Data Analysis 3: Assignment 2 - Technical Report

Github Repo: https://github.com/MarcellM01/Data-Analysis-3/tree/main/Assignment_2

Overview of functionality:

This report aims to present a comprehensive analysis of a code segment designed for data processing, exploratory data analysis (EDA), and predictive modelling using machine learning techniques. The primary focus of this code is to import, clean, and analyse Airbnb listing data to predict rental prices effectively. This report aims to focus on the technical aspects of the analysis as opposed to drawing conclusions from its results. The code black highlighted will not be those most complex, but rather those that dealt with unforeseen challenges, for further detail and granularity, please consult the full codebase, available at the GitHub link above.

Data Import and Cleaning

The initial phase of the project commenced with the importation of the Airbnb dataset directly from a GitHub repository, where the Airbnb Mallorca dataset was downloaded to and stored. This dataset drawn from the official Airbnb website, consists of 18,832 entries and 75 features, encapsulating a diverse range of information regarding Airbnb listings, including details about the accommodations, host information, review scores, and more. The cleaning process was meticulously designed to refine the dataset for a focused analysis. One of the first steps involved filtering listings to only include those that accommodate between 2 to 6 guests, as per the client requirement.

Following the accommodation filter, the dataset was further pruned to retain only relevant columns essential for the modelling process. A total of 16 crucial features were selected, including 'accommodates', 'beds', 'review_scores_rating', 'host_is_superhost', 'latitude', 'longitude', 'number_of_reviews', 'availability_365', 'minimum_nights', 'maximum_nights', 'property_type', 'room_type', 'price', 'amenities', 'neighbourhood_cleansed', and 'id'. This selection aimed to balance between providing a comprehensive view of the listings while eliminating redundant or less impactful information.

A significant step in the cleaning process was the transformation of the 'price' column. Originally stored as a string with currency symbols and commas (e.g., "$1,000"), these values were converted to numerical format to facilitate quantitative analysis. This conversion involved removing non-numeric characters and converting the cleaned strings into integers. Additionally, the dataset was filtered to exclude listings with a price of 0 or lower, ensuring the analysis focused on economically viable listings.

```
# Price column cleanup
processed_df['price'] = processed_df['price'].replace({'\$': '', ',': ''}, re-
gex=True).astype(float)
processed_df.dropna(subset=['price'], inplace=True)
processed_df = processed_df[processed_df['price'] > 0]
processed_df['price'] = processed_df['price'].astype(int)
```

Lastly, to maintain analytical coherence and relevance, the dataset was filtered to exclude rare property types. This was achieved by retaining only those property types with at least 100 listings, thereby focusing on the most common property types in the dataset.

```
# Filter out rare property types
property_counts = processed_df['property_type'].value_counts()
common_properties = property_counts[property_counts >= 100].index
processed_df = processed_df[processed_df['property_type'].isin(common_properties)]
```

Exploratory Data Analysis (EDA)

The cleaned dataset, now refined to 12,226 entries, offered a wealth of information through its descriptive statistics, please see initial descriptive statistics below for numerical only:

	accommodates	beds	review_scores_rating	latitude	longitude	number_of_reviews	availability_365	minimum_nights	maximum_nights	price	id
count	12226.000000	12157.000000	9354.000000	12226.000000	12226.000000	12226.000000	12226.000000	12226.000000	12226.000000	12226.000000	1.222600e+04
mean	4.494111	3.382249	4.648576	39.665043	3.004983	20.552756	193.950679	4.321773	702.568542	253.615901	2.591599e+17
std	1.500813	1.483669	0.508628	0.171349	0.243130	43.111580	126.688208	10.723649	480.353550	741.806476	3.794609e+17
min	2.000000	1.000000	0.000000	39.302070	2.347260	0.000000	0.000000	1.000000	1.000000	10.000000	1.068330e+05
25%	4.000000	2.000000	4.520000	39.551578	2.846342	1.000000	76.000000	1.000000	200.000000	125.000000	2.254076e+07
50%	4.000000	3.000000	4.800000	39.687100	3.054715	6.000000	198.000000	3.000000	1125.000000	183.000000	4.449842e+07
75%	6.000000	4.000000	4.970000	39.823498	3.147353	21.000000	318.000000	5.000000	1125.000000	264.000000	6.688053e+17
max	6.000000	14.000000	5.000000	39.921540	3.471520	1172.000000	365.000000	365.000000	1125.000000	50000.000000	9.766218e+17

