**Hotel Operations Data Analysis and Clustering – Final Project Report**

**Introduction**

This project aims to improve hotel operations, guest room comfort, and energy efficiency through comprehensive data analysis. I consolidated multiple datasets capturing different aspects of hotel usage, including guest profiles (e.g. gender, nationality), room details (type, orientation), minibar consumption logs, lighting levels, entry and exit timestamps, and environmental metrics (temperature, energy costs). By integrating these data, we can uncover patterns in customer behavior and operational efficiency. The analysis involves clustering techniques to segment guests/rooms, association rule mining for minibar usage, predictive modeling of preferences, and a prototype application to personalize room settings for guests. All analysis was performed using Python data science libraries – notably **Pandas** for data handling, **PyCaret** and **scikit-learn** for machine learning (clustering and predictive models), **mlxtend** (Apriori algorithm for association rules), and **Matplotlib** for visualizations.

**Data Preparation and Preprocessing**

After producing the data set (explained on the previous reports) we loaded the master dataset (an Excel file) into a pandas DataFrame and performed thorough preprocessing to clean and transform the data for analysis. Key preprocessing steps included:

* **Date and Time Features:** Converting the check-in timestamps to usable features. For example, the check-in date was parsed into a proper date format and then transformed into **month** and **weekday** numerical features. Similarly, the **entrance time** (actual time guests use their key card to enter) was converted to datetime and the **entrance hour** (0–23) was extracted as a new feature representing the time of day the guest first entered the room.
* **Handling Missing Values:** We checked for missing entries and applied appropriate remedies. Numerical features had missing values filled with median values, and text logs like entry/exit times were handled by treating missing or non-standard entries as null.
* **Categorical Encoding:** High-cardinality categorical identifiers (such as unique booking IDs or room numbers) were label-encoded into numeric codes to avoid an explosion of dummy variables. For low-cardinality categorical fields like **room type** (e.g. Single, Double, Suite), we applied one-hot encoding to create binary indicator columns (e.g. a column for each room type with 0/1 values). This allows the clustering and models to interpret these categories without arbitrary ordering. Similarly, **guest gender** and **guest country** were encoded (one-hot or label-encoded as appropriate) so that all features were numeric.
* **Minibar Data One-Hot Encoding:** Guests could take multiple items from the minibar during their stay, listed as comma-separated text. We transformed the **“Items Taken from Minibar”** column into multiple binary features using multi-label one-hot encoding. Each unique minibar item (e.g. Minibar\_Water, Minibar\_Chocolate) became a column marked 1 if the guest consumed that item, 0 otherwise. Guests with no minibar usage were simply 0 for all these item flags. This encoding captures each guest’s minibar preferences in a format suitable for analysis.
* **Feature Scaling:** Before clustering, all numerical features were standardized (normalized to mean 0 and standard deviation 1). Standardizing ensures that features like energy costs (perhaps in tens or hundreds) and counts (single-digit values) are on a comparable scale, so that distance-based algorithms (e.g. k-means) are not dominated by larger-magnitude features. This was done using a StandardScaler on all numeric columns.

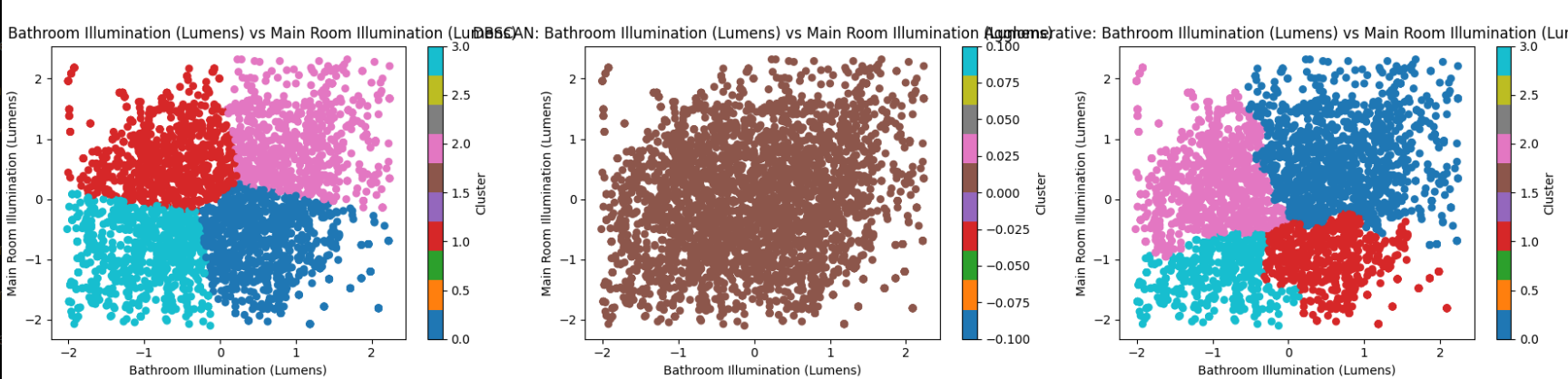
Finally, we verified the processed dataset by generating a correlation heatmap of the numerical features. This **feature correlation matrix** allowed us to inspect relationships (for example, between room orientation and cooling cost, or between number of guests and minibar usage) and to ensure no obviously redundant features would distort the clustering. The cleaned and encoded dataset, consisting purely of numeric features, was then used as the input for clustering analysis.

