Decision-making and Analysis for Intelligent Systems

Yasmine DALY

A report submitted as part of the requirements for our
Decision-making and Analysis for Intelligent Systems Project
M2 SIA
at Gustave Eiffel Univeristy

March 2024

Supervisor Mr. Mostafa AMELI

Abstract

In the pursuit of identifying the laptop offering the best value for money tailored to individual preferences and needs, this report presents a comprehensive analysis employing the Analytic Hierarchy Process (AHP) and the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS).

The study begins with a meticulous selection and preparation of a dataset encompassing a diverse range of laptop specifications and prices. Utilizing AHP, we establish a prioritization of criteria through a pairwise comparison matrix, ensuring consistency in preference judgments. The study further advances with the application of TOPSIS, a method that discerns the optimal choice by evaluating the geometric distance of each laptop from an ideal solution. This dual-method approach facilitates a robust cross-examination of rankings, allowing us to pinpoint the laptop that stands out in providing the highest return on investment when juxtaposed against specific user-defined criteria.

Our findings offer a nuanced perspective on decision-making in consumer electronics by illustrating the sensitivity of choice to varying priority scales and by highlighting the congruencies and divergences between two respected multi-criteria decision-making techniques. The report culminates in delivering strategic insights that guide consumers in navigating the complex market landscape to arrive at a purchase decision that aligns with their unique constellation of requirements and financial constraints.

Acknowledgements

I would like to extend my profound gratitude to Professor Mostafa Ameli, whose expertise and insights into decision-making and analysis for intelligent systems have been indispensable to the completion of this report. Professor Ameli's guidance through the intricacies of multi-criteria decision-making methodologies has not only enriched my analytical skills but has also provided a solid foundation for applying these complex techniques to real-world scenarios. His dedication to fostering a deep understanding of Analytic Hierarchy Process (AHP) and Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) within his students has been particularly instrumental in navigating the nuanced challenges presented in this analysis. I am truly appreciative of his unwavering support, constructive feedback, and encouragement throughout this academic endeavor.

Contents

List of Figures

Chapter 1

Introduction

The quest for the ideal laptop—one that marries performance with value—is a journey that most technology consumers will embark on. The essence of this expedition is not just to find a machine that fits within a budget, but one that also aligns perfectly with the user's specific demands for functionality and performance. This report is crafted to illuminate the path to such a discovery by employing rigorous multi-criteria decision-making tools that sift through the complexities of modern-day laptop features and pricing.

Our study leverages a rich dataset from Kaggle, a renowned platform for data science and machine learning, which offers an extensive array of laptop options currently available in the market. The dataset (Laptop Price Dataset) includes detailed specifications and pricing for a wide range of laptops, making it an invaluable resource for our analysis.

The granularity of the dataset provides us with an in-depth look at various factors that are vital in determining the best value for a laptop. The data points include:

- **Company**: The brand manufacturer, which can be indicative of the overall quality and reliability.
- **Product**: The specific brand and model, which can affect user preference based on reputation and performance history.
- **TypeName**: The category of laptop such as Notebook, Ultrabook, or Gaming, each tailored to different user needs.
- **Inches**: The screen size, affecting portability and the user's visual experience.
- **ScreenResolution**: The screen resolution, impacting the clarity and detail of

the display, an important aspect for graphic designers and gamers.

- **Cpu**: Details on the central processing unit, the heart of the laptop, dictating processing power and efficiency.
- **Ram**: The memory size, crucial for multitasking and speed.
- **Memory**: Storage capacity and type, affecting the speed of data retrieval and amount of data that can be stored.
- **GPU**: The graphics processing unit, especially important for gaming, video editing, and other graphic-intensive tasks.
- **OpSys**: The operating system, defining the interface and compatibility of software applications.
- **Weight**: The weight of the laptop, a consideration for users who travel frequently.
- **Price_euros**: The price in euros, providing a direct measure of cost against features and performance.

This data serves as the foundation upon which the Analytic Hierarchy Process (AHP) and the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) are applied. AHP assists in breaking down the decision into a hierarchy of criteria, assigning weights to each based on their importance to the user. TOPSIS complements this by evaluating how each laptop option measures up against an idealized version, based on the weighted criteria. Together, these tools form a robust framework for assessing which laptop stands as the paragon of value for money against the backdrop of the consumer's specific needs and preferences delineated within our dataset. The ensuing pages detail the methodologies, analyses, and insights gleaned from this endeavor, culminating in a concise guide for the prospective laptop buyer.

Chapter 2

Data Engineering

2.1 Introduction

The efficacy of multi-criteria decision-making methods such as AHP and TOPSIS is contingent upon the quality and preparation of the data upon which they are applied. Recognizing that decision-making is as robust as the data it rests on, meticulous data engineering steps are crucial to ensure that the dataset not only reflects accurate information but also aligns with the quantitative demands of the analysis tools. In this context, our exploration is predicated on a dataset that demands conversion and standardization of laptop specifications into quantifiable measures that can be processed by the chosen methodologies.

Given the scope of this report to focus on seven quantitative criteria and a representative sample size of 150 laptops, a series of data transformations were employed to curate the dataset to meet these specifications. The following subsections detail the computational procedures executed to translate raw data into a structured form amenable for subsequent analysis.

