WARWICK
THE UNIVERSITY OF WARWICK

WARWICK BUSINESS SCHOOL

IB2D40

PROGRAMMING FOR BUSINESS APPLICATION

INDIVIDUAL ASSIGNMENT, 2024-25

London's Accident and Emergency Admissions Analysis

5671431
Enric Fernández i Sala

INDEX

1. Data Cleaning and Analysis Process

The dataset contained in the excel document "AandE-Admissions_Dataset.xlsx" is not ready for direct data analysis with Python for several reasons:

- It contains **three unused top rows** for titles and labelling of column groups in excel.
- **Missing values are represented with an asterisk**, which is not a standard format for python libraries.
- The "Hour of Arrival" and "Weekday" **columns** are in a **mixed or string format** but should be converted to integer and categorical respectively to facilitate analysis and manipulation.

The steps taken to clean and prepare the data were the following:

1.1 Creating and understanding the data frame

Firstly, importing Numpy and Pandas libraries to access several functions and methods. Secondly, creating a data frame by importing the file with the "read_excel" method. The additional argument makes the program ignore the first three rows, setting the data frame correctly.

```
df = pd.read_excel(r'/content/AandE-Admissions_Dataset.xlsx', skiprows= 3)
```

As the first two columns have an empty first row because of grouped cells in the excel format, these were renamed with the "rename" method.

```
df = df.rename(columns={'Unnamed: 0': 'Weekday', 'Unnamed: 1': "Hour of arrival"})
```

Next, looking at the df.head() method output we can see the first five rows of the dataframe, which may show useful information.

```
This is the initial dataset:
```

	Weekday	Hour of arrival	Road traffic accident	Assault	Deliberate self-harm	Sports injury	Not known
0	Monday	00:00:00	101	127	44	36	230
1	Monday	01:00:00	59	92	46	8	185
2	Monday	02:00:00	37	84	*	*	150
3	Monday	03:00:00	32	45	27	11	118
4	Monday	04:00:00	24	56	16	14	93

We can observe that this dataset has **7 columns,** 5 of which count the number of admissions for each column's labeled reason, and the other 2 the day of the week and hour of their registration. Missing values are represented with an asterisk (*).

With the df.info() method, we can see basic information of the created dataframe:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 168 entries, 0 to 167
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Weekday                 168 non-null    object
 1   Hour of arrival         168 non-null    object
 2   Road traffic accident   168 non-null    int64
 3   Assault                 168 non-null    int64
 4   Deliberate self-harm    168 non-null    object
 5   Sports injury           168 non-null    object
 6   Not known               168 non-null    int64
dtypes: int64(3), object(4)
memory usage: 9.3+ KB
```

The data frame has **168 observations**. "Object" columns may suggest **mixed data types** or **strings**, for that reason columns 4 and 5, which should contain integers, are the ones with **missing values**. The weekday and Hour columns should be categorical and integer or categorical respectively, so they will have to be converted to these data types to perform the analysis.

1.2 Cleaning and preparing the data

After understanding the original data set, we only need to address incorrect values and modify the current variables (columns) or create new ones in order to begin the analysis. With this method we make sure that possible duplicate observations are erased:

```
df = df.drop_duplicates()
```

Afterwards, we may replace asterisks from the two columns that contain them with "NaN" (the standardized format of missing values). The usage of lambda functions allowed me to apply a short custom function to each element in the specified columns following a condition (if the element was an asterisk it would become NaN, otherwise an integer for admissions).

```
df['Deliberate self-harm '] = df[
    'Deliberate self-harm '].apply(lambda x: float('nan') if x == '*' else int(x))
df['Sports injury '] = df[
    'Sports injury '].apply(lambda x: float('nan') if x == '*' else int(x))
```

Since each observation corresponds to a specific hour, filling the missing values with an average of the hour before and the following may be the best approach to conserve as many observations as possible without altering the data's patterns. We can do so using the interpolation method with a for loop for the columns containing the missing values:

```
columns_to_fix = ['Deliberate self-harm ', 'Sports injury ']
for column in columns_to_fix:
    df[column] = df[column].interpolate(method='linear', limit_direction='both')
```

Additionally, because of the nature of our data, negative values should not be found in the dataset. The following code creates a variable selecting the numeric columns using a dataframe method and checks if any negative values are contained within these using a conditional statement. First "any()" function returns a boolean value for each column, the second one checks for any "true" values.

