Project Report: Best Moment to Buy a Dell Laptop – Prediction Model

1. Project Definition

1.1 Problem Statement

For many people, especially students and professionals, a high-performance laptop is a significant but necessary investment. Consumers try to acquire these devices at discounted prices in periods like Amazon Prime Day or Black Friday. However, the prices for these devices, particularly from major brands like Dell, are not static; they fluctuate for various reasons, such as: Seasonal sales, promotional events, or the release of new models. The issue is that even in dates like Black Friday, if the price of the device is higher in comparison to the usual trend. Then, the discount obtainable in such day will be negligible in comparison to discounts obtainable any other day. The core problem this project tackles is the challenge consumers face in navigating this price volatility. Without access to historical data and analytical tools, it's difficult to know the right time to buy, often leading to missed opportunities for substantial savings. My project was born out of a simple question: Can we use data to find the best time to buy a Dell laptop? I aimed to create a tool that could analyze price trends and forecast future sale periods, empowering consumers to make more informed and economically sound decisions.

1.2 Connection to Course Material

My project is a practical application of the entire data science pipeline as discussed in the lectures. I began by collecting raw data, transformed it into a structured format, stored it in a relational database, and then used that data to build a predictive model. In more detail:

- **Data Storage and SQL:** I chose to use a SQLite database for its simplicity and integration with Python. Although I used the Pandas library for convenience, it runs standard SQL commands in the background.
- **Data Transformation and Preprocessing:** This was a critical part of the project, especially after I pivoted it to using simulated data. I performed Feature Engineering by creating a new week column from the date data, which was essential for the analysis of it. I also directly addressed Handling Missing Values since the simulation intentionally created an imperfect data set by removing 7% of the data points. I then used Linear Interpolation to clean the dataset and prepare it for analysis.

2. Novelty and Importance

2.1 Importance of the Project

This project is important because it addresses a practical, real-world problem faced by most consumers. In fact, the problem about buying items in dates with discount is not particular to only tech, it is an issue shared by all kind of merchandise like apparel or even the food market.

However, tech devices are more prone to fluctuate in price in comparison to other kinds of merchandise. Therefore, a tool that can successfully predict price drops can result in savings of hundreds of dollars for a single purchase, making technology more accessible. Current data management practices for consumers in this area are inefficient, relying on manual price checking or setting up simple price-drop alerts on multiple retail websites. This approach fails to provide a consolidated historical view or any predictive insight, leaving the consumer to guess whether a current "deal" is truly the best possible price. This project directly tackles this issue by providing the necessary data and intelligence to empower the consumer.

3. Changes from Project Proposal & Implementation Details

A significant change was made to the project's methodology compared to the initial proposal. This change was a direct result of the technical challenges I encountered during the data acquisition phase.

Web Scraping Challenges: My original plan was to scrape two years of historical price data from third-party websites like Pangoly.com. However, I quickly ran into a lot of problems when trying to scrape the data related to the price history of the models. My attempts consistently met with:

1. **Anti-Scraping Measures:** The target websites have robust security that can detect and block automated scripts, leading to 403 Forbidden errors and SSL Handshake Failed errors.
2. **Dynamic Content Loading:** Price history charts on these sites are not loaded with the initial HTML. They are rendered dynamically using JavaScript. This meant that simpler scraping tools were ineffective, as the data I needed wasn't there when the script accessed the page.

Realistic Simulation: Given these challenges, I made the decision to pivot from trying to scrape historical data to creating a realistic, rule-based data simulation. This allowed me to build a robust and functional tool while still modeling the real-world price behaviors I intended to study.

The final implementation therefore follows this refined, three-step process:

Step 1: Live Scraping of Current Dell Models The application successfully uses selenium to scrape the official Dell website to get a list of the all the most recent laptop models and their current prices.

Step 2: Simulating Two Years of Price History (2023-2024) This is the core of the new methodology. For each laptop scraped in Step 1, a two-year price history will be simulated. Due to the large amount of data scraped. I decided to just pre-load 5 of the laptops and load the others when the user needs them.