2.2 Data Transformation and Criteria Selection

The raw dataset comprised varied formats and units that required normalization for quantitative analysis. The data engineering processes performed included:

1. Converting memory storage specifications to a consistent unit of gigabytes. The code block responsible for this transformation is showcased below.

2. Calculating total pixel counts from screen resolution data. The code block responsible for this transformation is showcased below.

```python
def convert_memory_to_gb(memory_str):
    size_part = memory_str.split()[0]
    if 'TB' in size_part:
        return float(size_part.replace('TB', '')) * 1000
    else:
        return float(size_part.replace('GB', ''))
```

Figure 2.1: Data transformation for memory.

```python
def calculate_pixels(resolution_str):
    numbers = re.findall(r'\d+', resolution_str)

    if len(numbers) >= 2:
        width, height = int(numbers[0]), int(numbers[1])
        return width * height
    else:
        return None
```

Figure 2.2: Data transformation for screen pixels.

3. Extracting and standardizing CPU clock speeds. The code block responsible for this transformation is showcased below.

```python
def extract_clock_speed(cpu_str):
    match = re.search(r'(\d+\.?\d*)GHz', cpu_str)
    if match:
        return float(match.group(1))
    else:
        match_mhz = re.search(r'(\d+\.?\d*)MHz', cpu_str)
        if match_mhz:
            return float(match_mhz.group(1)) / 1000
        else:
            return None
```

Figure 2.3: Data transformation for clock speed.

4. Assigning a performance score to different CPU series. The code block responsible for this transformation is showcased below.

```python
def enhanced_cpu_score(cpu_str):
    if 'i3' in cpu_str:
        base_score = 1
    elif 'i5' in cpu_str:
        base_score = 2
    elif 'i7' in cpu_str:
        base_score = 3
    elif 'i9' in cpu_str:
        base_score = 4
    elif 'Intel Core M' in cpu_str:
        base_score = 1.5
    elif 'AMD Ryzen' in cpu_str:
        base_score = 3  # Ryzen series generally offer high performance, but this can vary widely
    elif 'AMD A' in cpu_str:
        base_score = 1  # AMD A-Series are generally considered entry-level
    elif 'Intel Pentium' in cpu_str or 'Intel Celeron' in cpu_str:
        base_score = 0.5  # Generally lower performance
    elif 'Intel Xeon' in cpu_str:
        base_score = 3.5  # Xeon processors are high-end but vary widely
    elif 'AMD FX' in cpu_str:
        base_score = 2.5  # Positioning AMD FX series as mid- to high-range
    else:
        base_score = 0  # Unknown
    return base_score
```

Figure 2.4: Data transformation for CPU score.

5. Normalizing the weight data by removing textual units. This process, along with the subsequent sampling, is reflected in the following code blocks.

After applying these transformations, the resulting dataset, `criteria_df`, encapsulated

```
laptops_8gb = laptops[laptops['Ram'] == '8GB'].copy()
# Convert 'Weight' to numeric by stripping 'kg' and converting to float
laptops_8gb['Weight'] = laptops_8gb['Weight'].str.replace('kg', '').astype(float)
# Apply the conversion function to the 'Memory' column
laptops_8gb['Memory_Size_GB'] = laptops_8gb['Memory'].apply(convert_memory_to_gb)
# Apply the function to the 'ScreenResolution' column
laptops_8gb['Total_Pixels'] = laptops_8gb['ScreenResolution'].apply(calculate_pixels
# Apply the function to the 'Cpu' column
laptops_8gb['Cpu_Speed'] = laptops_8gb['Cpu'].apply(extract_clock_speed)
# Apply the function to the 'Cpu' column
laptops_8gb['Cpu_Series_Score'] = laptops_8gb['Cpu'].apply(enhanced_cpu_score)
```

Figure 2.5: Conversion and Standardization Steps.

```
print(laptops_8gb.shape)
laptops_8gb = laptops_8gb.sample(n=150, random_state=42)
laptops_8gb.reset_index(drop=True, inplace=True)
laptops_8gb
```

Figure 2.6: Selecting a random subset of 150 laptops.

the following criteria, which were deemed most relevant for the analysis:

- CPU Speed (in GHz)
- CPU Series Score
- Total Pixel Count
- Memory Size (in GB)
- Screen Size (in Inches)
- Weight (in kg)
- Price (in Euros)

2.3 Conclusion

The data engineering phase stands as a testament to the rigorous approach taken to ensure data integrity and relevance for the decision-making models applied. By methodically transforming qualitative and heterogeneous data into a standardized quantitative format, we laid a foundation for a reliable analysis. The subsequent steps, detailed in the following chapters, build upon this foundation to distill insights about the best value for money laptops, informed by empirical evidence and analytical rigor. This phase underscores our commitment to precision and appropriateness in the data preparation process, pivotal for the efficacy of any data-driven decision-making endeavor.

Chapter 3

AHP Methodology

3.1 Introduction

The Analytic Hierarchy Process (AHP) is a structured technique for organizing and analyzing complex decisions. It allows us to model a complex problem in a hierarchical structure showing the relationships of the goal, criteria, sub-criteria, and alternatives. Our study applies AHP to a curated dataset to determine the laptop offering the best value for money based on specific quantifiable criteria.

