Detailed Explanation of All 3 Files

File 1: data_extract.py

This script is like a little robot that goes to Amazon, looks at laptops, and writes down everything it sees in small notes (HTML files). It uses a browser that the robot can control.

Step-by-Step Explanation:

1. Import Libraries:

```
from selenium import webdriver

from selenium.webdriver.common.by import By

import time
```

- **Imagine: We are giving the robot tools to drive a car (browser), look for objects on the road (web elements), and take breaks (time).**

2. Configure the Browser:

```
from selenium.webdriver.chrome.options import Options

options = Options()

options.add_argument("--disable-blink-features=AutomationControlled")

options.add_argument("user-agent=Mozilla/5.0...")
```

- **Imagine: We are dressing the robot in a disguise so no one knows it's a robot. This makes it look like a real person using the browser.**

3. Initialize WebDriver:

```
driver = webdriver.Chrome(options=options)

file = 0
```

- **Imagine: We are turning on the robot's car (browser) and setting a counter to name its notes starting from 0.**

4. Loop Through Pages:

```
for i in range(1, 4):

  driver.get("https://www.amazon.com/s?k=laptop...page={i}")
```

- **Imagine: The robot is driving through three streets (pages) on Amazon to find laptops. It stops at each street and takes notes.**

5. Locate Product Containers:

```
elems = driver.find_elements(By.CLASS_NAME, "puis-card-container")
```

- Imagine: The robot looks for boxes (containers) that hold laptop information.

6. Save Data to Files:

```
with open(f"data/laptop_{file}.html", "w", encoding="utf-8") as f:

    f.write(d)

    file += 1
```

- Imagine: For every box the robot finds, it writes the details into a notebook (HTML file) and names it laptop_0.html, laptop_1.html, and so on.

7. Close Browser:

```
driver.close()
```

- Imagine: Once the robot finishes its job, it parks the car (browser) and goes to sleep.

File 2: collect.py

This script is like a librarian that organizes all the notes (HTML files) the robot wrote. It makes a nice, clean table (CSV file) with rows and columns for each laptop's details.

Step-by-Step Explanation:

1. Import Libraries:

```
import pandas as pd

from bs4 import BeautifulSoup

import os
```

- Imagine: We are giving the librarian tools to read the notes (BeautifulSoup), make a table (pandas), and find files in the cabinet (os).

2. Define get_data Function:

```
def get_data():

    data_dict = {'title': [], 'price': [], 'rating': [], 'disk_size': [], 'ram': [], 'link': []}
```

- Imagine: The librarian starts with an empty table with six columns: title, price, rating, disk_size, ram, and link.

3. Loop Through Files:

```
for file in os.listdir('data'):

   with open(f'data/{file}', 'r') as f:

      data = f.read()
```

- Imagine: The librarian opens each notebook (HTML file) from the robot’s cabinet (folder).

4. Extract Data Using BeautifulSoup:

```
title = soup.find_all('h2')[0].text

price = soup.find_all('span', attrs={'class': 'a-offscreen'})[0].text
```

- Imagine:The librarian reads each notebook and finds:
  - The name of the laptop (title).
  - Its price (price).
  - Other details like rating, disk_size, and ram.

5. Save Data to a DataFrame:

```
data_dict['title'].append(title)

data_dict['price'].append(price)
```

- Imagine: The librarian writes the details into the table, one row at a time.

6. Convert to CSV:

```
data = get_data()

data.to_csv('laptop.csv', index=False)
```

- Imagine: The librarian saves the table into a book (CSV file) called laptop.csv for anyone to read.

File 3: Final Project.py

This script is like a friendly shop assistant. It takes the librarian’s table and helps customers by predicting laptop prices or features. It also shows similar laptops they might like.

Step-by-Step Explanation:

1. Import Libraries:

```
import tkinter as tk
```

```
from tkinter import ttk

from sklearn.ensemble import RandomForestRegressor
```

- Imagine: We are giving the shop assistant a desk (GUI), a brain to make predictions (machine learning), and tools to organize their work.

2. Define LaptopPredictor Class:

Data Cleaning:

```
self.df['RAM'] = self.df['ram'].str.extract(r'(\d+)').astype(float)

self.df['Disk_Size'] = self.df['disk_size'].apply(self.clean_storage_value)
```

- Imagine:The assistant cleans up messy data. For example:
    - Turns 8GB into 8.
    - Turns 1TB into 1024 (because 1TB = 1024GB).

Model Training:

```
model = RandomForestRegressor()

model.fit(X_train, y_train)
```

- Imagine: The assistant learns by looking at past data (e.g., "A laptop with 8GB RAM and 256GB storage costs $500"). Now, it can guess prices for similar laptops.

3. Define PredictorGUI Class:

Setup GUI:

```
input_frame = ttk.LabelFrame(self.root, text="Input Parameters")
```

- Imagine: The assistant sets up a desk with fields where customers can type details like "8GB RAM" or "256GB storage."

Prediction Logic:

```
inputs = {k: float(v.get()) for k, v in self.entries.items()}

prediction = self.predictor.models[target].predict([inputs])
```

- Imagine: The customer types in some details, and the assistant uses its brain (model) to guess the missing information, like price.

4. Suggest Similar Laptops:

```
similar_laptop = df_temp.nsmallest(1, 'Similarity').iloc[0]
```

- **Imagine: If the customer asks, "What other laptops are like this one?" the assistant looks in its book (CSV) and finds the closest match.**

How to Use These Scripts Together:

1. **Run data_extract.py:**
   - **The robot goes to Amazon, looks at laptops, and writes notes (HTML files).**
2. **Run collect.py:**
   - **The librarian organizes these notes into a clean table (CSV file).**
3. **Run Final Project.py:**
   - **The shop assistant uses the table to predict laptop features and suggest similar options to customers.**

Let me know if you need even simpler explanations or more examples!