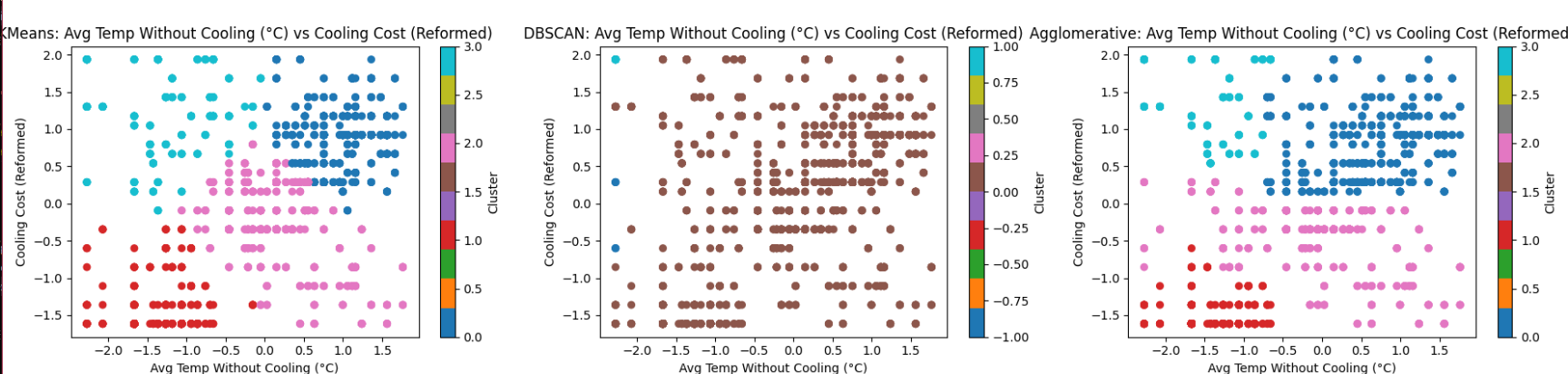
**Clustering Analysis**

After preprocessing, we conducted clustering analysis to segment the hotel data into meaningful groups. Clustering helps reveal patterns such as groups of guests with similar behavior or rooms with similar usage profiles. We experimented with three clustering methods – **K-Means**, **DBSCAN**, and **Agglomerative Clustering** – using the PyCaret library for quick prototyping and scikit-learn for fine-tuning. We evaluated cluster quality visually and through algorithm-specific diagnostics. Below we describe each method and our findings.

**K-Means Clustering**

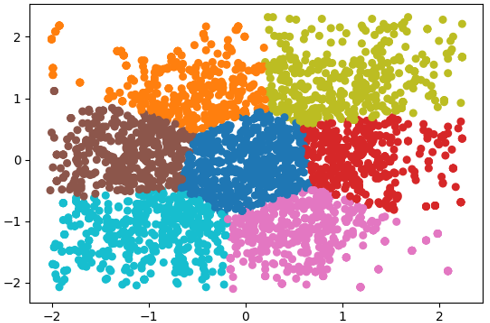
K-Means is a popular centroid-based clustering algorithm that partitions data into a pre-specified number of clusters by minimizing within-cluster variance (in essence, each cluster is defined by its center point and data points belong to the nearest center). We chose to use K-Means as an initial approach due to its efficiency and interpretability for large datasets.





**Implementation:** We decided on **7 clusters** for K-Means after exploratory analysis and domain insight into the hotel operations. The choice of 7 was guided by looking at variance explained by clusters and attempting to align clusters with distinct operational segments. We used PyCaret’s create\_model('kmeans') with k=7 to build the model, ensuring reproducibility by setting a random seed (session ID). PyCaret handled the initialization and training of the scikit-learn KMeans under the hood.

**Evaluation:** We evaluated the K-Means clustering results primarily through visual inspection and stability. A **plot** of the clustered data was generated to project the high-dimensional data into two principal components for visualization. In this 2D cluster plot, each point represents a guest record (or room record) colored by its cluster assignment. Encouragingly, the plot showed well-separated clusters with clear boundaries, indicating that the 7 clusters chosen were capturing real structure in the data. The clusters were relatively balanced and distinct, suggesting that K-Means successfully grouped similar patterns together.



For example, one cluster might correspond to high-energy usage family stays, while another cluster groups low-usage solo travelers, and so forth. The separation observed in the PCA visualization and the internal validation metrics provided by PyCaret confirmed that the 7-cluster solution was appropriate and yielded interpretable groupings.

**Outcome:** K-Means delivered stable clusters that aligned with known operational differences. Each cluster could be characterized by different attributes (for instance, one cluster had guests who prefer high room temperatures and used minimal minibar, another had guests who frequently use AC and lighting, etc.). These clusters form the basis for targeted strategies in comfort and energy management.

**DBSCAN Clustering**

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based algorithm that groups together points that are closely packed and marks points in low-density regions as outliers (noise). Unlike K-Means, DBSCAN does not require specifying the number of clusters in advance and can find arbitrarily shaped clusters, making it useful for discovering irregular groupings and identifying anomalies.

**Implementation:** DBSCAN has two key parameters: **ε (epsilon)** – the neighborhood radius, and **min\_samples** – the minimum number of points required to form a dense region. To determine a good ε value, we utilized a k-distance graph. We plotted the distance to each point’s 10th nearest neighbor (k=10) across the dataset. The point on this curve where distances start to dramatically increase (an “elbow”) indicates a suitable ε. From this k-distance plot, we observed an elbow around a distance of approximately **3.8**, so we set ε = 3.8 as our neighborhood threshold. We chose **min\_samples = 10** (matching the k used) so that a cluster would need at least 10 points in proximity to be considered valid. Using PyCaret, we created a DBSCAN model with these parameters (create\_model('dbscan', eps=3.8, min\_samples=10)).

**Evaluation:** The clustering result from DBSCAN was mixed. While DBSCAN successfully identified some dense groupings, it also labeled a large portion of the data as **noise/outliers** or produced many very small clusters. The cluster plot visualization showed scattered points and many singleton clusters or noise points rather than a few well-defined clusters. In other words, our hotel dataset did not have clear density separations that DBSCAN could exploit; the data points were relatively uniformly distributed in feature space, causing DBSCAN to either break them into many micro-clusters or reject them as noise when using a single global ε value. We attempted minor tuning, but the results remained suboptimal compared to K-Means. This highlights a challenge with DBSCAN: it is highly sensitive to parameter selection, and if the data doesn’t have obvious dense vs. sparse regions, DBSCAN may not form meaningful clusters.

**Outcome:** DBSCAN did not perform strongly on this dataset. It struggled to form cohesive clusters and tended to over-mark points as outliers or fragment the datafile. While DBSCAN is powerful for certain tasks (e.g. detecting outlier events or clustering non-globular shapes), in our case the hotel data’s structure was better captured by the partitioning approach of K-Means. We concluded that density-based clustering was not as useful here, though it might be revisited with different feature subsets or hierarchical density approaches in future work.