```
numeric_columns = df.select_dtypes(include=['number']).columns
if (df[numeric_columns]<0).any().any():
    print('There are negative values in the Dataset.')
else:
    print('\n Data cleaning completed.')
```

1.3 Extending the dataset

In order to generate and present information from the dataset, we can consider adding additional columns which may give us further useful information. A clear example would be adding a “Total” column to show the total affluence of patients amongst different reasons of attendance.

```
df["Total"] = df['Road traffic accident '] + df[
    'Sports injury '] + df["Deliberate self-harm "] + df["Assault "] + df['Not known ']
```

Additionally, to provide information relevant to human resource management, a variable encompassing different time frames representative of the traditional 8 hour shift split (from 8h to 16h, from 16h to 00h and from 00h to 8h) could enrich our analysis. For that reason, the hour variable was transformed from a string to an integer. Furthermore, a function that assigns a shift category (Morning, Evening or Night Shift) according to the mentioned hour values was defined and its application to the hour column added to the dataframe as a new column.

```
df['Hour of arrival'] = (
    df['Hour of arrival'].astype(str).str.replace(':', '').astype(int))/10000

def shift(hour):
    if 0 <= hour < 8:
        return 'Night shift'
    elif 8 <= hour < 16:
        return 'Morning shift'
    else:
        return 'Evening shift'
```

```
df['Shift'] = df['Hour of arrival'].apply(shift)
```

Note: There is evidence of both 12 and 8 hour shifts patterns in hospitals’ workforce (see references), but using three categories provides more granularity.

Finally, the “Weekday” and “Shift” columns were transformed to categorical values and ordered for clarity on our visualizations.

```
weekday_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
shift_order = ['Night shift', 'Morning shift', 'Evening shift']
df['Weekday'] = pd.Categorical(df['Weekday'], categories=weekday_order, ordered=True)
df['Shift'] = pd.Categorical(df['Shift'], categories=shift_order, ordered=True)
```

1.4 Data Analysis

With the df.head() method the format of the final cleaned and extended data frame can be visualized. We can observe filled missing values and the additional columns.

	Weekday	Hour of arrival	Road traffic accident	Assault	Deliberate self-harm	Sports injury	Not known	Shift	Total
0	Monday	0.0	101	127	44.0	36.0	230	Night shift	538.0
1	Monday	1.0	59	92	46.0	8.0	185	Night shift	390.0
2	Monday	2.0	37	84	36.5	9.5	150	Night shift	317.0
3	Monday	3.0	32	45	27.0	11.0	118	Night shift	233.0
			24	56	16.0	14.0	93	Night shift	203.0

```
display(df.head())
```

Moreover, the df.describe() method displays the **statistical summary**, which provides basic information about the dataset variables (tabulate() function was used for a clearer format).