These observations where further elaborated upon which included:

- **Accommodates**: The average number of guests accommodated was approximately 4.49, with a standard deviation of 1.50, indicating a moderate variance in the size of groups seeking rentals.
- **Price**: Prices varied significantly among listings, with a mean value of approximately $253.62. However, the standard deviation was quite high at $741.80, suggesting a wide range of rental prices. The minimum price was $10, with the 25th percentile at $125, the median at $183, and the 75th percentile at $264. The maximum price spiked to $50,000, highlighting extreme outliers in the dataset.
- **Reviews**: The number of reviews per listing also varied widely, with an average of 20.55 reviews and a standard deviation of 43.11, reflecting the diverse popularity and engagement levels of listings.

A key part of the EDA was addressing outliers in the 'price' variable. Through percentile analysis, it was observed that 99% of the listings were priced at or below $1,000, with a significant jump to $50,000 for the maximum price. To normalize the price distribution and mitigate the influence of extreme outliers, listings with prices beyond the interquartile range (IQR) were removed, focusing on a more representative price range for the analysis.

```
# Dealing with extreme values, create a function to remove them
def remove_outliers(df, column):
    return df[((df[column] >= df[column].quantile(0.25) - 1.5 *
                (df[column].quantile(0.75) - df[column].quantile(0.25))) &
               (df[column] <= df[column].quantile(0.75) + 1.5 *
                (df[column].quantile(0.75) - df[column].quantile(0.25))))]

# Remove extreme values from 'price'
cleaned_data = remove_outliers(cleaned_data, 'price')

cleaned_data['price'].describe()
```

The exploration of room and property types revealed insightful trends.

- **Room Type**: The dataset predominantly consisted of 'Entire home/apt' listings, accounting for a significant majority. This was followed by 'Private room' and a small percentage of 'Hotel room' listings, indicating a clear preference or availability trend towards entire homes or apartments for Airbnb listings.
- **Property Type**: Among the property types, 'Entire rental unit', 'Entire home', and 'Entire villa' were the most common, underscoring the variety of property types preferred by hosts and sought after by guests. This variety reflects the diverse needs and preferences of Airbnb users, from individual travelers to larger groups seeking different accommodation experiences.

Feature Engineering

Amenities provided by Airbnb listings were initially represented in a list format within each entry. From the initial analysis, a wide array of amenities was identified. However, for the sake of model simplicity and effectiveness, a selection of amenities was made based on their prevalence and potential impact on rental prices. This selection included features like "BBQ grill," "Free parking on premises," "Heating," and "Private entrance," among others. For instance, the presence of a "Pool" was deemed significant enough to be featured as a binary variable given its potential desirability to renters and impact on price.

To utilize this information effectively, the amenities were parsed and transformed into binary variables. For example, amenities such as "Wi-Fi," "Pool," "Air conditioning," and "Kitchen" were identified from the listings. Each of these amenities was then converted into a separate feature with a binary value (1 for presence and 0 for absence). This transformation allowed for a granular analysis of how each amenity impacts the rental price.

```
selected_amenities = [
        "BBQ grill", "Free parking on premises", "High chair", "Heating", "Private
entrance", "Crib", "Freezer",
        "Stove", "First aid kit", "Free street parking", "Toaster", "Hot water
kettle", "Fire extinguisher",
        "Patio or balcony", "Bathtub", "Backyard", "Outdoor furniture", "Pool",
"Dedicated workspace",
        "Self check-in", "Outdoor dining area", "Dining table", "Long term stays
allowed",
        "Extra pillows and blankets", "Private patio or balcony", "Lockbox", "Wine
glasses",
        "Drying rack for clothing", "Room-darkening shades", "Smoke alarm", "Indoor
fireplace", "Mountain view",
        "Shampoo", "Garden view", "Smoking allowed"
    ]

# Create dummy variables for each selected amenity
for amenity in selected_amenities:
    column_name = f"amenity_{amenity.replace(' ', '_').lower()}"
    df[column_name] = df['amenities'].str.contains(amenity, case=False,
na=False).astype(int)
```

The diversity in property and room types presented a challenge for numerical modelling. To address this and make it suitable for one-hot encoding further down the line, categorical data was converted into a format that could be provided to machine learning algorithms effectively.