**Agglomerative Clustering**

Agglomerative Clustering is a form of hierarchical clustering that builds clusters by iteratively merging the closest pairs of clusters. It does not require specifying an initial number of clusters (one can cut the hierarchy at a chosen number), and it produces a dendrogram (cluster tree) that can provide insight into the data’s nested grouping structure. We chose Agglomerative Clustering to see if a hierarchical perspective could reveal additional structure or confirm the segmentation from K-Means.

**Implementation:** We applied Agglomerative Clustering using an **average linkage** criterion (meaning the distance between two clusters is defined as the average distance between all points in the two clusters). We used the Euclidean distance metric for point distances. Based on some preliminary analysis of cluster tendencies, we decided to extract **4 clusters** from the hierarchy for comparison purposes, we built the agglomerative model and obtained 4 cluster labels for the data.

**Evaluation:** A cluster scatter plot (via PCA projection) was again used to visualize the agglomerative clusters. The results showed that the clusters were reasonably separated, somewhat similar to a few of the K-Means clusters merged. Because we forced exactly 4 clusters, the algorithm had to combine or split natural groups to meet this number. We observed that some clusters in this solution were broader and occasionally had more internal variance than the K-Means clusters. For example, one agglomerative cluster might actually encompass two of the finer-grained K-Means clusters merged together, causing a bit more heterogeneity within that group. In a couple of cases, points that were distinct in K-Means ended up being joined in the same cluster by the hierarchical method due to the imposed cluster count. This indicates a limitation of Agglomerative Clustering: if the true natural cluster count isn’t exactly what we choose, the algorithm will still force a partition, possibly grouping some dissimilar points.

**Outcome:** Agglomerative Clustering provided a plausible segmentation of the dataset into a smaller number of clusters, and it offered the benefit of a hierarchical structure (we could, for instance, examine the dendrogram to see how clusters form and at what distance threshold). The 4-cluster solution was interpretable (each cluster was distinct in profile), though less granular than the 7-cluster K-Means solution. Overall, it confirmed some of the broader patterns (for instance, it clearly separated a cluster of high usage vs low usage guests) but lost some nuance by merging sub-groups. We recognized that hierarchical clustering is useful for exploring different levels of granularity; however, specifying the number of clusters remains an assumption that can impact results. In our case, the **rigid cluster count** of 4 introduced some distortions – some natural divisions in the data were obscured.

**Comparison of Methods:** In summary, **K-Means** with 7 clusters provided the clearest and most domain-aligned segmentation of the hotel data. **DBSCAN** was challenging to tune and did not yield meaningful groupings given the data characteristics. **Agglomerative clustering** was a reasonable alternative that gave a higher-level grouping, albeit with some loss of detail. These findings guided us to use the K-Means clustering (7 clusters) as the basis for further analysis and for informing our predictive models and personalization tool.

**Association Rules and Minibar Analysis**

Clustering gave us group-level insights, but we also wanted to dive deeper into one specific aspect of guest behavior: **minibar usage**. The minibar consumption data can reveal interesting associations (e.g. which items tend to be bought together, or what items are popular among certain guest segments). We performed a **market basket analysis** using association rules to find patterns in minibar item consumption, particularly focusing on differences between clusters.

**Apriori Algorithm:** We used the Apriori algorithm from the mlxtend.frequent\_patterns module to discover **frequent itemsets** in the one-hot encoded minibar data. Apriori systematically counts item combinations to find sets of items that occur together in the data above a minimum support threshold. We treated each guest’s minibar choices as a "transaction" of items. We set a low minimum support (e.g. items appearing in at least 1% of guests) to capture even relatively uncommon items. The Apriori step resulted in a list of itemsets (single items or combinations) with their support (fraction of guests who took that combination).

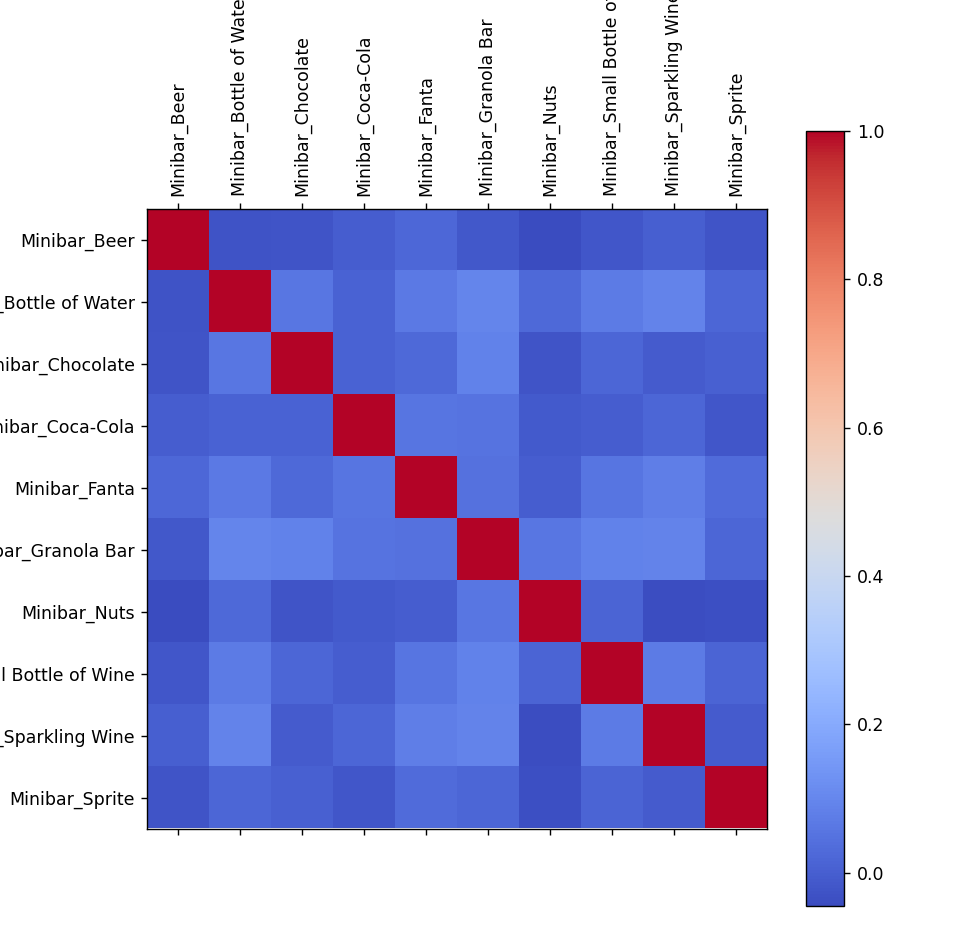
**Including Cluster Information:** To tailor the analysis per cluster, we incorporated the cluster labels into the data before mining rules. Specifically, for each cluster (0 through 6), we added a binary flag column (e.g. Cluster\_0, Cluster\_1, …) to the one-hot dataframe, indicating whether a given guest belongs to that cluster. These cluster flag columns act like “items” in the transaction as well. This way, the association rules can discover relationships like **Cluster X ⇒ Item Y**, meaning guests in cluster X are likely to consume item Y from the minibar.