```
print(tabulate(df.describe(), headers='keys', tablefmt='fancy_grid'))
```

	Hour of arrival	Road traffic accident	Assault	Deliberate self-harm	Sports injury	Not known	Total
count	168	168	168	168	168	168	168
mean	11.5	181.101	104.292	35.8214	183.417	212.113	716.744
std	6.94288	104.923	45.0202	15.9436	141.497	130.949	332.893
min	0	10	13	7	6	48	115
25%	5.75	78	80.75	24.375	28.25	110	394.75
50%	11.5	209.5	104	36.75	226.5	173.5	806
75%	17.25	264	127.25	48	282.5	286	981.5
max	23	365	269	81	558	619	1368

“Road Traffic Accident”, “Sports Injury” and “Not known” reasons for attendance have the highest **means** and **medians** of admissions per hour and “Deliberate self-harm” has the lowest. We can also see sport injuries and unknown reasons have had higher **maximums**, which indicates higher spikes observed. Finally, we can also observe that the types of admissions with highest **standard deviation** relative to their means were the same ones with higher means and medians. That means that the **most frequent causes** for attendance are also the ones with **more variability**.

2. Insights and Visualizations

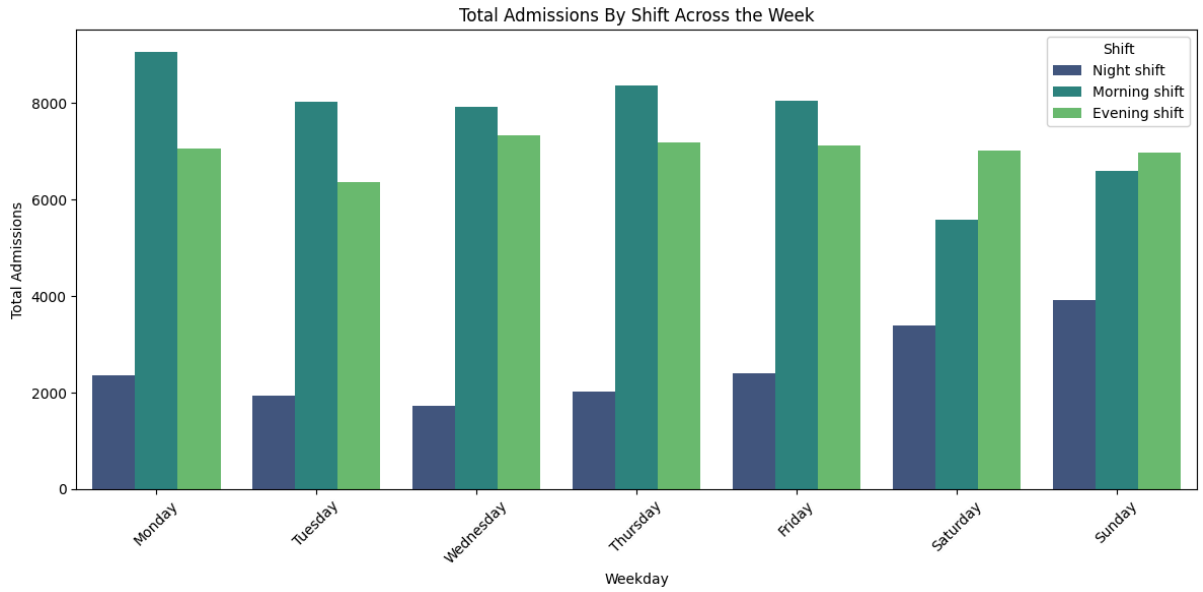
2.1 Total admissions by shift across the week

For the first insight, I grouped the data of the weekday, shift and total columns, which could be useful when assessing total need of shared human resources (nurses, ambulance drivers, etc) by shift across the week. This code creates a new dataframe from the sum of totals for each combination of weekday and shift:

```
shift_totals = (df.groupby(['Weekday', 'Shift'])['Total'].sum().reset_index())
```

From this new dataframe, we can visualize the results of total admissions per shift across the week using a **barplot**. The method used was from the seaborn library and titles and legend were added for better clarity of the information presented.

```
plt.figure(figsize=(12, 6))
sns.barplot(data=shift_totals,x='Weekday',y='Total',
            hue='Shift',palette='viridis',dodge=True,width=0.8)
plt.title('Total Admissions By Shift Across the Week')
plt.ylabel('Total Admissions')
plt.xlabel('Weekday')
plt.legend(title='Shift')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Interpreting the barplot, we can easily observe that **total admissions per shift patterns vary between working days and weekends**. From monday to friday, the morning shift is the one with most admissions (although close with evening shifts), but during the weekend evening shift hours accumulate higher admissions. It is also notable, besides being lower across the week, the increase of night shift admissions during the weekend.

For that reason, when planning efficient staffing, the NHS should consider increasing the workforce during the day (specially on morning shift hours) on working days. During the weekend, it should increase the evening and night workforce to adapt to the changing admissions pattern.