This process created a binary column for each category of the 'property_type' and 'room_type' features. The feature engineering process significantly enriched the dataset, providing a robust foundation for the subsequent modelling phase. By transforming the 'amenities' into binary variables, the models could understand and leverage the nuanced impact of various amenities on rental prices. Similarly, the transforming of 'property_type' and 'room_type' allowed the models to differentiate between the diverse types of listings in the dataset without conflating them into a single numerical spectrum.

Model Development

Ordinary Least Squares (OLS)

The implementation of the OLS model followed the creation of design matrices, which structured the predictor variables (X) and the target variable (price) in a format suitable for linear regression analysis. This step ensured that the model could effectively learn from the cleaned and engineered dataset.

The OLS model's RMSE was calculated to be approximately 70.26, serving as an initial gauge of model performance. This value represents the average error in the model's price predictions, indicating how closely the model's predictions align with the actual rental prices in the dataset.

Classification and Regression Trees (CART)

Following the baseline established by the Ordinary Least Squares (OLS) model, the project explored the use of Classification and Regression Trees (CART) as a more flexible approach to modeling the Airbnb rental prices. CART models are capable of capturing nonlinear relationships and interactions between variables, offering a potentially more accurate depiction of the factors influencing rental prices. To optimize the CART model, a crucial step involved tuning the model's complexity parameters, specifically the cost complexity pruning parameter (ccp_alpha). This parameter controls the tree's growth to prevent overfitting by pruning less important branches. The tuning was conducted through a cross-validation process using the RandomizedSearchCV method, which explored a range of ccp_alpha values to find the optimal setting that minimizes prediction error.

The tuning process identified an optimal ccp_alpha value that minimized the Root Mean Squared Error (RMSE) of the CART model. The optimized CART model achieved an RMSE of approximately 79.47. The RMSE value obtained from the CART model, while higher than the baseline OLS model, reflects the model's ability to capture more complex patterns in the data. However, the increase in RMSE suggests that despite its flexibility, the CART model may not have fully captured the underlying relationships within the Airbnb dataset.

Gradient Boosting Machine (GBM)

Continuing the exploration of ensemble methods, the project implemented a Gradient Boosting Machine (GBM) model. GBM is renowned for its ability to produce highly accurate predictive models by sequentially building decision trees, where each subsequent tree aims to correct the errors of its predecessor. This technique effectively minimizes prediction errors over iterations, making GBM a powerful tool for complex regression tasks.

To harness the full potential of the GBM model, a meticulous hyperparameter tuning process was conducted using Grid Search CV. This method involved evaluating a range of hyperparameter values to find the combination that yields the best performance, as measured

by the Root Mean Squared Error (RMSE). Key hyperparameters tuned included n_estimators, representing the number of trees in the forest, and max_depth, determining the maximum depth of the trees. The search considered n_estimators values of 200 and 300, and max_depth values of 5 and 10, aiming to identify an optimal balance between model complexity and overfitting risk.