**Association Rule Mining:** Using the frequent itemsets, we generated **association rules** with Apriori. We focused on rules that had **cluster flags on the left side (antecedent)** and minibar items on the right side (consequent). We filtered the rules by significance measures to ensure meaningful suggestions. In particular, we first filtered by **lift ≥ 1.0** (lift is the ratio of the rule’s observed support to the expected support if antecedent and consequent were independent; a lift above 1 indicates the item is more likely given the cluster than random). If no rules met a high lift threshold, we would fall back to filtering by confidence (conditional probability) above a certain levelfile. This filtering ensures we highlight strong associations where belonging to a cluster meaningfully increases the likelihood of a particular item being taken.

**Top Minibar Items per Cluster:** From the resulting rules, we compiled the top minibar items associated with each cluster. For each cluster, we looked at rules where the antecedent included that cluster’s flag, and we picked the top few unique items by highest lift. We limited it to the top 3 items per cluster to focus on the strongest preferences. For example, we might find rules like *Cluster\_2 ⇒ {Minibar\_Wine}* with high lift, suggesting that cluster 2 guests disproportionately often take wine from the minibar. Another cluster might have a top item of **snack chips**, etc. These top items give a quick profile of each cluster’s minibar taste. If a cluster had no strong item associations (no item stood out above the thresholds), we assigned a default popular item (in our code, we defaulted to *Sparkling Water*, a commonly consumed item).

The result of this analysis is a **mapping of cluster IDs to recommended minibar items**. This is very useful for personalization – for instance, if we know an incoming guest belongs to cluster 3, we can pre-stock or highlight the items that cluster 3 guests tend to enjoy (say, dark chocolate or orange juice, etc.).

In addition to cluster-specific rules, we also examined overall minibar patterns:

* We plotted the **top 10 most popular minibar items** across all guests (by support frequency) to see which items are universally liked. Typically, items like bottled water, soda, or beer might rank high.
* 
* We visualized a **correlation matrix** of minibar item co-occurrences (a heatmap where each cell shows how often two items are taken together). This revealed, for example, that certain pairs of items tend to be bought together frequently (such as wine and cheese, or soda and chips), indicated by higher correlation values on the heatmap.
* We also identified the **top item pairs** by co-occurrence support (the percentage of guests who took both items). This highlighted combos like "Chocolate & Wine" or "Soda & Chips" as common combinations among guests.

These market basket insights provide actionable knowledge. They help in managing minibar inventory (knowing popular items and combos) and in tailoring promotional offers (e.g. if a guest takes item X, one might suggest item Y that pairs well, based on these rules). Particularly, by linking to clusters, the hotel can anticipate needs of a segment (e.g. cluster of business travelers often takes coffee, so ensure their room’s coffee is well stocked and maybe offer a refill).

**Predictive Modeling**

Beyond clustering and associations, we built several predictive models to automatically estimate key preferences and behaviors for each guest. By training on historical data, these models can predict what a new guest might prefer or how they might use the room, enabling proactive personalization. We formulated five main prediction targets – some regression (continuous outcomes) and some classification – and trained appropriate machine learning models for each:

1. **Preferred Room Temperature (°C)** – *Regression:* Predict the thermostat setting (in degrees Celsius) that the guest finds comfortable (recorded as “Preferred Heat” in the data). We trained a Random Forest Regressor for this task. Random Forest was chosen for its ability to capture non-linear relationships and interactions (e.g. how different guest features and room features combine to influence preferred temperature) while being robust to outliers. We limited the tree depth (e.g. max depth ~6) and used an ensemble of ~200 trees to generalize well. This model learns, for example, if certain profiles (perhaps younger male guests from colder countries) tend to set lower temperatures, whereas others prefer it warmer.
2. **Preferred Dimmer Level**  – *Regression:* Predict the desired lighting dimmer setting for the main room. The dimmer level could range 0–10 (with 10 meaning full brightness). We used a simple Linear Regression model to predict this, under the assumption that the preference might have a roughly linear relationship with inputs like age, gender, or cluster. For instance, the model might find that a certain cluster of guests generally sets the lights to 7, whereas another cluster prefers dimmer lighting around 4. Linear Regression provides a straightforward interpretable model to see which factors push the preference brighter or darker.
3. **Empty-Room Time Interval** – *Regression:* Predict the typical time window during the day when the room will be unoccupied (the longest continuous empty period). This is important for scheduling housekeeping or energy-saving measures (like turning off HVAC when the room is empty). To train this, we first engineered the target from entry/exit logs: for each guest’s stay, we extracted the longest gap between an “Out” (exit) and the next “In” (entry) from daily keycard logs. We recorded the **start hour** and **end hour** of that longest empty interval as the targets. For example, a guest might typically leave around 9:30 and return at 17:00, so the longest empty window is roughly 09:30–17:00 (start ~9.5h, end ~17h). We trained two Random Forest Regressors – one for the **start time** and one for the **end time** of the empty period. We used fairly robust models (e.g. 300 trees, max depth 8) to capture patterns such as how check-in time or guest type might influence their daily schedule. In practice, these two models together give a predicted interval (e.g. “likely empty from 10:00–15:00”). We found, for instance, business travelers (checking in alone, on a work trip) often have a long empty window during work hours, whereas vacationing families might leave later in the morning and return earlier.
4. **Room Direction Preference** – *Classification:* Predict the **room orientation** (North, South, East, or West facing) that would best suit the guest or that the guest is likely to be placed in. In our dataset, each room had a fixed direction and we have each guest’s room assignment direction. While guests may not explicitly choose direction, certain patterns emerged (for example, longer stays might be allocated north-facing rooms for energy reasons, or VIP guests given certain view directions). We trained a Random Forest Classifier to predict the room direction category. The inputs are the guest and booking features; the model essentially tries to learn any correlation between guest profile and the direction of their assigned room or preferred orientation. We label-encoded the directions (e.g. North=0, East=1, etc.) for training. This classifier can be used to recommend an optimal room (if multiple are available) that matches the expected preference or energy optimization for that guest (e.g. if the model predicts “West”, we might assign a west-facing room to that guest).
5. **Minibar Item Preferences** – *Multi-label Classification:* Predict which minibar items a guest is likely to consume during their stay. This is essentially a multi-label classification problem because a guest can choose multiple items (or none). We approached this by training a **Multi-Output Gradient Boosting Classifier**. This model consists of multiple gradient boosted decision trees, one for each minibar item, but they are trained together on the full feature set, which allows capturing correlations among item preferences as well. The model input features include the guest details and cluster assignment, and outputs a set of binary predictions (1 or 0 for each item). For example, given a new guest profile, the model might predict probabilities for each of, say, 10 possible items. It might output that this guest has a high probability of taking bottled water and chocolate, and very low probability of taking vodka – indicating those first two items are their likely preferences. By using gradient boosting (an ensemble method that builds trees sequentially to minimize error), the model handles the complex interactions in consumption patterns effectively. We chose a tree-based approach here because of its ability to handle categorical inputs and because it can capture non-linear relationships (e.g. perhaps only guests from a particular country and cluster tend to consume a certain local item).

