OPTIMISING HOSPITAL STAFF SCHEDULING USING LINEAR AND INTEGER PROGRAMMING

By Nadia Isanga

Project Report

1. Introduction

Background

In today's business environment, timely resource allocation is essential. Whether it's scheduling staff or managing a supply chain, efficient resource allocation can make a significant difference. Linear Programming (LP) and Integer Programming (IP) offer smart solutions to these challenges.

This report focuses on staff scheduling—a common headache for many businesses. The primary goal is to develop a flexible system that can handle unexpected changes, ensuring that staffing needs are met efficiently. With this approach, I aim to build a smart and adaptable solution that meets the demands of modern businesses.

Problem Statement

In the healthcare sector, hospitals face the critical task of staff scheduling, a complex challenge that involves aligning nurses' and doctors' availability with the fluctuating demands of patient care. The optimization model I aim to develop tackles this by creating an efficient and cost-effective schedule that ensures all shifts are adequately covered. The goal is to avoid unnecessary overtime and prevent understaffing, which can compromise patient care. The desired outcome is a scheduling system that not only manages costs but also accommodates the unpredictable nature of healthcare needs, including emergency situations that require extremely quick responses. The expected improvements include a smoother operational flow, a happier and more balanced workforce, and enhanced patient care.

2. Data Collection and Preparation

Data Collection

I collected the hospital staff dataset from Kaggle, https://www.kaggle.com/datasets/mpwolke/cusersmarildownloadshospcsv. It contains information on hours worked by staff and hospital cost centre groupings in California. I also generated additional data using Python code for staff roles and IDs. Assumptions were made for shifts (Morning,evening, night), employment contracts(full time or part -time) , and hourly pay rates to simulate real-world scenarios. To maintain data integrity, and especially because the dataset was fairly large missing values were handled by removing them. Since the dataset was clean, minimal pre-processing was necessary.

Parameter Estimation

Fixed Parameters:

- Shift Types: Defined as Morning, Evening, and Night, these shifts are a constant operational requirement and were also assumed.
- Full-Time and Part-Time Contracts: The classification of staff into full-time or part-time contracts was a reasonable assumption.
- Hourly Pay Rates: Assumed based on standard rates for different roles, these rates are considered fixed for the model's scope.

Uncertain Parameters:

- Staff Availability: Variability in staff availability due to unforeseen circumstances like illnesses or personal emergencies requires predictive analysis.
- Patient Influx: The number of patients and the demand for care can fluctuate significantly, impacting staffing needs.

Staff Availability and Patient Influx:

The process of developing the model relied more on direct data utilization and assumptions for uncertain parameters rather than employing predictive analytics methods such as regression, time-series analysis, or machine learning.

Moving forward, incorporating predictive analytics could enhance the model's accuracy and responsiveness to changing conditions. Such advancements would involve:

- Gathering More Granular Data: Collecting detailed historical data on staff absences and patient admissions over time.
- Implementing Predictive Models: Applying time-series analysis, regression models, or machine learning to predict variations in staff availability and patient influx.
- Iterative Testing and Integration: Testing predictive models for accuracy and integrating their outputs as dynamic inputs into the optimization model.

3. Mathematical Modelling (up to 450 words)

Decision Variables

- $x_{i, j}$: Number of hours staff member *i* is scheduled to work during shift *j*.
  - *i*: Index for staff members.
  - *j*: Index for shifts, where 1 = Morning, 2 = Evening, 3 = Night.
  - Units: Hours.

For example, if staff member 1 (a registered nurse) is scheduled to work 8 hours during the night shift, this will be represented as $x_{1, 3} = 8$.

Parameters

- $c_{i, j}$: Cost of scheduling staff member *i* during shift *j*.
- D_j: Required staffing hours for shift *j*.
- F_i: Binary indicator for whether staff member *i* is full-time (1 if full-time, 0 otherwise).
- P_i: Binary indicator for whether staff member *i* is part-time (1 if part-time, 0 otherwise).
- H_i^{min}: Minimum required hours for full-time staff member *i* per scheduling period.
- H_i^{max}: Maximum allowable hours for part-time staff member *i* per scheduling period.
- $q_{i, k}$: Binary indicator if staff member *i* is qualified for role *k*.
- R_k: Required hours of role *k* for each shift.

Objective Function

Minimize $Z = \sum\sum(c_{i, j} * x_{i, j})$

The objective function seeks to minimize the total operational cost by optimizing the number of hours each full or part time staff member is scheduled for each shift, considering the cost associated with these hours. The costs may vary by shift due to different pay rates for evening and night shifts, as indicated in the shift differential. By minimizing the sum of these products, the hospital can achieve an optimal balance between cost efficiency and operational needs.