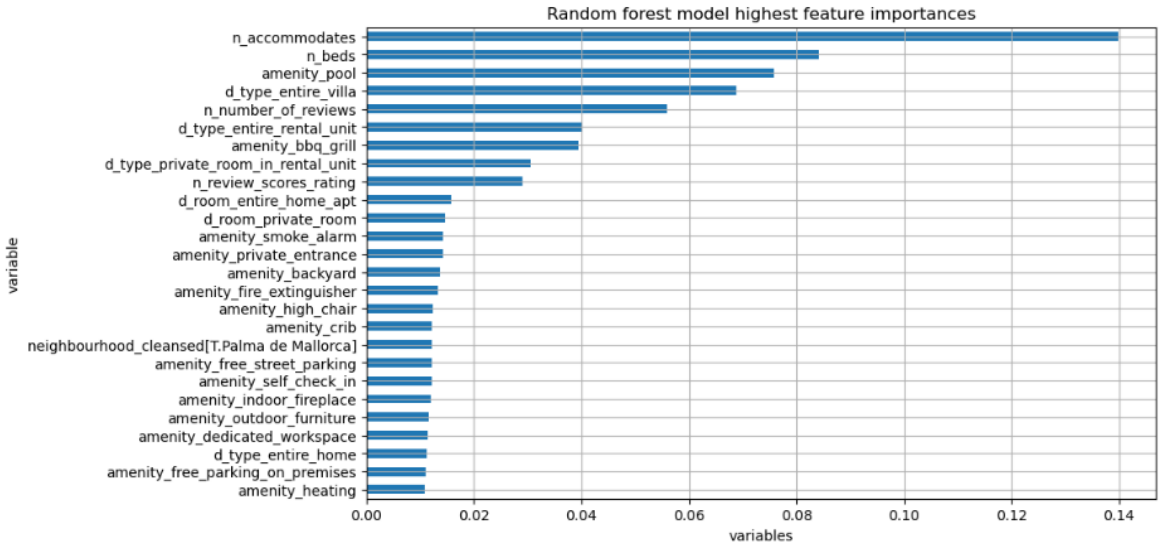
This optimized GBM model achieved an RMSE of approximately 65.01 on the validation set. This result represents a notable improvement over both the baseline OLS model and the other ensemble methods previously applied, such as the Random Forest model. The GBM's performance underscores its capability to capture the intricate relationships within the Airbnb dataset accurately.

Random Forest

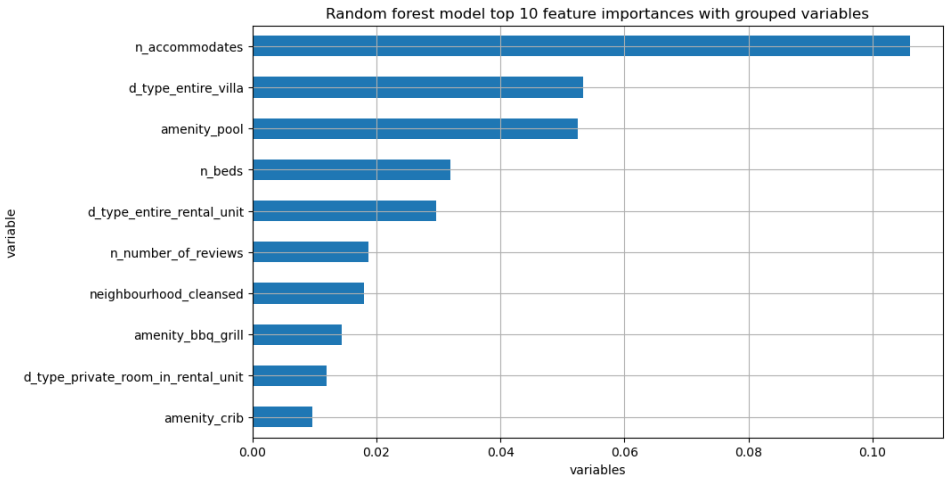
Advancing from the Classification and Regression Trees (CART) model, the project incorporated the Random Forest algorithm, an ensemble method known for its prediction accuracy. To maximize the Random Forest model's performance, hyperparameter tuning was executed using Grid Search CV. This process involved systematically working through multiple combinations of parameter values, cross-validating as it goes to determine which tune gives the best performance.

The grid search explored combinations of max_features ranging from 6 to 12 and min_samples_leaf from 5 to 15. This comprehensive search aimed to find the optimal balance between the model's complexity and its ability to generalize across new data. The grid search identified the optimal combination of hyperparameters that minimized the Root Mean Squared Error (RMSE) of the Random Forest model on the validation data. The best-performing model utilized max_features=12 and min_samples_leaf=5, indicating that considering a broader range of features at each split and maintaining a moderate level of leaf granularity led to the most accurate predictions.