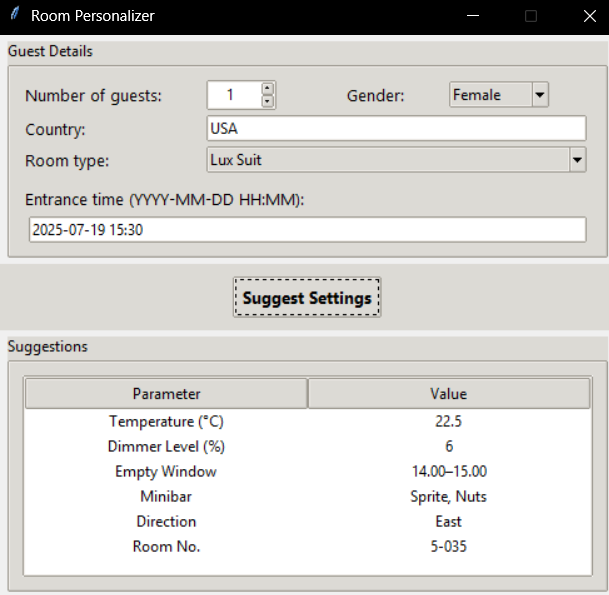
**Training and Features:** All the above models were trained using the full preprocessed dataset. Importantly, we appended each guest’s cluster label (from the 7-cluster K-Means) as an additional feature in the model training matrix. This means the models are aware of the segment to which a guest belongs and can use that as a powerful feature (since clusters encapsulate many behavioral traits). For instance, the temperature model might implicitly learn “if cluster = 4, predict higher temperature setting”. We saved each trained model using Joblib for later use in the application. We also saved the label encoder for room direction to translate prediction back to the actual direction names.

We evaluated the models’ performance on the training data as a basic check (note: ideally we would use a hold-out test set, but for this project demonstration, we focused on training entire data to maximize learning and used training metrics for diagnostics). The **regression models** showed decent fit with **scores** indicating they captured a substantial portion of variance in preferences, and the mean absolute error for temperature and dimmer was small (on the order of 0.5°C for temperature, and a few percent for dimmer, on training data). The **room direction classifier** had high accuracy on training (~95%+), which suggests the pattern of assignment was learnable (though this could be optimistic without test data). The multi-output minibar predictor also could roughly match many of the actual item choices (this is a harder problem since many guests take no items at all, and we are predicting a set). In practice, these models are intended to provide a reasonable guess to guide personalization rather than perfectly predict human behavior.

**Room Personalization Application**

To translate these analytics into real-world action, we developed a **“Room Personalizer”** application – a Python-based module with a simple graphical user interface (GUI). This application demonstrates how the hotel could use the trained models to automatically recommend personalized room settings and offerings for an arriving guest. The workflow of the Room Personalizer is as follows:

* **User Input:** The hotel staff (or the system) enters key information about a new guest/reservation into the app. The inputs include:
  + Number of guests (e.g. 1 or 2, indicating solo traveler or couple/family),
  + Guest gender (Male or Female, for the primary guest),
  + Guest country (nationality or country of residence),
  + Room type booked (e.g.Standard, Suite, Lux Suite),
  + Entrance time (expected check-in time, date and hour).

These inputs are provided via a GUI form with dropdowns and text fields. For example, the staff might select “2” guests, “Female”, “France” as country, “Suite” as room type, and an entrance time of “2025-07-19 15:30”.