Constraints

1. Coverage Constraint (Ensures adequate staffing for each shift):

   $\sum x_{i, j} \geq D_j$ for each shift *j*.

2. Full-Time Hours Constraint (Ensures full-time staff meet minimum hours):

   $\sum x_{i, j} \geq H_i^{min} * F_i$ for each full-time staff member *i*.

3. Part-Time Hours Constraint (Limits hours for part-time staff to their maximum):

   $\sum x_{i, j} \leq H_i^{max} * P_i$ for each part-time staff member *i*.

4. Qualification Constraint (Ensures staff qualifications are met for certain roles and shifts):

   $\sum x_{i, j} * q_{i, k} \geq R_k$ for each role *k* and shift *j*.

The model takes into account the cost of scheduling staff across different shifts, the minimum staffing requirements for each shift to maintain patient care, and contractual obligations regarding working hours for staff, all while ensuring staff are only scheduled for roles they are qualified for. It aims to create an efficient, cost-effective, and fair staffing schedule that supports both operational efficiency and staff satisfaction in a hospital setting.

Implementation and Solution

The development of the optimization model for hospital staff scheduling was carried out using the PuLP library. An iterative approach adopted for this was as follows;

- Initial Model Setup: Started with defining the core structure of the model, including decision variables x_{i, j} to represent the number of hours each staff member *i* is scheduled for each shift *j.* The initial objective was to minimize total staffing costs, balancing economic efficiency with the need for adequate shift coverage.
- Iterative Constraint Addition: Constraints were introduced in stages, allowing for the assessment of their individual and cumulative impacts on the model. This step-by-step inclusion covered;
    1. Shift coverage to ensure all shifts are adequately staffed.
    2. Full-time and part-time hours to respect contractual working limits.
    3. Staff qualifications to match staff members to shifts based on their skills.
- Solving the Model: At each iteration, after adding a new set of constraints, the model was solved using prob.solve(), which consistently returned 1, indicating a successful optimization. This was done to make it easy to identify any potential issues early on that could lead to infeasible or suboptimal distributions.
- Refinement and Final Solution: With each constraint added and the model solved, adjustments were made based on the outcomes. This iterative refinement continued until the model adequately represented the complex dynamics of hospital staff scheduling.

```
In [21]:  from pulp import *

          # Initialize the problem
          prob = LpProblem("Hospital_Staff_Scheduling", LpMinimize)
```

```
In [22]:  # Assumed roles and their costs
              'Registered Nurse': 35,
              'Technician': 30,
              'Aide': 30,
              'Management': 45,
              # Additional roles...
          }

          # Decision variables: Number of hours allocated to each role
          hours_vars = LpVariable.dicts("HoursAllocated", cost_per_hour.keys(), 0, None, cat='Integer')
```

```
In [23]:  prob += lpSum([cost_per_hour[role] * hours_vars[role] for role in cost_per_hour]), "TotalOperationalCost"
```

```
In [24]:  min_hours_needed = {
              'Registered Nurse': 500,
              'Technician': 300,
              'Aide': 200,
              'Management': 100,
              # Additional roles...
          }

          # Ensure each role meets its minimum required hours
          for role in min_hours_needed:
              prob += hours_vars[role] >= min_hours_needed[role], f"MinHoursReq_{role}"
```

```
In [25]:  total_productive_hours_available = 10000  # Example total available hours

          prob += lpSum([hours_vars[role] for role in cost_per_hour]) <= total_productive_hours_available, "TotalHoursAvailable"
```

Figure 1:Initial model set up

```
In [26]:  prob.solve()

          # Output the status and decision variable values
          print("Status:", LpStatus[prob.status])

          for v in prob.variables():
              print(v.name, "=", v.varValue)

          Status: Optimal
          HoursAllocated_Aide = 200.0
          HoursAllocated_Management = 100.0
          HoursAllocated_Registered_Nurse = 500.0
          HoursAllocated_Technician = 300.0
```