The dataset prepared for AHP analysis, shown in Figure 3.1, consists of seven key quantitative criteria for 150 laptops. These criteria are CPU Speed, CPU Series Score, Total Pixels, Memory Size, Screen Size, Weight, and Price in euros. This structured form is pivotal for carrying out the pairwise comparison required in AHP, where the importance of each criterion relative to each other will be evaluated systematically.

3.2 Step 1: Read Values from CSV & Build Comparison Matrix

The first step in applying the AHP methodology is to construct a pairwise comparison matrix. This matrix is fundamental to the AHP process as it captures the relative importance of each criterion against the others. The matrix was prepared by reading the values from a CSV file, which contains the initial comparison judgments made by experts or decision-makers.

Upon importing the CSV file into a DataFrame, the comparison matrix was constructed. In this matrix, the diagonal elements are set to 1, as every criterion is equally important to itself. The upper triangle of the matrix is filled with values that denote the relative

```
criteria_df = laptops_8gb[['Cpu_Speed','Cpu_Series_Score', 'Total_Pixels', 'Memory_Size_GB', 'Inches', 'Weight',
criteria_df
```

	Cpu_Speed	Cpu_Series_Score	Total_Pixels	Memory_Size_GB	Inches	Weight	Price_euros
0	1.6	2.0	2073600	256.0	13.3	1.50	847.0
1	2.5	2.0	2073600	1000.0	15.6	2.40	819.0
2	2.5	2.0	3110400	256.0	13.0	1.05	1349.0
3	2.3	1.0	1049088	1000.0	15.6	2.29	459.0
4	1.6	2.0	2073600	256.0	14.0	1.60	1149.0
...	...	...	...	...	...	...	...
145	2.8	3.0	2073600	128.0	15.6	2.20	1191.8
146	1.6	2.0	2073600	256.0	17.3	2.50	923.0
147	2.3	2.0	2073600	256.0	14.0	2.02	1340.0
148	2.7	3.0	2073600	512.0	12.5	1.26	1483.0
149	1.6	2.0	2073600	256.0	15.6	2.13	728.0

150 rows × 7 columns

Figure 3.1: Structured dataset prepared for AHP analysis, showcasing the selected seven quantitative criteria.

```
# Number of criteria
n = criteria_df.shape[1]
# Import the CSV file as a DataFrame
comparison_df = pd.read_csv("/kaggle/input/comparison-matrix/comparison_matrix.csv", dtype=float)
comparison_df
```

	Cpu_Speed	Cpu_Series_Score	Total_Pixels	Memory_Size_GB	Inches	Weight	Price_euros
0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0
1	3.0	0.000000	0.000000	0.0	0.0	0.0	0.0
2	5.0	3.000000	0.000000	0.0	0.0	0.0	0.0
3	1.0	0.333333	0.200000	0.0	0.0	0.0	0.0
4	3.0	1.000000	0.333333	3.0	0.0	0.0	0.0
5	7.0	5.000000	3.000000	7.0	5.0	0.0	0.0
6	9.0	7.000000	5.000000	9.0	7.0	3.0	0.0

Figure 3.2: Initial values of the pairwise comparison matrix read from a CSV file.

importance of one criterion over another, as assessed by the decision-maker.

To ensure the consistency of the matrix, the lower triangle was computed as the reciprocals of the upper triangle. This enforces a necessary condition of AHP where if criterion A is x times more important than criterion B, then criterion B must be 1/x as important as criterion A.

```
np.fill_diagonal(comparison_df.values, 1)
# Fill the upper triangle with the reciprocals of the lower triangle
for i in range(n):
    for j in range(i + 1, n):
        comparison_df.iloc[i, j] = 1 / comparison_df.iloc[j, i]
comparison_df
```

	Cpu_Speed	Cpu_Series_Score	Total_Pixels	Memory_Size_GB	Inches	Weight	Price_euros
0	1.0	0.333333	0.200000	1.0	0.333333	0.142857	0.111111
1	3.0	1.000000	0.333333	3.0	1.000000	0.200000	0.142857
2	5.0	3.000000	1.000000	5.0	3.000000	0.333333	0.200000
3	1.0	0.333333	0.200000	1.0	0.333333	0.142857	0.111111
4	3.0	1.000000	0.333333	3.0	1.000000	0.200000	0.142857
5	7.0	5.000000	3.000000	7.0	5.000000	1.000000	0.333333
6	9.0	7.000000	5.000000	9.0	7.000000	3.000000	1.000000

Figure 3.3: Completed pairwise comparison matrix after filling the lower triangle with reciprocals.

3.3 Step 2: Build Normalization Matrix

Once the pairwise comparison matrix is constructed, the next step is to normalize it. Normalization is crucial as it allows the matrix to be processed further to determine the priority of each criterion.

The code snippet in Figure 3.4 illustrates the normalization of the pairwise comparison matrix. By performing this step, we ensure that the relative weights are comparable and can be utilized in the calculation of the priority vector, which will be the next step in the AHP methodology.

```python
# Step 2: Normalize the pairwise comparison matrix
# Calculate the sum of each column
column_sums = comparison_df.sum(axis=0)
normalized_matrix = comparison_df.divide(column_sums, axis='columns')
normalized_matrix
```

	Cpu_Speed	Cpu_Series_Score	Total_Pixels	Memory_Size_GB	Inches	Weight	Price_euros
0	0.034483	0.018868	0.019868	0.034483	0.018868	0.028463	0.054432
1	0.103448	0.056604	0.033113	0.103448	0.056604	0.039848	0.069984
2	0.172414	0.169811	0.099338	0.172414	0.169811	0.066414	0.097978
3	0.034483	0.018868	0.019868	0.034483	0.018868	0.028463	0.054432
4	0.103448	0.056604	0.033113	0.103448	0.056604	0.039848	0.069984
5	0.241379	0.283019	0.298013	0.241379	0.283019	0.199241	0.163297
6	0.310345	0.396226	0.496689	0.310345	0.396226	0.597723	0.489891

Figure 3.4: Normalized pairwise comparison matrix.