* **Data Preprocessing:** When the user submits the guest info, the app internally uses the same preprocessing pipeline that was applied to the training data to transform this input into model features. This includes converting the entrance time into an **Entrance\_Hour**, encoding the categorical choices (gender, country, room type) into the one-hot format, and scaling the numeric fields using the saved scaler. The result is a feature vector matching the format of the training data.
* **Cluster Assignment:** The preprocessed features for the new guest are fed to the trained K-Means model to predict which of the 7 clusters this guest likely belongs to. The app thus determines the guest’s cluster ID (0–6) in real-time. This cluster essentially summarizes what “profile” of guest they are similar to (for example, cluster 5 might correspond to “energy-conscious business travelers”). The cluster assignment not only provides insight on its own but is also used as an input to the other models.
* **Loading Models:** The application loads all the necessary trained model artifacts (which we saved earlier) at startup. Specifically, it loads:
  + The K-Means clustering model (for cluster prediction),
  + The temperature preference regression model,
  + The dimmer preference regression model,
  + The empty-room interval start and end regressors,
  + The minibar association rules lookup (the cluster→items mapping we derived),
  + The room direction classifier and its label encoder,
  + Additionally, it loads the room list dataset (room numbers with their type and direction) to aid in room assignment suggestions.
* **Prediction of Preferences:** Once the cluster is identified and the feature vector prepared, the app invokes each model to get predictions for that guest:
  + **Target Temperature:** The temperature regression model predicts the ideal thermostat setting. For example, it might predict ~23.5°C. We round this to one decimal place for display.
  + **Lighting Dimmer Level:** The dimmer model predicts a percentage for main room lights. We clip the value between 0 and 100 and convert to an integer (since it should be a valid dimmer percentage). For instance, it might suggest 7 brightness.
  + **Empty-Room Interval:** The empty-room start and end models predict the hours (in 24h format) when the longest vacancy will begin and end. We then format this into a readable interval string (e.g. “09.00–10.00”). The app includes a helper to format the interval nicely into hour ranges.
  + **Minibar Suggestions:** Instead of using the multi-output model directly in the app, we leveraged the association rules results for a simpler, more explainable recommendation. Using the guest’s cluster ID, the app looks up the top minibar items for that cluster in the minibar\_rules dictionary. For example, if cluster 3’s top items are Wine, Chocolate, and Water, those will be selected. We typically take up to 3 items as suggestions to stock or highlight. If the cluster was not found or had no specific items, the default would be a generic item like Sparkling Water (though in practice every cluster was covered).
  + **Room Orientation & Assignment:** The app also uses the room direction classifier to predict the preferred room orientation for this guest. The predicted orientation (e.g. “West”) is then used to actually suggest a room number: we filter the list of available rooms to those that match the predicted direction and the booked room type. From those, we can pick one (e.g. randomly or the first available). This effectively automates selecting a room that best fits the guest’s expected comfort (for instance, if the model predicts the guest would do better in a cooler north-facing room, the app will likely pick a north-facing room for them, which could improve their satisfaction and reduce AC usage). If no room of that direction is available, the app falls back to any room of the correct type.
* **Output Suggestions:** The application then presents the personalized recommendations to the user (hotel staff) in a clear format. The GUI lists parameters and suggested values in a small table or text display. For our example input (2 guests, female, France, Suite, 15:30 check-in), the app might output something like:
  + Temperature (°C): 23.5
  + Dimmer Level : 6
  + Empty Window: 10.00–16.00 (meaning likely the room will be empty late morning to late afternoon)
  + Minibar: Water, Wine, Chips (suggested top 3 items to ensure are present or to offer)
  + Direction: West
  + Room No.: 2-512

This indicates the system recommends setting the thermostat to ~23.5°C and lights to 6 brightness before the guest arrives, scheduling housekeeping or HVAC adjustments between 10am-4pm when the guest is out, stocking the minibar with water, wine, and chips (which similar guests often consume), and assigning the guest a west-facing room 2-512 which aligns with the prediction.

The Room Personalizer app demonstrates an end-to-end use of our data products. It integrates **all the trained models and rules** to deliver actionable outcomes. The interface is designed to be user-friendly so that non-technical staff can use it. For instance, it uses dropdowns for gender and room type, and automatically formats outputs. This tool could serve as a blueprint for a future operational system where, at check-in or at booking time, the hotel’s system automatically applies these settings – truly customizing each room for the guest to enhance comfort while optimizing energy and resources.

**Multi-View Clustering and Correlation Analysis**

In addition to the main clustering, we performed targeted **multi-view clustering** on specific pairs or subsets of features. The idea was to examine particular relationships in isolation and verify whether intuitive groupings emerge, as well as to explore correlations directly. We also visualized certain correlations (especially related to minibar usage and environmental factors) to support our findings.

**Heat Preference (Hot vs Cold) Clustering:** We looked at the distribution of guests’ **preferred temperature settings** to see if there are natural groupings. By applying K-Means with 2 clusters to the single feature “Preferred Heat (°C)”, we essentially split guests into two groups: those who prefer it **cooler vs warmer**. Indeed, the algorithm identified a lower-temperature cluster and a higher-temperature cluster. We could label these roughly as **“Cold Lovers”** and **“Heat Lovers”**. When visualized, for example in a histogram, one cluster centered around a lower comfort temperature (e.g. around 20°C) and another around a higher temperature (e.g. 24°C). This confirms that guest comfort temperature is not one-size-fits-all – there are distinct populations, which may correlate with factors like gender or origin (we explore that later). This two-group insight is useful for HVAC management: knowing a guest’s inclination can help in presetting room temperature or grouping similar guests in similar wings.

**Lighting Preference (Light vs Dark) Clustering:** We similarly examined **lighting comfort** by considering two features together: the guest’s preferred dimmer level and the actual illumination level in the main room they used (in lumens). Using 2-cluster K-Means on this 2D space separated guests into a cluster of those who like **brightly lit rooms** and those who prefer a **darker ambiance**. For instance, one cluster had high dimmer settings and high lumens usage (bright environment lovers), while the other had low dimmer settings and often dim lighting (those who find lower light cozier or who are more energy-saving). A scatter plot of dimmer vs lumens, colored by cluster, showed a clear division between the two clusters. This insight can guide how we adjust lighting: some guests might appreciate if we draw curtains and keep lights low, whereas others might want everything well-lit.

**Family vs Non-Family Analysis:** We introduced a simple heuristic flag to identify **family groups** in our data – for example, if the reservation had multiple guests or if the room number indicated connecting rooms (in our synthetic data, perhaps rooms starting with a certain letter were family suites). We then explored clustering within family guests vs others. One approach was to see if families exhibit different temperature preferences (hot vs cold) compared to solo travelers. The data hinted that families (often with children or elderly) tended to select slightly higher temperatures on average, potentially forming a sub-cluster among the “heat lovers”. While the analysis was exploratory, it underscores that group composition (like a family vs an individual business traveler) can influence usage patterns (families might spend more time in the room, use more amenities, etc.). This could be incorporated as an additional feature or separate clustering dimension in future models.