2.2 Admissions by reason of attendance and day of the week

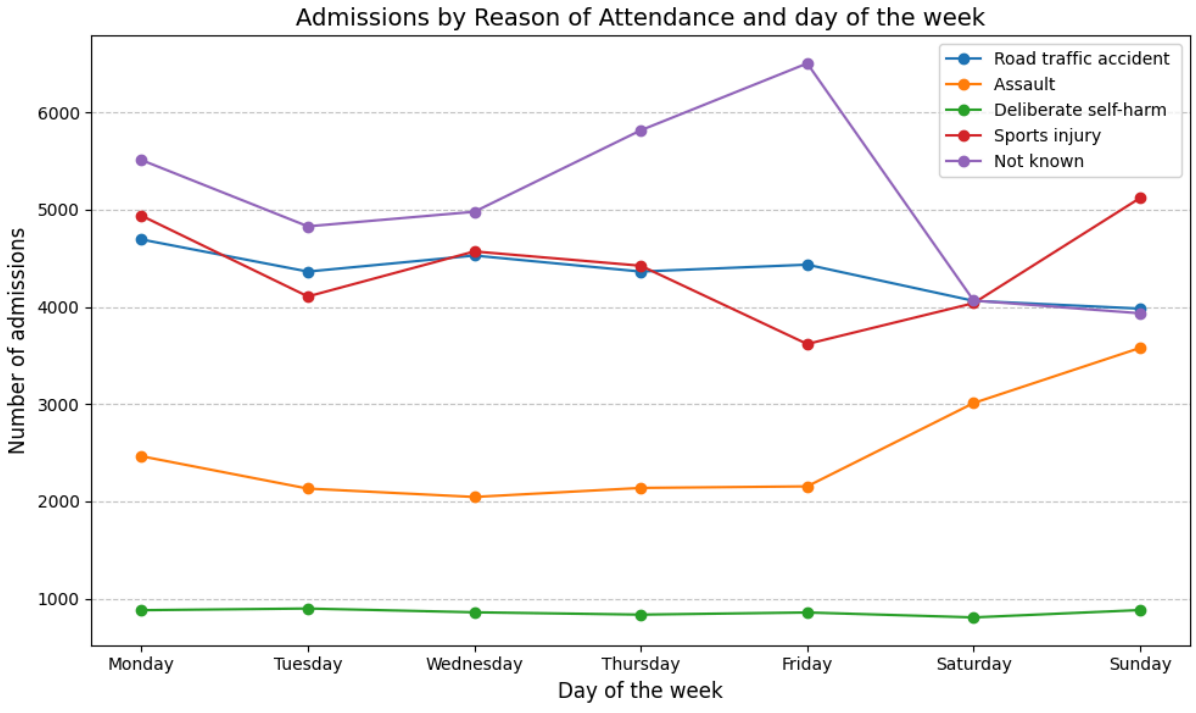
For the second insight, I considered analyzing how different reasons of attendance behave across the week. To do so, I created a new data frame grouping the total admissions for each reason by weekday:

```
grouped_data = df.groupby('Weekday')[['Road traffic accident ',
            'Assault ', 'Deliberate self-harm ', 'Sports injury ', 'Not known ']].sum()
```