Figure 2: Optimal Status and Staff allocated working hours.

```
    'Environmental & Food Services': {'Morning': 2, 'Evening': 2, 'Night': 2},
    'Other': {'Morning': 2, 'Evening': 2, 'Night': 2},
    'Contracted Registry Nursing': {'Morning': 1, 'Evening': 1, 'Night': 1},
    'Contracted Other': {'Morning': 1, 'Evening': 1, 'Night': 1},
    'Daily Cost Centers': {'Morning': 1, 'Evening': 1, 'Night': 1},
    'Ambulatory Cost Centers': {'Morning': 1, 'Evening': 1, 'Night': 1},
    'Ancillary Cost Centers': {'Morning': 1, 'Evening': 1, 'Night': 1},
    'Education Cost Centers': {'Morning': 1, 'Evening': 1, 'Night': 1},
    'General Services Cost Centers': {'Morning': 1, 'Evening': 1, 'Night': 1},
    'Fiscal Services Cost Centers': {'Morning': 1, 'Evening': 1, 'Night': 1},
    'Administrative Services Cost Centers': {'Morning': 1, 'Evening': 1, 'Night': 1},
}

In [34]: for role in roles:
             for shift in shifts:
                 # Ensure minimum staffing requirements are met for each role during each shift
                 prob += lpSum([hours_vars[(staff_id, shift)] for staff_id in staff_df[staff_df['Role'] == role]['StaffID']]) >= min_s

In [35]: # Maximum working hours per week for any staff member
         standard_max_hours_per_week = 40

         #  applying this standard uniformly across all staff.
         max_hours_per_staff = {staff_id: standard_max_hours_per_week for staff_id in staff_ids}

In [36]: for staff_id in staff_ids:
             prob += lpSum([hours_vars[(staff_id, shift)] for shift in shifts]) <= max_hours_per_staff[staff_id], f"MaxHours_{staff_id

In [37]: # Solve the problem
         prob.solve()

         # Output the status
         print("Status:", LpStatus[prob.status])

         Status: Optimal
```

Figure 3:Iterative approach to introduce shifts as a constraint.

```
4   AI005   Aide   Full-time
Status
Full-time    55
Part-time    55
Name: count, dtype: int64

In [45]: # Using the list of staff IDs for full-time staff
         full_time_staff_ids = staff_df[staff_df['Status'] == 'Full-time']['StaffID'].tolist()

         # Re-check the formulation of constraints for full-time staff minimum hours
         for staff_id in full_time_staff_ids:
             # Minimum hours constraint
             prob += lpSum([hours_vars[(staff_id, shift)] for shift in shifts]) >= FULL_TIME_MIN_HOURS, f"MinHoursFullTime_{staff_id}"

             # Maximum hours constraint - ensuring it does not conflict
             prob += lpSum([hours_vars[(staff_id, shift)] for shift in shifts]) <= FULL_TIME_MAX_HOURS, f"MaxHoursFullTime_{staff_id}"

In [46]: # Loop through part-time staff and add a constraint for minimum scheduled hours
         part_time_staff = staff_df[staff_df['Status'] == 'Part-time']['StaffID'].tolist()
         for staff_id in part_time_staff:
             prob += lpSum([hours_vars[(staff_id, shift)] for shift in shifts]) >= MIN_PART_TIME_HOURS, f"MinPartTime_{staff_id}"

In [47]: # Solve the problem
         prob.solve()

Out[47]: 1

In [48]: print("Status:", LpStatus[prob.status])

         Status: Optimal
```

Figure 4: Optimal solution obtained with employment contract constraint.

```
            staff_id, shift = match.groups()
            print(f"{staff_id} should work {v.varValue} hours in the {shift} shift.")
```

```
AI001 should work 35.0 hours in the Morning shift.
AI002 should work 35.0 hours in the Morning shift.
AI003 should work 35.0 hours in the Morning shift.
AI004 should work 35.0 hours in the Morning shift.
AI005 should work 35.0 hours in the Morning shift.
AI006 should work 35.0 hours in the Morning shift.
AI007 should work 35.0 hours in the Morning shift.
AI008 should work 35.0 hours in the Morning shift.
AI009 should work 35.0 hours in the Morning shift.
AI010 should work 35.0 hours in the Morning shift.
AI011 should work 5.0 hours in the Morning shift.
AI012 should work 5.0 hours in the Morning shift.
AI013 should work 5.0 hours in the Morning shift.
AI014 should work 5.0 hours in the Morning shift.
AI015 should work 5.0 hours in the Morning shift.
AI016 should work 5.0 hours in the Morning shift.
AI017 should work 5.0 hours in the Morning shift.
AI018 should work 5.0 hours in the Morning shift.
AI019 should work 5.0 hours in the Morning shift.
```