3.4 Step 3: Build Weight Matrix

Calculate the weight matrix involves determining the priority vector, which represents the weights of each criterion in the decision-making process. The priority vector is calculated using the eigenvector method, which involves finding the principal eigenvector of the normalized matrix.

The principal eigenvector corresponds to the largest eigenvalue, and it is used to provide an estimate of the relative weights. To ensure consistency in the comparison, the eigenvector is normalized so that the sum of its components equals one.

```python
# Step 3: Calculate the priority vector using the eigenvector method
eigenvalues, eigenvectors = np.linalg.eig(comparison_df)
principal_eigen = eigenvalues.argmax()
priority_vector = np.real(eigenvectors[:, principal_eigen])
priority_vector = priority_vector / np.sum(priority_vector)  # Normalize
```

Figure 3.5: Calculation of the priority vector using the eigenvector method from the normalized comparison matrix.

As shown in Figure 3.5, the priority vector is derived by taking the real part of the eigenvector associated with the largest eigenvalue of the comparison matrix. The resulting vector is then normalized to produce the final priority weights for each criterion.

3.5 Step 4: Calculate AW for all Matrices

The next step, following the determination of the priority vector, is to validate the consistency of the pairwise comparison matrix. This involves calculating the product of each matrix by its corresponding priority vector (AW). This vector AW represents the weighted sum of the comparisons for each criterion.

```
# Step 4: Calculate AW for all matrices
AW_vector = comparison_df.dot(priority_vector)
```

Figure 3.6: Calculation of the AW vector by multiplying the comparison matrix with the priority vector.

As depicted in Figure 3.6, the computation of AW is a crucial step in the consistency check. By multiplying the comparison matrix with the priority vector, we obtain a vector that should be proportional to the priority vector if the comparisons are consistent. The degree of consistency will be evaluated in the subsequent step by calculating the Consistency Index (CI) and the Consistency Ratio (CR).

3.6 Step 5: Calculate $\lambda_{\max}$ and Consistency Index (CI)

An essential step in ensuring the validity of the AHP results is to check for consistency in the pairwise comparisons. This is achieved by calculating the maximum eigenvalue ($\lambda_{\max}$) and subsequently the Consistency Index (CI). The maximum eigenvalue is estimated as the average of the elements in the AW vector divided by the corresponding elements in the priority vector.

```
# Calculate the maximum eigenvalue (λ_max) as the average of the AW vector divided by the priority vector
lambda_max = np.mean(AW_vector / priority_vector)

# Calculate the consistency index (CI)
CI = (lambda_max - n) / (n - 1)
print(f'The consistency index (CI) is: {CI}')
```

```
The consistency index (CI) is: 0.050735973747628726
```

Figure 3.7: Calculation of the maximum eigenvalue ($\lambda_{\max}$) and the Consistency Index (CI).

The CI is calculated using the formula $CI = (\lambda_{\max} - n)/(n - 1)$, where n is the number of criteria. The Consistency Index provides a measure of how much the pairwise comparison matrix deviates from consistency. If the value of CI is small, the level of inconsistency is acceptable, and the judgments can be considered reliable.

As displayed in Figure 3.7, the calculated CI in our analysis confirms that the pairwise comparisons are reasonably consistent, which suggests that the derived priorities are reliable for making decisions.

3.7 Step 6: Extract RI from External Table

To further evaluate the consistency of the pairwise comparison matrix, the Consistency Index (CI) needs to be compared to an appropriate Random Index (RI). The RI is derived from a table of random indices that corresponds to matrices of different sizes and provides a benchmark to assess the CI value obtained.

```
RI = ri_df.loc[ri_df['Matrix size'] == n, 'Random consistency index (RI)'].iloc[0]
print(f'Random consistency index (RI) is: {RI}')
```

```
Random consistency index (RI) is: 1.32
```

Figure 3.8: Extraction of the Random Consistency Index (RI) from an external table based on the matrix size.

As depicted in Figure 3.8, we extract the RI for a matrix of size n from an external source, which is typically a pre-calculated table of RI values. For our analysis, the RI value is determined to be 1.32, which is the standard value for a matrix of our specified size. This RI will be used in the final calculation of the Consistency Ratio (CR) to determine the acceptability of the consistency level in our pairwise comparisons.

3.8 Step 7: Calculate Consistency Ratio (CR)

The final step in assessing the consistency of the pairwise comparisons in the AHP methodology is to calculate the Consistency Ratio (CR). The CR is the result of dividing the Consistency Index (CI) by the Random Index (RI). This ratio is a measure of how much the pairwise comparison matrix deviates from a perfectly consistent matrix.