A **line plot** with all the reasons for attendance should be useful to visualize this dataframe. The following code uses a for loop to iterate through the different reasons for attendance and plotting them in the same visualization using the new dataframe. Labeling, title and legend were also added for more clarity.

```
plt.figure(figsize=(10, 6))
adtype_columns = ['Road traffic accident ',
                  'Assault ', 'Deliberate self-harm ', 'Sports injury ', 'Not known ']
for column in adtype_columns:
    plt.plot(grouped_data.index, grouped_data[column], label=column, marker='o')

plt.title('Admissions by Reason of Attendance and day of the week', fontsize=14)
plt.xlabel('Day of the week', fontsize=12)
plt.ylabel('Number of admissions', fontsize=12)
plt.legend()
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



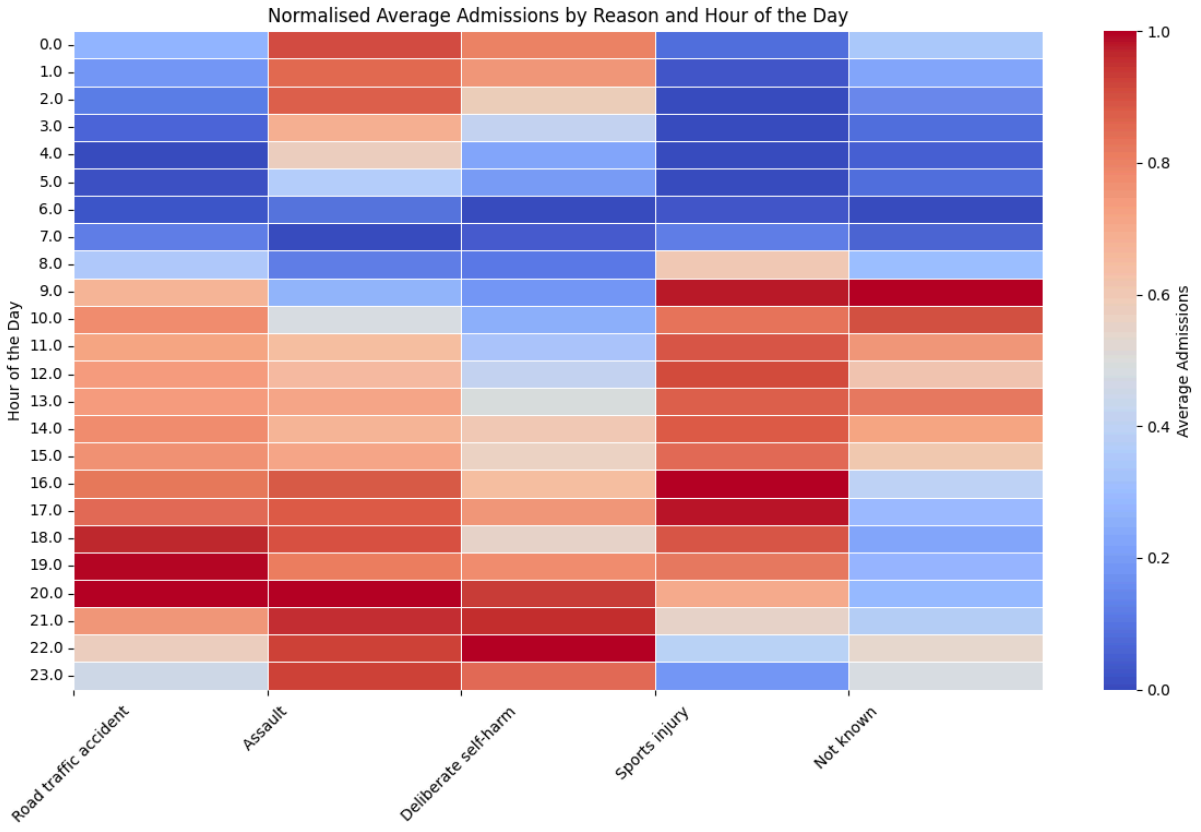
We can observe that **some reasons for attendance** number of admissions **vary significantly across weekdays**, showing also different behaviours.
Deliberate self-harm admissions stay steady across the week and assault admissions don't vary much during working days. All the other variables show significant overall fluctuations.
Assault admissions increase on weekends and peak on Sundays, unknown reasons increase significantly on Thursdays and specially on Fridays and sports injuries are high on Monday and peak on Sunday.
These insights help the NHS identify high attendance days for each reason. For example, assault admissions peaking on weekends or sports injuries on Sundays suggest targeted staffing needed during these periods.

2.3 Average admissions by reason and hour of the day

We saw from the summary that reasons with higher admissions tend to have greater variability. This is evident in 'Unknown' and 'Sports Injury' admissions (see previous insight), which fluctuate across weekdays. However, 'Traffic Accidents' show less weekday variation, suggesting their variability might be more pronounced within daily hours.
For that reason, to generate a third insight showing variability during the day, I decided to group the data by average admissions by reason and hour.