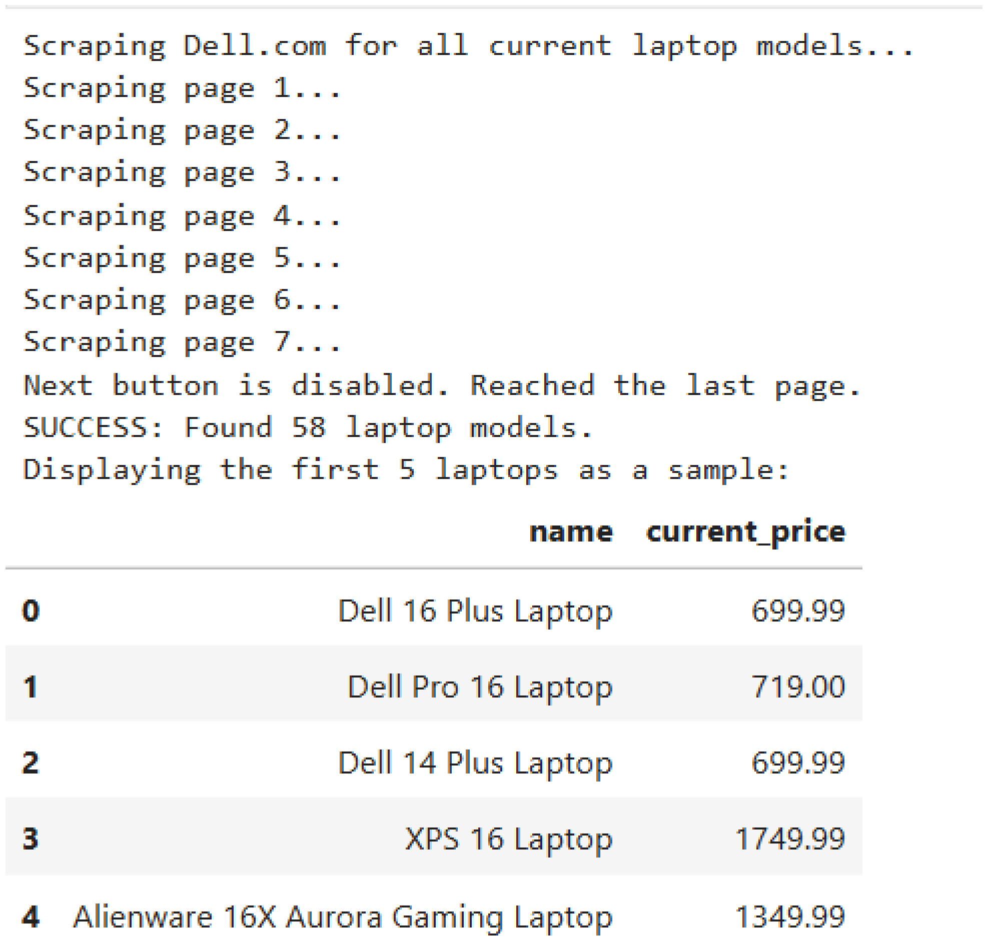
Step 3: Analysis and Prediction The simulated two-year dataset is analyzed to identify the top 5 weeks with the largest price drops.

4. Data and Techniques

4.1 Data Acquisition:

For the first step of the project (scraping the list of current laptops) I chose to use selenium and webdriver-manager libraries. While simpler tools like requests can fetch a website's HTML, they often fail on modern e-commerce sites like Dell.com. This is because much of the product information is loaded dynamically with JavaScript after the initial page loads. requests only see the initial, often incomplete, HTML.

Selenium solves this problem by allowing my Python script to control an actual web browser. It can load a page, wait for all the JavaScript to run and the content to fully appear, and *then* extract the data. This mimics human browsing behavior, making it far more effective for scraping complex, dynamic websites.



```
Scraping Dell.com for all current laptop models...
Scraping page 1...
Scraping page 2...
Scraping page 3...
Scraping page 4...
Scraping page 5...
Scraping page 6...
Scraping page 7...
Next button is disabled. Reached the last page.
SUCCESS: Found 58 laptop models.
Displaying the first 5 laptops as a sample:
```

	name	current_price
0	Dell 16 Plus Laptop	699.99
1	Dell Pro 16 Laptop	719.00
2	Dell 14 Plus Laptop	699.99
3	XPS 16 Laptop	1749.99
4	Alienware 16X Aurora Gaming Laptop	1349.99

4.2 Data Simulation:

To create the historical dataset, I developed a set certain price boundaries taking as base the price scraped from Dell.com. Then, I used several libraries to introduce real life like data into my dataset.