As shown in Figure 3.9, the CR value is calculated and then used to evaluate whether the comparisons made in the AHP model are consistent. If the CR is 0.1 or less, the model is considered to have an acceptable level of consistency. In our case, the calculated CR is 0.0384, which is well below the acceptable threshold, thereby indicating that the model is consistent.

The consistency of the model ensures that the relative weights derived from the pairwise comparison matrix are reliable, and thus, the subsequent decision-making process based on these weights is valid. A consistent AHP model is essential for deriving meaningful

```python
CR = CI / RI
print(f'The Consistency Ratio is: {CR}')
if CR<0.1:
    print('The model is consistent')
else:
    print('The model is not consistent')
```

```
The Consistency Ratio is: 0.03843634374820358
The model is consistent
```

Figure 3.9: Calculation of the Consistency Ratio (CR) and evaluation of the model's consistency.

and dependable results in multi-criteria decision analysis.

3.9 Step 8: Calculate Global Score & Rank the Alternatives

Having established a consistent priority vector, we proceed to the final step in the AHP methodology, which is to calculate a global score for each alternative and then rank them. The global score for an alternative is the result of multiplying its criteria values by the respective criteria weights and summing the results.

As illustrated in Figure 3.10, the global score for each laptop is calculated by dotting the criteria data frame with the priority vector. The laptops are then sorted based on their global scores in descending order to identify which laptop offers the highest value as per the established criteria. Additionally, a rank column is added to the dataset, ranking the laptops based on their global scores using the 'min' method, which assigns the same rank to ties.

The laptop with the highest global score is considered the best choice in terms of the criteria set out in the decision-making process. This global score reflects the performance of each laptop across all considered criteria, providing a comprehensive view of each option's strengths and weaknesses.

The methodology outlined in this report, culminating in the calculation of global scores

```python
criteria_df_copy=criteria_df.copy()

criteria_df['Global_Score'] = criteria_df.dot(priority_vector)

# Sort the laptops based on their global scores in descending order (highest score first)
criteria_df.sort_values(by='Global_Score', ascending=False, inplace=True)

# Add a rank column based on the global score
criteria_df['AHP_Rank'] = criteria_df['Global_Score'].rank(ascending=False, method='min')
criteria_df
```

	Cpu_Speed	Cpu_Series_Score	Total_Pixels	Memory_Size_GB	Inches	Weight	Price_euros	Global_Score	AHP_Rank
92	1.6	2.0	5760000	256.0	13.3	1.23	1869.00	756776.634146	1.0
60	2.9	2.0	4096000	512.0	13.3	1.37	1958.90	538439.702091	2.0
27	1.2	2.0	3840000	256.0	12.3	1.10	1559.00	504658.451082	3.0
26	1.2	2.0	3840000	128.0	12.3	1.10	1275.00	504529.641245	4.0
18	2.6	3.0	3686400	256.0	14.0	1.50	1962.98	484678.258223	5.0
...	...	...	...	...	...	...	...	...	...
96	1.2	1.5	15360	240.0	12.5	1.09	3100.00	3389.941848	146.0
44	2.8	3.0	15360	256.0	15.6	1.78	2712.00	3219.922901	147.0
86	2.3	2.0	15360	256.0	15.6	2.04	2250.68	3016.624380	148.0
42	1.6	2.0	15360	256.0	13.3	1.29	1499.00	2685.039087	149.0
56	2.8	3.0	15360	128.0	15.6	2.62	1498.00	2681.504072	150.0

Figure 3.10: Calculating the global scores for each alternative and ranking them accordingly.

and subsequent ranking, equips decision-makers with a rigorous and transparent tool to evaluate multiple alternatives against a set of weighted criteria.

3.10 Trade-off/Sensitivity Analysis

The Analytic Hierarchy Process (AHP) model not only provides a framework for decision-making but also offers a tool for trade-off or sensitivity analysis. This analysis is crucial as it allows us to understand the impact of changing criteria weights on the final decision. Sensitivity analysis tests the robustness of the top choice against changes in the weighting of the criteria, offering insights into how sensitive the outcome is to our subjective preferences.

In our study, we performed a sensitivity analysis on the weight of one specific criterion. We adjusted its weight from 0.1 to 0.9 in increments, redistributing the remaining weights among the other criteria proportionately. The goal was to observe how the global score of the initially selected best-value laptop changes with the varying importance of this criterion.