Figure 5:Staff assigned shifts.

```
In [31]:
shifts = ['Morning', 'Evening', 'Night']
staff_ids = staff_df['StaffID'].tolist()

# Re-initializing hours_vars with clear indexing by staff_id and shift
hours_vars = LpVariable.dicts("StaffShiftHours",
                              [(staff_id, shift) for staff_id in staff_ids for shift in shifts],
                              0,
                              None,
                              cat='Integer')

# Adjust how you access hours_vars in the objective function accordingly:
prob += lpSum([
    cost_per_hour[staff_df.loc[staff_df['StaffID'] == staff_id, 'Role'].values[0]] *
    shift_differential[shift] *
    hours_vars[(staff_id, shift)]
    for staff_id in staff_ids
    for shift in shifts
]), "TotalOperationalCost"

C:\Users\NADIA\anaconda3\Lib\site-packages\pulp\pulp.py:1668: UserWarning: Overwriting previously set objective.
  warnings.warn("Overwriting previously set objective.")
```

```
In [32]:
prob += lpSum([
    cost_per_hour[staff_df.loc[staff_df['StaffID'] == staff_id, 'Role'].values[0]] *
    shift_differential[shift] *
    hours_vars[(staff_id, shift)]
    for staff_id in staff_ids
    for shift in shifts
]), "TotalOperationalCost"
```

```python
for staff_id in part_time_staff:
    total_hours = scheduling_results.get(staff_id, 0)  # Directly retrieve total hours, default to 0 if not found
    if total_hours <= PART_TIME_MAX_HOURS:
        print(f"  - {staff_id}: {total_hours} hours scheduled (Max: {PART_TIME_MAX_HOURS}) - Compliant")
    else:
        print(f"  - {staff_id}: {total_hours} hours scheduled (Max: {PART_TIME_MAX_HOURS}) - Exceeds maximum hours")
```

```
Part-time staff compliance check:
  - AI011: 5.0 hours scheduled (Max: 20) - Compliant
  - AI012: 5.0 hours scheduled (Max: 20) - Compliant
  - AI013: 5.0 hours scheduled (Max: 20) - Compliant
  - AI014: 5.0 hours scheduled (Max: 20) - Compliant
  - AI015: 5.0 hours scheduled (Max: 20) - Compliant
  - AI016: 5.0 hours scheduled (Max: 20) - Compliant
  - AI017: 5.0 hours scheduled (Max: 20) - Compliant
  - AI018: 5.0 hours scheduled (Max: 20) - Compliant
  - AI019: 5.0 hours scheduled (Max: 20) - Compliant
  - AI020: 5.0 hours scheduled (Max: 20) - Compliant
  - MA006: 5.0 hours scheduled (Max: 20) - Compliant
  - MA007: 5.0 hours scheduled (Max: 20) - Compliant
  - MA008: 5.0 hours scheduled (Max: 20) - Compliant
  - MA009: 5.0 hours scheduled (Max: 20) - Compliant
  - MA010: 5.0 hours scheduled (Max: 20) - Compliant
  - RE026: 5.0 hours scheduled (Max: 20) - Compliant
  - RE027: 5.0 hours scheduled (Max: 20) - Compliant
  - RE028: 5.0 hours scheduled (Max: 20) - Compliant
  - RE029: 5.0 hours scheduled (Max: 20) - Compliant
  - RE030: 5.0 hours scheduled (Max: 20) - Compliant
```

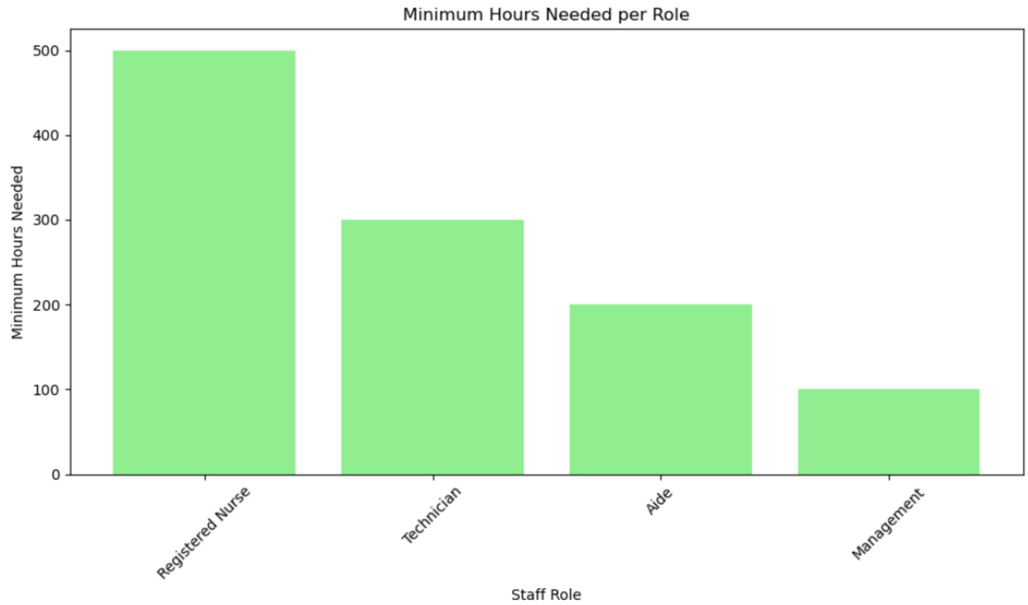
Figure 6: Compliance check for allocation of part -time employees.