Libraries Used:

- **random and numpy:** These libraries were essential for introducing realistic randomness into the price simulation. I used random.random() to decide *if* a price would change on a given day and numpy.random.uniform() to determine the *magnitude* of that change. This ensures the day-to-day price fluctuations are not predictable, mimicking real market noise.
- **datetime and timedelta:** These were used to systematically generate a time series of 730 days for the years 2023 and 2024, providing the chronological backbone for the dataset.
- **pandas:** The pandas library was the cornerstone of data processing and cleaning. Its DataFrame structure is ideal for handling time-series data. I specifically used it for its interpolate method for data cleaning.
- **sqlite3:** This library was used to store the final, cleaned dataset in a persistent, file-based database, allowing for easy access and querying.

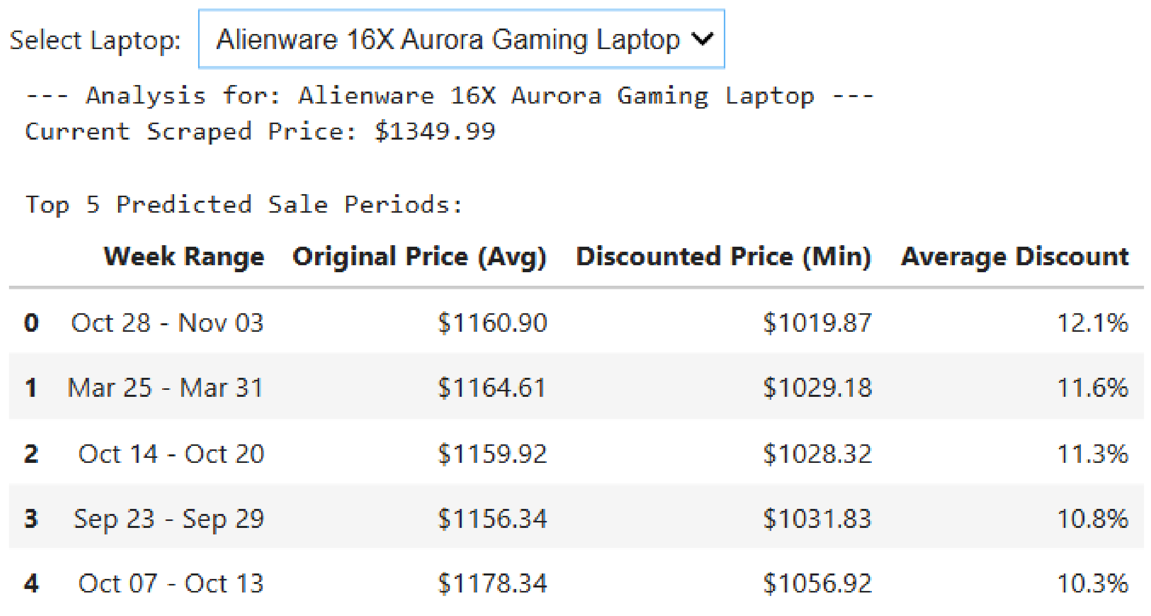
Data Science & Mathematical Methods:

1. **Data Storage:** The data is stored in a **SQLite database**. This is a lightweight, serverless database engine that is perfect for self-contained projects, as it doesn't require a separate server process and stores the entire database in a single file.
2. **Data Processing (Simulation):** The simulation is a rule-based mathematical model designed to mimic real-world price behavior:
    - **Baseline Price:** The simulation starts with the real, scraped price from Dell.com. For the first year (2023), this price is reduced by 10% to simulate the lower cost of an older model.
    - **Seasonal Sales:** A form of event-base modelig. I added major sale events such as: Black Friday, Prime Day, new releases, etc. These events are modeled as fixed-percentage discounts applied during specific weeks of the year.
    - **Price Boundaries:** The simulation enforces a floor price (85% of the baseline) and a price ceiling (130%).
    - **New Year Price Shock:** A special rule allows the price to increase by up to 40% on Dec 31/Jan 1, simulating the introduction of a new model year. The price on Jan 1 then becomes the new baseline for the next year's simulation.
3. **Data Cleaning:**
    - **Dirtying Dataset:** To simulate an imperfect dataset, 7% of the data points are randomly set to None.
    - **Linear Interpolation:** The pandas' interpolate function is used to fill these missing values. This is a mathematical method that estimates a missing value by connecting the known data points on either side of the gap with a straight line. It's

an effective technique for time-series data where values are expected to change smoothly over time.

5. Results and Outputs

The final application provides a clean, user-friendly interface to explore the data and view the predictions.