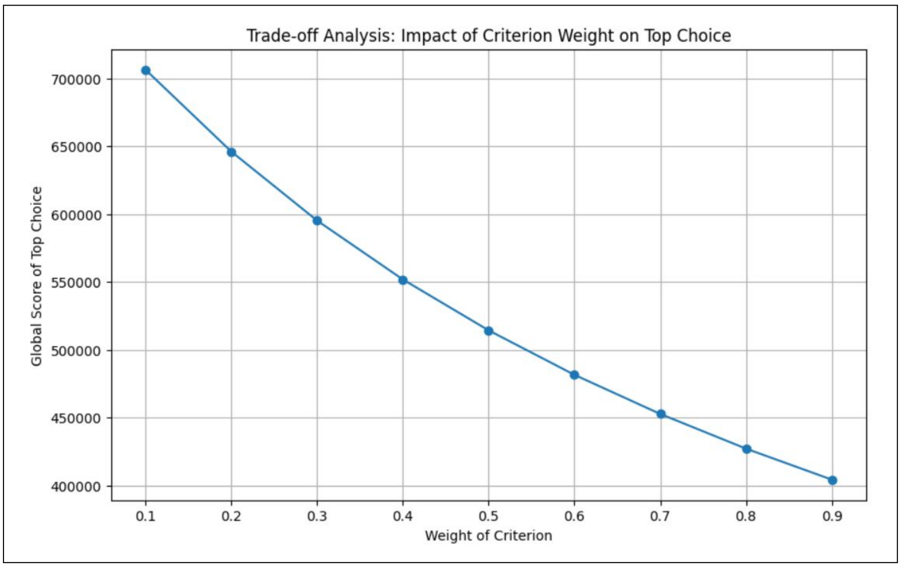

Figure 3.11: Sensitivity analysis depicting the impact of adjusting the weight of a single criterion on the global score of the top choice.

The plot in Figure 3.11 demonstrates the results of this analysis. It reveals a clear negative correlation: as the weight of the criterion increases, the global score of the top choice diminishes. This trend indicates that the top-ranked laptop is less ideal when the considered criterion's importance is heightened. It is worth noting that such a pattern underscores the necessity to carefully evaluate the weight assigned to each criterion.

This insight allows decision-makers to contemplate the stability of their choice and determine if the current preference holds across a range of weight adjustments. It also helps in understanding the potential trade-offs when prioritizing certain aspects over others.

3.11 Conclusion

Through meticulous analysis using the Analytic Hierarchy Process (AHP), we've discerned which laptop offers the best value for money across a range of criteria. The sensitivity analysis provided deeper insights, revealing the influence of criteria weight adjustments on the final decision, underscoring the necessity of balanced weighting. The robust AHP framework, combined with the thoroughness of our trade-off investigation, reinforces the importance of reflective deliberation in decision-making. Consequently, the study illustrates the power of AHP not just in reaching a decision but also in appreciating the subtleties and implications of the decision-making process itself.

Chapter 4

TOPSIS Methodology

4.1 Introduction

TOPSIS, an acronym for Technique for Order Preference by Similarity to Ideal Solution, is a multi-criteria decision-making (MCDM) method. It evaluates a set of alternatives based on their distance from an ideal solution and a nadir solution. This chapter outlines the application of TOPSIS to our dataset of laptops, as an extension to the insights provided by the Analytic Hierarchy Process (AHP).

4.2 Normalization of Decision Matrix

The TOPSIS method begins with the normalization of the decision matrix to remove the units and make the criteria comparable. We executed this by scaling the data, dividing each element by the square root of the sum of squares for each criterion's values, rendering a dimensionless matrix.

4.3 Weighting with AHP-Derived Priority Vector

The normalized decision matrix is then weighted using the priority vector derived from AHP, integrating the relative importance of each criterion into the TOPSIS framework. This step synthesizes the preference information from AHP with the comparative analysis of TOPSIS.

4.4 Ideal and Nadir Solutions

Subsequently, the ideal best (most desirable) and the ideal worst (least desirable) solutions are identified. These represent the hypothetical best and worst performance scores across all criteria.

4.5 Separation Measures

The separation measures for each alternative are computed next. This involves calculating the Euclidean distance of each alternative from the ideal and nadir solutions, known as the positive and negative ideal solutions, respectively.

4.6 Relative Closeness to the Ideal Solution

The relative closeness to the ideal solution is determined by the proportion of each alternative's distance from the nadir solution to the sum of its distances from both the ideal and nadir solutions. This metric quantifies how close each alternative is to the optimal performance.

4.7 TOPSIS Scores and Rankings

The final stage is to assign TOPSIS scores to each alternative based on their relative closeness to the ideal solution. Alternatives are then ranked in descending order of these scores, indicating their preference with a higher score denoting a better alternative.

```
# Apply TOPSIS Method
# 1. Normalize the decision matrix (criteria_df)
# Exclude 'Global_Score' and 'Rank' columns for TOPSIS normalization
criteria_columns = criteria_df.columns.difference(['Global_Score', 'AHP_Rank'])
normalized_criteria_df = criteria_df[criteria_columns].div(np.sqrt((criteria_df[criteria_columns]**2).sum()))

# Proceed with weighting the normalized decision matrix using the AHP-derived priority vector
weighted_normalized_criteria_df = normalized_criteria_df.multiply(priority_vector, axis=1)


# 2. Weight the normalized decision matrix using AHP-derived priority_vector
weighted_normalized_criteria_df = normalized_criteria_df.multiply(priority_vector, axis=1)

# 3. Determine the ideal best and worst solutions
ideal_best = weighted_normalized_criteria_df.max()
ideal_worst = weighted_normalized_criteria_df.min()

# 4. Calculate the separation measures for each alternative
separation_from_best = np.sqrt(((weighted_normalized_criteria_df - ideal_best)**2).sum(axis=1))
separation_from_worst = np.sqrt(((weighted_normalized_criteria_df - ideal_worst)**2).sum(axis=1))

# 5. Calculate the relative closeness to the ideal solution
relative_closeness = separation_from_worst / (separation_from_best + separation_from_worst)

# 6. Assign TOPSIS scores and ranks to alternatives
criteria_df['TOPSIS_Score'] = relative_closeness
criteria_df['TOPSIS_Rank'] = criteria_df['TOPSIS_Score'].rank(ascending=False, method='min')

criteria_df
```

Figure 4.1: Implementation of the TOPSIS method using Python.