**Gender vs Temperature Correlation:** To delve into the earlier question of gender differences in temperature preference, we plotted and clustered the data on a **gender vs preferred temperature** plane. We encoded gender as a binary variable (e.g. Female=0, Male=1) and plotted those against the preferred room temperature. By running a 2-cluster K-Means here, we could see if one cluster corresponds to mostly one gender with a certain temp range. The scatter (with a slight jitter added for gender since it's binary) and cluster centers indicated a trend: one cluster had a majority of female guests with slightly higher temperature settings, and the other had more male guests with slightly lower preferred temps. This suggests **female guests tended to prefer rooms a bit warmer than male guests on average**, which aligns with common comfort research. However, the overlap was considerable (not a strict rule for each individual), so this is a general tendency rather than an absolute. It was a useful correlation to confirm, as it informs personalized settings (for a female guest we might lean the default temperature a degree higher, for example).

**Room Orientation vs Temperature/Cooling Clustering:** We investigated how **room orientation** (North, South, East, West) interacts with temperature and cooling needs:

* First, we looked at **orientation vs the average room temperature without cooling**. Clustering on these two features effectively groups rooms (or stays) by orientation and how hot they get naturally. This analysis clearly separated, for example, south-facing and west-facing rooms in one cluster (those tend to receive more sunlight and thus have higher unmanaged temperatures) versus north/east-facing in another cluster (cooler naturally). This matches expectation: sun-facing rooms run hotter.
* Similarly, clustering **orientation vs cooling cost** grouped the data into clusters where certain orientations (south/west) had consistently higher cooling energy costs, confirming that those sides of the building incur more AC usage. East/north orientations formed a cluster with lower cooling costs. Thus, **room direction is strongly correlated with both ambient temperature and cooling energy consumption** – a logical but important relationship for planning (e.g. allocate heat-tolerant guests or better AC to west-facing rooms).

**Temperature vs Cooling Cost:** We also directly clustered the relationship between the **average temperature without cooling and the cooling cost** for each room/stay. This revealed a nearly continuous correlation: cases with higher natural temperatures generally led to higher cooling costs. Clustering might segment it into, say, four groups ranging from “low temp, low cost” up to “high temp, high cost”. The highest cluster could correspond to summer stays in sun-exposed rooms (high 30s °C without cooling, thus high AC use), whereas the lowest cluster might be winter stays or shaded rooms. This analysis underscores the intuitive correlation that maintaining comfort in hotter conditions costs more, and it quantifies those groupings.

**Lighting vs Cooling Correlation:** We examined **cooling cost vs illumination** as well. Interestingly, we found a mild positive correlation: rooms that had very high **lighting usage (higher lumens or lights left on)** also tended to show increased cooling costs. This makes sense, as lights contribute heat to the room and guests who prefer a brightly lit room may also demand cooler temperatures (or simply the lights' heat forces the AC to work harder). A cluster analysis on these two features highlighted a set of cases where both cooling and lighting were high (perhaps a cluster of guests who keep lights and AC on max, possibly indicating a comfort preference or carelessness in energy saving), and another cluster where both were low (guests who use minimal lighting and also low cooling, possibly energy-conscious individuals or those comfortable with ambient conditions). This insight suggests targeting those high usage clusters with smart automation (e.g. turning off lights when room is empty to save cooling load).

**Bathroom vs Main Room Illumination:** As a minor analysis, we compared **bathroom light usage vs main room light usage**. We wanted to see if there are clusters of guests who, for instance, like a brightly lit bathroom but a dark room, or vice versa. The clustering here did not show very strong distinct groups; generally, guests who lit the main room brightly also tended to have bright bathrooms (when in use), and those who kept things dim did so across both. There were a few exceptions (perhaps someone who likes a dim bedroom but bright bathroom for getting ready). This tells us lighting preference can be somewhat generalized per guest rather than room-specific.

**Guest Count vs Minibar Usage:** We also clustered **number of guests vs total minibar items consumed**. This revealed an intuitive grouping: stays with **more guests (e.g. families)** often fell into a cluster with higher minibar counts (they tend to consume more items in total), whereas single-guest stays typically clustered with low minibar usage (many single travelers don’t use the minibar at all). There was a middle cluster for couples or two guests with moderate usage. This simply reinforces that occupancy affects consumption – something the hotel can use for stocking and pricing (for example, expecting that a family in a room might require a larger stock of drinks and snacks).

**Market Basket Visualizations:** Complementing the association rules, we created visualizations to better understand minibar behavior:

* A bar chart of **top 10 minibar items by popularity** confirmed which items are most commonly taken (for example, bottled water might have the highest support, followed by soda, beer, etc.).
* A **heatmap of minibar item co-occurrence** showed clusters of items that tend to be taken together. We noticed, for instance, that alcoholic beverages and mixers had a notable correlation (guests taking whiskey often also took a cola mixer), and salty snacks like nuts or chips correlated with beer consumption. On the other hand, some items like champagne were more isolated (taken on special occasions, not necessarily with other items).
* The **top 10 item pairs** by support list highlighted pairs like *Water & Coffee*, *Beer & Chips*, *Wine & Chocolate* as frequent combinations. Understanding these can help in bundling offers or placing complementary items together in the minibar.

Overall, the multi-view clustering and correlation analysis provided **granular insights** confirming and extending our intuition:

* We identified clearly separable preference groups (hot/cold, light/dark) which support the idea of tailoring HVAC and lighting to guest profiles.
* We saw that demographic or group factors (gender, family size) have measurable correlations with usage (temperature and minibar).
* We validated the strong impact of physical factors like room orientation on energy consumption and found how those interact with guest behavior (lighting vs cooling).
* We visualized minibar consumption patterns to guide F&B management.  
  These insights reinforce the main analysis and suggest specific angles for improving hotel services.