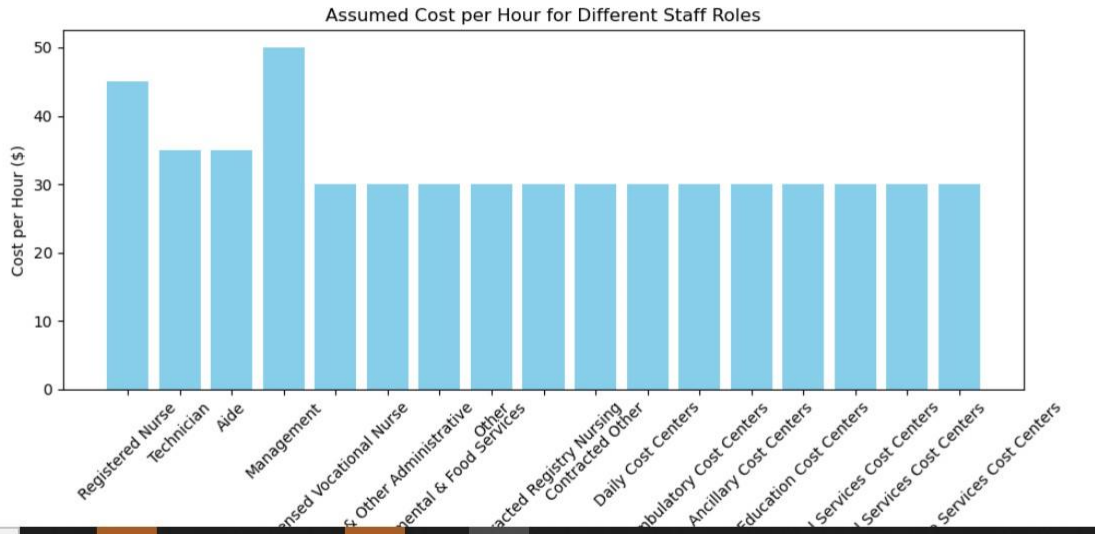
Every iteration's optimal solution, signified by the solver's return value of 1, demonstrates feasibility under the current constraints. The final model configuration provides a schedule that optimizes staffing costs while ensuring operational demands are met, respecting both full-time and part-time contractual hours, and aligning staff qualifications with shift requirements.

In practical terms, the optimal solution from this model translates into a staffing schedule that optimizes operational costs, enhances staff satisfaction by honouring their contractual preferences, and ensures patient care needs are met by appropriately qualified personnel.

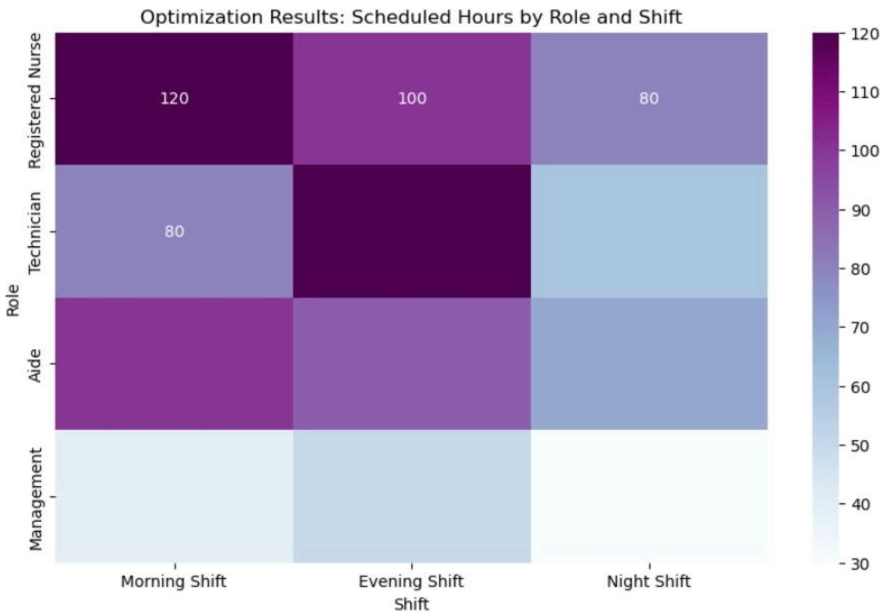
4. Data Visualization and Interpretation

Visualizations for the model parameters.