	Cpu_Speed	Cpu_Series_Score	Total_Pixels	Memory_Size_GB	Inches	Weight	Price_euros	Global_Score	AHP_Rank	TOPSIS_Score	TOPSIS_Rank
92	1.6	2.0	5760000	256.0	13.3	1.23	1869.00	756776.634146	1.0	0.563394	2.0
60	2.9	2.0	4096000	512.0	13.3	1.37	1958.90	538439.702091	2.0	0.484209	16.0
27	1.2	2.0	3840000	256.0	12.3	1.10	1559.00	504658.451082	3.0	0.429335	36.0
26	1.2	2.0	3840000	128.0	12.3	1.10	1275.00	504529.641245	4.0	0.426974	38.0
118	2.6	3.0	3686400	256.0	14.0	1.50	1962.98	484678.258223	5.0	0.465110	22.0
...	...	...	...	...	...	...	...	...	...	...	...
96	1.2	1.5	15360	240.0	12.5	1.09	3100.00	3389.941848	146.0	0.133764	149.0
144	2.8	3.0	15360	256.0	15.6	1.78	2712.00	3219.922901	147.0	0.225133	145.0
86	2.3	2.0	15360	256.0	15.6	2.04	2250.68	3016.624380	148.0	0.259715	142.0
42	1.6	2.0	15360	256.0	13.3	1.29	1499.00	2685.039087	149.0	0.105427	150.0
56	2.8	3.0	15360	128.0	15.6	2.62	1498.00	2681.504072	150.0	0.347709	92.0

150 rows × 11 columns

Figure 4.2: TOPSIS method results.

Chapter 5

Concluding Discussion

In the final analysis of our decision support system (DSS), we delve into a thorough comparison of the Analytic Hierarchy Process (AHP) and the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) methodologies. Our goal is to deduce the robustness and reliability of each method, as well as to understand the nuances that arise from their application to our dataset of laptops.

5.1 Ranking Correlation and Discrepancies

We commence our analysis by examining the correlation between the rankings generated by the two methods. Spearman's rank correlation coefficient provides a measure of the monotonic relationship between the AHP and TOPSIS ranks. A modest Spearman correlation of 0.2503 suggests that there is only a slight tendency for laptops that are ranked higher by AHP to also be ranked higher by TOPSIS.

```python
# Ranking Correlation
# Calculate Spearman's rank correlation
spearman_corr, _ = spearmanr(criteria_df['AHP_Rank'], criteria_df['TOPSIS_Rank'])
print(f"Spearman's Rank Correlation: {spearman_corr}")

# Rank Discrepancies
# Calculate discrepancies between rankings
criteria_df['Rank_Discrepancy'] = np.abs(criteria_df['AHP_Rank'] - criteria_df['TOPSIS_Rank'])

# Sort by largest discrepancies
discrepancies = criteria_df.sort_values(by='Rank_Discrepancy', ascending=False)

# Categorize discrepancies into 'Low', 'Medium', 'High'
criteria_df['Discrepancy_Category'] = pd.cut(criteria_df['Rank_Discrepancy'],
                                             bins=[0, 3, 7, np.inf],
                                             labels=['Low', 'Medium', 'High'])



# Display the DataFrame to check the new columns
print(criteria_df[['AHP_Rank', 'TOPSIS_Rank', 'Rank_Discrepancy', 'Discrepancy_Category']].head())
```

```
Spearman's Rank Correlation: 0.25026934578309395
     AHP_Rank  TOPSIS_Rank  Rank_Discrepancy Discrepancy_Category
92        1.0          2.0               1.0                  Low
60        2.0         16.0              14.0                 High
27        3.0         36.0              33.0                 High
26        4.0         38.0              34.0                 High
118       5.0         22.0              17.0                 High
```

Figure 5.1: Spearman's rank correlation coefficient between AHP and TOPSIS ranks.

The degree of ranking discrepancy was categorized into 'Low', 'Medium', and 'High' based on the absolute differences in ranks assigned by AHP and TOPSIS. This classification unveiled specific cases where the methodologies diverged significantly, suggesting that certain criteria or alternative evaluations were interpreted or weighted quite differently by each method.