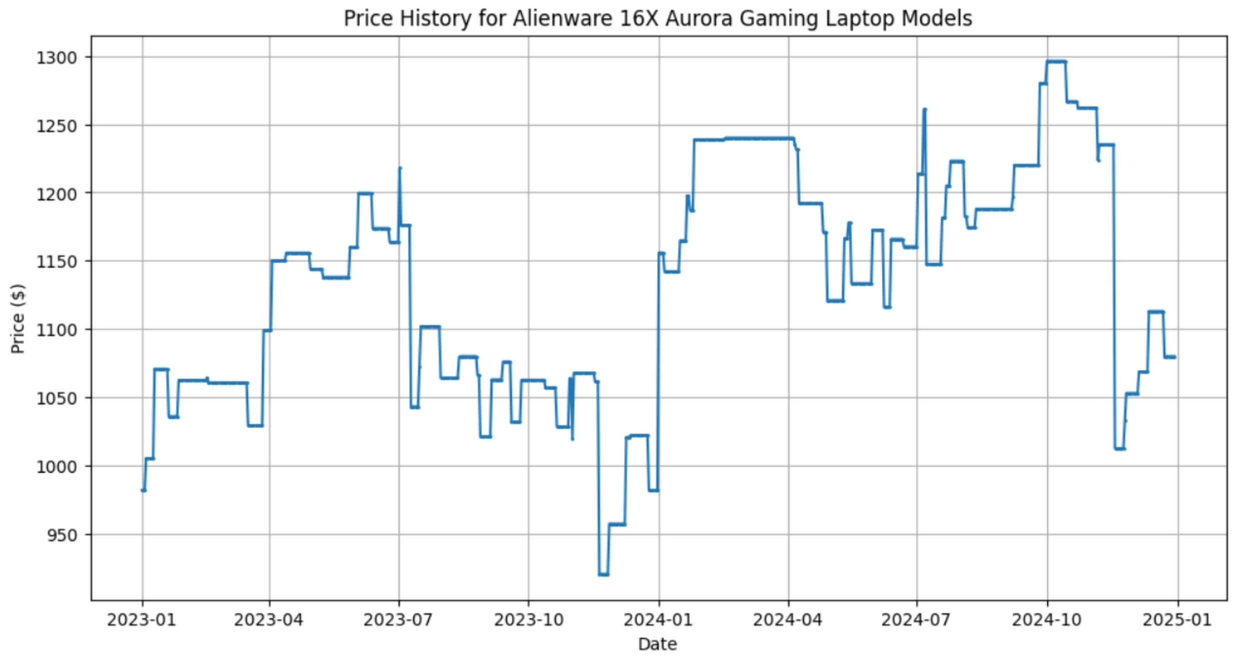


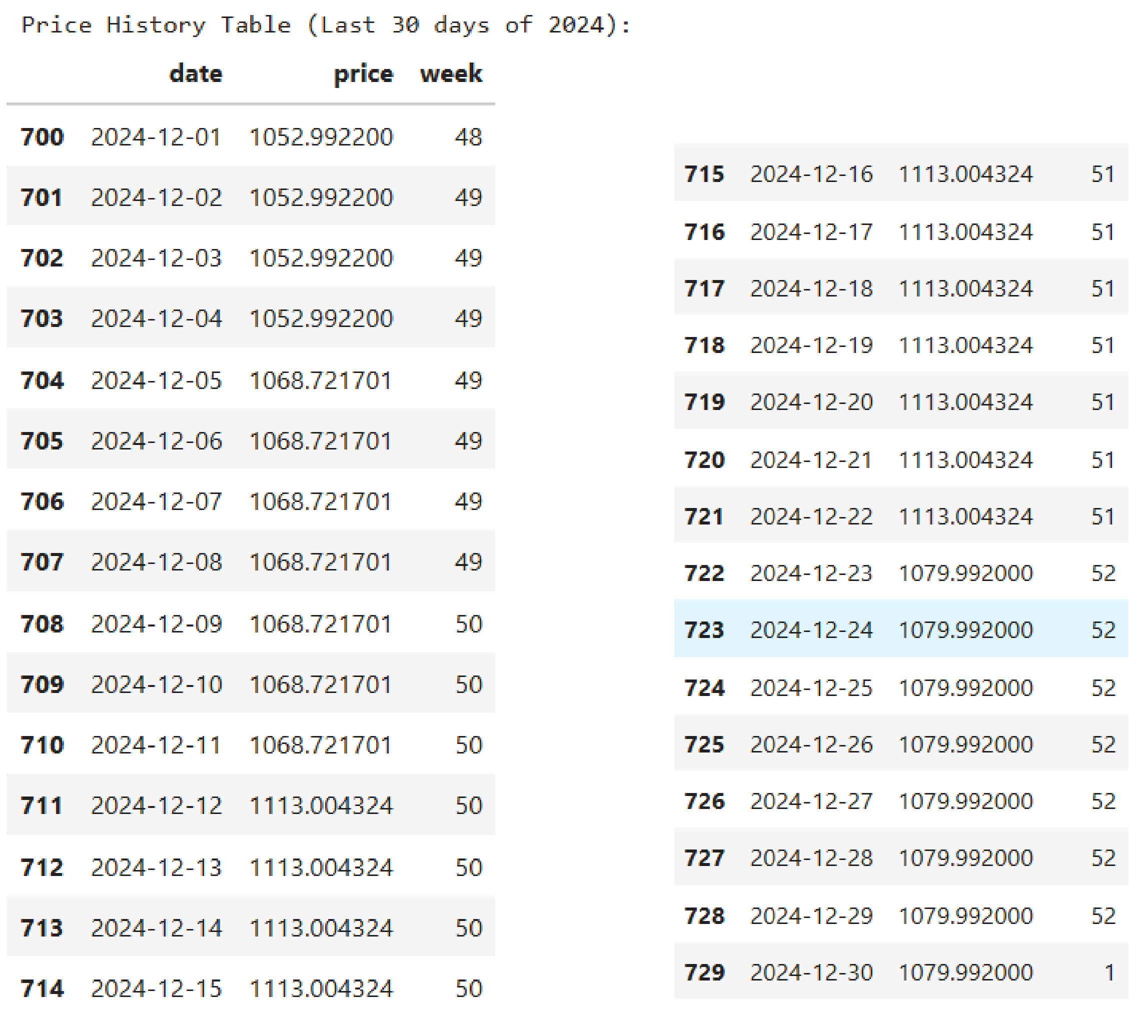
```
--- Analysis for: Alienware 16X Aurora Gaming Laptop ---
Current Scraped Price: $1349.99

Top 5 Predicted Sale Periods:
```

	Week Range	Original Price (Avg)	Discounted Price (Min)	Average Discount
0	Oct 28 - Nov 03	$1160.90	$1019.87	12.1%
1	Mar 25 - Mar 31	$1164.61	$1029.18	11.6%
2	Oct 14 - Oct 20	$1159.92	$1028.32	11.3%
3	Sep 23 - Sep 29	$1156.34	$1031.83	10.8%
4	Oct 07 - Oct 13	$1178.34	$1056.92	10.3%

```
Price History Graph (2023-2024):
```





```
Price History Table (Last 30 days of 2024):
```

	date	price	week
700	2024-12-01	1052.992200	48
701	2024-12-02	1052.992200	49
702	2024-12-03	1052.992200	49
703	2024-12-04	1052.992200	49
704	2024-12-05	1068.721701	49
705	2024-12-06	1068.721701	49
706	2024-12-07	1068.721701	49
707	2024-12-08	1068.721701	49
708	2024-12-09	1068.721701	50
709	2024-12-10	1068.721701	50
710	2024-12-11	1068.721701	50
711	2024-12-12	1113.004324	50
712	2024-12-13	1113.004324	50
713	2024-12-14	1113.004324	50
714	2024-12-15	1113.004324	50
715	2024-12-16	1113.004324	51
716	2024-12-17	1113.004324	51
717	2024-12-18	1113.004324	51
718	2024-12-19	1113.004324	51
719	2024-12-20	1113.004324	51
720	2024-12-21	1113.004324	51
721	2024-12-22	1113.004324	51
722	2024-12-23	1079.992000	52
723	2024-12-24	1079.992000	52
724	2024-12-25	1079.992000	52
725	2024-12-26	1079.992000	52
726	2024-12-27	1079.992000	52
727	2024-12-28	1079.992000	52
728	2024-12-29	1079.992000	52
729	2024-12-30	1079.992000	1

6. Conclusion and Future Work:

Even though the project successfully demonstrates a complete data science pipeline, from data acquisition and simulation to analysis and prediction. I am still a little bit dissatisfied due to the difficulty of scraping websites like Pangoly or CamelCamelCamel. I am still planning to modify the program in order to support web scraping from real life price histories of different models of laptops. In addition, this is an approach that could work with any kind of merchandise. We just need the price history, or develop one from scratch, scraping daily data from the website to analyze.