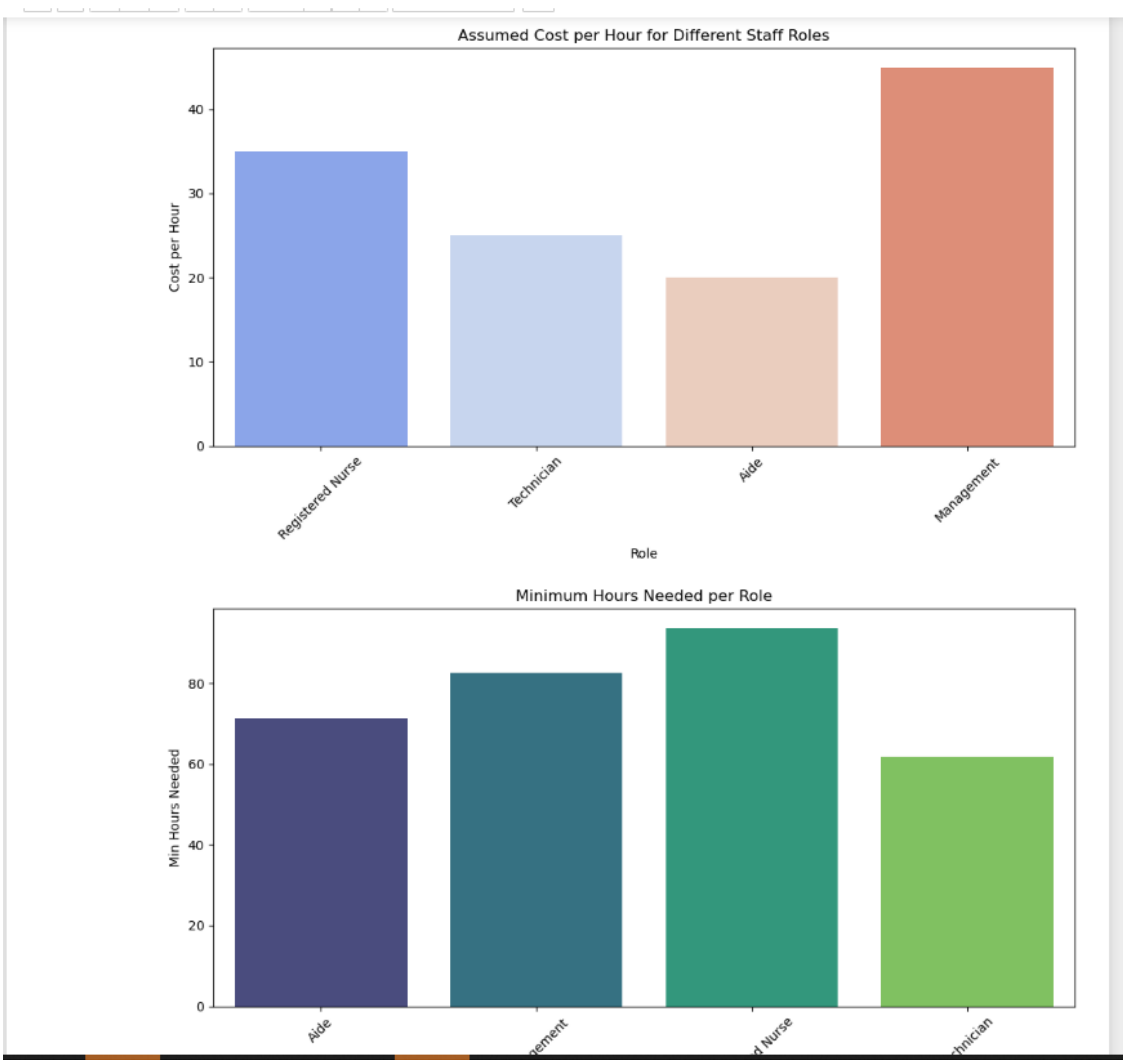




Optimization Visualizations.



The heatmap above provides a visual summary of the optimization results, showcasing how many hours each role is scheduled for across different shifts. The colour intensity indicates the volume of hours, allowing viewers to quickly identify which roles and shifts are most heavily allocated. This visualization effectively communicates the outcome of the optimization model, highlighting the distribution of staffing resources to meet hospital needs. It allows hospital administrators to see at a glance how the model has allocated staff across shifts, facilitating strategic decisions to ensure balanced coverage and operational efficiency.



This combined bar and line chart effectively communicates two key model parameters: the cost per hour for different staff roles (bar chart) and the minimum hours needed for those roles (line plot). By presenting both metrics in a single visualization, stakeholders can quickly grasp the financial and operational considerations that underpin the scheduling model. The use of two y-axes allows for distinct but related datasets to be compared directly, facilitating a comprehensive understanding of the staffing requirements and constraints.

5. Reflection

The optimization model developed for hospital staff scheduling represents a significant step towards addressing the complexities of real-world staffing needs. However, on critical analysis, there is a lot of room for future development.

One primary limitation is the model's reliance on assumptions, particularly regarding staff availability and patient demand. While these assumptions facilitated initial model development and problem-solving, they may not fully capture the unpredictable nature of healthcare operations. For instance, the model does not dynamically adjust for sudden increases in-patient admissions due to pandemics or natural disasters. It does not account for individual staff preferences and unforeseen absences, which can significantly impact staffing.