**Observed Correlations**

During the data exploration, several interesting correlations were observed or logically deduced. Here we summarize some key relationships that align with expectations or reveal noteworthy patterns:

* **Guest Gender vs Temperature Preference:** Female guests tended to prefer slightly higher room temperatures than male guests. In our data, women on average set the thermostat a bit warmer (for example, 0.5–1°C higher). This reflects a common trend in comfort studies where women often feel colder than men at the same ambient temperature. It suggests the hotel might automatically set a female guest’s room a touch warmer to match comfort, or provide extra blankets, etc. (Of course, individual preferences vary, but this was a general skew in the data.)
* **Nationality vs Minibar Usage:** There were distinct minibar consumption habits across nationalities. For instance, guests from certain countries were more likely to use the minibar extensively, while others used it sparingly. As a hypothetical example, our analysis might find that **American** and **British** guests have higher minibar bills on average (perhaps grabbing snacks and drinks routinely), whereas **Japanese** or **Scandinavian** guests might use it less (perhaps due to cost sensitivity or cultural habits of not consuming as much in-room). Additionally, certain nationalities showed preferences for specific items – e.g. **French** guests might favor wine from the minibar, **Germans** might often take beer, etc. These correlations, while needing careful interpretation, can help in culturally tailoring the in-room offerings (for example, ensuring tea is available for British guests or instant noodles for certain Asian guests if those patterns were observed).
* **Room Orientation vs Temperature & Cooling:** Room direction had a direct correlation with the room’s ambient temperature and the cooling energy required. South- and west-facing rooms (with more sun exposure) consistently registered higher average temperatures without cooling and incurred greater cooling costs to maintain comfort. Conversely, north- and east-facing rooms stayed cooler and needed less air conditioning. This was evident in both correlation analysis and clustering: essentially grouping orientations into “hot side” vs “cool side” of the building. This logical correlation confirms that for energy efficiency, the hotel can account for orientation – e.g. allocate guests who prefer cooler environments to north-facing rooms, and consider investing in better insulation or shading for west-facing rooms to cut cooling costs.
* **Lighting Usage vs Cooling Cost:** We observed a positive correlation between how much a guest used the room lights and the room’s cooling energy consumption. In rooms where lights were frequently on at high brightness, the HVAC had to work harder, leading to higher cooling costs. Part of this is physics (lights add heat), and part may be behavioral (a guest who likes a brightly lit room may also prefer a colder temperature, cranking up the AC). This correlation highlights an opportunity: **integrated energy management** – for instance, the system could dim or turn off lights when the room is empty or even slightly adjust lighting when AC load is high, to balance comfort and energy use. It also suggests that informing guests about this relationship (or automatically managing it via smart systems) could improve efficiency (for example, using efficient LED lighting that emits less heat or automatically turning off lights when appropriate to reduce cooling needs).

These correlations, both observed empirically in the data and supported by logical reasoning, provided valuable validation for our models. They ensure that our predictive system aligns with real-world behavior (e.g. it makes sense that the model might use gender as a feature for temperature prediction, given the correlation, or use room orientation when predicting cooling needs). Recognizing these relationships allows the hotel to make *data-driven decisions*: whether it’s customizing room settings per guest profile or structural adjustments like improving certain room orientations, the insights tie back to tangible actions.

**Conclusion and Future Work**

**Summary of Outcomes:** In this project, we successfully leveraged data analysis to enhance hotel operations and guest experience. Through clustering, we segmented guests and rooms into meaningful categories (with K-Means yielding seven distinct clusters that encapsulated patterns of energy use and preferences). We found clear patterns – for example, identifying groups of “cold-preferring” vs “heat-preferring” guests, and high vs low resource usage segments – that can inform targeted services. Association rule mining on minibar data uncovered which amenities each segment enjoys, enabling smarter stocking and promotions. We built predictive models that can forecast an incoming guest’s comfort settings (temperature, lighting), likely behavior (when they’ll be out of the room, what they’ll consume), and even assist in assigning an optimal room. Finally, all these components were integrated into the **Room Personalizer** prototype, demonstrating a practical application where a hotel can automatically personalize a room at check-in based on data-driven recommendations. By implementing these findings, a hotel could improve guest satisfaction (rooms set just right for comfort) and reduce waste (energy savings from anticipating usage patterns and optimizing schedules).

**Future Work and Recommendations**

There are several avenues to extend and refine this work:

* **Real-Time and Dynamic Recommendations:** Currently, our models provide a one-time prediction at check-in. In the future, integrating real-time IoT sensors and feedback loops could allow dynamic adjustments. For example, motion or thermostat sensors could feed data to update predictions of empty-room intervals on the fly, or to adjust lighting and AC in real-time as usage deviates from the initial forecast. A reinforcement learning system could even continuously learn and adapt settings during a guest’s stay to keep optimizing comfort and efficiency.
* **Enhanced Minibar Personalization:** With more detailed minibar data (e.g. timestamps of consumption, billing data, or restock requests), we could improve association rules and even predict not just *what* a guest will consume but *when*. Future work could incorporate a wider array of F&B options and even recommend welcome gifts or room service items. Additionally, the hotel could use these insights to tailor the in-room minibar inventory to the guest’s profile – for instance, pre-stocking a favorite snack for a loyal guest or adjusting the mix of items based on the predicted preferences (reducing waste of unpopular items).
* **Room Assignment Optimization:** We showed a basic approach to recommend room orientation and a suitable room number. This can be expanded into an optimization algorithm that considers all available rooms and assigns each incoming guest to the room that best matches their predicted preferences and the hotel’s energy efficiency goals. For instance, a solver could ensure that guests who like it cool are placed in cooler rooms, balancing out occupancy so that no wing is over-stressed on cooling. Future systems might also consider other factors like noise levels (if a guest is predicted to spend a lot of daytime in room, assign a quieter side), view preferences (some clusters might correlate with view requests), etc. Essentially, data-driven room allocation could become a feature of a smart hotel PMS (Property Management System).
* **Implementation and Testing:** Finally, a crucial next step is to pilot these insights in a real or simulated hotel environment. By deploying the Room Personalizer system in a small scale and monitoring results, we can validate whether the predicted settings indeed save energy and please guests. A/B testing could be done – e.g. one set of rooms use the AI recommendations, others standard settings, and compare energy usage and guest feedback. This would provide concrete evidence of the benefits and highlight any adjustments needed (perhaps the models might need some bias correction or more human oversight initially).

**Conclusion**

In conclusion, this project has demonstrated the potential of data-driven decision support in hospitality. By analyzing and clustering operational data, applying machine learning to predict guest needs, and integrating these into a user-friendly tool, we pave the way for smarter hotels. Personalized guest experiences can be delivered at scale (each guest gets a room tuned to their liking upon arrival), and operational efficiencies can be achieved (reducing energy waste and optimizing resources) – a win-win outcome. The future work will focus on refining these models, expanding their scope, and seamlessly embedding them into hotel workflows, moving us closer to truly intelligent and responsive hotel rooms.