED'
p.legend.location = 'top_left'

BokehDeprecationWarning: 'legend' keyword is deprecated, use explicit
'legend_label', 'legend_field', or 'legend_group' keywords instead

show(p)

""
```