```
hourly_data = df.groupby('Hour of arrival')[[
    'Road traffic accident ', 'Assault ', 'Deliberate self-harm ',
    'Sports injury ', 'Not known ']].mean()
```

Since there are many hours during the day, a **heat map** could help visualizing this dataframe in an understandable format (numbers were removed for simplicity of visualisation).



I **normalized** the data using a lambda function (the normalization function) to visualize variability relative to scale. This improves the heatmap by revealing patterns in reasons with both high and low admissions. For that reason the heat map's scale is from 0 to 1.

$$X_{normalized} = \frac{(X - X_{minimum})}{(X_{maximum} - X_{minimum})}$$

```
normalized_data = hourly_data.apply(
    lambda x: (x - x.min()) /
     (x.max() - x.min()), axis=0)
```

Analysing the visualization we can conclude that **high-risk hours vary by admission types** with **different patterns**. With more detail, we may observe that:

- Traffic accidents increase from 9h and peak during late afternoon and early evening (16h–20h).
- Assault peaks in the late evening and night (20h–03h), likely linked to social or nightlife activities.
- Deliberate self-harm admissions concentrate during the late evening and night hours (19h-2h), suggesting a period of emotional vulnerability.
- Sports Injuries are high during the day with peaks at 9h and early evening (16h-18h).
- Unknown reasons peak in the morning (9h-11h) and start decreasing through the day.

This data is useful when assessing allocation of specific resources (such as specialists or material dedicated for specific treatments) for each admission type throughout the 24 hours of the day. For example, as admissions for assault and self-harm are still relatively high at night, night hours require a higher proportion of psychiatrists, psychologists and emergency medicine physicians.

2.4 Bringing insights together

Even though these three insights are useful for themselves, the NHS could obtain better information for their decision making, analysing them together and identifying peaks of admissions by day, shift, reason, and hour. For example, combining insights reveals the need for more emergency physicians during weekend evening and night shifts, when assaults peak (both for the day and hour) and represent a higher proportion of total admissions. This integrated approach could bring better resource planning, reduced wait times, and improved patient care.

3. Proposal of data collection and program enhancement

The current program provides useful insights from the dataset but it **only contains observations from a specific week,** which does not allow us to identify general patterns (as this observed week may be altered by specific unknown variables), which would arise from collecting the same data for more weeks. Additionally, as the unknown reasons for attendance are a large proportion of total admissions and seem to show some patterns, the NHS should consider enforcing better data collection policies or adding categories to disaggregate and understand its hidden pattern.

The program already categorizes admissions based on shifts, so predictive models could be developed to forecast shift patterns more accurately. Furthermore, the program's shift-based categorization can be iteratively modified to look for optimal shift hour distribution. It is also possible to integrate this categorization into more detailed staffing predictions (using more advanced techniques) by combining real-time data on admissions with staffing metrics to continuously suggest adjustments.

From a software engineering perspective, adopting version control (for example, Git) would facilitate collaboration, track changes, and ensure code integrity. Moreover, an agile development approach arising from this basic code, extending and improving step by step from its core ideas, would allow the program to adapt to changing requirements and new ideas rather than if started with a very detailed and complete approach since the beginning (more difficult to change or adapt during development).

4. AI usage

The AI used to help in the code development was **ChatGPT**.

The first useful solution that it provided was an explanation of lambda functions when asked for how to transform the asterisks to NaN values in a fast way. Understanding lambda functions allowed me to perform quick transformations (as from "*" to NaN) and apply formulas to generate data frames such as the normalized hourly data.

Moreover, whenever I had a specific idea without knowing how to execute it (as filling NaN with the mean between the previous and posterior observation), the AI provided either standard built-in functions or methods easy to apply (as interpolate or rename methods) or complex code which I wouldn't understand and therefore not use.

Finally, the most useful application that I found was asking AI to set visualisation code arguments, since I didn't have specific knowledge of how each type of visualisation could be setted up (specially the heatmap).

Even though ChatGPT was a good complementary tool for some purposes, for other applications its performance was disappointing and leading to errors. I concluded that the key to using AI efficiently is knowing its limitations and bearing in mind all the information it does not have access to (context of the dataset, environment and variables, etc.).

5. References

Module slides and seminar materials.

Mober, Samra Marwi Ahmed, et al. "The Impact of Shift Work on Nurses' well-being and Patient Care Excellence in Inpatient Settings." British Journal of Nursing Studies 4.2 (2024): 95-105.

Nwoke, Judith. (2024). Healthcare Data Analytics and Predictive Modelling: Enhancing Outcomes in Resource Allocation, Disease Prevalence and High-Risk Populations. International Journal of Health Sciences. 7. 1-35. 10.47941/ijhs.2245.