Moreover, the current framework primarily focuses on minimizing staffing costs, potentially at the expense of other critical factors such as staff morale, patient satisfaction, and quality of care. These dimensions are crucial for a holistic approach to hospital management but are not directly considered in the objective function.

For future development, the model could be refined by integrating more complex predictive analytics to anticipate variations in staff availability and patient influx more accurately. Additionally, incorporating multi-objective optimization could balance cost minimization with goals related to care quality and staff well-being. Expanding the model to include preferences, qualifications, and emergency staffing requirements in a more nuanced manner would also enhance its practical applicability and effectiveness in a real-world setting.

Overall, while the model serves as a valuable tool for operational planning, its evolution will require addressing these limitations to more comprehensively meet the multifaceted challenges of hospital staff scheduling.

References

1. OpenAI, 2024. *ChatGPT*. Available at: https://openai.com/chatgpt [Accessed 20 March - 11 April 2024].
2. GitHub, 2024. *GitHub: Where the world builds software*. Available at: https://github.com [Accessed 20 March - 11 April 2024]

Appendices

Data Collection

Attached is the link to the dataset that I used to execute the project.

https://www.kaggle.com/datasets/mpwolke/cusersmarildownloadshospcsv and this the original source http://hcai.ca.gov/HID/DataFlow/.

Acknowledgments

In the development of this project on hospital staff scheduling, I utilized OpenAI's ChatGPT as a generative AI tool to assist in various stages of the project, including generating initial ideas, developing problem-solving strategies, and overcoming coding challenges.While ChatGPT offered valuable insights and suggestions , the final decisions and actual content of this report were my original and individual work.

Data Analytics

You
i had chosen a data set on hospital staff , would be it be complex enough for my assignment ? i can attach the assignment for your review.

Data Analytics
A dataset on hospital staff scheduling could indeed be complex and suitable for your assignment, given the various constraints and objectives involved in such a scenario. Staff scheduling, especially in a hospital setting, involves multiple layers of complexity, including but not limited to:

- Matching staff qualifications and roles with specific shifts and departments.
- Ensuring adequate coverage while considering staff preferences and labor laws.
- Minimizing overtime costs and ensuring high levels of patient care and staff satisfaction.

This problem can effectively demonstrate your understanding of linear programming (LP) or integer programming (IP) optimization techniques due to the necessity to balance various competing objectives and constraints.

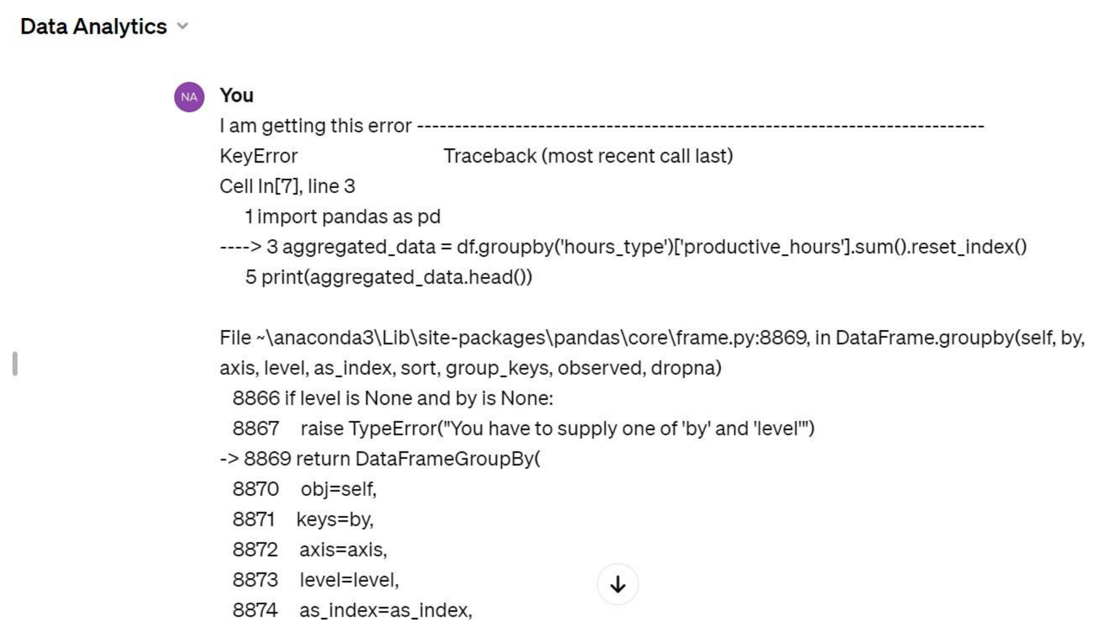
Figure 7: Sample of the usage of Generative AI.

relevant data is collected, cleaned, and prepared for analysis, including forecasting future departmental demand to inform staffing needs effectively.