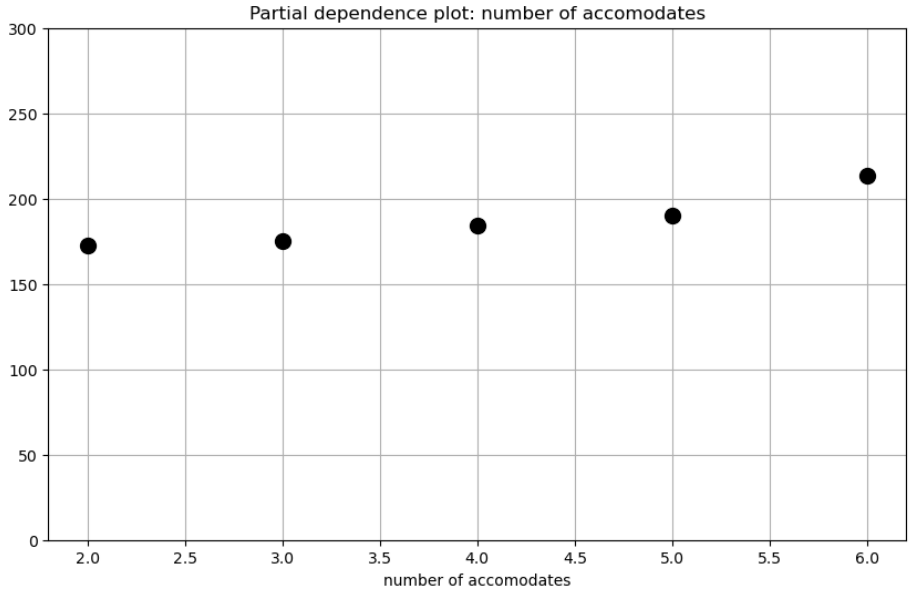
The graph bellow shows the variables with a more than 0.01 pct relevant. The capacity to accommodate guests is the most influential, followed by the number of beds, the inclusion of a pool, and the property being an entire villa. Guest reviews also play a significant role, indicating the combined impact of a property's attributes and guest experiences on pricing.



The graph below shows the relevant variables again, however now with the groupings created. Guest accommodation capacity tops the list, followed by the property being classified as an entire villa and having a pool. The number of beds and the type of rental unit also figure prominently, suggesting the significance of property size and amenities. Reviews and neighbourhood details are important as well, reflecting guest preferences and the influence of location.



The Partial Dependence Plot illustrates that as an Airbnb listing's capacity increases, so does its rental price, emphasizing the importance of accommodation size in pricing strategy. This visual representation, grounded in our Random Forest model's analysis, confirms that listings catering to more guests can command higher prices, providing a key insight for property owners and platform pricing algorithms.



The tuned Random Forest model achieved an RMSE of approximately 68.75 on the validation set. This performance represents a significant improvement over the baseline OLS model and the CART model, showcasing the Random Forest's effectiveness in handling the dataset's complexity and variability.

Model Evaluation

The comparison of RMSE scores across the models unequivocally demonstrates a trend where ensemble methods, specifically Random Forest with an RMSE of 67.81 and GBM with an RMSE of 65.01, significantly outperformed the more straightforward OLS and CART models, which recorded RMSE scores of 70.26 and 79.47, respectively. This pattern underscores the effectiveness of ensemble methods in navigating the complexities of datasets marked by nonlinear relationships and multifaceted feature interactions.

The standout performance of the GBM model, with its lowest RMSE of 65.01, highlights the critical advantage of adaptively refining predictions through learning from previous errors. This approach, coupled with precise hyperparameter tuning, elevates the GBM model as a premier choice for predictive modeling across diverse and intricate data landscapes. Such adaptability ensures that GBM remains sensitive to the dataset's underlying patterns, making it a powerful tool for achieving high prediction accuracy.

In essence, the model evaluation phase sheds light on the relative efficacy of various predictive modeling techniques, with the RMSE metric serving as a crucial comparative tool. The results distinctly illustrate the superiority of ensemble methods, particularly GBM, in yielding higher prediction accuracy. These insights not only guide the selection of the most appropriate model for forecasting Airbnb rental prices but also enrich the broader discourse on effective modeling strategies for complex datasets encountered in real-world scenarios.

	model	CV RMSE
0	OLS	70.256456
1	CART	79.473124
2	random forest	67.810000
3	GBM	65.012152