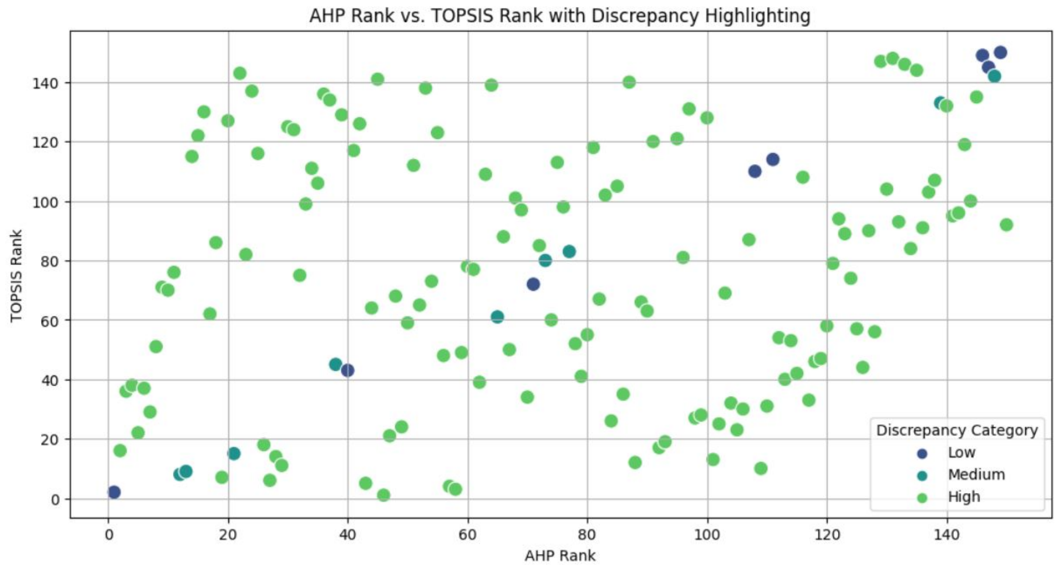

Figure 5.2: AHP Rank versus TOPSIS Rank with Discrepancy Highlighting.

The scatter plot in Figure 5.2 provides a visual representation of the rank discrepancies between AHP and TOPSIS. Alternatives that lie along the line y=x would indicate a perfect agreement between the two methods, which is clearly not the case here. The discrepancies, particularly those classified as 'High', warrant a detailed investigation into the decision-making criteria and the weight distribution used in each method. These discrepancies may reveal inherent biases or sensitivities in the methodologies, influencing the selection of the optimal alternative.

5.2 Monte Carlo Simulation and Ranking Stability

To evaluate the robustness of our ranking methods, we performed a Monte Carlo simulation by introducing random perturbations to the priority vector used in AHP. This approach simulates potential variability in the decision-maker's judgments, reflecting real-world uncertainties in the evaluation process.

The stability of the rankings was then analyzed by examining how these perturbations affected the rank positions of each laptop. Our simulation results indicate that while some laptops maintained stable positions across simulations, others exhibited significant fluctuations, as depicted in Figure 5.3.

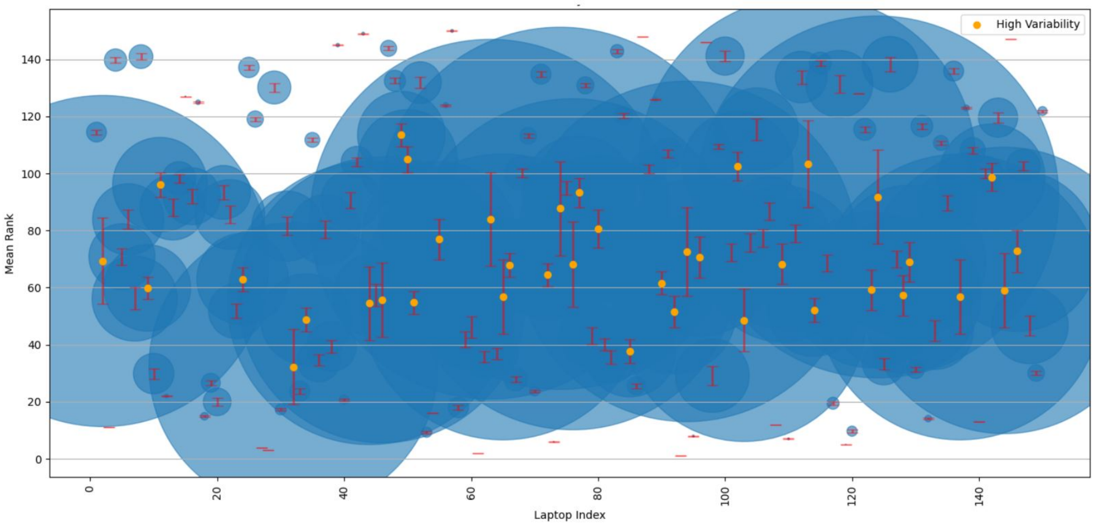

Figure 5.3: Monte Carlo simulation of ranking stability, highlighting laptops with high variability in ranks.

The plot reveals that laptops with high variability (indicated by the orange markers) may require additional consideration or weight adjustments to ensure a more definitive ranking. The blue shaded areas correspond to the density of rank variability, showcasing regions where the ranking order is particularly sensitive to changes in the priority vector.

5.2.1 Kendall's Tau Correlation

The Kendall's Tau correlation between AHP and TOPSIS ranks was found to be 0.18, indicating a weak correlation. This suggests that the two methods, while sharing some underlying agreement, can lead to considerably different rank orders for the set of alternatives. Such a finding underscores the importance of method selection based on the specific criteria and context of the decision problem at hand.

The modest correlation coefficient prompts us to consider the implications for decision-makers. For decisions where rank order is paramount, such as selecting a single best laptop, it may be advisable to investigate the criteria that contribute to ranking discrepancies and potentially revise the criteria weights or decision model accordingly. Conversely, when the goal is to identify a subset of top-performing laptops rather than a single best choice, the observed discrepancies might be less critical.

5.3 Implications for Decision Making

The insights gained from both the Monte Carlo simulation and the Kendall's Tau correlation analysis hold substantial implications for the decision-making process. They highlight the necessity of a comprehensive understanding of the decision context and the sensitivity of the chosen methods to the specific priorities of the stakeholders involved.