You
ok this dataset has records of various districts , so does it mean that our goal will be to optimize the resources across a whole country ?

Data Analytics
Given the dataset encompasses records from various districts, it indeed opens up an interesting dimension to the optimization problem. Optimizing hospital staff scheduling on a broader scale, such as across multiple districts or even country-wide, can significantly enhance the overall efficiency and resource allocation in healthcare services. However, this also introduces additional complexity in terms of variability in demand, logistical constraints, and differing regional regulations or staffing policies.

Figure 8:Sample 2 of the use of Generative AI.



Data Analytics

You

I am getting this error ---

KeyError Traceback (most recent call last)

Cell In[7], line 3

1 import pandas as pd

----> 3 aggregated_data = df.groupby('hours_type')['productive_hours'].sum().reset_index()

5 print(aggregated_data.head())

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:8869, in DataFrame.groupby(self, by, axis, level, as_index, sort, group_keys, observed, dropna)

8866 if level is None and by is None:

8867 raise TypeError("You have to supply one of 'by' and 'level'")

-> 8869 return DataFrameGroupBy(

8870 obj=self,

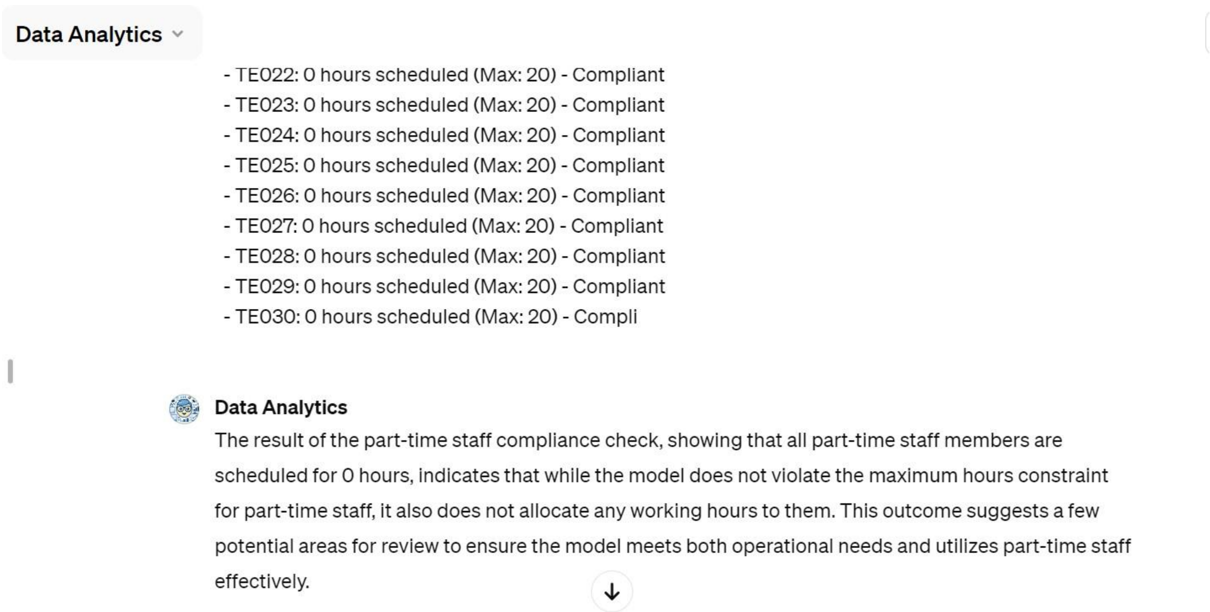
8871 keys=by,

8872 axis=axis,

8873 level=level,

8874 as_index=as_index,

Figure 9:Using Generative AI to troubleshoot my code.



Data Analytics

- TE022: 0 hours scheduled (Max: 20) - Compliant
- TE023: 0 hours scheduled (Max: 20) - Compliant
- TE024: 0 hours scheduled (Max: 20) - Compliant
- TE025: 0 hours scheduled (Max: 20) - Compliant
- TE026: 0 hours scheduled (Max: 20) - Compliant
- TE027: 0 hours scheduled (Max: 20) - Compliant
- TE028: 0 hours scheduled (Max: 20) - Compliant
- TE029: 0 hours scheduled (Max: 20) - Compliant
- TE030: 0 hours scheduled (Max: 20) - Compli

Data Analytics

The result of the part-time staff compliance check, showing that all part-time staff members are scheduled for 0 hours, indicates that while the model does not violate the maximum hours constraint for part-time staff, it also does not allocate any working hours to them. This outcome suggests a few potential areas for review to ensure the model meets both operational needs and utilizes part-time staff effectively.

Figure 10:Use of generative AI to understand why I failed to achieve desired results.